CONFERENCE PROCEEDINGS

SEPTEMBER 17-21

EMNLP 2015

CONFERENCE ON EMPIRICAL METHODS IN NATURAL LANGUAGE PROCESSING

LISBON

contact@emnlp2015.org

WWW.EMNLP2015.ORG

PLATINUM SPONSORS

Bloomberg Baidu 百度

GOLD SPONSORS

Google facebook LinkedIn

SILVER SPONSORS

RECRUIT Recruit Technologies Co., Ltd. NUANCE amazon AI @ ISI VoiceBox TECHNOLOGIES DigitalGenius Artificial Intelligence for Brands

EMNLP 2015 gratefully acknowledges the following sponsors for their support:

Platinum

Gold

Silver

Bronze

Best Student Paper

Supporters

Institutional partners

Official Airline

Order print-on-demand copies from:

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

ISBN: 978-1-941643-32-7

Table of Contents

Preface by the General Chair

August 25, 2015

Welcome to the 2015 Conference on Empirical Methods in Natural Language Processing. EMNLP is annually organized by SIGDAT, the Association for Computational Linguistics' special interest group on linguistic data and corpus-based approaches to NLP. This year the conference will be held on September 17–21 in the enchanting city of Lisbon, Portugal.[1]

EMNLP has continued to increase in prominence as one of the most important conferences in Natural Language Processing (NLP). This year the conference has experienced an unprecedented boost in submitted papers. I believe that this reflects both the growth of the NLP field and also the health and strength of the conference itself, with a history of many years of solid work. With this level of interest at submission time, we are also expecting a record attendance. The conference will span a five-day period this year, and it requires a growing organization structure.

Some of the features introduced in EMNLP 2014 will continue this year (e.g., tutorials, new chairs, posters as parallel sessions, flat rates and flexibility for tutorials and workshops, etc.). We also introduce some innovations, like a revised selection process for which talks are presented as talks versus posters.

This year I had the privilege of coordinating the conference from my General Chair position. This has been a very instructive and enriching exercise which showed me the conference as a whole, from many different angles. Prefaces in the proceedings invariably praise the team of organizers. This one will not be an exception. Organizing a large conference as EMNLP requires excellent people working as a team in multiple interrelated tasks. I have been lucky to work with an outstanding team of people, from whom I learnt a lot. These aren't empty words. I would like to thank each and every chair for the hard work that made the conference a reality.

The Program Chairs, Jian Su and Chris Callison-Burch, did an excellent job at putting together a very interesting program with over 300 papers. They had to deal with a very large number of submissions, which exceeded even our most optimistic expectations. As a consequence, they were forced to be creative and to find solutions on the fly to adapt to the situation. They recruited the largest ever program committee and successfully managed a huge reviewing and decision making process under a very tight schedule. A real gift for the general chair. They complemented the program with very interesting keynote speakers, Yoshua Bengio and Justin Grimmer who will present exciting research topics for our community.

The EMNLP 2015 main conference is accompanied by 7 workshops and 8 tutorials during the first two days. The Workshops Chairs, Zornitsa Kozareva and Jörg Tiedemann, and the Tutorials Chairs, Maggie Li and Khalil Sima'an, conducted the selection processes in a joint effort with the other ACL conferences in 2015 (NAACL and ACL-IJCNLP). This has been the standard procedure from last years. It has the advantage of starting early, avoiding duplicated reviewing and allowing a more balanced selection among conferences. EMNLP attracted a varied and interesting set of workshops and tutorials, which gives more value to the conference.

Daniele Pighin and Yuval Marton were responsible for the always difficult and sometimes thankless task of putting together the conference publications. This is a very complex effort which involves coordination with almost everyone in the team under the pressure of hard publication deadlines. Yuval is serving in this position for a second year. Staggered two year terms for publication chairs is a new addition for

[1] Conference website: `http://www.emnlp2015.org`

EMNLP starting this year, and we hope that it will be a permanent feature. In the first year, publication chairs will learn and do the bulk of proceedings compilation. During the second year their role will be more advisory, instructing and helping the first-year chair. This procedure will help the transmission of the necessary know-how from year to year. Thanks to Yuval and Daniele for accepting the challenge and making it work wonderfully. Finally, this is the second year that EMNLP uses a mobile app for the conference program (Conference4me). The publication chairs also coordinated the integration of the app with SoftConf, which is now smoother and more seamless.

The local organization team was led by André Martins and João Graça. They did an amazing job, working hard and with all the complexities and subtleties of local arrangements. One of the keys for the success was the creation of a large team of local organizers with clearly defined roles and responsibilities. They appointed very committed people: Isabel Trancoso (Local Publicity Chair), Fernando Batista (Handbook Chair), Bruno Martins (Website and App Chair), Luísa Coheur (Student Volunteer Coordinator), and Helena Moniz (Local Arrangements Chair). Thanks to all. I am especially pleased about the new website, which was revamped and looks more professional everyday. This is certainly a good investment for the future.

A large conference as EMNLP needs to focus on dissemination activities too. Barbara Plank acted as the international Publicity Chair. She did a fantastic job and coordinated very well with the local publicity and the website chairs. The calls for papers, calls for participation, and main news of the conference were timely distributed through ACL, the usual distribution lists, and also through the conference website and two Facebook and Twitter accounts. The EMNLP2015 Twitter account garnered more followers than in previous years.

I am really grateful to SIGDAT. Its secretary, Noah Smith, acted as the liasion between SIGDAT and the conference organizers. He was always available and ready to help with very good advice. SIGDAT also provided the funds for the student scholarship program. These grants help covering traveling expenses to a dozen of students. The committee appointed for collecting the applications and making the decisions was formed by Francisco Guzmán and Lluís Padró, who had to analyze all the information and decide the awardees in only a few days.

Another sign of the health of EMNLP and the field in general is the interest showed by sponsors. Thanks to the work of our sponsorship team, formed by João Graça and Hang Li, in coordination with the ACL International Sponsorship Committee, we got a record number of 13 sponsors for EMNLP 2015 (2 platinum, 3 gold, 6 silver and 2 bronze). In addition to these direct sponsors, we also have several smaller supporters, exhibitors, and institutional partners. We are extremely grateful to all these companies and institutions, which make a better conference possible at a more affordable registration fee.

Additionally, we counted on the invaluable help of Priscilla Rasmussen, supporting the local organization in all fronts with her broad experience. She took care of the registration process too. We also got very good advice, know-how, and helpful software and forms from last year general chair and local organizers, Alessandro Moschitti and Kareem Darwish. Thank you.

Finally, I would like to thank the authors of submitted and accepted papers, and all the attendees to the conference, who will be the main actors from September 17 to September 21, 2015. I am convinced that we will experience a fantastic conference, scientifically exciting and full of fond memories, in the unique environment of Lisbon.

Lluís Màrquez
EMNLP 2015 General Chair

Preface by the Program Committee Co-Chairs

August 25, 2015

Welcome to the 2015 Conference on Empirical Methods in Natural Language Processing! This year we received a record number of submissions. There were 1300 valid submissions. The 600 long papers and 700 short papers were allocated to one of 15 areas. The most popular areas this year were Semantics, Statistical Models and Machine Learning Methods, Text Mining and NLP applications, and Machine Translation.

Reviewing for a conference this size involves an enormous volunteer effort from many individuals. We are very grateful to our 30 area chairs and to the more than 900 researchers who reviewed the submissions. We accepted 312 papers (157 long and 155 short papers), representing a global acceptance rate of 24%. An additional 17 papers accepted by the TACL journal were presented at the conference as well.

To decide whether the accepted papers should be presented as talks or posters, we asked the area chairs, the reviewers, and the authors of accepted papers to vote on which papers they would like to attend. We showed the title of each paper and its abstract, but not its authors. 400 people provided their input. We selected talks based on popularity, while ensuring that each area was represented by at least one session. Our rationale for taking a vote was that papers that many people wanted to attend would be better served by presenting a talk in a large room, while papers with more specialized interest would benefit from the one-on-one interactions facilitated by posters. Rather than doing large plenary poster sessions, we have scheduled two parallel poster sessions with small batches of thematically similar papers that will be run simultaneously with the talks.

We selected best papers from a shortlist of 20 papers that were nominated by the area chairs. The best paper committee ranked the nominees, and based on their rankings we selected the following papers for the best paper awards:

- Best paper - *Broad-coverage CCG Semantic Parsing with AMR* by Yoav Artzi, Kenton Lee and Luke Zettlemoyer.
- Best paper - *Semantically Conditioned LSTM-based Natural Language Generation for Spoken Dialogue Systems* by Tsung-Hsien Wen, Milica Gasic, Nikola Mrkšić, Pei-Hao Su, David Vandyke and Steve Young.

IBM has provided a cash scholarship for us to award to the best student paper. This will go to Tsung-Hsien Wen, since he is currently a student. The following papers received an honorable mention for the best paper award:

- Honorable mention for best paper - *Traversing Knowledge Graphs in Vector Space* by Kelvin Guu, John Miller and Percy Liang.
- Honorable mention for best paper - *Building a shared world: mapping distributional to model-theoretic semantic spaces* by Aurélie Herbelot and Eva Maria Vecchi.
- Honorable mention for best paper - *Language Understanding for Text-based Games using Deep Reinforcement Learning* by Karthik Narasimhan, Tejas Kulkarni and Regina Barzilay.
- Honorable mention for best short paper - *Joint Lemmatization and Morphological Tagging with Lemming* by Thomas Müller, Ryan Cotterell, Alexander Fraser and Hinrich Schütze.

- Honorable mention for best short paper - *Semi-Supervised Bootstrapping of Relationship Extractors with Distributional Semantics* by David S. Batista, Bruno Martins and Mário J. Silva.

This year we created a new "Best data set or resource" award, since so much work in our community is driven by data. The paper that receiving this inaugural distinction is:

- Best data set or resource - *A large annotated corpus for learning natural language inference* by Samuel R. Bowman, Gabor Angeli, Christopher Potts and Christopher D. Manning.

With two honorable mentions:

- Notable data set or resource - *That's So Annoying!!!: A Lexical and Frame-Semantic Embedding Based Data Augmentation Approach to Automatic Categorization of Annoying Behaviors using #petpeeve Tweets* by William Yang Wang and Diyi Yang.
- Notable data set or resource - *Modeling Reportable Events as Turning Points in Narrative* by Jessica Ouyang and Kathy McKeown.

We decided to give more awards than in past years by recognizing papers with honorable mentions and by creating the new best data or resource award. Our goal was to recognize roughly the top 1% of all of the submissions to the conference with awards (recognizing approximately the top 5% of accepted papers). We are very grateful to our invited speakers Yoshua Bengio and Justin Grimmer.

Yoshua Bengio is professor of Computer Science and Operations Research at the Université de Montréal. He is the author of two books and more than 200 publications, the most cited being in the areas of deep learning, recurrent neural networks, probabilistic learning algorithms, natural language processing and manifold learning. He co-directs the Canadian Institute for Advanced Research's program on deep learning. He is on the board of NIPS. Professor Bengio's research into deep learning has had a dramatic impact on the field of NLP in the past few years, and has invigorated interest in AI through machine learning.

Justin Grimmer is an associate professor of Political Science at Stanford University. His research uses statistical methods to examine American politics. He is the author of two books on the topic "Representational Style in Congress: What Legislators Say and Why It Matters" and "The Impression of Influence: How Legislator Communication and Government Spending Cultivate a Personal Vote." His work has appeared in the American Political Science Review, American Journal of Political Science, Journal of Politics, Political Analysis, Proceedings of the National Academy of Sciences, Regulation and Governance, and Poetics. Professor Grimmer's research points to exciting new directions for computational social science and how the field of NLP can facilitate research in many areas.

We thank them in advance for coming to the conference and sharing their insights.

We would also like to thank our general chair Lluís Màrquez, André Martins and João Graça and colleagues for their excellent work with the local organization, and Yuval Marton and Daniele Pighin for doing an excellent job assembling these proceedings.

We thank SIGDAT for inviting us to serve as Program Co-Chairs of EMNLP 2015. We hope that the conference is an excellent one. Enjoy your stay in Lisbon!

Chris Callison-Burch and Jian Su
EMNLP 2015 Program Committee Co-Chairs

Organizing Committee

General Chair
Lluís Màrquez, Qatar Computing Research Institute

Program Co-Chairs
Chris Callison-Burch, University of Pennsylvania
Jian Su, Institute for Infocomm Research (I2R)

Workshops Co-Chairs
Zornitsa Kozareva, Yahoo! Labs
Jörg Tiedemann, University of Helsinki

Tutorial Co-Chairs
Maggie Li, Hong Kong Polytechnic University
Khalil Sima'an, University of Amsterdam

Publication Co-Chairs
Daniele Pighin, Google Inc.
Yuval Marton, Microsoft Corp.

Publicity Chair
Barbara Plank, University of Copenhagen

Sponsorship Team
João Graça, Unbabel Inc.
Hang Li, Huawei Technologies (ISC Representative for EMNLP)

Student Scholarships Co-Chairs
Fancisco Guzmán, Qatar Computing Research Institute
Lluís Padró, Technical University of Catalonia

SIGDAT Liaison
Noah Smith, University of Washington

Local Organizing Committee

Local Co-Chairs
André Martins, Priberam
João Graça, Unbabel Inc.

Local Publicity Chair
Isabel Trancoso, University of Lisbon

Handbook Chair
Fernando Batista, University Institute of Lisbon (ISCTE-IUL)

Website and App Chair
Bruno Martins, University of Lisbon

Student Volunteer Coordinator
Luísa Coheur, University of Lisbon

Local Arrangements Chair
Helena Moniz, Instituto de Engenharia de Sistemas e Computadores - I&D (INESC-ID)

Program Committee

Program Co-Chairs

Chris Callison-Burch, University of Pennsylvania
Jian Su, Institute for Infocomm Research (I2R)

Area Chairs

Computational Psycholinguistics
Vera Demberg, Saarland University

Discourse, Dialogue, and Pragmatics
Kentaro Inui, Tohoku University
Oliver Lemon, Heriot Watt University

Information Extraction
Heng Ji, Rensselaer Polytechnic Institute
Sebastian Riedel, University College London
Jing Jiang, Singapore Management University

Information Retrieval and Question Answering
Zhu Xiaoyan, Tsinghua University
Liu Ting, Harbin Institute of Technology

Language and Vision
Julia Hockenmaier, University of Illinois Urbana Champaign

Machine Translation and Multilinguality
Zhang Min, SooChow University
Lucia Specia, University of Sheffield
Marine Carpuat, University of Maryland

NLP for the Web and Social Media, including Computational Social Science
Alice Oh, KAIST
Brendan O'Connor, University of Massachusetts Amherst

Phonology, Morphology, and Word Segmentation
Greg Kondrak, University of Alberta
Xu Sun, Peking University

Semantics
Marco Baroni, University of Trento
Shiqi Zhao, Baidu
Percy Liang, Stanford University

Sentiment Analysis and Opinion Mining
Ku Lun-Wei, Institute of Information Science, Academia Sinica
He Yulan, Aston University

Spoken Language Processing
Mari Ostendorf, University of Washington

Statistical Models and Machine Learning Methods
Trevor Cohn, University of Melbourne
Jordan Boyd-Graber, University of Colorado

Summarization and Generation
Meg Mitchell, Microsoft Research
Furu Wei, Microsoft Research Asia

Tagging, Chunking, Syntax and Parsing
Shay Cohen, University of Edinburgh
Yue Zhang, Singapore University of Technology and Design
Yusuke Miyao, National Institute of Informatics, Japan

Text Mining and NLP applications
Sophia Ananiadou, University of Manchester

Best Paper Award Committee
Douglas Downey, Northwestern University
Kevin Knight, University of Southern California
Qun Liu, Dublin City University
Miles Osborne, Bloomberg
Mari Ostendorf, University of Washington
Xiaojun Wan, Peking University
Janyce Weibe, University of Pittsburgh

Primary Reviewers

Balamurali A R; Omri Abend; Erman Acar; Karteek Addanki; Stergos Afantenos; Željko Agić; Lars Ahrenberg; Vicent Alabau; Mohammed Alam; Nikolaos Aletras; Alexandre Allauzen; Mariana S. C. Almeida; Yasemin Altun; Bharat Ram Ambati; Hadi Amiri; Waleed Ammar; Sophia Ananiadou; Jacob Andreas; Nicholas Andrews; David Andrzejewski; Gabor Angeli; Roitberg Anna; Stephen Anthony; Marianna Apidianaki; Ioannis Arapakis; Yuki Arase; Mihael Arcan; Yoav Artzi; Isabelle Augenstein; Eleftherios Avramidis; AiTi Aw; Wilker Aziz; Amittai Axelrod;

Xiao Bai; JinYeong Bak; Timothy Baldwin; Kalika Bali; Miguel Ballesteros; David Bamman; Pratyush Banerjee; Mohit Bansal; Libby Barak; Marco Baroni; Regina Barzilay; Roberto Basili; Emanuele Bastianelli; Fernando Batista; Riza Theresa Batista-Navarro; Daniel Bauer; Daniel Beck; Beata Beigman Klebanov; Núria Bel; David Belanger; Yonatan Belinkov; Islam Beltagy; Emily M. Bender; Meriem Bendris; Fabrício Benevenuto; Kepa Bengoetxea; Jonathan Berant; Taylor Berg-Kirkpatrick; Raffaella Bernardi; Nicola Bertoldi; Steven Bethard; Klinton Bicknell; Lidong Bing; Alexandra Birch; Arianna Bisazza; Yonatan Bisk; Alan W Black; Frédéric Blain; Eduardo Blanco; Bernd Bohnet; Gemma Boleda; Francis Bond; Kalina Bontcheva; Georgeta Bordea; Elizabeth Boschee; Houda Bouamor; Gerlof Bouma; Samuel R. Bowman; Jordan Boyd-Graber; S.R.K. Branavan; Jonathan Brennan; Chris Brew; Ted Briscoe; Chris Brockett; William Bryce; Christian Buck; Razvan Bunescu; David Burkett; David Buttler; Antoine Bordes;

José G. C. de Souza; Aoife Cahill; Chris Callison-Burch; Berkant Barla Cambazoglu; Nicola Cancedda; Marie Candito; Hailong Cao; Cornelia Caragea; Marine Carpuat; Xavier Carreras; Ben Carterette; Francisco Casacuberta; Tommaso Caselli; Vittorio Castelli; Giuseppe Castellucci; Asli Celikyilmaz; Daniel Cer; Mauro Cettolo; Arun Chaganty; Nathanael Chambers; Yee Seng Chan; Baobao Chang; Jonathan Chang; Kai-Wei Chang; Ming-Wei Chang; Yin-Wen Chang; Wanxiang Che; Bin Chen; Boxing Chen; Chen Chen; Danqi Chen; Hsin-Hsi Chen; Liwei Chen; Wenliang Chen; Yun-Nung Chen; Zhiyuan Chen; Colin Cherry; Sean Chester; Jackie Chi Kit Cheung; David Chiang; Christian Chiarcos; Hai Leong Chieu; Do Kook Choe; Eunsol Choi; Jinho D. Choi; Yejin Choi; Monojit Choudhury; Christos Christodoulopoulos; Grzegorz Chrupała; Chenhui Chu; Tagyoung Chung; Mark Cieliebak; Stephen Clark; Ann Clifton; Martin Cmejrek; Moreno Coco; Shay B. Cohen; Trevor Cohn; Paul Cook; Bonaventura Coppola; Greg Coppola; Marta R. Costa-Jussà; Benoit Crabbé; Danilo Croce; Fabien Cromieres; James Cross; Montse Cuadros; Heriberto Cuayahuitl; Lei Cui; Ronan Cummins;

Zhicheng Dou; Walter Daelemans; Daniel Dahlmeier; Van Dang; William Darling; Dipanjan Das; Pradipto Das; Rajarshi Das; Sajib Dasgupta; Pradeep Dasigi; Steve DeNeefe; John DeNero; Judith Degen; Felice Dell'Orletta; Vera Demberg; Dina Demner-Fushman; Pascal Denis; Michael Denkowski; Tejaswini Deoskar; Leon Derczynski; Aliya Deri; Nina Dethlefs; Laura Dietz; Georgiana Dinu; Qing Dou; Doug Downey; Eduard Dragut; Markus Dreyer; Gregory Druck; Lan Du; Xiangyu Duan; Kevin Duh; Nadir Durrani; Greg Durrett; Chris Dyer; Marc Dymetman; Bharath Dandala; Adrià de Gispert; Gerard de Melo; Ann Devitt;

Richard Eckart de Castilho; Patrick Ehlen; Carsten Eickhoff; Andreas Eisele; Jacob Eisenstein; Ismail El Maarouf; Michael Elhadad; Micha Elsner; Tomaž Erjavec; Arash Eshghi; Cristina España-Bonet; Luis Espinosa Anke; Miquel Esplà-Gomis;

Licheng Fang; Benamara Farah; Noura Farra; Manaal Faruqui; Benoit Favre; Anna Feldman; Minwei Feng; Song Feng; Yang Feng; Yansong Feng; Raquel Fernandez; Daniel Fernández-González; Francis Ferraro; Michele Filannino; Simone Filice; Denis Filimonov; Nicholas FitzGerald; Darja Fišer; Jeffrey

Flanigan; Tiziano Flati; Radu Florian; John Foley; José A. R. Fonollosa; Mikel Forcada; Lluís Formiga; Eric Fosler-Lussier; George Foster; Stefan L. Frank; Stella Frank; Markus Freitag; Lea Frermann; Annemarie Friedrich; Guohong Fu; Hagen Fuerstenau; Alona Fyshe;

Hongyu Guo; Michel Galley; Matthias Gallé; Michael Gamon; Kavita Ganesan; Wei Gao; Claire Gardent; Dan Garrette; Milica Gasic; Jon Gauthier; Tao Ge; Spandana Gella; Kallirroi Georgila; Alborz Geramifard; Debanjan Ghosh; Daniel Gildea; Kevin Gimpel; Filip Ginter; Roxana Girju; Koldo Gojenola; Yoav Goldberg; Dan Goldwasser; Zhengxian Gong; Graciela Gonzalez; Jesús González-Rubio; Michael Wayne Goodman; Amit Goyal; Natalia Grabar; Mark Granroth-Wilding; Spence Green; Stephan Greene; Edward Grefenstette; Cyril Grouin; Stig-Arne Grönroos; Fritz Guenther; Jiafeng Guo; Jiang Guo; Weiwei Guo; Sonal Gupta; Iryna Gurevych; Jose Manuel Gómez; Carlos Gómez-Rodríguez;

Ben Hachey; Barry Haddow; Gholamreza Haffari; Masato Hagiwara; Dilek Hakkani-Tur; David Hall; Keith Hall; Thierry Hamon; Bo Han; Shuguang Han; Xianpei Han; Greg Hanneman; Yu Hao; Sanda Harabagiu; Christian Hardmeier; Saša Hasan; Chikara Hashimoto; Kazuma Hashimoto; Eva Hasler; Jun Hatori; He He; Xiaodong He; Yifan He; Yulan He; Kenneth Heafield; Larry Heck; Carmen Heger; Michael Heilman; James Henderson; John Henderson; Aurélie Herbelot; Karl Moritz Hermann; Felix Hieber; Derrick Higgins; Gerold Hintz; Graeme Hirst; Hieu Hoang; Julia Hockenmaier; Yu Hong; Ales Horak; Julian Hough; Dirk Hovy; Chun-Nan Hsu; Yuening Hu; Fei Huang; Hongzhao Huang; Liang Huang; Minlie Huang; Ruihong Huang; Xuanjing Huang; Yi-Ting Huang; Zhongqiang Huang; Matthias Huck; Rebecca Hwa;

Gonzalo Iglesias; Ryu Iida; Shajith Ikbal; Iustina Ilisei; Diana Inkpen; Naoya Inoue; Kentaro Inui; Aminul Islam;

Guillaume Jacquet; Line Jakubiec-Jamet; Myungha Jang; Peter Jansen; Sujay Kumar Jauhar; Arun Kumar Jayapal; Laura Jehl; Heng Ji; Yangfeng Ji; Jing Jiang; Wenbin Jiang; Peng Jin; Anders Johannsen; Richard Johansson; Melvin Johnson Premkumar; Dan Jurafsky; David Jurgens;

Mijail Kabadjov; Nobuhiro Kaji; Nal Kalchbrenner; Justine Kao; Dimitri Kartsaklis; Saurabh Kataria; Daisuke Kawahara; Anna Kazantseva; Simon Keizer; Frank Keller; Casey Kennington; Maxim Khalilov; Ankur Khanna; Mitesh M. Khapra; Douwe Kiela; Halil Kilicoglu; Dongho Kim; Dongwoo Kim; Jin-Dong Kim; Seungyeon Kim; Suin Kim; Chunyu Kit; Alexandre Klementiev; Roman Klinger; Ari Kobren; Łukasz Kobyliński; Philipp Koehn; Rob Koeling; Mamoru Komachi; Grzegorz Kondrak; Fang Kong; Lingpeng Kong; Ioannis Konstas; Georgios Kontonatsios; Ioannis Korkontzelos; Zornitsa Kozareva; Mikhail Kozhevnikov; Martin Krallinger; Jayant Krishnamurthy; Kriste Krstovski; Canasai Kruengkrai; Germán Kruszewski; Lun-Wei Ku; Roland Kuhn; Anjishnu Kumar; Shankar Kumar; Jonathan K. Kummerfeld; Tsung-Ting Kuo; Nate Kushman; Tom Kwiatkowski;

Mathias Lambert; Patrik Lambert; Vasileios Lampos; Gerasimos Lampouras; Man Lan; Yanyan Lan; Jun Lang; Phillippe Langlais; Mirella Lapata; First Name Last Name; Jey Han Lau; Angeliki Lazaridou; Svetlana Lazebnik; Dieu-Thu Le; Joseph Le Roux; Kenton Lee; Lung-Hao Lee; Sungjin Lee; Tao Lei; Oliver Lemon; Alessandro Lenci; Chee Wee Leong; Gina-Anne Levow; Omer Levy; Mike Lewis; Binyang Li; Chen Li; Chenliang Li; Fangtao Li; Haibo Li; Hao Li; Jing Li; Jiwei Li; Junhui Li; Peifeng Li; Qi Li; Shoushan Li; Sujian Li; Wenjie Li; Yanen Li; Zhenghua Li; Maria Liakata; Percy Liang; Zhen Liao; Chenghua Lin; Chu-Cheng Lin; Xi Victoria Lin; Wang Ling; Xiao Ling; Tal Linzen; Bing Liu; Fei Liu; Jialu Liu; Jing Liu; Kang Liu; Qun Liu; Rui Liu; Ting Liu; Xiaodong Liu; Yang Liu; Yang Liu; Yiqun Liu; Zhanyi Liu; Zhiyuan Liu; Nikola Ljubešić; Chi-kiu Lo; Pavel Logacev; Varvara Logacheva; Oier Lopez de Lacalle; Bin Lu; Wei Lu; Xiaofei Lu; Marco Lui; Zhunchen Luo; Thang Luong;

Véronique Moriceau; Ji Ma; Yanjun Ma; Zongyang Ma; Wolfgang Macherey; Pranava Swaroop Madhyastha; Nitin Madnani; Pierre Magistry; Gideon Mann; Christopher D. Manning; Amin Mantrach; Xian-Ling Mao; Georgiana Marsic; Scott Martin; André F. T. Martins; Yuval Marton; Héctor Martínez Alonso; Prashant Mathur; Takuya Matsuzaki; Arne Mauser; Jonathan May; David McClosky; John Philip McCrae; Christopher McCubbin; Ryan McDonald; Louise McNally; Avihai Mejer; Yelena Mejova; Oren Melamud; Pablo Mendes; Florian Metze; Haitao Mi; Rada Mihalcea; Claudiu Mihăilă; David Mimno; Bonan Min; Zhao-Yan Ming; Shachar Mirkin; Seyed Abolghasem Mirroshandel; Paramita Mirza; Margaret Mitchell; Makoto Miwa; Yusuke Miyao; Daichi Mochihashi; Saif Mohammad; Mitra Mohtarami; Karo Moilanen; Christian Monson; Manuel Montes; Christof Monz; Taesun Moon; Raymond Mooney; Roser Morante; Alessandro Moschitti; Arjun Mukherjee; Philippe Muller; Dragos Munteanu; Yugo Murawaki; Brian Murphy; Lluís Màrquez;

Jian-Yun Nie; Maria Nadejde; Vinita Nahar; Preslav Nakov; Karthik Narasimhan; Shashi Narayan; Tahira Naseem; Vivi Nastase; Borja Navarro; Roberto Navigli; Raheel Nawaz; Mark-Jan Nederhof; Arvind Neelakantan; Sapna Negi; Aida Nematzadeh; Graham Neubig; Hwee Tou Ng; Vincent Ng; Nhung Nguyen; Viet-An Nguyen; Garrett Nicolai; Jan Niehues; Rodney Nielsen; Zheng-Yu Niu; Joakim Nivre; Joel Nothman; Scott Nowson; Pierre Nugues; Malte Nuhn;

Brendan O'Connor; Timothy O'Donnell; Stephan Oepen; Kemal Oflazer; Alice Oh; Naoaki Okazaki; Desmond Ong; Constantin Orasan; Vicente Ordonez; Daniel Ortiz-Martínez; Miles Osborne; Mari Ostendorf;

Deepak P; Ulrike Pado; Muntsa Padró; Gustavo Paetzold; Georgios Paliouras; Martha Palmer; Alessio Palmero Aprosio; Sinno J. Pan; Denis Paperno; Aasish Pappu; Rebecca J. Passonneau; Panupong Pasupat; Siddharth Patwardhan; Michael J. Paul; Adam Pauls; Ellie Pavlick; Lisa Pearl; Stephan Peitz; Filipa Peleja; Xiaochang Peng; Sergio Penkale; Gerald Penn; Bianca Pereira; Bryan Perozzi; Johann Petrak; Slav Petrov; Anselmo Peñas; Nghia The Pham; Lawrence Phillips; Karl Pichotta; Daniele Pighin; Mohammad Taher Pilehvar; Manfred Pinkal; Juan Pino; Yuval Pinter; Emily Pitler; Barbara Plank; Massimo Poesio; Tamara Polajnar; Heather Pon-Barry; Simone Paolo Ponzetto; Andrei Popescu-Belis; Maja Popović; Vinodkumar Prabhakaran; John Prager; Rashmi Prasad; Daniel Preoţiuc-Pietro; Prokopis Prokopidis; Matthew Purver; Verónica Pérez-Rosas;

Longhua Qian; Xian Qian; Lu Qin; Likun Qiu; Long Qiu; Xipeng Qiu; Lizhen Qu; Chris Quirk;

Ella Rabinovich; Will Radford; Preethi Raghavan; Altaf Rahman; Rafal Rak; Carlos Ramisch; Mohammad Sadegh Rasooli; Siva Reddy; Sravana Reddy; Marek Rei; Roi Reichart; Nils Reiter; Xiang Ren; Philip Resnik; Corentin Ribeyre; Kyle Richardson; Sebastian Riedel; Martin Riedl; Stefan Riezler; German Rigau; Laura Rimell; Fabio Rinaldi; Miguel Rios; Alan Ritter; Brian Roark; Angus Roberts; Kirk Roberts; Tim Rocktäschel; Horacio Rodriguez; Marcus Rohrbach; Stephen Roller; Francesco Ronzano; Andrew Rosenberg; Paolo Rosso; Michael Roth; Johann Roturier; Mickael Rouvier; Alla Rozovskaya; Eugen Ruppert;

Jun Sun; Markus Saers; Saurav Sahay; Hassan Saif; Hassan Sajjad; Keisuke Sakaguchi; Mohammad Salameh; Diarmuid Ó Séaghdha; Bahar Salehi; Tanja Samardzic; Mark Sammons; Germán Sanchis Trilles; Federico Sangati; Baskaran Sankaran; Motoki Sano; Felix Sasaki; Ryohei Sasano; Asad Sayeed; Carolina Scarton; David Schlangen; Natalie Schluter; Helmut Schmid; Nathan Schneider; Michael Schuhmacher; William Schuler; H. Andrew Schwartz; Lane Schwartz; Roy Schwartz; Holger Schwenk; Hinrich Schütze; Djamé Seddah; Satoshi Sekine; Jean Senellart; Rico Sennrich; Hendra Setiawan; Burr

Settles; Aliaksei Severyn; Serge Sharoff; Shuming Shi; Chaitanya Shivade; Maryam Siahbani; Carina Silberer; Yanchuan Sim; Khalil Sima'an; Patrick Simianer; Kiril Simov; Sameer Singh; Kevin Small; Peter Smit; Noah A. Smith; Jan Šnajder; Artem Sokolov; Swapna Somasundaran; Linfeng Song; Lucia Specia; Vivek Srikumar; Miloš Stanojević; Mark Steedman; Pontus Stenetorp; Brandon Stewart; Veselin Stoyanov; Carlo Strapparava; Karl Stratos; Emma Strubell; Jannik Strötgen; Jian Su; Qi Su; Kazunari Sugiyama; Le Sun; Weiwei Sun; Xu Sun; Mihai Surdeanu; Hisami Suzuki; Swabha Swayamdipta; György Szarvas; Stan Szpakowicz; Idan Szpektor; Felipe Sánchez-Martínez; Anders Søgaard; Patrick Saint-Dizier; Jinsong Su;

Hiroya Takamura; Partha P. Talukdar; Duyu Tang; Jian Tang; Jiliang Tang; Xavier Tannier; Kapil Thadani; Sam Thomson; Jörg Tiedemann; Christoph Tillmann; Ivan Titov; Nadi Tomeh; Sara Tonelli; Fatemeh Torabi Asr; Antonio Toral; Kristina Toutanova; Jun'ichi Tsujii; Oren Tsur; Yoshimasa Tsuruoka; Yulia Tsvetkov; Gokhan Tur; Francis Tyers; Oscar Täckström;

Raghavendra Udupa; Lyle Ungar; Olga Uryupina; Jakob Uszkoreit;

Sowmya Vajjala; Benjamin Van Durme; Daniele Vannella; Ashish Vaswani; Marc Verhagen; Yannick Versley; Laure Vieu; David Vilar; Luke Vilnis; Sami Virpioja; Karthik Visweswariah; Andreas Vlachos; Rob Voigt; Svitlana Volkova; Ivan Vulić; Vinod Vydiswaran; Marieke van Erp; Marten van Schijndel; Antal van den Bosch;

Houfeng Wang; Joachim Wagner; Aurelien Waite; Xiaojun Wan; Chi Wang; Chuan Wang; Hongling Wang; Hongning Wang; Lu Wang; Rui Wang; Sida I. Wang; Wen Wang; William Yang Wang; Zhiguo Wang; Zhongqing Wang; Zhuoran Wang; Taro Watanabe; Aleksander Wawer; Kellie Webster; Furu Wei; Zhongyu Wei; Daniel Weld; Janyce Wiebe; Michael Wiegand; John Wieting; Jason D Williams; Steven Wilson; Shuly Wintner; Guillaume Wisniewski; Silke Witt-Ehsani; Travis Wolfe; Kam-Fai Wong; Alina Wróblewska; Stephen Wu; Xianchao Wu; Yuanbin Wu; Joern Wuebker;

Yunqing Xia; Tong Xiao; Boyi Xie; Pengtao Xie; Deyi Xiong; Jia Xu; Liheng Xu; Ruifeng Xu; Wei Xu; Wenduan Xu; Ying Xu;

Rui Yan; Bin Yang; Weiwei Yang; Yaqin Yang; Yi Yang; Yi Yang; Helen Yannakoudakis; Jin-ge Yao; Xuchen Yao; Ainur Yessenalina; Xing Yi; Wen-tau Yih; Seid Muhie Yimam; Dani Yogatama; Bei Yu; Dianhai Yu; Liang-Chih Yu; Mo Yu; Zhou Yu; François Yvon; Tae Yano;

Deyu Zhou; Manzil Zaheer; Omar Zaidan; Fabio Massimo Zanzotto; Alessandra Zarcone; Sina Zarrieß; Rabih Zbib; Richard Zens; Torsten Zesch; Luke Zettlemoyer; Feifei Zhai; Ke Zhai; Jiajun Zhang; Kevin Zhang; Meishan Zhang; Min Zhang; Min Zhang; Qi Zhang; Wei Zhang; Wei-Nan Zhang; Yu Zhang; Yuan Zhang; Yuchen Zhang; Yue Zhang; Zhe Zhang; Zhichang Zhang; Hai Zhao; Jun Zhao; Kai Zhao; Qiuye Zhao; Shiqi Zhao; Wayne Xin Zhao; Xin Zhao; Yanyan Zhao; Zhi Zhong; Guangyou Zhou; Guodong Zhou; Junsheng Zhou; Mianwei Zhou; Xinjie Zhou; Jingbo Zhu; Muhua Zhu; Xiaodan Zhu; Imed Zitouni; Bowei Zou; Arkaitz Zubiaga; Pierre Zweigenbaum; Xiaoyan Zhu;

Additional Reviewers

Abeed Sarker; Abouenour Lahsen; Afra Alishahi; Afshin Rahimi; Aitor García Pablos; Aitziber Atutxa Salazar; Alan Lee; Alicia Pérez Ramírez; Anders Johannsen; Anders Søgaard; Andreas Grivas; Anna Jørgensen; Annemarie Friedrich; Antonio Uva; Austin Matthews; Avihai Mejer; Baolin Peng; Benjamin Marie; Benjamin Yap; Bo Han; Bonnie Webber; Bradley Hauer; Camilo Thorne; Chong Zhou; Daniel Torregrosa; David Buttler; Davy Weissenbacher; Desmond Ong; Ehsan Emazadeh; Elias Zavitsanos; Enrique Vidal; Fandong Meng; Federico Nanni; Filipa Peleja; Francis Ferraro; Francisco Rangel; Gözde Özbal; Henry Anaya-Sanchez; Hieu Pham; Ignacio Iacobacci; I-Ta Lee; Jack Hessel; James Cross; Jing Li; Jing Ma; Jintao Ye; Joost Bastings; Kai Zhao; Kateryna Tymoshenko; Kazuya Kawakami; Ke Tao; Ke Zhai; Keith Stevens; Lane Schwartz; Lasse Borgholt; Leila Wehbe; Long Duong; Lucie Flekova; Marc Franco-Salvador; Marco Del Tredici; Maximin Coavoux; Meng Zhang; Michal Lukasik; Mingxuan Wang; Mohamed Mouine; Mohammad Salameh; Muhua Zhu; Nadir Durrani; Nan Yang; Nichola Lubold; Nima Pourdamghani; Nina Zhou; Phil Williams; Prodromos Malakasiotis; Rachel Rudinger; Rui Xia; Serra Sinem Tekiroglu; Shachi H Kumar; Shruti Rijhwani; Songxian Xie; Stas Semeniuta; Stefano Faralli; Stephan Gouws; Stephen Maddock; Sungjoon Park; Susana Palmaz; Tobias Domhan; Tomer Levinboim; Vaggelis Michelioudakis; Vassilis Papavassiliou; Vivek Kulkarni; Wang Ling; Wen Wang; Weston Feely; Wu Kui; Xanda Schofield; Xiang Li; Xiaodong He; Yan Fu; Yanran Li; Yi Luan; Yijia Liu; Yulia Tsvetkov; Zhihua Zhang.

Invited Speaker: Yoshua Bengio

Deep Learning of Semantic Representations

Abstract: The core ingredient of deep learning is the notion of distributed representation. This talk will start by explaining its theoretical advantages, in comparison with non-parametric methods based on counting frequencies of occurrence of observed tuples of values (like with n-grams). The talk will then explain how having multiple levels of representation, i.e., depth, can in principle give another exponential advantage. Neural language models have been extremely successful in recent years but extending their reach from language modeling to machine translation is very appealing because it forces the learned intermediate representations to capture meaning, and we found that the resulting word embeddings are qualitatively different. Recently, we introduced the notion of attention-based encoder-decoder systems, with impressive results on machine translation several language pairs and for mapping an image to a sentence, and these results will conclude the talk.

Biography: Yoshua Bengio received a PhD in Computer Science from McGill University, Canada in 1991. After two post-doctoral years, one at M.I.T. with Michael Jordan and one at AT&T Bell Laboratories with Yann LeCun and Vladimir Vapnik, he became professor at the Department of Computer Science and Operations Research at Université de Montréal. He is the author of two books and more than 200 publications, the most cited being in the areas of deep learning, recurrent neural networks, probabilistic learning algorithms, natural language processing and manifold learning. He is among the most cited Canadian computer scientists and is or has been associate editor of the top journals in machine learning and neural networks. Since '2000 he holds a Canada Research Chair in Statistical Learning Algorithms, since '2006 an NSERC Industrial Chair, since '2005 his is a Senior Fellow of the Canadian Institute for Advanced Research and since 2014 he co-directs its program focused on deep learning. He is on the board of the NIPS foundation and has been program chair and general chair for NIPS. He has co-organized the Learning Workshop for 14 years and co-created the new International Conference on Learning Representations. His current interests are centered around a quest for AI through machine learning, and include fundamental questions on deep learning and representation learning, the geometry of generalization in high-dimensional spaces, manifold learning, biologically inspired learning algorithms, and challenging applications of statistical machine learning.

Invited Speaker: Justin Grimmer

Measuring How Elected Officials and Constituents Communicate

Abstract: This talk will show how elected officials use communication to cultivate support with constituents, how constituents express their views to elected officials, and why biases in both kinds of communication matter for political representation. To demonstrate the bias and its effects, I propose to use novel collections of political texts and new text as data methods. Using the new data and methods, I will show how the incentives of communication contribute to perceptions of an angry public and vitriolic politicians. Among elected officials, the ideologically extreme members of Congress disproportionately participate in policy debates, resulting in political debates that occur between the most extreme members of each party. Among constituents, the most ideologically extreme and angry voters disproportionately contact their member of Congress, creating the impression of a polarized and vitriolic public. The talk will explain how the findings help us to understand how representation occurs in American politics, while also explaining how computational tools can help address questions in the social sciences.

Biography: Justin Grimmer is an associate professor of political science at Stanford University. His research examines how representation occurs in American politics using new statistical methods. His first book Representational Style in Congress: What Legislators Say and Why It Matters (Cambridge University Press, 2013) shows how senators define the type of representation they provide constituents and how this affects constituents' evaluations and was awarded the 2014 Richard Fenno Prize. His second book The Impression of Influence: How Legislator Communication and Government Spending Cultivate a Personal Vote (Princeton University Press, 2014 with Sean J. Westwood and Solomon Messing) demonstrates how legislators ensure they receive credit for government actions. His work has appeared in the American Political Science Review, American Journal of Political Science, Journal of Politics, Political Analysis, Proceedings of the National Academy of Sciences, Regulation and Governance, and Poetics.

Conference Program

Friday, September 18, 2015

18:30–20:00 Welcome Reception

Saturday, September 19, 2015

07:30–18:00 Registration

08:40–10:00 Session P1: Plenary Session

08:40–09:00 *Opening Remarks and Introductory Speeches*
General Chair, Program Co-Chairs and Local Co-Chairs

09:00–10:00 *Invited Talk: Deep Learning of Semantic Representations*
Yoshua Bengio

10:00–10:30 *Coffee break*

10:30–12:10 Session 1A: Semantics 1 (Long + TACL Papers)

10:30–10:55 *Language Understanding for Text-based Games using Deep Reinforcement Learning*
Karthik Narasimhan, Tejas Kulkarni and Regina Barzilay

10:55–11:20 *Distributional vectors encode referential attributes*
Abhijeet Gupta, Gemma Boleda, Marco Baroni and Sebastian Padó

11:20–11:45 *Building a shared world: mapping distributional to model-theoretic semantic spaces*
Aurélie Herbelot and Eva Maria Vecchi

11:45–12:10 *[TACL] Deriving Boolean Structures from Distributional Vectors*
Germán Kruszewski, Denis Paperno and Marco Baroni

Saturday, September 19, 2015 (continued)

10:30–12:10 Session 1B: Machine Translation 1 (Long + TACL Papers)

10:30–10:55 *Dependency Graph-to-String Translation*
Liangyou Li, Andy Way and Qun Liu

10:55–11:20 *Reordering Grammar Induction*
Miloš Stanojević and Khalil Sima'an

11:20–11:45 *Syntax-based Rewriting for Simultaneous Machine Translation*
He He, Alvin Grissom II, John Morgan, Jordan Boyd-Graber and Hal Daumé III

11:45–12:10 *[TACL] Modelling and Optimizing on Syntactic N-Grams for Statistical Machine Translation*
Rico Sennrich

10:30–12:10 Session 1C: NLP for the Web and Social Media, including Computational Social Science (Long Papers)

10:30–10:55 *Identifying Political Sentiment between Nation States with Social Media*
Nathanael Chambers, Victor Bowen, Ethan Genco, Xisen Tian, Eric Young, Ganesh Harihara and Eugene Yang

10:55–11:20 *Open Extraction of Fine-Grained Political Statements*
David Bamman and Noah A. Smith

11:20–11:45 *Using Personal Traits For Brand Preference Prediction*
Chao Yang, Shimei Pan, Jalal Mahmud, Huahai Yang and Padmini Srinivasan

11:45–12:10 *Semantic Annotation for Microblog Topics Using Wikipedia Temporal Information*
Tuan Tran, Nam Khanh Tran, Asmelash Teka Hadgu and Robert Jäschke

10:30–12:10 Session 1D (P1-4): Summarization (Long Paper Posters)

System Combination for Multi-document Summarization
Kai Hong, Mitchell Marcus and Ani Nenkova

Phrase-based Compressive Cross-Language Summarization
Jin-ge Yao, Xiaojun Wan and Jianguo Xiao

Re-evaluating Automatic Summarization with BLEU and 192 Shades of ROUGE
Yvette Graham

Indicative Tweet Generation: An Extractive Summarization Problem?
Priya Sidhaye and Jackie Chi Kit Cheung

Saturday, September 19, 2015 (continued)

10:30–12:10 Session 1D (P5): Language and Vision (Long Paper Posters)

Visual Bilingual Lexicon Induction with Transferred ConvNet Features
Douwe Kiela, Ivan Vulić and Stephen Clark

10:30–12:10 Session 1D (P6-9): Sentiment Analysis and Opinion Mining (Long Paper Posters)

Cross Lingual Sentiment Analysis using Modified BRAE
Sarthak Jain and Shashank Batra

Monotone Submodularity in Opinion Summaries
Jayanth Jayanth, Jayaprakash Sundararaj and Pushpak Bhattacharyya

Joint Prediction for Entity/Event-Level Sentiment Analysis using Probabilistic Soft Logic Models
Lingjia Deng and Janyce Wiebe

Learning to Recognize Affective Polarity in Similes
Ashequl Qadir, Ellen Riloff and Marilyn Walker

10:30–12:10 Session 1E (P1-3): Language and Vision (Short Paper Posters)

Cross-document Event Coreference Resolution based on Cross-media Features
Tongtao Zhang, Hongzhi Li, Heng Ji and Shih-Fu Chang

A Survey of Current Datasets for Vision and Language Research
Francis Ferraro, Nasrin Mostafazadeh, Ting-Hao Huang, Lucy Vanderwende, Jacob Devlin, Michel Galley and Margaret Mitchell

Combining Geometric, Textual and Visual Features for Predicting Prepositions in Image Descriptions
Arnau Ramisa, Josiah Wang, Ying Lu, Emmanuel Dellandrea, Francesc Moreno-Noguer and Robert Gaizauskas

10:30–12:10 Session 1E (P4-16): Statistical Models and Machine Learning Methods for NLP (Short Paper Posters)

On A Strictly Convex IBM Model 1
Andrei Simion, Michael Collins and Cliff Stein

Factorization of Latent Variables in Distributional Semantic Models
Arvid Österlund, David Ödling and Magnus Sahlgren

Non-lexical neural architecture for fine-grained POS Tagging
Matthieu Labeau, Kevin Löser and Alexandre Allauzen

Online Representation Learning in Recurrent Neural Language Models
Marek Rei

A Model of Zero-Shot Learning of Spoken Language Understanding
Majid Yazdani and James Henderson

Modeling Tweet Arrival Times using Log-Gaussian Cox Processes
Michal Lukasik, P. K. Srijith, Trevor Cohn and Kalina Bontcheva

Pre-Computable Multi-Layer Neural Network Language Models
Jacob Devlin, Chris Quirk and Arul Menezes

Birds of a Feather Linked Together: A Discriminative Topic Model using Link-based Priors
Weiwei Yang, Jordan Boyd-Graber and Philip Resnik

Aligning Knowledge and Text Embeddings by Entity Descriptions
Huaping Zhong, Jianwen Zhang, Zhen Wang, Hai Wan and Zheng Chen

An Empirical Analysis of Optimization for Max-Margin NLP
Jonathan K. Kummerfeld, Taylor Berg-Kirkpatrick and Dan Klein

Learning Better Embeddings for Rare Words Using Distributional Representations
Irina Sergienya and Hinrich Schütze

Composing Relationships with Translations
Alberto Garcia-Duran, Antoine Bordes and Nicolas Usunier

Noise or additional information? Leveraging crowdsource annotation item agreement for natural language tasks.
Emily Jamison and Iryna Gurevych

12:10–13:30 ***Lunch***

Saturday, September 19, 2015 (continued)

13:30–15:10 Session 2A: Statistical Modeling and Machine Learning 1 (Long + TACL Papers)

13:30–13:55 *Evaluation methods for unsupervised word embeddings*
Tobias Schnabel, Igor Labutov, David Mimno and Thorsten Joachims

13:55–14:20 *Efficient Methods for Incorporating Knowledge into Topic Models*
Yi Yang, Doug Downey and Jordan Boyd-Graber

14:20–14:45 *Traversing Knowledge Graphs in Vector Space*
Kelvin Guu, John Miller and Percy Liang

14:45–15:10 *[TACL] Improving Topic Models with Latent Feature Word Representations*
Dat Quoc Nguyen, Richard Billingsley, Lan Du and Mark Johnson

13:30–15:10 Session 2B: Tagging, Chunking and Parsing 1 (Long +TACL Papers)

13:30–13:55 *Density-Driven Cross-Lingual Transfer of Dependency Parsers*
Mohammad Sadegh Rasooli and Michael Collins

13:55–14:20 *A Neural Network Model for Low-Resource Universal Dependency Parsing*
Long Duong, Trevor Cohn, Steven Bird and Paul Cook

14:20–14:45 *Improved Transition-based Parsing by Modeling Characters instead of Words with LSTMs*
Miguel Ballesteros, Chris Dyer and Noah A. Smith

14:45–15:10 *[TACL] Approximation-Aware Dependency Parsing by Belief Propagation*
Matthew R. Gormley, Mark Dredze and Jason Eisner

13:30–15:10 Session 2C: Summarization (Long Papers)

13:30–13:55 *Sentence Compression by Deletion with LSTMs*
Katja Filippova, Enrique Alfonseca, Carlos A. Colmenares, Lukasz Kaiser and Oriol Vinyals

13:55–14:20 *An Empirical Comparison Between N-gram and Syntactic Language Models for Word Ordering*
Jiangming Liu and Yue Zhang

14:20–14:45 *A Neural Attention Model for Abstractive Sentence Summarization*
Alexander M. Rush, Sumit Chopra and Jason Weston

14:45–15:10 *Scientific Article Summarization Using Citation-Context and Article's Discourse Structure*
Arman Cohan and Nazli Goharian

13:30–15:10 Session 2D (P1-9): Text Mining and NLP Applications (Long Paper Posters)

Hashtag Recommendation Using Dirichlet Process Mixture Models Incorporating Types of Hashtags
Yeyun Gong, Qi Zhang and Xuanjing Huang

A Graph-based Readability Assessment Method using Word Coupling
Zhiwei Jiang, Gang Sun, Qing Gu, Tao Bai and Daoxu Chen

More Features Are Not Always Better: Evaluating Generalizing Models in Incident Type Classification of Tweets
Axel Schulz, Christian Guckelsberger and Benedikt Schmidt

Flexible Domain Adaptation for Automated Essay Scoring Using Correlated Linear Regression
Peter Phandi, Kian Ming A. Chai and Hwee Tou Ng

Show Me Your Evidence - an Automatic Method for Context Dependent Evidence Detection
Ruty Rinott, Lena Dankin, Carlos Alzate Perez, Mitesh M. Khapra, Ehud Aharoni and Noam Slonim

Spelling Correction of User Search Queries through Statistical Machine Translation
Saša Hasan, Carmen Heger and Saab Mansour

Human Evaluation of Grammatical Error Correction Systems
Roman Grundkiewicz, Marcin Junczys-Dowmunt and Edward Gillian

Learning a Deep Hybrid Model for Semi-Supervised Text Classification
Alexander Ororbia II, C. Lee Giles and David Reitter

Joint Embedding of Query and Ad by Leveraging Implicit Feedback
Sungjin Lee and Yifan Hu

Saturday, September 19, 2015 (continued)

13:30–15:10 Session 2E (P1-11): Information Extraction (Short Paper Posters)

Automatic Extraction of Time Expressions Accross Domains in French Narratives
Mike Donald Tapi Nzali, Xavier Tannier and Aurelie Neveol

Semi-Supervised Bootstrapping of Relationship Extractors with Distributional Semantics
David S. Batista, Bruno Martins and Mário J. Silva

Extraction and generalisation of variables from scientific publications
Erwin Marsi and Pinar Öztürk

Named entity recognition with document-specific KB tag gazetteers
Will Radford, Xavier Carreras and James Henderson

"A Spousal Relation Begins with a Deletion of engage and Ends with an Addition of divorce": Learning State Changing Verbs from Wikipedia Revision History
Derry Tanti Wijaya, Ndapandula Nakashole and Tom Mitchell

Improving Distant Supervision for Information Extraction Using Label Propagation Through Lists
Lidong Bing, Sneha Chaudhari, Richard Wang and William Cohen

An Entity-centric Approach for Overcoming Knowledge Graph Sparsity
Manjunath Hegde and Partha P. Talukdar

Semantic Relation Classification via Convolutional Neural Networks with Simple Negative Sampling
Kun Xu, Yansong Feng, Songfang Huang and Dongyan Zhao

A Baseline Temporal Tagger for all Languages
Jannik Strötgen and Michael Gertz

Named Entity Recognition for Chinese Social Media with Jointly Trained Embeddings
Nanyun Peng and Mark Dredze

Inferring Binary Relation Schemas for Open Information Extraction
Kangqi Luo, Xusheng Luo and Kenny Zhu

13:30–15:10 **Session 2E (P12-16): Information Retrieval and Question Answering (Short Paper Posters)**

LDTM: A Latent Document Type Model for Cumulative Citation Recommendation
Jingang Wang, Dandan Song, Zhiwei Zhang, Lejian Liao, Luo Si and Chin-Yew Lin

Online Sentence Novelty Scoring for Topical Document Streams
Sungjin Lee

Global Thread-level Inference for Comment Classification in Community Question Answering
Shafiq Joty, Alberto Barrón-Cedeño, Giovanni Da San Martino, Simone Filice, Lluís Màrquez, Alessandro Moschitti and Preslav Nakov

Key Concept Identification for Medical Information Retrieval
Jiaping Zheng and Hong Yu

Image-Mediated Learning for Zero-Shot Cross-Lingual Document Retrieval
Ruka Funaki and Hideki Nakayama

15:10–15:40 ***Coffee break***

15:40–17:20 **Session 3A: Sentiment Analysis and Opinion Mining 1 (Long Papers)**

15:40–16:05 *Detecting Risks in the Banking System by Sentiment Analysis*
Clemens Nopp and Allan Hanbury

16:05–16:30 *Sentiment Flow - A General Model of Web Review Argumentation*
Henning Wachsmuth, Johannes Kiesel and Benno Stein

16:30–16:55 *Neural Networks for Open Domain Targeted Sentiment*
Meishan Zhang, Yue Zhang and Duy Tin Vo

16:55–17:20 *Extracting Condition-Opinion Relations Toward Fine-grained Opinion Mining*
Yuki Nakayama and Atsushi Fujii

Saturday, September 19, 2015 (continued)

15:40–17:20 Session 3B: Semantics 2 (Long +TACL Papers)

15:40–16:05 *A large annotated corpus for learning natural language inference*
Samuel R. Bowman, Gabor Angeli, Christopher Potts and Christopher D. Manning

16:05–16:30 *Question-Answer Driven Semantic Role Labeling: Using Natural Language to Annotate Natural Language*
Luheng He, Mike Lewis and Luke Zettlemoyer

16:30–16:55 *[TACL] It's All Fun and Games until Someone Annotates: Video Games with a Purpose for Linguistic Annotation.*
David Jurgens and Roberto Navigli

16:55–17:20 *[TACL] Semantic Proto-Roles*
Drew Reisinger, Rachel Rudinger, Francis Ferraro, Kyle Rawlins and Benjamin Van Durme

15:40–17:20 Session 3C: Information Retrieval and Question Answering (Long Papers)

15:40–16:05 *Name List Only? Target Entity Disambiguation in Short Texts*
Yixin Cao, Juanzi Li, Xiaofei Guo, Shuanhu Bai, Heng Ji and Jie Tang

16:05–16:30 *Biography-Dependent Collaborative Entity Archiving for Slot Filling*
Yu Hong, Xiaobin Wang, Yadong Chen, Jian Wang, Tongtao Zhang and Heng Ji

16:30–16:55 *Stochastic Top-k ListNet*
Tianyi Luo, Dong Wang, Rong Liu and Yiqiao Pan

16:55–17:20 *Exploring Markov Logic Networks for Question Answering*
Tushar Khot, Niranjan Balasubramanian, Eric Gribkoff, Ashish Sabharwal, Peter Clark and Oren Etzioni

15:40–17:20 Session 3D (P1-9): Information Extraction (Long + TACL Paper Posters)

Language and Domain Independent Entity Linking with Quantified Collective Validation
Han Wang, Jin Guang Zheng, Xiaogang Ma, Peter Fox and Heng Ji

Modeling Relation Paths for Representation Learning of Knowledge Bases
Yankai Lin, Zhiyuan Liu, Huanbo Luan, Maosong Sun, Siwei Rao and Song Liu

Corpus-level Fine-grained Entity Typing Using Contextual Information
Yadollah Yaghoobzadeh and Hinrich Schütze

Knowledge Base Unification via Sense Embeddings and Disambiguation
Claudio Delli Bovi, Luis Espinosa Anke and Roberto Navigli

Open-Domain Name Error Detection using a Multi-Task RNN
Hao Cheng, Hao Fang and Mari Ostendorf

Extracting Relations between Non-Standard Entities using Distant Supervision and Imitation Learning
Isabelle Augenstein, Andreas Vlachos and Diana Maynard

Sieve-Based Spatial Relation Extraction with Expanding Parse Trees
Jennifer D'Souza and Vincent Ng

[TACL] Cross-Document Co-Reference Resolution using Sample-Based Clustering with Knowledge Enrichment
Sourav Dutta and Gerhard Weikum

[TACL] Combining Minimally-supervised Methods for Arabic Named Entity Recognition
Maha Althobaiti, Udo Kruschwitz and Massimo Poesio

Saturday, September 19, 2015 (continued)

15:40–17:20 Session 3E (P1-13): Text Mining and NLP Applications (Short Paper Posters)

Mr. Bennet, his coachman, and the Archbishop walk into a bar but only one of them gets recognized: On The Difficulty of Detecting Characters in Literary Texts
Hardik Vala, David Jurgens, Andrew Piper and Derek Ruths

Convolutional Sentence Kernel from Word Embeddings for Short Text Categorization
Jonghoon Kim, Francois Rousseau and Michalis Vazirgiannis

Predicting the Structure of Cooking Recipes
Jermsak Jermsurawong and Nizar Habash

TSDPMM: Incorporating Prior Topic Knowledge into Dirichlet Process Mixture Models for Text Clustering
Linmei Hu, Juanzi Li, Xiaoli Li, Chao Shao and Xuzhong Wang

Sentence Modeling with Gated Recursive Neural Network
Xinchi Chen, Xipeng Qiu, Chenxi Zhu, Shiyu Wu and Xuanjing Huang

Learning Timeline Difference for Text Categorization
Fumiyo Fukumoto and Yoshimi Suzuki

Summarizing Topical Contents from PubMed Documents Using a Thematic Analysis
Sun Kim, Lana Yeganova and W John Wilbur

Recognizing Biographical Sections in Wikipedia
Alessio Palmero Aprosio and Sara Tonelli

Learn to Solve Algebra Word Problems Using Quadratic Programming
Lipu Zhou, Shuaixiang Dai and Liwei Chen

An Unsupervised Method for Discovering Lexical Variations in Roman Urdu Informal Text
Abdul Rafae, Abdul Qayyum, Muhammad Moeenuddin, Asim Karim, Hassan Sajjad and Faisal Kamiran

Component-Enhanced Chinese Character Embeddings
Yanran Li, Wenjie Li, Fei Sun and Sujian Li

Multi-label Text Categorization with Joint Learning Predictions-as-Features Method
Li Li, Houfeng Wang, Xu Sun, Baobao Chang, Shi Zhao and Lei Sha

A Framework for Comparing Groups of Documents
Arun Maiya

Saturday, September 19, 2015 (continued)

Sunday, September 20, 2015

07:30–18:00 Registration

09:00–10:00 Session P2: Plenary Session

Invited Talk: Measuring How Elected Officials and Constituents Communicate
Justin Grimmer

10:00–10:30 ***Coffee break***

10:30–12:10 Session 4A: Information Extraction 1 (Long Papers)

10:30–10:55 *C3EL: A Joint Model for Cross-Document Co-Reference Resolution and Entity Linking*
Sourav Dutta and Gerhard Weikum

10:55–11:20 *Joint Mention Extraction and Classification with Mention Hypergraphs*
Wei Lu and Dan Roth

11:20–11:45 *FINET: Context-Aware Fine-Grained Named Entity Typing*
Luciano Del Corro, Abdalghani Abujabal, Rainer Gemulla and Gerhard Weikum

11:45–12:10 *Joint Entity Recognition and Disambiguation*
Gang Luo, Xiaojiang Huang, Chin-Yew Lin and Zaiqing Nie

10:30–12:10 Session 4B: Statistical Modeling and Machine Learning 2 (Long Papers)

10:30–10:55 *How Much Information Does a Human Translator Add to the Original?*
Barret Zoph, Marjan Ghazvininejad and Kevin Knight

10:55–11:20 *Hierarchical Recurrent Neural Network for Document Modeling*
Rui Lin, Shujie Liu, Muyun Yang, Mu Li, Ming Zhou and Sheng Li

11:20–11:45 *Auto-Sizing Neural Networks: With Applications to n-gram Language Models*
Kenton Murray and David Chiang

11:45–12:10 *Dual Decomposition Inference for Graphical Models over Strings*
Nanyun Peng, Ryan Cotterell and Jason Eisner

Sunday, September 20, 2015 (continued)

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

10:30–10:55 *Discourse parsing for multi-party chat dialogues*
Stergos Afantenos, Eric Kow, Nicholas Asher and Jérémy Perret

10:55–11:20 *Joint prediction in MST-style discourse parsing for argumentation mining*
Andreas Peldszus and Manfred Stede

11:20–11:45 *[TACL] One Vector is Not Enough: Entity-Augmented Distributed Semantics for Discourse Relations*
Yangfeng Ji and Jacob Eisenstein

11:45–12:10 *[TACL] Latent Structures for Coreference Resolution*
Sebastian Martschat and Michael Strube

10:30–12:10 Session 4D (P1-9): Semantics (Long Paper Posters)

Feature-Rich Two-Stage Logistic Regression for Monolingual Alignment
Md Arafat Sultan, Steven Bethard and Tamara Sumner

Semantic Role Labeling with Neural Network Factors
Nicholas FitzGerald, Oscar Täckström, Kuzman Ganchev and Dipanjan Das

RELLY: Inferring Hypernym Relationships Between Relational Phrases
Adam Grycner, Gerhard Weikum, Jay Pujara, James Foulds and Lise Getoor

Mise en Place: Unsupervised Interpretation of Instructional Recipes
Chloé Kiddon, Ganesa Thandavam Ponnuraj, Luke Zettlemoyer and Yejin Choi

Semantic Framework for Comparison Structures in Natural Language
Omid Bakhshandeh and James Allen

Sarcastic or Not: Word Embeddings to Predict the Literal or Sarcastic Meaning of Words
Debanjan Ghosh, Weiwei Guo and Smaranda Muresan

Incorporating Trustiness and Collective Synonym/Contrastive Evidence into Taxonomy Construction
Tuan Luu Anh, Jung-jae Kim and See Kiong Ng

Learning to Automatically Solve Logic Grid Puzzles
Arindam Mitra and Chitta Baral

Sunday, September 20, 2015 (continued)

10:30–12:10 Session 4E (P1-13): Machine Translation and Multilinguality (Short Paper Posters)

Improving fast_align by Reordering
Chenchen Ding, Masao Utiyama and Eiichiro Sumita

Touch-Based Pre-Post-Editing of Machine Translation Output
Benjamin Marie and Aurélien Max

A Discriminative Training Procedure for Continuous Translation Models
Quoc-Khanh DO, Alexandre Allauzen and François Yvon

System Combination for Machine Translation through Paraphrasing
Wei-Yun Ma and Kathleen McKeown

Hierarchical Incremental Adaptation for Statistical Machine Translation
Joern Wuebker, Spence Green and John DeNero

ReVal: A Simple and Effective Machine Translation Evaluation Metric Based on Recurrent Neural Networks
Rohit Gupta, Constantin Orasan and Josef van Genabith

Investigating Continuous Space Language Models for Machine Translation Quality Estimation
Kashif Shah, Raymond W. M. Ng, Fethi Bougares and Lucia Specia

Supervised Phrase Table Triangulation with Neural Word Embeddings for Low-Resource Languages
Tomer Levinboim and David Chiang

Translation Invariant Word Embeddings
Kejun Huang, Matt Gardner, Evangelos Papalexakis, Christos Faloutsos, Nikos Sidiropoulos, Tom Mitchell, Partha P. Talukdar and Xiao Fu

Hierarchical Phrase-based Stream Decoding
Andrew Finch, Xiaolin Wang, Masao Utiyama and Eiichiro Sumita

Rule Selection with Soft Syntactic Features for String-to-Tree Statistical Machine Translation
Fabienne Braune, Nina Seemann and Alexander Fraser

Motivating Personality-aware Machine Translation
Shachar Mirkin, Scott Nowson, Caroline Brun and Julien Perez

Trans-gram, Fast Cross-lingual Word-embeddings
Jocelyn Coulmance, Jean-Marc Marty, Guillaume Wenzek and Amine Benhalloum

Sunday, September 20, 2015 (continued)

10:30–12:10 Session 4E (P14-16): Computational Psycholinguistics (Short Paper Posters)

The Overall Markedness of Discourse Relations
Lifeng Jin and Marie-Catherine de Marneffe

Experiments in Open Domain Deception Detection
Verónica Pérez-Rosas and Rada Mihalcea

A model of rapid phonotactic generalization
Tal Linzen and Timothy O'Donnell

12:10–12:50 *Lunch*

12:50–13:30 Session P3: SIGDAT business meeting

13:30–15:10 Session 5A: Text Mining and NLP Applications 1 (Long + TACL Papers)

13:30–13:55 *[TACL] Unsupervised Identification of Translationese*
Ella Rabinovich and Shuly Wintner

13:55–14:20 *Automatically Solving Number Word Problems by Semantic Parsing and Reasoning*
Shuming Shi, Yuehui Wang, Chin-Yew Lin, Xiaojiang Liu and Yong Rui

14:20–14:45 *[TACL] Which Step Do I Take First? Troubleshooting with Bayesian Models*
Annie Louis and Mirella Lapata

14:45–15:10 *[TACL] Problems in Current Text Simplification Research: New Data Can Help*
Wei Xu, Chris Callison-Burch and Courtney Napoles

Sunday, September 20, 2015 (continued)

13:30–15:10 Session 5B: Semantics 3 (Long +TACL Papers)

13:30–13:55 *Parsing English into Abstract Meaning Representation Using Syntax-Based Machine Translation*
Michael Pust, Ulf Hermjakob, Kevin Knight, Daniel Marcu and Jonathan May

13:55–14:20 *The Forest Convolutional Network: Compositional Distributional Semantics with a Neural Chart and without Binarization*
Phong Le and Willem Zuidema

14:20–14:45 *Alignment-Based Compositional Semantics for Instruction Following*
Jacob Andreas and Dan Klein

14:45–15:10 *[TACL] Context-aware Frame-Semantic Role Labeling*
Michael Roth and Mirella Lapata

13:30–15:10 Session 5C: Phonology and Word Segmentation (Long Papers)

13:30–13:55 *Do we need bigram alignment models? On the effect of alignment quality on transduction accuracy in G2P*
Steffen Eger

13:55–14:20 *Keyboard Logs as Natural Annotations for Word Segmentation*
Fumihiko Takahasi and Shinsuke Mori

14:20–14:45 *Long Short-Term Memory Neural Networks for Chinese Word Segmentation*
Xinchi Chen, Xipeng Qiu, Chenxi Zhu, Pengfei Liu and Xuanjing Huang

14:45–15:10 *Semi-supervised Chinese Word Segmentation based on Bilingual Information*
Wei Chen and Bo Xu

13:30–15:10 Session 5D (P1-8): Machine Translation and Multilinguality (Long Paper Posters)

Hierarchical Back-off Modeling of Hiero Grammar based on Non-parametric Bayesian Model
Hidetaka Kamigaito, Taro Watanabe, Hiroya Takamura, Manabu Okumura and Eiichiro Sumita

Consistency-Aware Search for Word Alignment
Shiqi Shen, Yang Liu, Maosong Sun and Huanbo Luan

Graph-Based Collective Lexical Selection for Statistical Machine Translation
Jinsong Su, Deyi Xiong, Shujian Huang, Xianpei Han and Junfeng Yao

Bilingual Correspondence Recursive Autoencoder for Statistical Machine Translation
Jinsong Su, Deyi Xiong, Biao Zhang, Yang Liu, Junfeng Yao and Min Zhang

How to Avoid Unwanted Pregnancies: Domain Adaptation using Neural Network Models
Shafiq Joty, Hassan Sajjad, Nadir Durrani, Kamla Al-Mannai, Ahmed Abdelali and Stephan Vogel

Detecting Content-Heavy Sentences: A Cross-Language Case Study
Junyi Jessy Li and Ani Nenkova

Search-Aware Tuning for Hierarchical Phrase-based Decoding
Feifei Zhai, Liang Huang and Kai Zhao

Part-of-speech Taggers for Low-resource Languages using CCA Features
Young-Bum Kim, Benjamin Snyder and Ruhi Sarikaya

Sunday, September 20, 2015 (continued)

13:30–15:10 Session 5E (P1-12): Tagging, Chunking, Sytnax and Parsing (Short Paper Posters)

An Improved Tag Dictionary for Faster Part-of-Speech Tagging
Robert Moore

Improving Arabic Diacritization through Syntactic Analysis
Anas Shahrour, Salam Khalifa and Nizar Habash

Combining Discrete and Continuous Features for Deterministic Transition-based Dependency Parsing
Meishan Zhang and Yue Zhang

Efficient Inner-to-outer Greedy Algorithm for Higher-order Labeled Dependency Parsing
Xuezhe Ma and Eduard Hovy

Online Updating of Word Representations for Part-of-Speech Tagging
Wenpeng Yin, Tobias Schnabel and Hinrich Schütze

Empty Category Detection using Path Features and Distributed Case Frames
Shunsuke Takeno, Masaaki Nagata and Kazuhide Yamamoto

Foreebank: Syntactic Analysis of Customer Support Forums
Rasoul Kaljahi, Jennifer Foster, Johann Roturier, Corentin Ribeyre, Teresa Lynn and Joseph Le Roux

Semi-supervised Dependency Parsing using Bilexical Contextual Features from Auto-Parsed Data
Eliyahu Kiperwasser and Yoav Goldberg

Improved Transition-Based Parsing and Tagging with Neural Networks
Chris Alberti, David Weiss, Greg Coppola and Slav Petrov

Syntactic Parse Fusion
Do Kook Choe, David McClosky and Eugene Charniak

Not All Contexts Are Created Equal: Better Word Representations with Variable Attention
Wang Ling, Yulia Tsvetkov, Silvio Amir, Ramon Fermandez, Chris Dyer, Alan W Black, Isabel Trancoso and Chu-Cheng Lin

An Improved Non-monotonic Transition System for Dependency Parsing
Matthew Honnibal and Mark Johnson

Sunday, September 20, 2015 (continued)

15:10–15:40 ***Coffee break***

15:40–17:20 **Session 6A: Machine Translation 2 (Long Papers)**

15:40–16:05 *Improving Statistical Machine Translation with a Multilingual Paraphrase Database*
Ramtin Mehdizadeh Seraj, Maryam Siahbani and Anoop Sarkar

16:05–16:30 *Learning Semantic Representations for Nonterminals in Hierarchical Phrase-Based Translation*
Xing Wang, Deyi Xiong and Min Zhang

16:30–16:55 *A Comparison between Count and Neural Network Models Based on Joint Translation and Reordering Sequences*
Andreas Guta, Tamer Alkhouli, Jan-Thorsten Peter, Joern Wuebker and Hermann Ney

16:55–17:20 *Effective Approaches to Attention-based Neural Machine Translation*
Thang Luong, Hieu Pham and Christopher D. Manning

15:40–17:20 **Session 6B: Sentiment Analysis and Opinion Mining 2 / Tagging, Chunking and Parsing 2 (Long Papers)**

15:40–16:05 *Document Modeling with Gated Recurrent Neural Network for Sentiment Classification*
Duyu Tang, Bing Qin and Ting Liu

16:05–16:30 *Fine-grained Opinion Mining with Recurrent Neural Networks and Word Embeddings*
Pengfei Liu, Shafiq Joty and Helen Meng

16:30–16:55 *Joint A* CCG Parsing and Semantic Role Labelling*
Mike Lewis, Luheng He and Luke Zettlemoyer

16:55–17:20 *Improving Semantic Parsing with Enriched Synchronous Context-Free Grammar*
Junhui Li, Muhua Zhu, Wei Lu and Guodong Zhou

15:40–17:20 Session 6C: Language and Vision / Information Extraction 2 (Long Papers)

15:40–16:05 *Solving Geometry Problems: Combining Text and Diagram Interpretation*
Minjoon Seo, Hannaneh Hajishirzi, Ali Farhadi, Oren Etzioni and Clint Malcolm

16:05–16:30 *Do You See What I Mean? Visual Resolution of Linguistic Ambiguities*
Yevgeni Berzak, Andrei Barbu, Daniel Harari, Boris Katz and Shimon Ullman

16:30–16:55 *Efficient and Expressive Knowledge Base Completion Using Subgraph Feature Extraction*
Matt Gardner and Tom Mitchell

16:55–17:20 *Representing Text for Joint Embedding of Text and Knowledge Bases*
Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury and Michael Gamon

15:40–17:20 Session 6D (P1-11): Statistical Models and Machine Learning Methods for NLP (Long Paper Posters)

A Utility Model of Authors in the Scientific Community
Yanchuan Sim, Bryan Routledge and Noah A. Smith

Finding Function in Form: Compositional Character Models for Open Vocabulary Word Representation
Wang Ling, Chris Dyer, Alan W Black, Isabel Trancoso, Ramon Fermandez, Silvio Amir, Luis Marujo and Tiago Luis

Syntax-Aware Multi-Sense Word Embeddings for Deep Compositional Models of Meaning
Jianpeng Cheng and Dimitri Kartsaklis

Conversation Trees: A Grammar Model for Topic Structure in Forums
Annie Louis and Shay B. Cohen

Fast, Flexible Models for Discovering Topic Correlation across Weakly-Related Collections
Jingwei Zhang, Aaron Gerow, Jaan Altosaar, James Evans and Richard Jean So

Molding CNNs for text: non-linear, non-consecutive convolutions
Tao Lei, Regina Barzilay and Tommi Jaakkola

Multi-Perspective Sentence Similarity Modeling with Convolutional Neural Networks
Hua He, Kevin Gimpel and Jimmy Lin

Posterior calibration and exploratory analysis for natural language processing models
Khanh Nguyen and Brendan O'Connor

A Generative Word Embedding Model and its Low Rank Positive Semidefinite Solution
Shaohua Li, Jun Zhu and Chunyan Miao

Reading Documents for Bayesian Online Change Point Detection
Taehoon Kim and Jaesik Choi

15:40–17:20 Session 6E (P1-13): Semantics (Short Paper Posters)

Recognizing Textual Entailment Using Probabilistic Inference
Lei Sha, Sujian Li, Baobao Chang, Zhifang Sui and Tingsong Jiang

Chinese Semantic Role Labeling with Bidirectional Recurrent Neural Networks
Zhen Wang, Tingsong Jiang, Baobao Chang and Zhifang Sui

Unsupervised Negation Focus Identification with Word-Topic Graph Model
Bowei Zou, Guodong Zhou and Qiaoming Zhu

Reverse-engineering Language: A Study on the Semantic Compositionality of German Compounds
Corina Dima

Event Detection and Factuality Assessment with Non-Expert Supervision
Kenton Lee, Yoav Artzi, Yejin Choi and Luke Zettlemoyer

Large-Scale Acquisition of Entailment Pattern Pairs by Exploiting Transitivity
Julien Kloetzer, Kentaro Torisawa, Chikara Hashimoto and Jong-Hoon Oh

Context-Dependent Knowledge Graph Embedding
Yuanfei Luo, Quan Wang, Bin Wang and Li Guo

Learning to Identify the Best Contexts for Knowledge-based WSD
Evgenia Wasserman Pritsker, William Cohen and Einat Minkov

Sunday, September 20, 2015 (continued)

Measuring Prerequisite Relations Among Concepts
Chen Liang, Zhaohui Wu, Wenyi Huang and C. Lee Giles

Adapting Phrase-based Machine Translation to Normalise Medical Terms in Social Media Messages
Nut Limsopatham and Nigel Collier

Script Induction as Language Modeling
Rachel Rudinger, Pushpendre Rastogi, Francis Ferraro and Benjamin Van Durme

Online Learning of Interpretable Word Embeddings
Hongyin Luo, Zhiyuan Liu, Huanbo Luan and Maosong Sun

A Strong Lexical Matching Method for the Machine Comprehension Test
Ellery Smith, Nicola Greco, Matko Bosnjak and Andreas Vlachos

19:00–23:00 Conference Dinner

Monday, September 21, 2015

07:30–18:00 Registration

09:00–10:00 Session P4: Plenary Session

09:00–09:05 *Best Paper Awards*
Chris Callison-Burch and Jian Su

09:05–09:30 *Broad-coverage CCG Semantic Parsing with AMR*
Yoav Artzi, Kenton Lee and Luke Zettlemoyer

09:30–09:55 *Semantically Conditioned LSTM-based Natural Language Generation for Spoken Dialogue Systems*
Tsung-Hsien Wen, Milica Gasic, Nikola Mrkšić, Pei-Hao Su, David Vandyke and Steve Young

09:55–10:05 *A Large Annotated Corpus for Learning Natural Language Inference*
Samuel R. Bowman, Gabor Angeli, Christopher Potts and Christopher D. Manning

10:05–10:30 ***Coffee break***

Monday, September 21, 2015 (continued)

10:30–12:10 Session 7A: Semantics 4 (Long +TACL Papers)

10:30–10:55 *Do Multi-Sense Embeddings Improve Natural Language Understanding?*
Jiwei Li and Dan Jurafsky

10:55–11:20 *Learning Semantic Composition to Detect Non-compositionality of Multiword Expressions*
Majid Yazdani, Meghdad Farahmand and James Henderson

11:20–11:45 *Solving General Arithmetic Word Problems*
Subhro Roy and Dan Roth

11:45–12:10 *[TACL] From Paraphrase Database to Compositional Paraphrase Model and Back*
John Wieting, Mohit Bansal, Kevin Gimpel, Karen Livescu and Dan Roth

10:30–12:10 Session 7B: Information Extraction 3 (Long Papers)

10:30–10:55 *Distant Supervision for Relation Extraction via Piecewise Convolutional Neural Networks*
Daojian Zeng, Kang Liu, Yubo Chen and Jun Zhao

10:55–11:20 *CORE: Context-Aware Open Relation Extraction with Factorization Machines*
Fabio Petroni, Luciano Del Corro and Rainer Gemulla

11:20–11:45 *Improved Relation Extraction with Feature-Rich Compositional Embedding Models*
Matthew R. Gormley, Mo Yu and Mark Dredze

11:45–12:10 *Classifying Relations via Long Short Term Memory Networks along Shortest Dependency Paths*
Yan Xu, Lili Mou, Ge Li, Yunchuan Chen, Hao Peng and Zhi Jin

10:30–12:10 Session 7C: Computational Psycholinguistics / Machine Translation 3 (Long Papers)

10:30–10:55 *A Computational Cognitive Model of Novel Word Generalization*
Aida Nematzadeh, Erin Grant and Suzanne Stevenson

10:55–11:20 *Personality Profiling of Fictional Characters using Sense-Level Links between Lexical Resources*
Lucie Flekova and Iryna Gurevych

11:20–11:45 *Leave-one-out Word Alignment without Garbage Collector Effects*
Xiaolin Wang, Masao Utiyama, Andrew Finch, Taro Watanabe and Eiichiro Sumita

11:45–12:10 *Generalized Agreement for Bidirectional Word Alignment*
Chunyang Liu, Yang Liu, Maosong Sun, Huanbo Luan and Heng Yu

10:30–12:10 Session 7D (P1-6): Word Segmentation, Tagging and Parsing (Long +TACL Paper Posters)

A Transition-based Model for Joint Segmentation, POS-tagging and Normalization
Tao Qian, Yue Zhang, Meishan Zhang, Yafeng Ren and Donghong Ji

Multilingual discriminative lexicalized phrase structure parsing
Benoit Crabbé

Hierarchical Low-Rank Tensors for Multilingual Transfer Parsing
Yuan Zhang and Regina Barzilay

Diversity in Spectral Learning for Natural Language Parsing
Shashi Narayan and Shay B. Cohen

Transition-based Dependency Parsing Using Two Heterogeneous Gated Recursive Neural Networks
Xinchi Chen, Yaqian Zhou, Chenxi Zhu, Xipeng Qiu and Xuanjing Huang

[TACL] A Graph-based Lattice Dependency Parser for Joint Morphological Segmentation and Syntactic Analysis
Wolfgang Seeker and Özlem Çetinoğlu

10:30–12:10 Session 7E (P1-3): Spoken Language Processing (Short Paper Posters)

Turn-taking phenomena in incremental dialogue systems
Hatim Khouzaimi, Romain Laroche and Fabrice Lefevre

Hierarchical Latent Words Language Models for Robust Modeling to Out-Of Domain Tasks
Ryo Masumura, Taichi Asami, Takanobu Oba, Hirokazu Masataki, Sumitaka Sakauchi and Akinori Ito

A Coarse-Grained Model for Optimal Coupling of ASR and SMT Systems for Speech Translation
Gaurav Kumar, Graeme Blackwood, Jan Trmal, Daniel Povey and Sanjeev Khudanpur

10:30–12:10 Session 7E (P4-18): Summarization (Short Paper Posters)

Abstractive Multi-document Summarization with Semantic Information Extraction
Wei Li

Concept-based Summarization using Integer Linear Programming: From Concept Pruning to Multiple Optimal Solutions
Florian Boudin, Hugo Mougard and Benoit Favre

GhostWriter: Using an LSTM for Automatic Rap Lyric Generation
Peter Potash, Alexey Romanov and Anna Rumshisky

Better Summarization Evaluation with Word Embeddings for ROUGE
Jun-Ping Ng and Viktoria Abrecht

Krimping texts for better summarization
Marina Litvak, Mark Last and Natalia Vanetik

From the Virtual to the RealWorld: Referring to Objects in Real-World Spatial Scenes
Dimitra Gkatzia, Verena Rieser, Phil Bartie and William Mackaness

An Unsupervised Bayesian Modelling Approach for Storyline Detection on News Articles
Deyu Zhou, Haiyang Xu and Yulan He

Topical Coherence for Graph-based Extractive Summarization
Daraksha Parveen, Hans-Martin Ramsl and Michael Strube

Monday, September 21, 2015 (continued)

Summarizing Student Responses to Reflection Prompts
Wencan Luo and Diane Litman

Extractive Summarization by Maximizing Semantic Volume
Dani Yogatama, Fei Liu and Noah A. Smith

LCSTS: A Large Scale Chinese Short Text Summarization Dataset
Baotian Hu, Qingcai Chen and Fangze Zhu

Discourse Planning with an N-gram Model of Relations
Or Biran and Kathleen McKeown

Experiments with Generative Models for Dependency Tree Linearization
Richard Futrell and Edward Gibson

Summarization Based on Embedding Distributions
Hayato Kobayashi, Masaki Noguchi and Taichi Yatsuka

Reversibility reconsidered: finite-state factors for efficient probabilistic sampling in parsing and generation
Marc Dymetman, Sriram Venkatapathy and Chunyang Xiao

12:10–13:30 ***Lunch***

13:30–15:15 **Session 8A: Fun and Quirky Topics (Short Papers)**

13:30–13:45 *A quantitative analysis of gender differences in movies using psycholinguistic normatives*
Anil Ramakrishna, Nikolaos Malandrakis, Elizabeth Staruk and Shrikanth Narayanan

13:45–14:00 *EMNLP versus ACL: Analyzing NLP research over time*
Sujatha Das Gollapalli and Xiaoli Li

14:00–14:15 *Answering Elementary Science Questions by Constructing Coherent Scenes using Background Knowledge*
Yang Li and Peter Clark

14:15–14:30 *WikiQA: A Challenge Dataset for Open-Domain Question Answering*
Yi Yang, Wen-tau Yih and Christopher Meek

Monday, September 21, 2015 (continued)

14:30–14:45 *Personalized Machine Translation: Predicting Translational Preferences*
Shachar Mirkin and Jean-Luc Meunier

14:45–15:00 *Talking to the crowd: What do people react to in online discussions?*
Aaron Jaech, Victoria Zayats, Hao Fang, Mari Ostendorf and Hannaneh Hajishirzi

15:00–15:15 *What Your Username Says About You*
Aaron Jaech and Mari Ostendorf

13:30–15:15 Session 8B: Semantics 5 (Short Papers)

13:30–13:45 *Knowledge Base Inference using Bridging Entities*
Bhushan Kotnis, Pradeep Bansal and Partha P. Talukdar

13:45–14:00 *Specializing Word Embeddings for Similarity or Relatedness*
Douwe Kiela, Felix Hill and Stephen Clark

14:00–14:15 *Evaluation of Word Vector Representations by Subspace Alignment*
Yulia Tsvetkov, Manaal Faruqui, Wang Ling, Guillaume Lample and Chris Dyer

14:15–14:30 *Higher-order logical inference with compositional semantics*
Koji Mineshima, Pascual Martínez-Gómez, Yusuke Miyao and Daisuke Bekki

14:30–14:45 *Any-language frame-semantic parsing*
Anders Johannsen, Héctor Martínez Alonso and Anders Søgaard

15:00–15:15 *What's in an Embedding? Analyzing Word Embeddings through Multilingual Evaluation*
Arne Köhn

13:30–15:15 Session 8C: Statistical Modeling, Machine Learning / Machine Translation (Short Papers)

13:30–13:45 *Joint Event Trigger Identification and Event Coreference Resolution with Structured Perceptron*
Jun Araki and Teruko Mitamura

13:45–14:00 *A Joint Dependency Model of Morphological and Syntactic Structure for Statistical Machine Translation*
Rico Sennrich and Barry Haddow

14:00–14:15 *Variable-Length Word Encodings for Neural Translation Models*
Rohan Chitnis and John DeNero

14:15–14:30 *A Binarized Neural Network Joint Model for Machine Translation*
Jingyi Zhang, Masao Utiyama, Eiichiro Sumita, Graham Neubig and Satoshi Nakamura

14:30–14:45 *Bayesian Optimization of Text Representations*
Dani Yogatama, Lingpeng Kong and Noah A. Smith

14:45–15:00 *A Comparative Study on Regularization Strategies for Embedding-based Neural Networks*
Hao Peng, Lili Mou, Ge Li, Yunchuan Chen, Yangyang Lu and Zhi Jin

15:00–15:15 *Efficient Hyper-parameter Optimization for NLP Applications*
Lidan Wang, Minwei Feng, Bowen Zhou, Bing Xiang and Sridhar Mahadevan

13:30–15:15 Session 8D (P1-6): NLP for the Web and Social Media, including Computational Social Science (Long Paper Posters)

Improved Arabic Dialect Classification with Social Media Data
Fei Huang

Exploiting Debate Portals for Semi-Supervised Argumentation Mining in User-Generated Web Discourse
Ivan Habernal and Iryna Gurevych

Confounds and Consequences in Geotagged Twitter Data
Umashanthi Pavalanathan and Jacob Eisenstein

Modeling Reportable Events as Turning Points in Narrative
Jessica Ouyang and Kathleen McKeown

Towards the Extraction of Customer-to-Customer Suggestions from Reviews
Sapna Negi and Paul Buitelaar

Using Content-level Structures for Summarizing Microblog Repost Trees
Jing Li, Wei Gao, Zhongyu Wei, Baolin Peng and Kam-Fai Wong

13:30–15:15 Session 8D (P7-9): Discourse (Long Paper Posters)

Intra-sentential Zero Anaphora Resolution using Subject Sharing Recognition
Ryu Iida, Kentaro Torisawa, Chikara Hashimoto, Jong-Hoon Oh and Julien Kloetzer

Estimation of Discourse Segmentation Labels from Crowd Data
Ziheng Huang, Jialu Zhong and Rebecca J. Passonneau

Comparing Word Representations for Implicit Discourse Relation Classification
Chloé Braud and Pascal Denis

13:30–15:15 Session 8E (P1-9): Discourse (Short Paper Posters)

Better Document-level Sentiment Analysis from RST Discourse Parsing
Parminder Bhatia, Yangfeng Ji and Jacob Eisenstein

Closing the Gap: Domain Adaptation from Explicit to Implicit Discourse Relations
Yangfeng Ji, Gongbo Zhang and Jacob Eisenstein

Wikification of Concept Mentions within Spoken Dialogues Using Domain Constraints from Wikipedia
Seokhwan Kim, Rafael E. Banchs and Haizhou Li

Shallow Convolutional Neural Network for Implicit Discourse Relation Recognition
Biao Zhang, Jinsong Su, Deyi Xiong, Yaojie Lu, Hong Duan and Junfeng Yao

On the Role of Discourse Markers for Discriminating Claims and Premises in Argumentative Discourse
Judith Eckle-Kohler, Roland Kluge and Iryna Gurevych

Fatal or not? Finding errors that lead to dialogue breakdowns in chat-oriented dialogue systems
Ryuichiro Higashinaka, Masahiro Mizukami, Kotaro Funakoshi, Masahiro Araki, Hiroshi Tsukahara and Yuka Kobayashi

Learning Word Meanings and Grammar for Describing Everyday Activities in Smart Environments
Muhammad Attamimi, Yuji Ando, Tomoaki Nakamura, Takayuki Nagai, Daichi Mochihashi, Ichiro Kobayashi and Hideki Asoh

Discourse Element Identification in Student Essays based on Global and Local Cohesion
Wei Song, Ruiji Fu, Lizhen Liu and Ting Liu

Adapting Coreference Resolution for Narrative Processing
Quynh Ngoc Thi Do, Steven Bethard and Marie-Francine Moens

13:30–15:15 Session 8E (P10-15): Phonology, Morphology and Word Segmentation (Short Paper Posters)

Joint Lemmatization and Morphological Tagging with Lemming
Thomas Müller, Ryan Cotterell, Alexander Fraser and Hinrich Schütze

Transducer Disambiguation with Sparse Topological Features
Gonzalo Iglesias, Adrià de Gispert and Bill Byrne

Arabic Diacritization with Recurrent Neural Networks
Yonatan Belinkov and James Glass

Automatic Diacritics Restoration for Hungarian
Attila Novák and Borbála Siklósi

Morphological Analysis for Unsegmented Languages using Recurrent Neural Network Language Model
Hajime Morita, Daisuke Kawahara and Sadao Kurohashi

Can Symbol Grounding Improve Low-Level NLP? Word Segmentation as a Case Study
Hirotaka Kameko, Shinsuke Mori and Yoshimasa Tsuruoka

15:15–15:40 ***Coffee break***

Monday, September 21, 2015 (continued)

15:40–17:20 Session 9A: Statistical Modeling and Machine Learning 3 (Long + TACL Papers)

15:40–16:05 *When Are Tree Structures Necessary for Deep Learning of Representations?*
Jiwei Li, Thang Luong, Dan Jurafsky and Eduard Hovy

16:05–16:30 *Discriminative Neural Sentence Modeling by Tree-Based Convolution*
Lili Mou, Hao Peng, Ge Li, Yan Xu, Lu Zhang and Zhi Jin

16:30–16:55 *Multi-Timescale Long Short-Term Memory Neural Network for Modelling Sentences and Documents*
Pengfei Liu, Xipeng Qiu, Xinchi Chen, Shiyu Wu and Xuanjing Huang

16:55–17:20 *[TACL] Learning Structural Kernels for Natural Language Processing*
Daniel Beck, Trevor Cohn, Christian Hardmeier and Lucia Specia

15:40–17:20 Session 9B: Text Mining and NLP Applications 2 (Long Papers)

15:40–16:05 *Verbal and Nonverbal Clues for Real-life Deception Detection*
Verónica Pérez-Rosas, Mohamed Abouelenien, Rada Mihalcea, Yao Xiao, CJ Linton and Mihai Burzo

16:05–16:30 *Social Media Text Classification under Negative Covariate Shift*
Geli Fei and Bing Liu

16:30–16:55 *Co-Training for Topic Classification of Scholarly Data*
Cornelia Caragea, Florin Bulgarov and Rada Mihalcea

16:55–17:20 *Humor Recognition and Humor Anchor Extraction*
Diyi Yang, Alon Lavie, Chris Dyer and Eduard Hovy

Monday, September 21, 2015 (continued)

15:40–17:20 Session 9C: Spoken Language Processing and Language Modeling (Long Papers)

15:40–16:05 *Topic Identification and Discovery on Text and Speech*
Chandler May, Francis Ferraro, Alan McCree, Jonathan Wintrode, Daniel Garcia-Romero and Benjamin Van Durme

16:05–16:30 *A Dynamic Programming Algorithm for Computing N-gram Posteriors from Lattices*
Dogan Can and Shrikanth Narayanan

16:30–16:55 *Bilingual Structured Language Models for Statistical Machine Translation*
Ekaterina Garmash and Christof Monz

16:55–17:20 *Compact, Efficient and Unlimited Capacity: Language Modeling with Compressed Suffix Trees*
Ehsan Shareghi, Matthias Petri, Gholamreza Haffari and Trevor Cohn

15:40–17:20 Session 9D (P1-8): Semantics (Long Paper Posters)

ERSOM: A Structural Ontology Matching Approach Using Automatically Learned Entity Representation
Chuncheng Xiang, Tingsong Jiang, Baobao Chang and Zhifang Sui

A Single Word is not Enough: Ranking Multiword Expressions Using Distributional Semantics
Martin Riedl and Chris Biemann

Syntactic Dependencies and Distributed Word Representations for Analogy Detection and Mining
Likun Qiu, Yue Zhang and Yanan Lu

Navigating the Semantic Horizon using Relative Neighborhood Graphs
Amaru Cuba Gyllensten and Magnus Sahlgren

Multi- and Cross-Modal Semantics Beyond Vision: Grounding in Auditory Perception
Douwe Kiela and Stephen Clark

Automatic recognition of habituals: a three-way classification of clausal aspect
Annemarie Friedrich and Manfred Pinkal

Distributed Representations for Unsupervised Semantic Role Labeling
Kristian Woodsend and Mirella Lapata

A Tableau Prover for Natural Logic and Language
Lasha Abzianidze

15:40–17:20 Session 9E (P1-9): Sentiment Analysis and Opinion Mining (Short Paper Posters)

JEAM: A Novel Model for Cross-Domain Sentiment Classification Based on Emotion Analysis
Kun-Hu Luo, Zhi-Hong Deng, Hongliang Yu and Liang-Chen Wei

PhraseRNN: Phrase Recursive Neural Network for Aspect-based Sentiment Analysis
Thien Hai Nguyen and Kiyoaki Shirai

ASTD: Arabic Sentiment Tweets Dataset
Mahmoud Nabil, Mohamed Aly and Amir Atiya

Adjective Intensity and Sentiment Analysis
Raksha Sharma, Mohit Gupta, Astha Agarwal and Pushpak Bhattacharyya

The Rating Game: Sentiment Rating Reproducibility from Text
Lasse Borgholt, Peter Simonsen and Dirk Hovy

A Multi-lingual Annotated Dataset for Aspect-Oriented Opinion Mining
Salud M. Jiménez-Zafra, Giacomo Berardi, Andrea Esuli, Diego Marcheggiani, María Teresa Martín-Valdivia and Alejandro Moreo Fernández

Deep Convolutional Neural Network Textual Features and Multiple Kernel Learning for Utterance-level Multimodal Sentiment Analysis
Soujanya Poria, Erik Cambria and Alexander Gelbukh

SLSA: A Sentiment Lexicon for Standard Arabic
Ramy Eskander and Owen Rambow

Reinforcing the Topic of Embeddings with Theta Pure Dependence for Text Classification
Ning Xing, Yuexian Hou, Peng Zhang, Wenjie Li and Dawei Song

15:40–17:20 **Session 9E (P10-17): NLP for the Web and Social Media, including Computational Social Science (Short Paper Posters)**

That's So Annoying!!!: A Lexical and Frame-Semantic Embedding Based Data Augmentation Approach to Automatic Categorization of Annoying Behaviors using #petpeeve Tweets
William Yang Wang and Diyi Yang

Detection of Steganographic Techniques on Twitter
Alex Wilson, Phil Blunsom and Andrew Ker

#SupportTheCause: Identifying Motivations to Participate in Online Health Campaigns
Dong Nguyen, Tijs van den Broek, Claudia Hauff, Djoerd Hiemstra and Michel Ehrenhard

An Analysis of Domestic Abuse Discourse on Reddit
Nicolas Schrading, Cecilia Ovesdotter Alm, Ray Ptucha and Christopher Homan

Twitter-scale New Event Detection via K-term Hashing
Dominik Wurzer, Victor Lavrenko and Miles Osborne

Classifying Tweet Level Judgements of Rumours in Social Media
Michal Lukasik, Trevor Cohn and Kalina Bontcheva

Identification and Verification of Simple Claims about Statistical Properties
Andreas Vlachos and Sebastian Riedel

17:30–17:50 **Session P5: Closing Remarks**

18:30–20:00 **Farewell Drink**

List of Papers

Language Understanding for Text-based Games using Deep Reinforcement Learning

Karthik Narasimhan[*]
CSAIL, MIT
karthikn@csail.mit.edu

Tejas D Kulkarni[*]
CSAIL, BCS, MIT
tejask@mit.edu

Regina Barzilay
CSAIL, MIT
regina@csail.mit.edu

Abstract

In this paper, we consider the task of learning control policies for text-based games. In these games, all interactions in the virtual world are through text and the underlying state is not observed. The resulting language barrier makes such environments challenging for automatic game players. We employ a deep reinforcement learning framework to jointly learn state representations and action policies using game rewards as feedback. This framework enables us to map text descriptions into vector representations that capture the semantics of the game states. We evaluate our approach on two game worlds, comparing against baselines using bag-of-words and bag-of-bigrams for state representations. Our algorithm outperforms the baselines on both worlds demonstrating the importance of learning expressive representations. [1]

State 1: The old bridge
You are standing very close to the bridge's eastern foundation. If you go east you will be back on solid ground ... The bridge sways in the wind.

Command: **Go east**

State 2: Ruined gatehouse
The old gatehouse is near collapse. Part of its northern wall has already fallen down ... East of the gatehouse leads out to a small open area surrounded by the remains of the castle. There is also a standing archway offering passage to a path along the old southern inner wall.
Exits: Standing archway, castle corner, Bridge over the abyss

Figure 1: Sample gameplay from a Fantasy World. The player with the quest of finding a secret tomb, is currently located on an *old bridge*. She then chooses an action to *go east* that brings her to a *ruined gatehouse* (State 2).

1 Introduction

In this paper, we address the task of learning control policies for text-based strategy games. These games, predecessors to modern graphical ones, still enjoy a large following worldwide.[2] They often involve complex worlds with rich interactions and elaborate textual descriptions of the underlying states (see Figure 1). Players read descriptions of the current world state and respond with natural language commands to take actions. Since the underlying state is not directly observable, the player has to understand the text in order to act, making it challenging for existing AI programs to play these games (DePristo and Zubek, 2001).

In designing an autonomous game player, we have considerable latitude when selecting an adequate state representation to use. The simplest method is to use a bag-of-words representation derived from the text description. However, this scheme disregards the ordering of words and the finer nuances of meaning that evolve from composing words into sentences and paragraphs. For instance, in State 2 in Figure 1, the agent has to understand that going *east* will lead it to the castle whereas moving *south* will take it to the standing archway. An alternative approach is to convert text descriptions to pre-specified representations using annotated training data, commonly used in

[*]Both authors contributed equally to this work.
[1]Code is available at http://people.csail.mit.edu/karthikn/mud-play.
[2]http://mudstats.com/

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1–11,
Lisbon, Portugal, 17-21 September 2015.

language grounding tasks (Matuszek et al., 2013; Kushman et al., 2014).

In contrast, our goal is to learn useful representations in conjunction with control policies. We adopt a reinforcement learning framework and formulate game sequences as Markov Decision Processes. An agent playing the game aims to maximize rewards that it obtains from the game engine upon the occurrence of certain events. The agent learns a policy in the form of an action-value function $Q(s, a)$ which denotes the long-term merit of an action a in state s.

The action-value function is parametrized using a deep recurrent neural network, trained using the game feedback. The network contains two modules. The first one converts textual descriptions into vector representations that act as proxies for states. This component is implemented using Long Short-Term Memory (LSTM) networks (Hochreiter and Schmidhuber, 1997). The second module of the network scores the actions given the vector representation computed by the first.

We evaluate our model using two Multi-User Dungeon (MUD) games (Curtis, 1992; Amir and Doyle, 2002). The first game is designed to provide a controlled setup for the task, while the second is a publicly available one and contains human generated text descriptions with significant language variability. We compare our algorithm against baselines of a random player and models that use bag-of-words or bag-of-bigrams representations for a state. We demonstrate that our model LSTM-DQN significantly outperforms the baselines in terms of number of completed quests and accumulated rewards. For instance, on a fantasy MUD game, our model learns to complete 96% of the quests, while the bag-of-words model and a random baseline solve only 82% and 5% of the quests, respectively. Moreover, we show that the acquired representation can be reused across games, speeding up learning and leading to faster convergence of Q-values.

2 Related Work

Learning control policies from text is gaining increasing interest in the NLP community. Example applications include interpreting help documentation for software (Branavan et al., 2010), navigating with directions (Vogel and Jurafsky, 2010; Kollar et al., 2010; Artzi and Zettlemoyer, 2013; Matuszek et al., 2013; Andreas and Klein, 2015) and playing computer games (Eisenstein et al., 2009; Branavan et al., 2011a).

Games provide a rich domain for grounded language analysis. Prior work has assumed perfect knowledge of the underlying state of the game to learn policies. Gorniak and Roy (2005) developed a game character that can be controlled by spoken instructions adaptable to the game situation. The grounding of commands to actions is learned from a transcript manually annotated with actions and state attributes. Eisenstein et al. (2009) learn game rules by analyzing a collection of game-related documents and precompiled traces of the game. In contrast to the above work, our model combines text interpretation and strategy learning in a single framework. As a result, textual analysis is guided by the received control feedback, and the learned strategy directly builds on the text interpretation.

Our work closely relates to an automatic game player that utilizes text manuals to learn strategies for Civilization (Branavan et al., 2011a). Similar to our approach, text analysis and control strategies are learned jointly using feedback provided by the game simulation. In their setup, states are fully observable, and the model learns a strategy by combining state/action features and features extracted from text. However, in our application, the state representation is not provided, but has to be inferred from a textual description. Therefore, it is not sufficient to extract features from text to supplement a simulation-based player.

Another related line of work consists of automatic video game players that infer state representations directly from raw pixels (Koutník et al., 2013; Mnih et al., 2015). For instance, Mnih et al. (2015) learn control strategies using convolutional neural networks, trained with a variant of Q-learning (Watkins and Dayan, 1992). While both approaches use deep reinforcement learning for training, our work has important differences. In order to handle the sequential nature of text, we use Long Short-Term Memory networks to automatically learn useful representations for arbitrary text descriptions. Additionally, we show that decomposing the network into a representation layer and an action selector is useful for transferring the learnt representations to new game scenarios.

3 Background

Game Representation We represent a game by the tuple $\langle H, A, T, R, \Psi \rangle$, where H is the set of

all possible game states, $A = \{(a, o)\}$ is the set of all commands (action-object pairs), $T(h' \mid h, a, o)$ is the stochastic transition function between states and $R(h, a, o)$ is the reward function. The game state H is *hidden* from the player, who only receives a varying textual description, produced by a stochastic function $\Psi : H \rightarrow S$. Specifically, the underlying state h in the game engine keeps track of attributes such as the player's location, her health points, time of day, etc. The function Ψ (also part of the game framework) then converts this state into a textual description of the location the player is at or a message indicating low health. We do not assume access to either H or Ψ for our agent during both training and testing phases of our experiments. We denote the space of all possible text descriptions s to be S. Rewards are generated using R and are only given to the player upon completion of in-game quests.

Q-Learning Reinforcement Learning is a commonly used framework for learning control policies in game environments (Silver et al., 2007; Amato and Shani, 2010; Branavan et al., 2011b; Szita, 2012). The game environment can be formulated as a sequence of state transitions (s, a, r, s') of a Markov Decision Process (MDP). The agent takes an action a in state s by consulting a state-action value function $Q(s, a)$, which is a measure of the action's expected long-term reward. Q-Learning (Watkins and Dayan, 1992) is a model-free technique which is used to learn an optimal $Q(s, a)$ for the agent. Starting from a random Q-function, the agent continuously updates its Q-values by playing the game and obtaining rewards. The iterative updates are derived from the Bellman equation (Sutton and Barto, 1998):

$$Q_{i+1}(s, a) = \mathrm{E}[r + \gamma \max_{a'} Q_i(s', a') \mid s, a] \quad (1)$$

where γ is a discount factor for future rewards and the expectation is over all game transitions that involved the agent taking action a in state s.

Using these evolving Q-values, the agent chooses the action with the highest $Q(s, a)$ to maximize its expected future rewards. In practice, the trade-off between exploration and exploitation can be achieved following an ϵ-greedy policy (Sutton and Barto, 1998), where the agent performs a random action with probability ϵ.

Deep Q-Network In large games, it is often impractical to maintain the Q-value for all possible state-action pairs. One solution to this problem is to approximate $Q(s, a)$ using a parametrized function $Q(s, a; \theta)$, which can generalize over states and actions by considering higher-level attributes (Sutton and Barto, 1998; Branavan et al., 2011a). However, creating a good parametrization requires knowledge of the state and action spaces. One way to bypass this feature engineering is to use a Deep Q-Network (DQN) (Mnih et al., 2015). The DQN approximates the Q-value function with a deep neural network to predict $Q(s, a)$ for all possible actions a simultaneously given the current state s. The non-linear function layers of the DQN also enable it to learn better value functions than linear approximators.

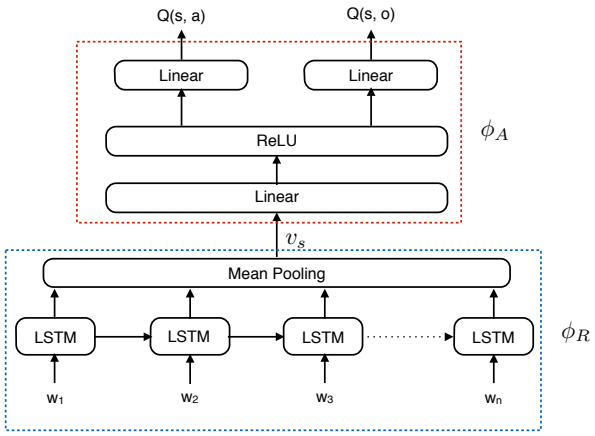


Figure 2: Architecture of LSTM-DQN: The Representation Generator (ϕ_R) (bottom) takes as input a stream of words observed in state s and produces a vector representation v_s, which is fed into the action scorer (ϕ_A) (top) to produce scores for all actions and argument objects.

4 Learning Representations and Control Policies

In this section, we describe our model (DQN) and describe its use in learning good Q-value approximations for games with stochastic textual descriptions. We divide our model into two parts. The first module is a *representation generator* that converts the textual description of the current state into a vector. This vector is then input into the second module which is an *action scorer*. Figure 2 shows the overall architecture of our model. We learn the parameters of both the representation generator and the action scorer jointly, using the in-game reward feedback.

Representation Generator (ϕ_R) The representation generator reads raw text displayed to the agent and converts it to a vector representation v_s. A bag-of-words (BOW) representation is not sufficient to capture higher-order structures of sentences and paragraphs. The need for a better semantic representation of the text is evident from the average performance of this representation in playing MUD-games (as we show in Section 6).

In order to assimilate better representations, we utilize a Long Short-Term Memory network (LSTM) (Hochreiter and Schmidhuber, 1997) as a representation generator. LSTMs are recurrent neural networks with the ability to connect and recognize long-range patterns between words in text. They are more robust than BOW to small variations in word usage and are able to capture underlying semantics of sentences to some extent. In recent work, LSTMs have been used successfully in NLP tasks such as machine translation (Sutskever et al., 2014) and sentiment analysis (Tai et al., 2015) to compose vector representations of sentences from word-level embeddings (Mikolov et al., 2013; Pennington et al., 2014). In our setup, the LSTM network takes in word embeddings w_k from the words in a description s and produces output vectors x_k at each step.

To get the final state representation v_s, we add a *mean pooling* layer which computes the element-wise mean over the output vectors x_k.[3]

$$v_s = \frac{1}{n}\sum_{k=1}^{n} x_k \tag{2}$$

Action Scorer (ϕ_A) The action scorer module produces scores for the set of possible actions given the current state representation. We use a multi-layered neural network for this purpose (see Figure 2). The input to this module is the vector from the representation generator, $v_s = \phi_R(s)$ and the outputs are scores for actions $a \in A$. Scores for all actions are predicted simultaneously, which is computationally more efficient than scoring each state-action pair separately. Thus, by combining the representation generator and action scorer, we can obtain the approximation for the Q-function as $Q(s,a) \approx \phi_A(\phi_R(s))[a]$.

An additional complexity in playing MUD-games is that the actions taken by the player are multi-word natural language *commands* such as *eat apple* or *go east*. Due to computational constraints, in this work we limit ourselves to consider commands to consist of one action (e.g. *eat*) and one argument object (e.g. *apple*). This assumption holds for the majority of the commands in our worlds, with the exception of one class of commands that require two arguments (e.g. *move red-root right*, *move blue-root up*). We consider all possible actions and objects available in the game and predict both for each state using the same network (Figure 2). We consider the Q-value of the entire command (a, o) to be the average of the Q-values of the action a and the object o. For the rest of this section, we only show equations for $Q(s,a)$ but similar ones hold for $Q(s,o)$.

Parameter Learning We learn the parameters θ_R of the representation generator and θ_A of the action scorer using stochastic gradient descent with *RMSprop* (Tieleman and Hinton, 2012). The complete training procedure is shown in Algorithm 1. In each iteration i, we update the parameters to reduce the discrepancy between the predicted value of the current state $Q(s_t, a_t; \theta_i)$ (where $\theta_i = [\theta_R; \theta_A]_i$) and the expected Q-value given the reward r_t and the value of the next state $\max_a Q(s_{t+1}, a; \theta_{i-1})$.

We keep track of the agent's previous experiences in a *memory* $\mathcal{D}$.[4] Instead of performing updates to the Q-value using transitions from the current episode, we sample a random transition $(\hat{s}, \hat{a}, s', r)$ from $\mathcal{D}$. Updating the parameters in this way avoids issues due to strong correlation when using transitions of the same episode (Mnih et al., 2015). Using the sampled transition and (1), we obtain the following loss function to minimize:

$$\mathcal{L}_i(\theta_i) = \mathrm{E}_{\hat{s},\hat{a}}[(y_i - Q(\hat{s}, \hat{a}; \theta_i))^2] \tag{3}$$

where $y_i = \mathrm{E}_{\hat{s},\hat{a}}[r + \gamma \max_{a'} Q(s', a'; \theta_{i-1}) \mid \hat{s}, \hat{a}]$ is the target Q-value with parameters θ_{i-1} fixed from the previous iteration.

The updates on the parameters θ can be performed using the following gradient of $\mathcal{L}_i(\theta_i)$:

$$\nabla_{\theta_i}\mathcal{L}_i(\theta_i) = \mathrm{E}_{\hat{s},\hat{a}}[2(y_i - Q(\hat{s}, \hat{a}; \theta_i))\nabla_{\theta_i} Q(\hat{s}, \hat{a}; \theta_i)]$$

For each epoch of training, the agent plays several episodes of the game, which is restarted after every terminal state.

[3] We also experimented with considering just the output vector of the LSTM after processing the last word. Empirically, we find that mean pooling leads to faster learning, so we use it in all our experiments.

[4] The memory is limited and rewritten in a first-in-first-out (FIFO) fashion.

Algorithm 1 Training Procedure for DQN with prioritized sampling

Initialize experience memory $\mathcal{D}$
Initialize parameters of representation generator (ϕ_R) and action scorer (ϕ_A) randomly
for $episode = 1, M$ **do**
    Initialize game and get start state description s_1
    for $t = 1, T$ **do**
        Convert s_t (text) to representation v_{s_t} using ϕ_R
        if $random() < \epsilon$ **then**
            Select a random action a_t
        else
            Compute $Q(s_t, a)$ for all actions using $\phi_A(v_{s_t})$
            Select $a_t = \text{argmax}\ Q(s_t, a)$
        Execute action a_t and observe reward r_t and new state s_{t+1}
        Set priority $p_t = 1$ if $r_t > 0$, else $p_t = 0$
        Store transition $(s_t, a_t, r_t, s_{t+1}, p_t)$ in $\mathcal{D}$
        Sample random mini batch of transitions $(s_j, a_j, r_j, s_{j+1}, p_j)$ from $\mathcal{D}$, with fraction ρ having $p_j = 1$
        Set $y_j = \begin{cases} r_j & \text{if } s_{j+1} \text{ is terminal} \\ r_j + \gamma \max_{a'} Q(s_{j+1}, a'; \theta) & \text{if } s_{j+1} \text{ is non-terminal} \end{cases}$
        Perform gradient descent step on the loss $\mathcal{L}(\theta) = (y_j - Q(s_j, a_j; \theta))^2$

Mini-batch Sampling In practice, online updates to the parameters θ are performed over a mini batch of state transitions, instead of a single transition. This increases the number of experiences used per step and is also more efficient due to optimized matrix operations.

The simplest method to create these mini-batches from the experience memory $\mathcal{D}$ is to sample uniformly at random. However, certain experiences are more valuable than others for the agent to learn from. For instance, rare transitions that provide positive rewards can be used more often to learn optimal Q-values faster. In our experiments, we consider such positive-reward transitions to have higher *priority* and keep track of them in $\mathcal{D}$. We use *prioritized sampling* (inspired by Moore and Atkeson (1993)) to sample a fraction ρ of transitions from the higher priority pool and a fraction $1 - \rho$ from the rest.

5 Experimental Setup

Game Environment For our game environment, we modify Evennia,[5] an open-source library for building online textual MUD games. Evennia is a Python-based framework that allows one to easily create new games by writing a batch file describing the environment with details of rooms, objects and actions. The game engine keeps track of the game state internally, presenting textual descriptions to the player and receiving text commands from the player. We conduct experiments on two worlds - a smaller *Home world* we created ourselves, and a larger, more complex *Fantasy world* created by Evennia's developers. The motivation behind Home world is to abstract away high-level planning and focus on the language understanding requirements of the game.

Stats	Home World	Fantasy World
Vocabulary size	84	1340
Avg. words / description	10.5	65.21
Max descriptions / room	3	100
# diff. quest descriptions	12	-
State transitions	Deterministic	Stochastic
# states (underlying)	16	≥ 56
Branching factor (# commands / state)	40	222

Table 1: Various statistics of the two game worlds

Table 1 provides statistics of the game worlds. We observe that the Fantasy world is moderately sized with a vocabulary of 1340 words and up to 100 different descriptions for a room. These descriptions were created manually by the game developers. These diverse, engaging descriptions are designed to make it interesting and exciting for human players. Several rooms have many alternative descriptions, invoked randomly on each visit by

[5] http://www.evennia.com/

the player.

Comparatively, the Home world is smaller: it has a very restricted vocabulary of 84 words and the room descriptions are relatively structured. However, both the room descriptions (which are also varied and randomly provided to the agent) and the quest descriptions were adversarially created with negation and conjunction of facts to force an agent to actually understand the state in order to play well. Therefore, this domain provides an interesting challenge for language understanding.

In both worlds, the agent receives a positive reward on completing a quest, and negative rewards for getting into bad situations like falling off a bridge, or losing a battle. We also add small deterministic negative rewards for each non-terminating step. This incentivizes the agent to learn policies that solve quests in fewer steps. The supplementary material has details on the reward structure.

Home World We created *Home world* to mimic the environment of a typical house.[6] The world consists of four rooms - a living room, a bedroom, a kitchen and a garden with connecting pathways. Every room is reachable from every other room. Each room contains a representative object that the agent can interact with. For instance, the kitchen has an *apple* that the player can *eat*. Transitions between the rooms are deterministic. At the start of each game episode, the player is placed in a random room and provided with a randomly selected quest. The text provided to the player contains both the description of her current state and that of the quest. Thus, the player can begin in one of 16 different states (4 rooms $\times$ 4 quests), which adds to the world's complexity.

An example of a quest given to the player in text is *Not you are sleepy now but you are hungry now*. To complete this quest and obtain a reward, the player has to navigate through the house to reach the kitchen and eat the apple (i.e type in the command *eat apple*). More importantly, the player should interpret that the quest does not require her to take a nap in the bedroom. We created such misguiding quests to make it hard for agents to succeed without having an adequate level of language understanding.

[6] An illustration is provided in the supplementary material.

Fantasy World The Fantasy world is considerably more complex and involves quests such as navigating through a broken bridge or finding the secret tomb of an ancient hero. This game also has stochastic transitions in addition to varying state descriptions provided to the player. For instance, there is a possibility of the player falling from the bridge if she lingers too long on it.

Due to the large command space in this game,[7] we make use of cues provided by the game itself to narrow down the set of possible objects to consider in each state. For instance, in the MUD example in Figure 1, the game provides a list of possible exits. If the game does not provide such clues for the current state, we consider all objects in the game.

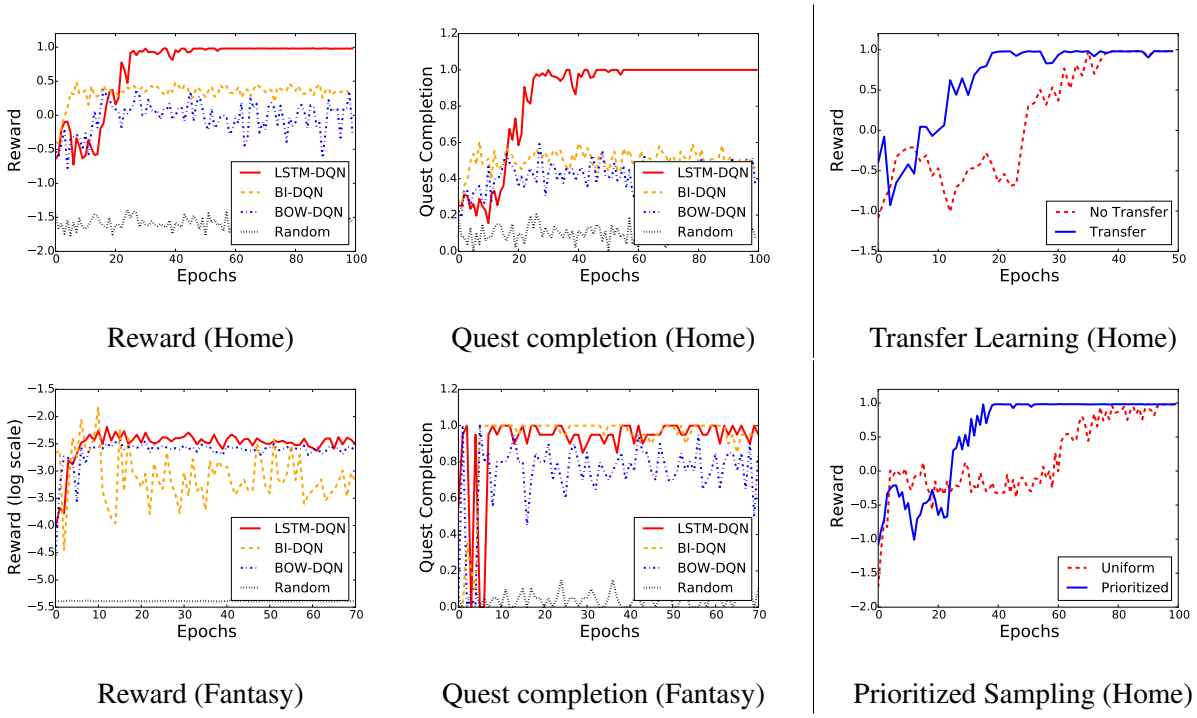
Evaluation We use two metrics for measuring an agent's performance: (1) the cumulative reward obtained per episode averaged over the episodes and (2) the fraction of quests completed by the agent. The evaluation procedure is as follows. In each epoch, we first train the agent on M episodes of T steps each. At the end of this training, we have a testing phase of running M episodes of the game for T steps. We use $M = 50, T = 20$ for the Home world and $M = 20, T = 250$ for the Fantasy world. For all evaluation episodes, we run the agent following an ϵ-greedy policy with $\epsilon = 0.05$, which makes the agent choose the best action according to its Q-values 95% of the time. We report the agent's performance at each epoch.

Baselines We compare our LSTM-DQN model with three baselines. The first is a *Random* agent that chooses both actions and objects uniformly at random from all available choices.[8] The other two are BOW-DQN and BI-DQN, which use a bag-of-words and a bag-of-bigrams representation of the text, respectively, as input to the DQN action scorer. These baselines serve to illustrate the importance of having a good representation layer for the task.

Settings For our DQN models, we used $\mathcal{D} = 100000, \gamma = 0.5$. We use a learning rate of 0.0005 for RMSprop. We anneal the ϵ for ϵ-greedy from 1 to 0.2 over 100000 transitions. A mini-batch gradient update is performed every 4 steps of the gameplay. We roll out the LSTM (over words) for

[7] We consider 222 possible command combinations of 6 actions and 37 object arguments.

[8] In the case of the Fantasy world, the object choices are narrowed down using game clues as described earlier.



Reward (Home) | Quest completion (Home) | Transfer Learning (Home)

Reward (Fantasy) | Quest completion (Fantasy) | Prioritized Sampling (Home)

Figure 3: **Left:** Graphs showing the evolution of average reward and quest completion rate for BOW-DQN, LSTM-DQN and a Random baseline on the Home world (**top**) and Fantasy world (**bottom**). Note that the reward is shown in log scale for the Fantasy world. **Right:** Graphs showing effects of transfer learning and prioritized sampling on the Home world.

a maximum of 30 steps on the Home world and for 100 steps on the Fantasy world. For the prioritized sampling, we used $\rho = 0.25$ for both worlds. We employed a mini-batch size of 64 and word embedding size $d = 20$ in all experiments.

6 Results

Home World Figure 3 illustrates the performance of LSTM-DQN compared to the baselines. We can observe that the Random baseline performs quite poorly, completing only around 10% of quests on average[9] obtaining a low reward of around -1.58. The BOW-DQN model performs significantly better and is able to complete around 46% of the quests, with an average reward of 0.20. The improvement in reward is due to both greater quest success rate and a lower rate of issuing invalid commands (e.g. *eat apple* would be invalid in the bedroom since there is no apple). We notice that both the reward and quest completion graphs of this model are volatile. This is because the model fails to pick out differences between quests like *Not you are hungry now but you are sleepy now* and *Not you are sleepy now but you are hungry now*. The BI-DQN model suffers from the same issue although it performs slightly better than BOW-DQN by completing 48% of quests. In contrast, the LSTM-DQN model does not suffer from this issue and is able to complete 100% of the quests after around 50 epochs of training, achieving close to the optimal reward possible.[10] This demonstrates that having an expressive representation for text is crucial to understanding the game states and choosing intelligent actions.

In addition, we also investigated the impact of using a deep neural network for modeling the action scorer ϕ_A. Figure 4 illustrates the performance of the BOW-DQN and BI-DQN models along with their simpler versions BOW-LIN and BI-LIN, which use a single linear layer for ϕ_A. It can be seen that the DQN models clearly achieve better performance than their linear counterparts, which points to them modeling the control policy better.

Fantasy World We evaluate all the models on the Fantasy world in the same manner as before and report reward, quest completion rates and Q-

[9] Averaged over the last 10 epochs.

[10] Note that since each step incurs a penalty of -0.01, the best reward (on average) a player can get is around 0.98.

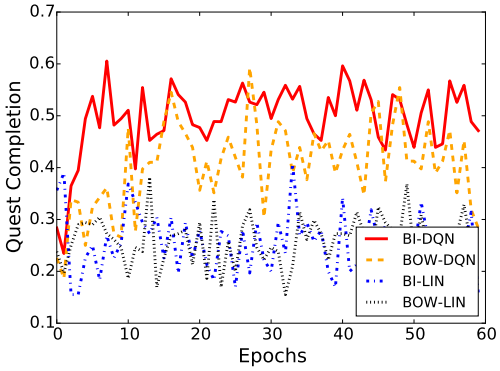


Figure 4: Quest completion rates of DQN vs. Linear models on Home world.

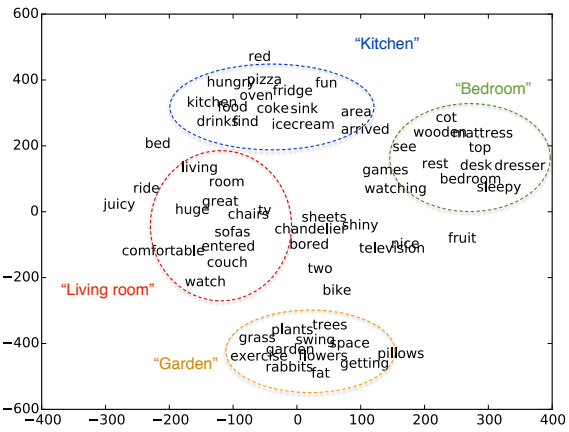


Figure 5: t-SNE visualization of the word embeddings (except stopwords) after training on Home world. The embedding values are initialized randomly.

values. The quest we evaluate on involves crossing the broken bridge (which takes a minimum of five steps), with the possibility of falling off at random (a 5% chance) when the player is on the bridge. The game has an additional quest of reaching a secret tomb. However, this is a complex quest that requires the player to memorize game events and perform high-level planning which are beyond the scope of this current work. Therefore, we focus only on the first quest.

From Figure 3 (bottom), we can see that the Random baseline does poorly in terms of both average per-episode reward[11] and quest completion rates. BOW-DQN converges to a much higher average reward of -12.68 and achieves around 82% quest completion. Again, the BOW-DQN is often confused by varying (10 different) descriptions of the portions of the bridge, which reflects in its erratic performance on the quest. The BI-DQN performs very well on quest completion by finishing 97% of quests. However, this model tends to find sub-optimal solutions and gets an average reward of -26.68, even worse than BOW-DQN. One reason for this is the negative rewards the agent obtains after falling off the bridge. The LSTM-DQN model again performs best, achieving an average reward of -11.33 and completing 96% of quests on average. Though this world does not contain descriptions adversarial to BOW-DQN or BI-DQN, the LSTM-DQN obtains higher average reward by completing the quest in fewer steps and showing more resilience to variations in the state descriptions.

Transfer Learning We would like the representations learnt by ϕ_R to be generic enough and *transferable* to new game worlds. To test this, we created a second Home world with the same rooms, but a completely different map, changing the locations of the rooms and the pathways between them. The main differentiating factor of this world from the original home world lies in the high-level planning required to complete quests.

We initialized the LSTM part of an LSTM-DQN agent with parameters θ_R learnt from the original home world and trained it on the new world.[12] Figure 3 (top right) demonstrates that the agent with transferred parameters is able to learn quicker than an agent starting from scratch initialized with random parameters (*No Transfer*), reaching the optimal policy almost 20 epochs earlier. This indicates that these simulated worlds can be used to learn good representations for language that transfer across worlds.

Prioritized sampling We also investigate the effects of different minibatch sampling procedures on the parameter learning. From Figure 3 (bottom right), we observe that using prioritized sampling significantly speeds up learning, with the agent achieving the optimal policy around 50 epochs faster than using uniform sampling. This shows promise for further research into different schemes of assigning priority to transitions.

Representation Analysis We analyzed the representations learnt by the LSTM-DQN model on the Home world. Figure 5 shows a visualization

[11] Note that the rewards graph is in log scale.

[12] The parameters for the Action Scorer (θ_A) are initialized randomly.

Description	Nearest neighbor
You are halfways out on the unstable bridge. From the castle you hear a distant howling sound, like that of a large dog or other beast.	The bridge slopes precariously where it extends westwards towards the lowest point - the center point of the hang bridge. You clasp the ropes firmly as the bridge sways and creaks under you.
The ruins opens up to the sky in a small open area, lined by columns. ... To the west is the gatehouse and entrance to the castle, whereas southwards the columns make way for a wide open courtyard.	The old gatehouse is near collapse. .... East the gatehouse leads out to a small open area surrounded by the remains of the castle. There is also a standing archway offering passage to a path along the old southern inner wall.

Table 2: Sample descriptions from the Fantasy world and their nearest neighbors (NN) according to their vector representations from the LSTM representation generator. The NNs are often descriptions of the same or similar (nearby) states in the game.

of learnt word embeddings, reduced to two dimensions using t-SNE (Van der Maaten and Hinton, 2008). All the vectors were initialized randomly before training. We can see that semantically similar words appear close together to form coherent subspaces. In fact, we observe four different subspaces, each for one type of room along with its corresponding object(s) and quest words. For instance, food items like *pizza* and rooms like *kitchen* are very close to the word *hungry* which appears in a quest description. This shows that the agent learns to form meaningful associations between the semantics of the quest and the environment. Table 2 shows some examples of descriptions from Fantasy world and their nearest neighbors using cosine similarity between their corresponding vector representations produced by LSTM-DQN. The model is able to correlate descriptions of the same (or similar) underlying states and project them onto nearby points in the representation subspace.

7 Conclusions

We address the task of end-to-end learning of control policies for text-based games. In these games, all interactions in the virtual world are through text and the underlying state is not observed. The resulting language variability makes such environments challenging for automatic game players. We employ a deep reinforcement learning framework to jointly learn state representations and action policies using game rewards as feedback. This framework enables us to map text descriptions into vector representations that capture the semantics of the game states. Our experiments demonstrate the importance of learning good representations of text in order to play these games well. Future directions include tackling high-level planning and strategy learning to improve the performance of intelligent agents.

Acknowledgements

We are grateful to the developers of Evennia, the game framework upon which this work is based. We also thank Nate Kushman, Clement Gehring, Gustavo Goretkin, members of MIT's NLP group and the anonymous EMNLP reviewers for insightful comments and feedback. T. Kulkarni was graciously supported by the Leventhal Fellowship. We would also like to acknowledge MIT's Center for Brains, Minds and Machines (CBMM) for support.

References

Christopher Amato and Guy Shani. 2010. High-level reinforcement learning in strategy games. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: Volume 1*, pages 75–82. International Foundation for Autonomous Agents and Multiagent Systems.

Eyal Amir and Patrick Doyle. 2002. Adventure games: A challenge for cognitive robotics. In *Proc. Int. Cognitive Robotics Workshop*, pages 148–155.

Jacob Andreas and Dan Klein. 2015. Alignment-based compositional semantics for instruction following. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Yoav Artzi and Luke Zettlemoyer. 2013. Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association for Computational Linguistics*, 1(1):49–62.

SRK Branavan, Luke S Zettlemoyer, and Regina Barzilay. 2010. Reading between the lines: Learning to map high-level instructions to commands. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1268–1277. Association for Computational Linguistics.

SRK Branavan, David Silver, and Regina Barzilay. 2011a. Learning to win by reading manuals in a monte-carlo framework. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 268–277. Association for Computational Linguistics.

SRK Branavan, David Silver, and Regina Barzilay. 2011b. Non-linear monte-carlo search in Civilization II. AAAI Press/International Joint Conferences on Artificial Intelligence.

Pavel Curtis. 1992. Mudding: Social phenomena in text-based virtual realities. *High noon on the electronic frontier: Conceptual issues in cyberspace*, pages 347–374.

Mark A DePristo and Robert Zubek. 2001. being-in-the-world. In *Proceedings of the 2001 AAAI Spring Symposium on Artificial Intelligence and Interactive Entertainment*, pages 31–34.

Jacob Eisenstein, James Clarke, Dan Goldwasser, and Dan Roth. 2009. Reading to learn: Constructing features from semantic abstracts. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 958–967, Singapore, August. Association for Computational Linguistics.

Peter Gorniak and Deb Roy. 2005. Speaking with your sidekick: Understanding situated speech in computer role playing games. In R. Michael Young and John E. Laird, editors, *Proceedings of the First Artificial Intelligence and Interactive Digital Entertainment Conference, June 1-5, 2005, Marina del Rey, California, USA*, pages 57–62. AAAI Press.

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

Thomas Kollar, Stefanie Tellex, Deb Roy, and Nicholas Roy. 2010. Toward understanding natural language directions. In *Human-Robot Interaction (HRI), 2010 5th ACM/IEEE International Conference on*, pages 259–266. IEEE.

Jan Koutník, Giuseppe Cuccu, Jürgen Schmidhuber, and Faustino Gomez. 2013. Evolving large-scale neural networks for vision-based reinforcement learning. In *Proceedings of the 15th annual conference on Genetic and evolutionary computation*, pages 1061–1068. ACM.

Nate Kushman, Yoav Artzi, Luke Zettlemoyer, and Regina Barzilay. 2014. Learning to automatically solve algebra word problems. *ACL (1)*, pages 271–281.

Cynthia Matuszek, Evan Herbst, Luke Zettlemoyer, and Dieter Fox. 2013. Learning to parse natural language commands to a robot control system. In *Experimental Robotics*, pages 403–415. Springer.

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

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. 2015. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 02.

Andrew W Moore and Christopher G Atkeson. 1993. Prioritized sweeping: Reinforcement learning with less data and less time. *Machine Learning*, 13(1):103–130.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. *Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014)*, 12.

David Silver, Richard S Sutton, and Martin Müller. 2007. Reinforcement learning of local shape in the game of go. In *IJCAI*, volume 7, pages 1053–1058.

Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112.

Richard S Sutton and Andrew G Barto. 1998. *Introduction to reinforcement learning*. MIT Press.

István Szita. 2012. Reinforcement learning in games. In *Reinforcement Learning*, pages 539–577. Springer.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1556–1566, Beijing, China, July. Association for Computational Linguistics.

Tijmen Tieleman and Geoffrey Hinton. 2012. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning*, 4.

Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(2579-2605):85.

Adam Vogel and Dan Jurafsky. 2010. Learning to follow navigational directions. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 806–814. Association for Computational Linguistics.

Christopher JCH Watkins and Peter Dayan. 1992. Q-learning. *Machine learning*, 8(3-4):279–292.

Distributional vectors encode referential attributes

Abhijeet Gupta* **Gemma Boleda†** **Marco Baroni†** **Sebastian Padó***
*Institut für Maschinelle Sprachverarbeitung
Universität Stuttgart
†Center for Mind/Brain Sciences
University of Trento

Abstract

Distributional methods have proven to excel at capturing fuzzy, graded aspects of meaning (*Italy* is more similar to *Spain* than to *Germany*). In contrast, it is difficult to extract the values of more specific attributes of word referents from distributional representations, attributes of the kind typically found in structured knowledge bases (*Italy* has *60 million* inhabitants). In this paper, we pursue the hypothesis that distributional vectors also implicitly encode referential attributes.

We show that a standard supervised regression model is in fact sufficient to retrieve such attributes to a reasonable degree of accuracy: When evaluated on the prediction of both categorical and numeric attributes of countries and cities, the model consistently reduces baseline error by 30%, and is not far from the upper bound. Further analysis suggests that our model is able to "objectify" distributional representations for entities, anchoring them more firmly in the external world in measurable ways.

1 Introduction

Distributional models induce vector-based semantic representations of words from their contextual distributions in corpora, exploiting the observation that words with related meanings tend to occur in similar linguistic contexts (Turney and Pantel, 2010; Erk, 2012). Since the approach only requires raw text as input, it can be used to harvest word representations on a very large scale. By encoding the rich knowledge that is present in text, these representations are able to capture many aspects of word meaning. Moreover, approximating semantic similarity by graded geometric distance in a vector space is an effective strategy to address the many linguistic phenomena that are better characterized in gradient rather than discrete terms, such as synonymy, selectional preferences, and semantic priming (Baroni and Lenci, 2010; Erk et al., 2010; Padó and Lapata, 2007, among others).

However, not all aspects of human semantic knowledge are satisfactorily captured in terms of fuzzy relations and graded similarity. In particular, our knowledge of the meaning of words denoting specific entities involves a number of "hard facts" about the referents they denote that are best formalized as attribute-value pairs, of the sort that are stored in manually-curated knowledge bases, such as FreeBase or Wikidata.[1] While distributional vectors can capture the useful fact that, say, *Italy* is in many ways more similar to *Spain* than to *Germany*, as humans we also know (or we can easily look up) a set of objective facts about Italy, such as what is its capital, its area, its official language and GDP, that are difficult to express in the language of vector algebra and geometry.

In this paper, we explore the hypothesis that distributional vectors implicitly encode such attributes of referential entities, which we will call *referential attributes*. We show that a simple supervised algorithm applied to vectors can retrieve them so that they can be expressed in the explicit language of structured knowledge bases. Concretely, we train a logistic regression model to predict the values of both numeric and categorical FreeBase attributes of countries and cities from their distributional vectors. This model makes predictions that are significantly better than an informed baseline, in-between the latter and an upper-bound method. Qualitative analysis of the results points both to the inherent difficulty of correctly retrieving certain classes of attributes, and to some intriguing properties of the conceptual nature of the knowledge encoded in distributional data, that bias their predictions about certain objective attributes of geographic entities.

[1] `www.freebase.com`, `www.wikidata.org`.

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 12–21,
Lisbon, Portugal, 17-21 September 2015.

We see our experiment as a first step towards integrating conceptual and referential aspects of meaning in distributional semantics, as we further discuss in the conclusion.

2 Method

2.1 Distributional Representations

Mikolov et al.'s (2013) skip-gram model is a state-of-the-art "predictive" distributional semantic model which represents each word in a space of latent dimensions optimized to predict the contexts of the word's occurrences. For our study, we adopt the pre-trained 1,000-dimensional skip-gram model for Named Entities that is available at `https://code.google.com/p/word2vec/` and was produced from a 100-billion token news corpus. We refer to this model as WORD2VEC.

2.2 Referential Representations

As our source of referential attributes, we use FreeBase (see footnote 1), a knowledge base of structured information on a wide range of entities of different semantic types (people, geographical entities, etc.). The information in FreeBase comes from various sources, including Wikipedia and domain-specific databases, plus user content generation and correction. FreeBase currently records at least 2 attributes for over 47 million entities, and it has been used fairly extensively in NLP before (Mintz et al., 2009; Socher et al., 2013a, among others).

For each *entity*, FreeBase contains a list of *attribute-value* tuples (where values can in turn be entities, allowing a graph view of the data that we do not exploit here). Table 1 shows a sample of the attributes that FreeBase records for countries. Note that some attributes are simple (e.g., `date_founded`), while other can be called complex, in the sense that they are attributes of attributes (e.g., `geolocation::latitude`). We use a double-colon notation to refer to complex attributes. The values of all attributes can be either numeric or categorical. The numeric attributes in particular are often strongly correlated, both within attributes types across years (e.g., fertility rate in different years) and across attributes within years (e.g., absolute GDP and GDP per capita in a given year).

We built two datasets for our experiments, one for countries and one for cities, with data automatically extracted from FreeBase.[2] We consider two datasets in order to check that the mapping we seek can be established not just for one, possible hand-picked, type of entities; we leave it to future work to study very different kinds of entities, such as people or institutions.

Attribute	Value
`geolocation::latitude`	52.52
`geolocation::longitude`	13.38
`fertility_rate::1960`	2.37
`fertility_rate::1994`	1.24
`fertility_rate::2010`	1.39
`date_founded`	1871-01-18
`containedBy`	Western Europe
`containedBy`	Europe
`containedBy`	Eurasia
`adjectival_form`	German

Table 1: Sample of numeric and binary FreeBase attributes for *Germany*.

The *Countries* dataset consists of the 260 countries for which we have a distributional vector. Some countries do not exist anymore, like Yugoslavia, but, since this does not impact our method, we keep them in the dataset. The dataset records all simple attributes as well as complex attributes of at most two hops in the FreeBase graph, without manual inspection. We linearly rescale all numeric attributes to [0..1] and translate all categorical attributes into a binary representation by suffixing the original value to the original attribute name. For example, the attribute `member-of::organization` with the value `world_bank` results in a binary attribute `member-of::organization::world_bank` having value 1 for all and only those countries that are members of the World Bank, 0 for the others.[3] Attributes that occur less than 15 times are discarded, since they are either not consistently recorded or rare. This results in a total of 707 numeric and 247 binary attributes. Finally, we partition the data into training, validation, and test set, using a 60-20-20 percent split.

We apply the same process to the *Cities* dataset, which consists of 1645 cities from the intersection of the distributional and FreeBase city lists. In

[2]Both datasets are publicly available at `http://www.ims.uni-stuttgart.de/forschung/ressourcen/korpora/CityCountry.html`.

[3]We considered treating some categorical attributes as multi-valued, but decided against it since the cases in which alternative values are mutually exclusive are rare (e.g., the same country can be `containedBy` multiple entities, cf. Table 1).

this case, we have 211 numeric and 106 binary attributes – the numbers are smaller because countries have a richer representation in FreeBase than cities.

2.3 Attribute Prediction

We do zero-shot learning of full FreeBase attribute-based country/city representations, based on distributional (WORD2VEC) representations. It is zero-shot learning in the sense of Palatucci et al. (2009): We split the datasets at the entity, rather than attribute level, such that at test time our system must predict the full attribute set of countries and cities that were not seen during training at all.

We use logistic regression. In effect, we predict each output variable (FreeBase attribute) with an independent logistic regression model based on a constant set of input features (WORD2VEC distributional dimensions). We call this model DIST2REF. DIST2REF does *not* take advantage of the correlations between the output attributes mentioned in Section 2.2.

The dependent variables are binary as well as numeric FreeBase attributes, and our model does not distinguish between them. For binary attributes, we interpret the value returned by the model as the probability of "success" of a binary Bernoulli trial. In the numeric case, we view the probability returned by the model as directly representing normalized attribute values.

2.4 Experimental Setup

We design the model using the Countries dataset, and apply it to Cities without further tuning to test its robustness. We optimize the parameters with gradient descent, using the Cross Entropy error function. We considered L_2 regularization to address possible overfitting, but experiments on validation set showed that the model performs best without any regularization.

As for baselines, for binary features we predict the majority class (0 or 1), and for numeric features we predict the mean value of the feature in the training set. These are of course strong baselines to beat.

As an upper bound, we train a model that uses the same architecture as described above but uses as input not distributional vectors but the FreeBase attributes themselves. In other words, this model has to learn "only" an identity mapping. This is not trivial, though, for example due to the presence of strong correlations among attributes, in particular the time series attributes (cf. Section 2.2). We call this model REF2REF.

2.5 Evaluation

Since there is no appropriate unified evaluation measure that covers both numeric and binary attributes, we evaluate them separately. For binary attributes, we report the attributes' mean accuracy.

For numeric attributes, we consider attribute prediction a ranking task. As an example, take the `population::2011::number` attribute, and imagine that we only have three countries (Germany: 80M; Spain: 36M; and Netherlands: 17M). If we predict 56M for Spain's population, it is still (correctly) predicted as the second most populous country (rank difference of 0); a prediction of 16M, however, would push Spain to third place (rank difference of 1).

This suggests the use of rank correlation coefficients like Spearman's ρ. However, we want to measure not only how well the model can rank the countries in the test set, but also whether these predictions are consistent with the training set (which makes evaluation both more challenging and more realistic). One way of achieving this goal would be to use ρ on the union of training and test instances, but this could lead to misleadingly high correlation coefficients since this method would include the labels of the training instances in the evaluation.

Consequently, we define our own evaluation measure, following a rationale similar to Frome et al.'s (2013) evaluation of a zero-shot learning scenario. What we evaluate, for each attribute, is the rank of the test countries in the whole country list. Note that this makes our task harder, as there are more confounders: If we only evaluated on the test set, there would be shorter lists and therefore less chances of getting bad rankings. So, concretely, we first define the prediction quality of each attribute, $Q(a)$, as the median of the rank difference between the prediction and the gold standard in a list that includes both training and test countries (we use the median to give less weight to outlier countries). We also normalize the rank difference to obtain a number between zero and one. In a second step, we define the quality of the complete model, the *normalized rank score (NRS)*, as the mean of all attribute quality scores, in parallel to our evaluation on binary attributes.

Let the set of instances I be partitioned into training instances Tr and test instances Ts. Let $a \in A$

Attribute Type	Model	Countries	Cities
Binary (Acc)	Most Frequent Class Baseline	0.86	0.97
	DIST2REF	0.90	0.99
	REF2REF (upper bound)	0.96	1.00
Numeric (NRS)	Mean Value Baseline	0.35	0.35
	DIST2REF	0.22	0.25
	REF2REF (upper bound)	0.14	0.21

Table 2: Results for predicting FreeBase attributes from distributional vectors on the test sets. Both evaluation measures range between 0 and 1. For accuracy, 1 is best. For normalized rank score (NRS), 0 is best. All pairwise differences between models are significant ($p<0.001$, bootstrap resampling).

denote an attribute. We write $p_a(i)$ for the predicted value of attribute a for instance i and $g_a(i)$ for the gold standard value. Finally, let $r(v, S)$ denote the rank of value v in the list resulting when ordering the set S. Now we can define:

$$Q(a) = \frac{1}{||I||} med\{|r(p_a(i), I) - r(g_a(i), I)| - 1 \mid i \in Ts\} \quad (1)$$

$$NRS = \frac{1}{||A||} \sum_{a \in A} Q(a) \quad (2)$$

This measure can be interpreted similarly to Mean Reciprocal Rank (Manning et al., 2008): It has range [0..1], with smaller numbers indicating better ranking: 0.1, for example, means that, on average, the prediction is 10% of the ranks off (e.g., by four countries in a forty-country list).[4]

Note that, when evaluating each instance i, we use gold-standard values for all other instances, so that there the baseline is not hampered by ties.

3 Results

Table 2 shows the results of our experiments on the two test sets. For accuracy 1 is best, but for NRS 0 is best. Recall from Section 2.2 that we perform model selection on the Countries dataset only.

The baseline is relatively high, in particular for the binary attributes, many of which are positive for a small subset of entities only. The amount of skew differs considerably between the two datasets, though. For Countries, the baseline yields an accuracy of 0.86, but it achieves 0.97 on Cities. The increase stems from very sparse categorical City features such as `containedBy`, which includes all levels of administrative divisions – that is, for the US, all counties appear as values and are transformed into sparse binary features (cf. Section 2.2). Of course, the predictions of the baseline are useless, since it always predicts the absence of any features. On numeric features, where the baseline predicts the mean, its performance is 0.35 NRS on both datasets. In other words, its average prediction is off by about one third the length of the ranked list for each attribute.

Recall that the upper bound model, REF2REF, uses FreeBase attributes to predict FreeBase attributes. All it has to learn is that there is one feature in the input that corresponds ideally to the output. This works almost perfectly for binary attributes, with accuracy values of 0.96 (Countries) and 1.00 (Cities). However, its performance on numeric features (with NRS at 0.14 and 0.21, respectively) is not quite perfect. We attribute this to the presence of correlations (cf. Section 2.2).

The model whose performance we are actually interested in, DIST2REF, in which we map from distributional information to FreeBase features, performs with remarkable consistency between these two extremes. In fact, we see a consistent error reduction of around 30% over the baseline, with a similar distance to the upper bound. A significance test with bootstrap resampling (Efron and Tibshirani, 1994) showed that all pairwise comparisons (Baseline vs. DIST2REF, DIST2REF vs. REF2REF) are statistically significant at $p<0.001$.

To rule out that we misinterpret our accuracy-based evaluation for the binary features in the face of a highly skewed class distribution, we also computed precision, recall, and F-Score values. The relative patterns match those of the accuracy-based evaluation well (Countries: baseline F=0.13, DIST2REF F=0.51, REF2REF F=0.77) and indicate that generally precision is higher than recall.

[4] Subtracting 1 in Equation (1) ensures that, when the predicted and gold value of an attribute are adjacent in the ranking, their rank difference is 0, capturing the intuition of rank difference as counting the number of falsely intervening items.

We think that these are overall promising results, given that the FreeBase attributes we predict are fairly fine-grained, and we only use generic distributional information as input.

4 Analysis

We take the overall results just presented to suggest that we are able to learn referential attributes from distributional information to a large extent. In this section we take a closer look at what kind of information we are able to learn, what is beyond the scope of our model, and what are the differences between the entity representations in WORD2VEC and the ones our model produces. All the data concerns the test sets only.

4.1 Attribute Groups

We start with a qualitative analysis of the Countries dataset. Due to the large number of attributes, we sort all individual attributes into *attribute groups* by their base name (i.e. the leftmost component of their name, cf. Section 2.2), which offers an accessible level of granularity for inspection. We obtain 34 numeric and 40 binary attribute groups with median sizes of 8.5 and 2 attributes per group, respectively.

Table 3 shows the attribute groups for both types sorted by quality. For each group, we report average normalized rank score (NRS) and accuracy, respectively, for both DIST2REF and the baseline.

The analysis suggests that there are two main factors that account for the results: (1) The degree to which an attribute is *contextually supported*, that is, to what extent its values can be identified on the basis of the contextual information that is captured in a distributional model, and (2) general properties of the data that affect Machine Learning, most notably data sparseness, possibly also feature value distributions.

Attributes that are contextually supported include for instance those related to socioeconomic development (see below for details); people talk (and so write) about countries being more or less developed, rich, having one or another kind of laws, and this is captured in the abstractions over textual context that distributional models perform. As an extreme example of an attribute that is *not* contextually supported, consider the numeric ISO code of a country (`iso_numeric`), whose values are arbitrary: They do not correspond to facts about the world that are reflected in the way people use language, and so can't be picked up by the distributional model. For this reason, DIST2REF does worse than the baseline.

Note that, in a sufficiently large corpus, we might indeed encounter statements like *The numeric ISO code for Spain is 724*. However, since distributional models represent words as *aggregated* distributions of their contexts, and compute semantic similarity from these context distributions, the contexts that they use need to be generic enough to yield meaningful overlap between concepts (e.g., words). As a result, distributional models cannot easily represent knowledge of the form "the value for property Y of word/concept X is Z".

Fortunately, we find that many FreeBase attributes are contextually supported to a substantial degree, even some seemingly arbitrary ones. An example is calling codes, which we predict very well. They turn out to be correlated with geolocations: 2X calling codes are located in Africa, 3X calling codes in Southern and Eastern Europe and 4X calling codes in Western and Northern Europe (for comparison, ISO codes are assigned in a roughly alphabetical order).

Numeric Attributes. Our best numeric attributes belong to the `geolocation` group (latitude and longitude). We provide a more detailed analysis of these attributes below (Section 4.2). As mentioned above, we also excel at many attributes related to a country's economic and social development (broadly construed), such as GNI, GDP, CO_2 emissions, internet usage (each per capita), or fertility rate. These attributes can be expected to be contextually grounded – e.g., Luxembourg will occur with contexts like "broadband" or "rich" more than India.

Note, however, that the information contained in the vectors is surprisingly subtle: For instance, the fertility rate is a function of both general development status (lower rates in more developed countries) and of specific social factors (higher rates in countries with more support for families, such as France and Finland compared countries with less support, such as Germany or Italy).

Around the middle of the table, we find the absolute versions of the developmental cluster above (GNI in $, real and nominal GDP). Evidently, the absolute versions of these attributes are substantially less contextually supported than the relative versions. This is not surprising: While India and China have high absolute GDPs because they are

Numeric Attributes (Normalized Rank Score: lower is better)					Binary Attributes (Accuracy: higher is better)				
Attribute Group	DIST2REF	BL	#A	f(A)	Attribute Group	DIST2REF	BL	#A	f(A)
geolocation	0.07	0.30	2	250	continent	0.98	0.84	4	45
gdp_nominal_per_capita	0.11	0.27	1	172	time_zones	0.98	0.93	2	26
gni_per_capita_in_ppp_dollars	0.12	0.28	32	155	containedBy	0.98	0.81	9	49
co2_emissions_per_capita	0.12	0.25	49	157	casualties!	0.96	0.97	2	17
fertility_rate	0.12	0.24	52	178	places_exported_to!	0.96	0.98	2	17
calling_code	0.12	0.27	1	205	member_of	0.95	0.86	25	27
internet_users_percent_pop	0.13	0.32	22	184	championships_athletes!	0.94	0.96	1	22
entry	0.14	0.23	2	140	military_conflicts	0.94	0.94	2	18
gni_in_ppp_dollars	0.16	0.31	32	154	organizations	0.94	0.93	8	20
broadband_penetration_rate	0.17	0.68	15	23	entry	0.94	0.81	5	30
population_growth_rate	0.19	0.31	52	201	minimum_wage	0.93	0.93	2	20
military_expenditure_perc_gdp	0.20	0.27	24	128	gdp_nominal	0.92	0.85	1	213
gdp_real	0.20	0.34	51	149	religions	0.92	0.93	3	23
life_expectancy	0.20	0.24	52	179	tournaments_participated_in	0.91	0.91	2	27
electricity_cons._per_capita	0.22	0.36	50	105	places_imported_from	0.91	0.91	2	18
gdp_nominal	0.22	0.34	52	157	athletic_performances	0.91	0.89	30	26
energy_use_per_capita	0.23	0.39	51	104	medals_won	0.91	0.89	29	31
population	0.25	0.42	54	202	gdp_nominal_per_capita	0.90	0.85	1	215
places_imported_from	0.26	0.29	2	18	currency_used	0.89	0.89	2	26
iso_numeric!	0.26	0.23	1	220	official_language	0.89	0.81	4	32
national_anthem_since	0.27	0.43	1	97	administrative_area_type	0.89	0.69	1	185
championships_athletes	0.28	0.33	1	18	companies_founded	0.89	0.83	3	39
gdp_growth_rate	0.28	0.41	51	154	organizations_founded	0.89	0.83	3	39
government_debt_percent_gdp!	0.33	0.19	17	24	schools_founded	0.89	0.83	3	39
casualties!	0.39	0.35	1	33	olympics_participated_in	0.88	0.81	9	55
athletic_performances_rank	0.43	0.43	1	34	tour_operators	0.88	0.89	3	40
date_founded!	0.46	0.41	1	61	athletes	0.88	0.86	48	36
date_dissolved	0.48	0.48	1	21	languages_spoken	0.88	0.84	5	38
climate_avg_rainfall!	0.50	0.38	1	4	government_bodies	0.88	0.87	2	34
force_deployments	0.53	0.58	2	20	administrative_parent	0.87	0.69	1	185
religions_percentage	0.58	0.66	2	14	gdp_real	0.87	0.73	1	189
minimum_wage	0.63	0.82	28	17	gni_in_ppp_dollars	0.87	0.62	1	170
					gni_per_capita_in_ppp_dollars	0.87	0.62	1	170
					is_clear	0.87	0.87	1	23
					governing_officials	0.86	0.82	14	34
					form_of_government	0.84	0.81	11	42
					equivalent_instances	0.79	0.75	1	200
					exceptions	0.69	0.67	1	87
					loc_type	0.69	0.58	1	146
					adjectival_form!	0.65	0.69	1	65

Table 3: Results for all attribute groups on the Countries test set, in descending order of performance. DIST2REF, BL: models; **#A**: number of attributes in group; **f(A)**: median number of countries instantiating each attribute in the dataset (260 countries); **!**: attribute group where model performs worse than baseline.

large countries, and for instance Luxembourg has a much smaller one, these numbers are not indicative of the actual conditions in these countries, and therefore also not so clearly correlated with what people write about them. This provides another interesting angle on the difference between distributional and formal knowledge representation. In a formal system, absolute GDP, relative GDP, and population stand in a fixed linear relationship and knowing any two of the three uniquely determines the third – thus, all three attributes have equal status. In our distributional space, their status is clearly different, determined by the conceptual relevance of the different attributes.

Towards the end of the table, we find more attributes related to socioeconomic development, such as `government_percent_debt` and `minimum_wage`. While these should be contextually supported, too, the problem here is factor (2) mentioned above, namely severe data sparsity (see column **f(A)** in Table 3, which lists the median number of datapoints that exhibit each attribute group). The same goes for the remaining attribute groups, for instance `casualties` (describing the

total number of military casualties incurred in history), date_founded and date_dissolved,[5] or climate_avg_rainfall.

Binary Attributes. The binary attributes show a similar picture, albeit somewhat less sharp. We again find contextually unsupported groups, many of them arising from our fully automatic attribute mining from FreeBase (cf. Section 2.2). There are many categorical attributes that store metadata about numeric attributes (such as the currency in the gdp and gni groups) as well as meta-information of FreeBase: exceptions is a specific marker of potentially inconsistent entries about Ghana, and equivalent_instances is a flag concerning links between FreeBase and OpenCyc. Fortunately, almost all contextually unsupported groups are small, with only one or two attributes, and do not have a large impact on the overall performance. We decided not to exclude them from evaluation for robustness' sake, since there is no automatic way to identify contextually unsupported attributes in a new dataset.

We obtain good results on meaningful attributes that are arguably strongly contextually grounded, such as geographical and geopolitical attributes (member_of: membership in international organizations; location on a continent, etc.). However, we fare relatively badly on government-related attributes (form of government, governing officials). While this seems surprising at first glance, the form_of_government attribute in FreeBase makes very fine-grained distinctions: Its values include "unitary state", "presidential system", "parliamentary system" and "republic", which are not mutually exclusive, and misses obvious alternatives like "authoritarian system". It is not surprising that distributional models cannot make such subtle distinction between presidential and parliamentary systems. The attribute governing_official presents a similar case. Other bad attributes are very domain-specific, including athletes, encoding the athletic disciplines that countries participate in (such as swimming, judo, running, etc.), and the data sparsity issue is certainly worse for the binary attributes.

[5]Note that date-based attributes can be contextually supported: We do better on national_anthem_since, for which we have more datapoints, 97.

Model	Countries	Cities
WORD2VEC	-0.36	-0.45
DIST2REF	0.49	0.88

Table 4: Pearson correlation coefficients of model-predicted vs. ground truth distances between countries and cities in the test sets. WORD2VEC correlations are negative because we use cosines.

4.2 Geolocation

To analyze the difference between the distributional representations and the output of our model, we focus on geolocation, our best attribute group.

It has already been shown that geometric distance in distributional space captures, to a certain extent, physical distance between locations in the real world (Louwerse and Zwaan, 2009). Table 4 shows that DIST2REF extracts even more precise distance information from distributional vectors. The table reports the correlation between real and model-predicted distances for countries and cities. Ground-truth *great circle distances* (Kern and Bland, 1948) between items are computed using the FreeBase longitude and latitude values; for DIST2REF we use its predicted latitude and longitude values; for WORD2VEC, the cosines between the corresponding distributional vectors.

We obtain highly significant correlations in all cases ($p<10^{-14}$), but much higher for DIST2REF. For countries, as shown in Table 4, the correlation is -0.36 for WORD2VEC (negative, because cosine is a *similarity* measure), 0.49 for DIST2REF. For cities, WORD2VEC reaches -0.45 correlation, and DIST2REF distances are at 0.88, showing that the method can estimate city positions to a perhaps unexpectedly high degree of accuracy.[6]

This result suggests that we manage to objectify the information in the distributional model, anchoring the entities more firmly in the external world. Indeed, distributional models are known to be subject to conceptual or cultural effects in their distance estimations. For instance, in WORD2VEC German and Spanish cities are much farther away than in the physical world, while cities within Spain and within Germany are predicted to be a bit closer than they actually are. Note that these effects have

[6]The results are confirmed when the analysis is repeated using the Spearman correlation measure: The DIST2REF coefficients are stable, whereas those of WORD2VEC go down to 0.22 (countries) and 0.40 (cities), respectively. The good results for Spearman, as a rank-based measure, indicate that our success is not dominated by outliers.

an actual cognitive basis: Human intuitions about objective physical distance between countries and cities are biased by cognitive, cultural and socio-economic factors, as explored for example in Friedman et al. (2002), who report that Texans locate Canadian cities closer to the US border relative to Mexican cities, despite their proximity to the latter, and that they place Southern US cities further south than they really are.

Interestingly, DIST2REF does also show some cultural effects in its geolocation errors: For example, some Pacific island states with lesser-known identities (e.g., Nauru and French Polynesia) are placed in the Indian Ocean, where we find the perhaps prototypes of beautiful islands, like Seychelles and Mauritius; also, Central American countries (such as Panama, El Salvador, and Nicaragua) move towards their "cultural center of gravity", South America.

However, this kind of cultural bias is much more prominent in the original WORD2VEC distributional representation. The Spain/Germany effect discussed above is not found in the DIST2REF model at all. And while both DIST2REF and WORD2VEC place Mexican and Spanish cities in our test set closer to each other than they actually are, WORD2VEC does so to a much larger extent. In line with our goal to extract referential attributes, thus, we are satisfied to see that DIST2REF manages to minimize this bias and distill the referential part from the distributional representations.

5 Related Work

There is a large literature on exploiting corpus evidence, sometimes through distributional semantic methods, in order to construct and populate structured knowledge bases (KBs) (e.g., Buitelaar and Cimiano (2008) and references therein). This line of work, however, does not attempt to *connect* entity representations extracted from corpora and from KBs, as we do. Moreover, it focuses on harvesting relations between entities or between entities and a limited number of discrete attributes, rather than predicting full-fledged KB representations of specific entities, like we do. Freitas and Curry (2014) and Freitas et al. (2014) embed relational graphs from KBs in a distributional semantic space to support various forms of search and reasoning about the KB. The focus is again on relations between discrete entities, and on exploiting distributional semantics to navigate among them.

Socher et al. (2013a) represent WordNet and FreeBase entities with corpus-based distributional vectors. They train a tensor for each relation of interest to return high scores when combined with the vectors of two entities that hold the intended relation. At test time, the system is used to classify relational tuples as true or false, as well as to predict new entities that hold a certain relationship with a target entity. This is quite close in spirit to what we do, except that, given an *entity1-relation-entity2* tuple, we treat *relation-entity2* as a binary attribute of *entity1*, and we try to induce such attributes on a larger scale (Socher et al. consider seven relations in total). Moreover, we rely on the same architecture to learn discrete features denoting relations with entities and numerical features, to induce full attribute-based descriptions of entities.

Our proposal is only distantly related to methods to embed words tokens and KB entities and relationships in a vector space, e.g., for better relation extraction (see Weston et al. (2013) and references therein). This line of work does not use distributional semantics to induce word vectors, and ignores numerical attributes.

The broader goal of getting at referential information with distributional semantics is shared with Herbelot (2015). However, the specific approach is different, as she constructs vectors for individual entities (literary characters) by contextualizing generic noun vectors with distributional properties of those entities. Finally, we share our methodology with work on mapping between corpus-based word representations and other representational spaces, such as subject-generated concept properties (Johns and Jones, 2012; Hill et al., 2014; Făgărăşan et al., 2015), visual features (Frome et al., 2013; Socher et al., 2013b; Lazaridou et al., 2014) or brain signals (Mitchell et al., 2008; Murphy et al., 2012). In all these settings, the focus is entirely on predicting numerical attributes, whereas we treat both numerical and binary attributes. Rubinstein et al. (2015) use distributional vectors to predict binary conceptual attributes of common nouns, as well as a continuous score measuring saliency of such attributes. Our target features are conceptually very different from those of all these studies.

6 Discussion and Conclusion

We have shown that a simple model can learn to predict, to a reasonable degree of accuracy, ref-

erential attributes of an entity that are typically seen in a knowledge base from the corresponding corpus-based distributional representation. The results suggest that, while distributional semantic vectors can be used "as-is" to capture generic word similarity, with some supervision it is also possible to extract other kinds of information from them, including structured factual statements of the sort encoded in manually-curated knowledge bases. This makes distributional vectors very attractive as general-purpose word meaning representations.

We have also shown that some of the errors in the predictions can be explained on cultural grounds, but that these effects are more pronounced in the input of our model, a standard distributional semantic model, than in its output. In this sense, our model manages to objectify the information that it is provided with. Our analyses also suggest that the main limiting factor in learning referential attributes, apart from good old data sparseness, is the degree to which they are *contextually supported*, that is, to what extent they are expressed with consistent and specific linguistic means in the context of their target words. This determines whether they are actually represented in the distributional model in the first place.

More generally, we see our work as a small step towards the more general goal of bridging the concept-referent gap in distributional semantics. A common noun such as *dog* denotes a *concept*, based on a prototype with fuzzy boundaries, susceptible of metaphorical extensions, and bearing all the other hallmarks of generic conceptual knowledge (Carlson, 2009; Murphy, 2002). These might be adequately captured by the properties of the *dog* vector in distributional semantic space. However, when used in a specific discourse, words and more complex linguistic expressions often denote specific *referents* with fixed, "hard" properties, such as *this dog*, or *Amur*, when used for my neighbor's dog at 3.31pm on May 29th 2015 in Novosibirsk, a 61cm-tall black-and-tan foxhound. Amur is more easily characterized by a set of precise attribute-value pairs than by a vector in a generic conceptual space. Our experiment suggests that distributional vectors encode both generic conceptual knowledge and more precise attributes of specific referents. Of course, while we can use FreeBase and other knowledge bases to gather training data about public-domain entities, such as countries or cities, it is still not clear where we could gather appropriate training data to learn about the specific properties of "private-discourse" referents such as Amur. Moreover, it remains to be seen whether the properties of common named entities, such as countries and cities, that are in a sense "hybrid" between the conceptual and referential domains, also transfer to entities of a more specific and private kind. Finally, it is still not clear how to extend the current approach beyond words and phrases directly denoting an entity (*Amur*) to other kinds of definite descriptions (*this dog*).

Acknowledgments: This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement No 655577 (LOVe); ERC 2011 Starting Independent Research Grant n. 283554 (COMPOSES); DFG (SFB 732, Project D10); and Spanish MINECO (grant FFI2013-41301-P). This paper reflects the authors' view only, and the EU is not responsible for any use that may be made of the information it contains. Special thanks to Christian Scheible for help with the Machine Learning part, to the anonymous reviewers for insightful and constructive feedback, and to the FLOSS reading group for helping us shape our ideas on the topic of this paper.

References

Marco Baroni and Alessandro Lenci. 2010. Distributional Memory: A general framework for corpus-based semantics. *Computational Linguistics*, 36(4):673–721.

Paul Buitelaar and Philipp Cimiano. 2008. *Bridging the Gap between Text and Knowledge*. IOS, Amsterdam.

Greg Carlson. 2009. Generics and concepts. In Jefrey Pelletier, editor, *Kinds, Things and Stuff*, pages 16–35. Oxford University Press, Oxford, UK.

Bradley Efron and Robert Tibshirani. 1994. *An Introduction to the Bootstrap*. Chapman and Hall, Boca Raton, FL.

Katrin Erk, Sebastian Padó, and Ulrike Padó. 2010. A flexible, corpus-driven model of regular and inverse selectional preferences. *Computational Linguistics*, 36(4):723–763.

Katrin Erk. 2012. Vector space models of word meaning and phrase meaning: A survey. *Language and Linguistics Compass*, 6(10):635–653.

Luana Făgărăşan, Eva Maria Vecchi, and Stephen Clark. 2015. From distributional semantics to feature norms: grounding semantic models in human perceptual data. In *Proceedings of IWCS*, pages 52–57, London, UK.

André Freitas and Edward Curry. 2014. Natural language queries over heterogeneous linked data graphs: A distributional-compositional semantics approach. In *Proceedings of IUI*, pages 279–288, Haifa, Israel.

André Freitas, João Carlos Pereira da Silva, Edward Curry, and Paul Buitelaar. 2014. A distributional semantics approach for selective reasoning on commonsense graph knowledge bases. In *Proceedings of NLDB*, pages 21–32, Montpellier, France.

Alinda Friedman, Dennis Kerkman, and Norman Brown. 2002. Spatial location judgments: A cross-national comparison of estimation bias in subjective North American geography. *Psychonomic Bulletin & Review*, 9(3):615–623.

Andrea Frome, Greg Corrado, Jon Shlens, Samy Bengio, Jeff Dean, Marc'Aurelio Ranzato, and Tomas Mikolov. 2013. DeViSE: A deep visual-semantic embedding model. In *Proceedings of NIPS*, pages 2121–2129, Lake Tahoe, NV.

Aurélie Herbelot. 2015. Mr Darcy and Mr Toad, gentlemen: distributional names and their kinds. In *Proceedings of IWCS*, pages 151–161, London, UK.

Felix Hill, Roi Reichart, and Anna Korhonen. 2014. Multi-modal models for concrete and abstract concept meaning. *Transactions of the Association for Computational Linguistics*, 2:285–296.

Brendan Johns and Michael Jones. 2012. Perceptual inference through global lexical similarity. *Topics in Cognitive Science*, 4(1):103–120.

Willis F. Kern and James R. Bland. 1948. *Solid Mensuration with Proofs*. Wiley, New York, 2nd edition.

Angeliki Lazaridou, Elia Bruni, and Marco Baroni. 2014. Is this a wampimuk? Cross-modal mapping between distributional semantics and the visual world. In *Proceedings of ACL*, pages 1403–1414, Baltimore, MD.

Max Louwerse and Rolf Zwaan. 2009. Language encodes geographical information. *Cognitive Science*, 33:51–73.

Chris Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press, Cambridge, UK.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013. Linguistic regularities in continuous space word representations. In *Proceedings of NAACL*, pages 746–751, Atlanta, Georgia.

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

Tom Mitchell, Svetlana Shinkareva, Andrew Carlson, Kai-Min Chang, Vincente Malave, Robert Mason, and Marcel Just. 2008. Predicting human brain activity associated with the meanings of nouns. *Science*, 320:1191–1195.

Brian Murphy, Partha Talukdar, and Tom Mitchell. 2012. Selecting corpus-semantic models for neurolinguistic decoding. In *Proceedings of *SEM*, pages 114–123, Montreal, Canada.

Gregory Murphy. 2002. *The Big Book of Concepts*. MIT Press, Cambridge, MA.

Sebastian Padó and Mirella Lapata. 2007. Dependency-based construction of semantic space models. *Computational Linguistics*, 33(2):161–199.

Mark Palatucci, Dean Pomerleau, Geoffrey Hinton, and Tom Mitchell. 2009. Zero-shot learning with semantic output codes. In *Proceedings of NIPS*, pages 1410–1418, Vancouver, Canada.

Dana Rubinstein, Effi Levi, Roy Schwartz, and Ari Rappoport. 2015. How well do distributional models capture different types of semantic knowledge? In *Proceedings of ACL (Volume 2: Short Papers)*, pages 726–730, Beijing, China.

Richard Socher, Danqi Chen, Christopher Manning, and Andrew Ng. 2013a. Reasoning with neural tensor networks for knowledge base completion. In *Proceedings of NIPS*, pages 926–934, Lake Tahoe, NV.

Richard Socher, Milind Ganjoo, Christopher Manning, and Andrew Ng. 2013b. Zero-shot learning through cross-modal transfer. In *Proceedings of NIPS*, pages 935–943, Lake Tahoe, NV.

Peter Turney and Patrick Pantel. 2010. From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research*, 37:141–188.

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

Building a shared world: Mapping distributional to model-theoretic semantic spaces

Aurélie Herbelot
Universität Stuttgart
Institut für Maschinelle Sprachverarbeitung
Stuttgart, Germany
aurelie.herbelot@cantab.net

Eva Maria Vecchi
University of Cambridge
Computer Laboratory
Cambridge, UK
eva.vecchi@cl.cam.ac.uk

Abstract

In this paper, we introduce an approach to automatically map a standard distributional semantic space onto a set-theoretic model. We predict that there is a functional relationship between distributional information and vectorial concept representations in which dimensions are predicates and weights are generalised quantifiers. In order to test our prediction, we learn a model of such relationship over a publicly available dataset of feature norms annotated with natural language quantifiers. Our initial experimental results show that, at least for domain-specific data, we can indeed map between formalisms, and generate high-quality vector representations which encapsulate set overlap information. We further investigate the generation of natural language quantifiers from such vectors.

1 Introduction

In recent years, the complementarity of distributional and formal semantics has become increasingly evident. While distributional semantics (Turney and Pantel, 2010; Clark, 2012; Erk, 2012) has proved very successful in modelling lexical effects such as graded similarity and polysemy, it clearly has difficulties accounting for logical phenomena which are well covered by model-theoretic semantics (Grefenstette, 2013).

A number of proposals have emerged from these considerations, suggesting that an overarching semantics integrating both distributional and formal aspects would be desirable (Coecke et al., 2011; Bernardi et al., 2013; Grefenstette, 2013; Baroni et al., 2014a; Garrette et al., 2013; Beltagy et al., 2013; Lewis and Steedman, 2013). We will use the term 'Formal Distributional Semantics' (FDS) to refer to such proposals. This paper follows this line of work, focusing on one central question: the formalisation of the systematic dependencies between lexical and set-theoretic levels.

Let us consider the following examples.

1. Kim writes books.
2. Kim likes books.

The preferred reading of 1 has a logical form where the object is treated as an existential, while the object in 2 has a generic reading:

- $\exists x^*[book'(x^*) \wedge write'(Kim, x^*)]$
- $GEN\, x[book'(x) \rightarrow like'(Kim, x)]$

with x^* indicating a plurality and GEN the generic quantifier.

It is generally accepted that the appropriate choice of quantifier for an ambiguous bare plural object depends, amongst other things, on the lexical semantics of the verb (e.g. Glasbey (2006)). This type of interaction implies the existence of systematic influences of the lexicon over logic, which could in principle be formalised.

A model of the lexicon/logic interface would be desirable to explain how speakers resolve standard cases of ambiguity like the bare plural in 1 and 2, but more generally, it could be the basis for answering a more fundamental question: how do speakers construct a model of a sentence for which they have no prior perceptual data?

People can make complex inferences about statements without having access to their real-world reference. As an example, consider the sentence *The kouprey is a mammal*. English speakers have no problem ascertaining that if x is a kouprey, x is a mammal (which set-theoretic semantics would express as $\forall x[kouprey'(x) \rightarrow mammal'(x)]$), regardless of whether they have ever encountered a kouprey. The inference is supported by the lexical semantics of *mammal*, which applies a property (being a mammal) to *all* instances of a class. Much more complex inferences are routinely performed by speakers, down to estimating the cardinality of the entities involved in a particular situation. Compare e.g. *The cats are on the sofa* (2 / a few cats?), *I picked pears today* (a few / a few dozen?) and *The protesters were blocking the entire avenue* (hundreds/thousands of protesters?).

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 22–32,
Lisbon, Portugal, 17-21 September 2015.

Understanding how this process works would not only give us an insight into a complex cognitive process, but also make a crucial contribution to NLP tasks relying on inference (e.g. the Recognising Textual Entailment challenge, RTE: Dagan et al. (2009)). Indeed, while systems have successfully been developed to model entailment between quantifiers, ranging from natural logic approaches (MacCartney and Manning, 2008) to distributional semantics solutions (Baroni et al., 2012), they rely on an explicit representation of quantification. That is, they can model the entailment *All koupreys are mammals* $\models$ *This kouprey is a mammal*, but not *Koupreys are mammals* $\models$ *This kouprey is a mammal*.

In this work, we assume the existence of a mapping between language (distributional models) and world (set-theoretic models), or to be more precise, between language and a shared set of beliefs about the world, as negotiated by a group of speakers. To operationalise this mapping, we propose that set-theoretic models, like distributions, can be expressed in terms of vectors – giving us a common representation across formalisms. Using a publicly available dataset of feature norms annotated with quantifiers[1] (Herbelot and Vecchi, 2015), we show that human-like intuitions about the quantification of simple subject/predicate pairs can be induced from standard distributional data.

This paper is structured as follows. §2 reviews related work, focusing in turn on approaches to formal distributional semantics, computational work on quantification, and mapping between semantic spaces. In §3, we describe our dataset. §4 and §5 describe our experiments, reporting correlation against human annotations. We discuss our results in §6 and end with an attempt at generating natural language quantifiers from our mapped vectors (§7).

2 Related Work

2.1 Formal Distributional Semantics

The relation between distributional and formal semantics has been the object of a number of studies in recent years. Proposals for a FDS, i.e. a combination of both formalisms, roughly fall into two groups: a) the fully distributional approaches, which redefine the concepts of formal semantics in distributional terms (Coecke et al., 2011; Bernardi et al., 2013; Grefenstette, 2013; Hermann et al., 2013; Baroni et al., 2014a; Clarke, 2012); b) the hybrid approaches, which try to keep the set-theoretic apparatus for function words and integrate distributions as content words representations (Erk, 2013; Garrette et al., 2013; Beltagy et al., 2013; Lewis and Steedman, 2013). This paper follows the hybrid frameworks in that we fully preserve the principles of set theory and do not attempt to give a distributional interpretation to phenomena traditionally catered for by formal semantics such as quantification or negation.

Our account is also similar to that proposed by Erk (2015). Erk suggests that distributional data influences semantic 'knowledge'[2]: specifically, while a speaker may not know the extension of the word *alligator*, they maintain an information state which models properties of alligators (for instance, that they are animals). This information state is described in terms of probabilistic logic, which accounts for an agent's uncertainty about what the world is like. The probability of a sentence is the summed probability of the possible worlds that make it true. Similarly, we assume a systematic relation between distributional information and world knowledge, expressed set-theoretically. The knowledge representation we derive is not a model proper: it cannot be said to be a description of a world – either the real one or a speaker's set of beliefs (c.f. §4 for more details). But it is a good approximation of the shared intuitions people have about the world, in the way that distributional representations are an averaged representation of how a group of speakers use their language.

2.2 Generalised quantifiers

Computational semantics has traditionally focused on very specific aspects of quantification. There is a large literature on the computational formalisation of quantifiers as automata, starting with Van Benthem (1986). In parallel to this work, much research has been done on drawing inferences from explicitly quantified statements – i.e. statements quantified with determiners such as *some/most/all*, which give information about the set overlap of a subject-predicate pair (Cooper et al., 1996; Alshawi and Crouch, 1992; MacCartney and Manning, 2008). Recent work in this area has even shown that entailment between explicit quantifiers can be modelled distributionally (Baroni et al., 2012). A complementary object of focus, actively pursued in the 1990s, has been inference between generic statements (Bacchus, 1989; Vogel, 1995).

Beside those efforts, computational approaches have been developed to convert arbitrary text into logical forms. The techniques range from completely supervised (Baldwin et al., 2004; Bos, 2008) to lightly supervised (Zettlemoyer and Collins, 2005). Such work has shown that it was possible to automatically give complex formal semantics analyses to large amounts of data. But the formalisation of quantifiers in those systems either remains very much underspecified (e.g. bare plurals are not resolved into either existentials or generics) or relies on some grounded information, for example in the form of a database.

To the best of our knowledge, no existing system is able to universally predict the generalised quantification of noun phrases, including those introduced by the (in)definite singulars *a/the* and definite plurals *the*. The closest attempt is Herbelot (2013), who suggests that

[1] Data available at `http://www.cl.cam.ac.uk/~ah433/mcrae-quantified-majority.txt`

[2] We use the term *knowledge* loosely, to refer to a speaker's beliefs about the world or a state of affairs.

Concept	*Feature*	
ape	is_muscular	ALL
	is_wooly	MOST
	lives_on_coasts	SOME
	is_blind	FEW
tricycle	has_3_wheels	ALL
	used_by_children	MOST
	is_small	SOME
	used_for_transportation	FEW
	a_bike	NO

Table 1: Example annotations for concepts.

'model-theoretic vectors' can be built out of distributional vectors supplemented with manually annotated training data. The proposed implementation, however, fails to validate the theory.

Our work follows the intuition that distributions can be translated into set-theoretic equivalents. But it implements the mapping as a systematic linear transformation. Our approach is similar to Gupta et al. (2015), who predict numerical attributes for unseen concepts (countries and cities) from distributional vectors, getting comparably accurate estimates for features such as the GDP or CO_2 emissions of a country. We complement such research by providing a more formal interpretation of the mapping between language and world knowledge. In particular, we offer a) a vectorial representation of set-theoretic models; b) a mechanism for predicting the application of generalised quantifiers to the sets in a model.

2.3 Mapping between Semantic Spaces

The mapping between different semantic modalities or semantic spaces has been explored in various aspects. In cognitive science, research by Riordan and Jones (2011) and Andrews et al. (2009) show that models that map between and integrate perceptual and linguistic information perform better at fitting human semantic intuition. In NLP, Mikolov et al. (2013b) show that a linear mapping between vector spaces of different languages can be learned to infer missing dictionary entries by relying on a small amount of bilingual information. Frome et al. (2013) learn a linear regression to transform vector-based image representations onto vectors representing the same concepts in a linguistic semantic space, and Lazaridou et al. (2014) explore mapping techniques to learn a cross-modal mapping between text and images with promising performance. We follow the basic intuition introduced by these previous studies: a simple linear function can map between semantic spaces, in this case between a linguistic (distributional) semantic space and a model-theoretic space.

3 Annotated datasets

3.1 The quantified McRae norms

The McRae norms (McRae et al., 2005) are a set of feature norms elicited from 725 human participants for 541 concepts covering living and non-living entities (e.g. alligator, chair, accordion). The annotators were given concepts and asked to provide features for them, covering physical, functional and other properties. The result is a set of 7257 concept-feature pairs such as *airplane used-for-passengers* or *bear is-brown*.

In our work, we use the annotation layer produced by Herbelot and Vecchi (2015) for the McRae norms (henceforth QMR): for each concept-feature pair (C, f), the annotation provides a natural language quantifier expressing the ratio of instances of C having the feature f, as elicited by three coders. The quantifiers in use are NO, FEW, SOME, MOST, ALL. Table 1 provides example annotations for concept-feature pairs (reproduced from the original paper). An additional label, KIND, was introduced for usages of the concept as a kind, where quantification does not apply (e.g. *beaver symbol-of-Canada*). A subset of the annotation layer is available for training computational models, corresponding to all instances with a majority label (i.e. those where two or three coders agreed on a label). The reported average weighted Cohen kappa on this data is $\kappa = 0.59$.

In the following, we use a derived gold standard including all 5 quantified classes in QMR (removing the KIND items), with the annotation set to majority opinion (6156 instances). The natural language quantifiers are converted to a numerical format (see §4 for details). Using the numerical data, we can calculate the mean Spearman rank correlation between the three annotators, which comes to 0.63.

3.2 Additional animal data

QMR gives us an average of 11 features per concept. This results in fairly sparse vectors in the model-theoretic semantic space (see §4). In order to remedy data sparsity, we consider the use of additional data in the form of the animal dataset from Herbelot (2013) (henceforth AD). AD[3] is a set of 72 animal concepts with quantification annotations along 54 features. The main differences between QMR and AD are as follows:

- Nature of features: the features in AD are not human elicited norms, but linguistic predicates obtained from a corpus analysis.
- Comprehensiveness of annotation: the 72 concepts were annotated along all 54 features. This ensures the availability of a large number of negatively quantified pairs (e.g. *cat is-fish*).

We manually align the AD concepts and features to the QMR format, changing e.g. *bat* to *bat_(animal)*. The QMR and AD sets have an overlap of 39 concepts and 33 features.

[3]Data available at `http://www.cl.cam.ac.uk/~ah433/material/herbelot_iwcs13_data.txt`.

4 Semantic spaces

We construct two distinct semantic spaces (distributional and model-theoretic), as described below.

4.1 The distributional semantic space

We consider two distributional semantic space architectures which have each shown to have considerable success in a number of semantic tasks. First, we build a co-occurrence based space ($\mathbf{DS}_{cooc}$), in which a word is represented by co-occurrence counts with content words (nouns, verbs, adjectives and adverbs). As a source corpus, we use a concatenation of the ukWaC, a 2009 dump of the English Wikipedia and the BNC[4], which consists of about 2.8 billion tokens. We select the top 10K content words for the contexts, using a bag-of-words approach and counting co-occurrences within a sentence. We then apply positive Pointwise Mutual Information to the raw counts, and reduce the dimensions to 300 through Singular Value Decomposition.[5]

Next we consider the context-predicting vectors ($\mathbf{DS}_{Mikolov}$) available as part of the word2vec[6] project (Mikolov et al., 2013a). We use the publicly available vectors which were trained on a Google News dataset of circa 100 billion tokens. Baroni et al. (2014b) showed that vectors constructed under this architecture outperform the classic count-based approaches across many semantic tasks, and we therefore explore this option as a valid distributional representation of a word's semantics.

4.2 The model-theoretic space

Our 'model-theoretic space' differs in a couple of important respects from traditional formal semantics models. So it may be helpful to first come back to the standard definition of a model, which relies on two components: an ontology and a denotation function (Cann, 1993). The ontology describes a world (which can be a simple situation or 'state of affairs'), with everything that is contained in that world. Ontologies can be represented in various ways, but in this paper, we assume they are formalised in terms of *sets* of entities. The denotation function associates words with their *extensions* in the model, i.e. the sets they refer to. Thanks to the availability of the ontology, it is possible to define a truth function for sentences, which computes whether a particular statement corresponds to the model or not.

In our account, we do not have an *a priori* model of the world: we wish to infer it from our observation of language data. We believe this to be an advantage over traditional formal semantics, which requires full ontological data to be available in order to account for reference and truth conditions, but never spells out how this data comes into being. This however implies that our produced ontology will necessarily be partial: we can only model what can be inferred from language use. This has consequences for the denotation function.

Let's imagine a world with three cats and two horses. In model theory, the word *horse* has an extension in that world which is the set of horses, with a cardinality of two. This can be trivially derived because the world is fully described in the ontology. In our approach, however, it is unlikely we might be able to learn the cardinality of *any* set in *any* world. And in fact, it is clear that 'in real life', speakers do miss this information for many sets (how many horses are there in the world?) Note that we do not in principle reject the possibility to learn cardinalities from distributional data (for an example of this, see Gupta et al. (2015)). We simply remark that this will not always possible, or even desirable from a cognitive point of view. By extension, this means that a model built from distributional data does not support denotation in the standard way, and thus precludes the definition of a truth function: we cannot verify the truth of the sentence *There are 25,957 white horses in the world*. Our 'model-theoretic' space may then be described as an underspecified set-theoretic representation of some shared beliefs about the world.

Our 'ontology' can be defined as follows. To each word w_k in vocabulary $V = w_{1...m}$ corresponds a set w'_k with underspecified cardinality. A number of predicates $p'_{1...n}$ are similarly defined as sets with an unknown number of elements. Our claim is that this very underspecified model can be further specified by learning a function F from distributions to generalised quantifiers. Specifically, $F(\vec{w_k}) = \{Q_1(w'_k, p'_1), Q_2(w'_k, p'_2)...Q_n(w'_k, p'_n)\}$, where $\vec{w_k}$ is the distribution of w_k and $Q_1...Q_n \in \{no, few, some, most, all\}$. That is, F takes a distribution $\vec{w_k}$ and returns a quantifier for each predicate in the model, corresponding to the set overlap between w'_k and $p'_{1...n}$. Note that we focus here on 5 quantifiers only, but as mentioned above, we do not preclude the possibility of learning others (including cardinals in appropriate cases).

$F(\vec{w_k})$ lives in a model-theoretic space which broadly follows the representation suggested by Herbelot (2013). We assume a space with n dimensions $d_1...d_n$ which correspond to predicates $p'_{1...n}$ (e.g. *is fluffy*, *used for transportation*). In that space, $F(\vec{w_k})$ is weighted along the dimension d_m in proportion to the set overlap $w'_k \cap p'_m$.[7] The following shows a toy vector with only four dimensions for the concept *horse*.

a_mammal	1
has_four_legs	0.95
is_brown	0.35
is_scaly	0

[4] `http://wacky.sslmit.unibo.it`, `http://www.natcorp.ox.ac.uk`

[5] All semantic spaces, both distributional and model-theoretic, were built using the DISSECT toolkit (Dinu et al., 2013).

[6] `https://code.google.com/p/word2vec`

[7] In Herbelot (2013), weights are taken to be probabilities, but we prefer to talk of quantifiers, as the notion models our data more directly.

This vector tells us that the set of horses includes the set of mammals (the number of horses that are also mammals divided by the number of horses comes to 1, i.e. *all horses are mammals*), and that the set of horses and the set of things that are scaly are disjoint (*no horse is scaly*). We also learn that a great majority of horses have four legs and that some are brown.

In the following, we experiment with 3 model-theoretic spaces built from the McRae and AD datasets described in §3. As both datasets are annotated with natural language quantifiers rather than cardinality ratios, we convert the annotation into a numerical format, where ALL → 1, MOST → 0.95, SOME → 0.35, FEW → 0.05, and NO → 0. These values correspond to the weights giving the best inter-annotator agreement in Herbelot and Vecchi (2015), when calculating weighted Cohen's kappa on QMR.

In each model-theoretic space, a concept is represented as a vector in which the dimensions are features (*has_buttons, is_green*), and the values of the vectors along each dimension are quantifiers (in numerical format). When a feature does not occur with a concept in QMR, the concept's vector receives a weight of 0 on the corresponding dimension.[8] We define 3 spaces as follows. The McRae-based model-theoretic space (**MT**$_{QMR}$) contains 541 concepts, as described in §3.1. The second space is constructed specifically for the additional animal data from §3.2 (**MT**$_{AD}$). Finally, we merge the two into a single space of 555 unique concepts (**MT**$_{QMR+AD}$).

5 Experiments

5.1 Experimental setup

To map from one semantic representation to another, we learn a function f: **DS** → **MT** that transforms a distributional semantic vector for a concept to its model-theoretic equivalent.

Following previous research showing that similarities amongst word representations can be maintained within linear transformations (Mikolov et al., 2013b; Frome et al., 2013), we learn the mapping as a linear relationship between the distributional representation of a word and its model-theoretic representation. We estimate the coefficients of the function using (multivariate) partial least squares regression (PLSR) as implemented in the R pls package (Mevik and Wehrens, 2007).

We learn a function from the distributional space to each of the model-theoretic spaces (c.f. §4). The distribution of training and test items is outlined in Table 2, expressed as a number of concept vectors. We also include the number of quantified instances in the test set (i.e. the number of actual concept-feature pairs that were explicitly annotated in QMR/AD and that we can thus evaluate – this is a portion of each concept vector in the spaces including QMR data).

Space	*# train vec.*	*# test vec.*	*# dims*	*# test inst.*
MT$_{QMR}$	400	141	2172	1570
MT$_{AD}$	60	12	54	648
MT$_{QMR+AD}$	410	145	2193	1595

Table 2: Distribution of training/test items for each model-theoretic semantic space. We also provide the number of dimensions for each space, and the actual number of concept-feature instances tested on.

5.2 Results

We first consider a preliminary quantitative analysis to better understand the behavior of the transformations, while a more qualitative analysis is provided in §6. The results in Table 3 show the degree to which predicted values for each dimension in a model-theoretic space correlate with the gold annotations, operationalised as the Spearman ρ (rank-order correlation). Wherever appropriate, we also report the mean Spearman correlation between the three human annotators for the particular test set under consideration, showing how much they agreed on their judgements.[9] These figures provide an upper bound performance for the system, i.e. we will consider having reached human performance if the correlation between system and gold standard is in the same range as the agreement between humans. For each mapping tested, Table 3 provides details about the training data used to learn the mapping function and the test data for the respective results. Also for each mapping, results are reported when learned from either the co-occurrence distributional space (DS$_{cooc}$) or the context-predicting distributional space (DS$_{Mikolov}$).

The top section of the table reports results for the QMR and AD dataset taken separately, as well as their concatenation. Performance on the domain-specific AD is very promising, at 0.641 correlation, calculated over 648 test instances. The results when trained on just the QMR features (MT$_{QMR}$) are much lower (0.35 over 1570 test instances), which we put down to the wider variety of concepts in that dataset; we however observe a substantial increase in performance when we train and test over the two datasets (MT$_{QMR+AD}$: 0.569 over 1595 instances).

We investigate whether merging the datasets generally benefits QMR concepts or just the animals (see middle section in Table 3). The result on the MT$_{animals}$ test set, which includes animals from the AD and QMR datasets, shows that this category fares indeed very well, at $\rho = 0.663$. But while augmenting the training data with category-specific datapoints benefits that category, it does not improve the results

[8] No transformations or dimensionality reductions were performed on the MT spaces.

[9] These figures are only available for the QMR dataset, as AD only contains one annotation per subject-predicate pair.

Model-Theoretic		*Distributional*		
train	*test*	DS_{cooc}	$DS_{Mikolov}$	*human*
MT_{QMR}	MT_{QMR}	0.350	0.346	0.624
MT_{AD}	MT_{AD}	**0.641**	0.634	–
MT_{QMR+AD}	MT_{QMR+AD}	0.569	0.523	–
MT_{QMR+AD}	$MT_{animals}$	**0.663**	0.612	–
MT_{QMR+AD}	$MT_{no\text{-}animals}$	0.353	0.341	–
MT_{QMR}	$MT_{QMR^{animals}}$	0.419	0.405	–
MT_{QMR+AD}	$MT_{QMR^{animals}}$	**0.666**	0.600	0.663

Table 3: (Spearman) correlations of mapped dimensions with gold annotations for all test items. The table reports results (ρ) when mapped from a distributional space (DS_{cooc} or $DS_{Mikolov}$) to each MT space, as well as the correlation with human annotations when available. The train/test data for the mappings is specified in Table 2. For further analysis we report the results when tested *only* on animal test items ($animals$), or on all test items *but* animals ($no\text{-}animals$). $MT_{animals}$ contains test items from both AD and the animal section of the McRae norms. See text for more details.

for concepts of other classes (c.f. compare $MT_{animals}$ with $MT_{no\text{-}animals}$).

Finally, we quantify the specific improvement to the QMR animal concepts by comparing the correlation obtained on $MT_{QMR^{animals}}$ (a test set consisting only of QMR animal features) after training on a) the QMR data alone and b) the merged dataset (third section of Table 3). Performance increases from 0.419 to 0.666 on that specific set. This is in line with the inter-annotator agreement (0.663).

To summarise, we find that the best correlations with the gold annotations are seen when we include the animal-only dataset in training (MT_{AD} and MT_{QMR+AD}) and test on just animal concepts (MT_{AD}, $MT_{animals}$ and $MT_{QMR^{animals}}$). As one might expect, category-specific training data yields high performance when tested on the same category. Although this expectation seems intuitive, it is worth noting that our system produces promisingly high correlations, reaching human-performance on a subset of our data. The assumption we can draw from these results is that, given a reasonable amount of training data for a category, we can proficiently generate model-theoretic representations for concept-feature pairs from a distributional space. The empirical question remains whether this can be generalized for all categories in the QMR dataset.

It is important to keep in mind that the MT spaces are not full matrices, meaning that we have 'missing values' for various dimensions when a concept is converted into a vector. For example, the feature has_a_tail is not among the annotated features for *bear* in QMR and has a weight of 0, even though *most* bears have a tail. This is a consequence of the original McRae dataset, rather than the design of our approach. But it follows that in this quantitative analysis, we are not able to confirm the accuracy of the predicted values on dimensions for which we do not have gold annotations. This may also affect the performance of the system by including 'false' 0 weights in the training data. Although this does not affect our reported correlation results – we test the correlations on those values for which we have gold annotations only – it does open the door to a natural next step in the evaluation. In order to judge the performance of the system on the missing gold dimensions, we need a manual analysis to assess the quality of the whole vectors, which goes hand-in-hand with obtaining additional annotations for the missing dimensions. It seems, therefore, that an active learning strategy would allow us to not only evaluate the model-theoretic vectors more fully, but also improve the system by capturing new data.[10]

	% of gold in...
top 5 neighbours	19% (27/145)
top 10 neighbours	29% (42/145)
top 20 neighbours	46% (67/145)

Table 4: Percentage of gold vectors found in the top neighbours of the mapped concepts, shown for the $DS_{cooc} \rightarrow MT_{QMR+AD}$ transformation.

In this analysis, we focused primarily on the comparison between transformations using various truth-theoretic datasets for training and generation. We leave it to further work to extensively compare the effect of varying the type of the distributional space. Our results show, however, that the $Mikolov$ model performs slightly worse than the co-occurrence space ($cooc$), disproving the idea that predictive models always outperform count-based models.

6 Discussion

To further assess the quality of the produced space, we perform a nearest-neighbour analysis of our results to evaluate the coherence of the estimated vectors: for

[10] As suggested by a reviewer, one could also treat the missing entries as latent dimensions and define the loss function on only the known entries. We leave it to future work to test this promising option to resolve the issue of data sparsity.

axe	*hatchet*
a tool	a tool
is sharp	is sharp
has a handle	has a handle
used for cutting	used for cutting
has a metal blade	made of metal
a weapon	an axe
has a head	is small
used for chopping	–
has a blade	–
is dangerous	–
is heavy	–
used by lumberjacks	–
used for killing	–

Table 5: McRae feature norms for *axe* and *hatchet*

each concept in our test set, we return its nearest neighbours from the gold dataset, as given by the cosine similarity measure, hoping to find that the estimated vector is close to its ideal representation (see Făgărăşan et al. (2015) for a similar evaluation on McRae norms). Results are shown in Table 4. We find that the gold vector is among the top 5 nearest neighbours to the predicted equivalent in nearly 20% of concepts, with the percentage of gold items in the top neighbours improving as we increase the size of the neighbourhood. We perform a more in-depth analysis of the neighbourhoods for each concept to gain a better understanding of their behaviour and quality.

We discover that, in many cases, the mapped vector is close to a similar concept in the gold standard, but not to itself. So for instance, $\overrightarrow{alligator}_{mapped}$ is very close to $\overrightarrow{crocodile}_{gold}$, but not to $\overrightarrow{alligator}_{gold}$. Similar findings are made for *church/cathedral*, *axe/hatchet*, *dishwasher/fridge*, etc. A further investigation show that in the gold standard itself, those pairs are not as close to each other as they should be. Here are some relevant cosine similarities:

$alligator - crocodile$	0.47
$church - cathedral$	0.45
$axe - hatchet$	0.50
$dishwasher - fridge$	0.21

Two reasons can be identified for these comparatively low[11] similarities. First, the McRae norms do not make for a consistent semantic space because a feature that – from an extensional point of view – seems relevant to two concepts may only have been produced by the annotators for one of them. As an example of this, see Table 5, which shows the feature norms for *axe* and *hatchet* after processing (§3). Although the concepts share 4 features, they also differ quite strongly, an *axe* being seen as a weapon with a blade, while the *hatchet* is itself referred to as an axe. Extensionally, of course, there is no reason to think that a hatchet does not have a blade or might not be dangerous, but those features do not appear in the norms for the concept. This results in the two vectors being clearly separated in the set-theoretic space. This means that the distribution of *axe* may well be mapped to a region close to *hatchet*, but thereby ends up separated from the gold *axe* vector.

The second, related issue is that the animal concepts in the McRae norms are annotated along fewer dimensions than in AD. For example, *alligator* – which only appears in the McRae set – has 13 features, while *crocodile* (in both sets) has 70. Given that features which are not mentioned for a concept receive a weight of 0, this also results in very different vectors.

In Table 6, we provide the top weighted features for a small set of concepts. As expected, the animal representations (*bear, housefly*) have higher quality than the other two (*plum, cottage*). But overall, the ranking of dimensions is sensible. We see also that these representations have 'learnt' features for which we do not have values in our gold data – thereby correcting some of the 0 values in the training vectors.

7 Generating natural language quantifiers

In a last experiment, we attempt to map the set-theoretic vectors obtained in §5 back to natural language quantifiers. This last step completes our pipeline, giving us a system that produces quantified statements of the type *All dogs are mammals* or *Some bears are brown* from distributional data.

For each mapped vector $F(\vec{w_k}) = \vec{v_k}$ and a set of dimensions $d_{1...n}$ corresponding to properties $p'_{1...n}$, the value of $\vec{v_k}$ along each dimension is indicative of the proportion of instances of w'_k having the property signalled by the dimension. The smaller the value, the smaller the overlap between the set of instances of w'_k and the set of things having the property. Deriving natural language quantifiers from these values involves setting four thresholds $t_{all}, t_{most}, t_{some}$ and t_{few} so that for instance, if the value of $\vec{v_k}$ along d_m is more than t_{all}, it is the case that *all* instances of $\vec{w_k}$ have property p_m, and similarly for the other quantifiers (*no* has a special status as it is not entailed by any of the other quantifiers under consideration). We set the t-thresholds by a systematic search on a training set (see below).

To evaluate this step, we propose a function that calculates precision while taking into account the two following factors: a) some errors are worse than others: the system shouldn't be overly penalised for classifying a property as MOST rather than ALL, but much more for classifying a gold standard ALL as SOME; b) errors that are conducive to false inferences should be strongly penalised, e.g. generating *all dogs are black* is more serious than *some dogs are mammals*, because the former might lead to incorrect inferences with respect to individual dogs while the latter is true, even though it is pragmatically odd.

[11] Compare with e.g. *ape* - *monkey*, $Sim = 0.97$.

bear	*housefly*	*plum*	*cottage*
an_animal	an_insect	a_fruit	has_a_roof
a_mammal	is_small	grows_on_trees	used_for_shelter*
has_eyes	flies	tastes_sweet	has_doors*
is_muscular	is_slender*	is_edible	a_house
has_a_head	crawls*	is_round	has_windows
has_4_legs	stings*	is_small	is_small
has_a_heart	has_legs	has_skin	a_building*
is_terrestrial	is_large*	is_juicy	used_for_living_in
has_hair	a_bug*	tastes_good	made_of_wood*
is_brown	has_wings	has_seeds*	made_by_humans*
walks	is_black	is_green*	worn_on_feet*
is_wooly	is_terrestrial*	has_peel*	has_rooms*
has_a_tail*	hibernates*	is_orange*	used_for_storing_farm_equipment*
a_carnivore	has_a_heart*	is_citrus*	found_on_farms*
is_large	has_eyes	is_yellow*	found_in_the_country
a_predator	has_antennae*	has_vitamin_C*	an_appliance*
is_furry*	bites*	has_leaves*	has_tenants*
roosts	jumps*	has_a_pit	has_a_bathroom*
is_stout	has_a_head*	has_a_stem*	requires_rent*
hunted_by_people	is_grey*	grows_in_warm_climates*	requires_a_landlord*

Table 6: Example of 20 most weighted contexts in the predicted model-theoretic vectors for 4 test concepts, shown for the $\text{DS}_{cooc} \rightarrow \text{MT}_{McRae+AD}$ transformation. Features marked with an asterisk ($*$) are not among the concept's features in the gold data.

		Gold				
		no	few	some	most	all
Mapped	no	0	-0.05	-0.35	-0.95	-1
	few	-0.05	0	0.2	0.9	0.95
	some	-0.35	-0.2	0	0.6	0.65
	most	-0.95	-0.9	-0.6	0	0.05
	all	-1	-0.95	-0.65	-0.05	0

Table 7: Distance matrix for the evaluation of the natural language quantifiers generation step.

		Gold				
		no	few	some	most	all
Mapped	no	238	66	20	4	2
	few	53	45	30	19	12
	some	6	1	2	3	2
	most	4	6	4	16	56
	all	0	0	0	2	3

Table 8: Confusion matrix for the results of the natural language quantifiers generation.

We set a distance matrix, which we will use for penalising errors. This matrix, shown in Table 7, is basically equivalent to the matrix used by Herbelot and Vecchi (2015) to calculate weighted kappa between annotators, with the difference that all errors involving NO cause incorrect inferences and receive special treatment. Cases where the gold quantifier entails the mapped quantifier (*all cats* $\models$ *some cats*) have positive distances, while cases where the entailment doesn't hold have negative distances. Using the distance matrix, we give a score to each instance in our test data as follows:

$$s = \begin{cases} 1 - d & \text{if } d \geq 0 \\ d & \text{if } d < 0 \end{cases} \quad (1)$$

where d is obtained from the distance matrix.

This has the effect that when the mapped quantifier equals the gold quantifier, the system scores 1; when the mapped value deviates from the gold standard but produces a true sentence (*some dogs are mammals*), the system gets a partial score proportional to the distance between its output and the gold data; when the mapping results in a false sentence (*all dogs are black*), the system is penalised with minus points.

In what follows, we report the average performance of the system as $P = \frac{\sum s_m}{N}$ where s_m is the score assigned to a particular test instance, and N is the number of test instances. We evaluate on the 648 test instances of MT_{AD}, as this is the only dataset containing a fair number of negatively quantified concept-predicate pairs. We perform 5-fold cross-evaluation on this data, using 4 folds to set the t thresholds, and testing on one fold. We obtain an average P of 0.61. Inference is preserved in 73% of cases (also averaged over the 5 folds).

Table 8 shows the confusion matrix for our results. We note that the system classifies NO-quantified instances with good accuracy (72% – most confusions being with FEW). Because of the penalty given to instances that violate proper entailment, the system is conservative and prefers FEW to SOME, as well as MOST to ALL. Table 9 shows randomly selected instances, together with their mapped quantifier and the label from the gold standard.

Instance	Mapped	Gold
raven a_bird	most	all
pigeon has_hair	few	no
elephant has_eyes	most	all
crab is_blind	few	few
snail a_predator	no	no
octopus is_stout	no	few
turtle roosts	no	few
moose is_yellow	no	no
cobra hunted_by_people	some	some
snail forages	few	no
chicken is_nocturnal	few	no
moose has_a_heart	most	all
pigeon hunted_by_people	no	few
cobra bites	few	most

Table 9: Examples of mapped concept-predicate pairs

8 Conclusion

In this paper, we introduced an approach to map from distributional to model-theoretic semantic vectors. Using traditional distributional representations for a concept, we showed that we are able to generate vectorial representations that encapsulate generalised quantifiers.

We found that with a relatively "cheap" linear function – cheap in that it is easy to learn and requires modest training data – we can reproduce the quantifiers in our gold annotation with high correlation, reaching human performance on a domain-specific test set. In future work, we will however explore the effect of more powerful functions to learn the transformations from distributional to model-theoretic spaces.

Our qualitative analysis showed that our predicted model-theoretic vectors sensibly model the concepts under consideration, even for features which do not have gold annotations. This is not only a promising result for our approach, but it provides potential as a next step to this work: expanding our training data with non-zero dimensions in an active learning procedure. We also experimented with generating natural language quantifiers from the mapped vectorial representations, producing 'true' quantified sentences with a 73% accuracy.

We note that our approach gives a systematic way to disambiguate non-explicitly quantified sentences such as generics, opening up new possibilities for improved semantic parsing and recognising entailment. Right now, many parsers give the same broad analysis to *Mosquitoes are insects* and *Mosquitoes carry malaria*, involving an underspecified/generic quantifier. This prevents inferring, for instance, that Mandie the mosquito is definitely an insect but may or may not carry malaria. In contrast, our system would attribute the most plausible quantifiers to those sentences (*all/few*), allowing us to produce correct inferences.

The focus of this paper was concept-predicate pairs out of context. That is, we considered quantified sentences where the restrictor was the entire set denoted by a lexical item. A natural next step is to investigate the quantification of statements involving contextualised subsets. For instance, we should obtain a different quantifier for *taxis are yellow* depending on whether the sentence starts with *In London...* or *In New York...* In future work, we will test our system on such context-specific examples, using contextualised vector representations such as the ones proposed by e.g. Erk and Padó (2008) and Dinu and Lapata (2010).

We conclude by noting again that the set-theoretic models produced in this work differ from formal semantics models in important ways. They do not represent the world *per se*, but rather some shared beliefs about the world, induced from an annotated dataset of feature norms. This calls for a modified version of the standard denotation function and for the replacement of the truth function with a 'plausibility' function, which would indicate how likely a stereotypical speaker might be to agree with a particular sentence. While this would be a fundamental departure from the core philosophy of model theory, we feel that it may be a worthwhile endeavour, allowing us to preserve the immense benefits of the set-theoretic apparatus in a cognitively plausible fashion. Following this aim, we hope to expand the preliminary framework presented here into a more expressive vector-based interpretation of set theory, catering for aspects not covered in this paper (e.g. cardinality, non-intersective modification) and refining our notion of a model, together with its relation to meaning.

Acknowledgments

We thank Marco Baroni, Stephen Clark, Ann Copestake and Katrin Erk for their helpful comments on a previous version of this paper, and the three anonymous reviewers for their thorough feedback on this work. Eva Maria Vecchi is supported by ERC Starting Grant DisCoTex (306920).

References

Hiyan Alshawi and Richard Crouch. 1992. Monotonic semantic interpretation. In *Proceedings of the 30th annual meeting on Association for Computational Linguistics*, pages 32–39. Association for Computational Linguistics.

Mark Andrews, Gabriella Vigliocco, and David Vinson. 2009. Integrating experiential and distributional data to learn semantic representations. *Psychological Review*, 116(3):463–498.

Fahiem Bacchus. 1989. A modest, but semantically well founded, inheritance reasoner. In *Proceedings of the 11th International Joint Conference on Artificial Intelligence*, pages 1104–1109, Detroit, MI.

Timothy Baldwin, Emily M Bender, Dan Flickinger, Ara Kim, and Stephan Oepen. 2004. Road-testing

the English Resource Grammar over the British National Corpus. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC2004)*, Lisbon, Portugal.

Marco Baroni, Raffaella Bernardi, Ngoc-Quynh Do, and Chung-chieh Shan. 2012. Entailment above the word level in distributional semantics. In *Proceedings of the fifteenth Conference of the European Chapter of the Association for Computational Linguistics (EACL2012)*, pages 23–32.

Marco Baroni, Raffaela Bernardi, and Roberto Zamparelli. 2014a. Frege in space: A program of compositional distributional semantics. *Linguistic Issues in Language Technology*, 9.

Marco Baroni, Georgiana Dinu, and Germán Kruszewski. 2014b. Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL2014)*, pages 238–247, Baltimore, Maryland.

Islam Beltagy, Cuong Chau, Gemma Boleda, Dan Garrette, Katrin Erk, and Raymond Mooney. 2013. Montague meets Markov: Deep semantics with probabilistic logical form. In *Second Joint Conference on Lexical and Computational Semantics (*SEM2013)*, pages 11–21, Atlanta, Georgia, USA.

Raffaella Bernardi, Georgiana Dinu, Marco Marelli, and Marco Baroni. 2013. A relatedness benchmark to test the role of determiners in compositional distributional semantics. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL2013)*, Sofia, Bulgaria.

Johan Bos. 2008. Wide-coverage semantic analysis with Boxer. In *Proceedings of the 2008 Conference on Semantics in Text Processing (STEP2008)*, pages 277–286.

Ronnie Cann. 1993. *Formal semantics*. Cambridge University Press.

Stephen Clark. 2012. Vector space models of lexical meaning. In Shalom Lappin and Chris Fox, editors, *Handbook of Contemporary Semantics – second edition*. Wiley-Blackwell.

Daoud Clarke. 2012. A context-theoretic framework for compositionality in distributional semantics. *Computational Linguistics*, 38(1):41–71.

Bob Coecke, Mehrnoosh Sadrzadeh, and Stephen Clark. 2011. Mathematical foundations for a compositional distributional model of meaning. *Linguistic Analysis: A Festschrift for Joachim Lambek*, 36(1–4):345–384.

Robin Cooper, Dick Crouch, JV Eijckl, Chris Fox, JV Genabith, J Japars, Hans Kamp, David Milward, Manfred Pinkal, Massimo Poesio, et al. 1996. A framework for computational semantics (FraCaS). Technical report, The FraCaS Consortium.

Ido Dagan, Bill Dolan, Bernardo Magnini, and Dan Roth. 2009. Recognizing textual entailment: Rational, evaluation and approaches. *Natural Language Engineering*, 15:459–476.

Georgiana Dinu and Mirella Lapata. 2010. Measuring distributional similarity in context. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP2010)*, pages 1162–1172.

Georgiana Dinu, Nghia The Pham, and Marco Baroni. 2013. DISSECT: DIStributional SEmantics Composition Toolkit. In *Proceedings of the System Demonstrations of ACL 2013*, Sofia, Bulgaria.

Katrin Erk and Sebastian Padó. 2008. A structured vector space model for word meaning in context. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing (EMNLP2008)*, pages 897–906, Honolulu, HI.

Katrin Erk. 2012. Vector space models of word meaning and phrase meaning: a survey. *Language and Linguistics Compass*, 6:635–653.

Katrin Erk. 2013. Towards a semantics for distributional representations. In *Proceedings of the Tenth International Conference on Computational Semantics (IWCS2013)*, Potsdam, Germany.

Katrin Erk. 2015. What do you know about an alligator when you know the company it keeps? Unpublished draft. `https://utexas.box.com/s/ekznoh08afi1kpkbf0hb`.

Andrea Frome, Greg S Corrado, Jon Shlens, Samy Bengio, Jeff Dean, Tomas Mikolov, et al. 2013. Devise: A deep visual-semantic embedding model. In *Advances in Neural Information Processing Systems*, pages 2121–2129.

Luana Făgărăşan, Eva Maria Vecchi, and Stephen Clark. 2015. From distributional semantics to feature norms: Grounding semantic models in human perceptual data. In *Proceedings of the 11th International Conference on Computational Semantics (IWCS 2015)*, London, UK.

Dan Garrette, Katrin Erk, and Raymond Mooney. 2013. A formal approach to linking logical form and vector-space lexical semantics. In Harry Bunt, Johan Bos, and Stephen Pulman, editors, *Computing Meaning*, volume 4. Springer.

Sheila Glasbey. 2006. Bare plurals in object position: which verbs fail to give existential readings, and why? In Liliane Tasmowski and Svetlana Vogeleer, editors, *Non-definiteness and Plurality*, pages 133–157. Amsterdam: Benjamins.

Edward Grefenstette. 2013. Towards a formal distributional semantics: Simulating logical calculi with tensors. In *Proceedings of the Second Joint Conference on Lexical and Computational Semantics (*SEM2013)*, Atlanta, GA.

Abhijeet Gupta, Gemma Boleda, Marco Baroni, and Sebastian Padó. 2015. Distributional vectors encode referential attributes. In *Proceedingsof the Conference on Empirical Methods in Natural Language Processing (EMNLP2015)*, Lisboa, Portugal.

Aurélie Herbelot and Eva Maria Vecchi. 2015. From concepts to models: some issues in quantifying feature norms. *Linguistic Issues in Language Technology*. To appear.

Aurélie Herbelot. 2013. What is in a text, what isn't, and what this has to do with lexical semantics. In *Proceedings of the Tenth International Conference on Computational Semantics (IWCS2013)*, Potsdam, Germany.

Karl Moritz Hermann, Edward Grefenstette, and Phil Blunsom. 2013. "Not not bad" is not "bad": A distributional account of negation. In *Proceedings of the 2013 Workshop on Continuous Vector Space Models and their Compositionality (ACL2013)*, Sofia, Bulgaria.

Angeliki Lazaridou, Elia Bruni, and Marco Baroni. 2014. Is this a wampimuk? Cross-modal mapping between distributional semantics and the visual world. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL2014)*, pages 1403–1414, Baltimore, Maryland.

Mike Lewis and Mark Steedman. 2013. Combined Distributional and Logical Semantics. *Transactions of the Association for Computational Linguistics*, 1:179–192.

Bill MacCartney and Christopher D Manning. 2008. Modeling semantic containment and exclusion in natural language inference. In *Proceedings of the 22nd International Conference on Computational Linguistics (COLING08)*, pages 521–528, Manchester, UK.

Ken McRae, George S Cree, Mark S Seidenberg, and Chris McNorgan. 2005. Semantic feature production norms for a large set of living and nonliving things. *Behavior research methods*, 37(4):547–559.

Björn-Helge Mevik and Ron Wehrens. 2007. The pls package: Principal component and partial least squares regression in R. *Journal of Statistical Software*, 18(2). Published online: `http://www.jstatsoft.org/v18/i02/`.

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

Tomas Mikolov, Quoc V Le, and Ilya Sutskever. 2013b. Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168*.

Brian Riordan and Michael N Jones. 2011. Redundancy in perceptual and linguistic experience: Comparing feature-based and distributional models of semantic representation. *Topics in Cognitive Science*, 3(2):303–345.

Peter D. Turney and Patrick Pantel. 2010. From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research*, 37:141–188.

J.F.A.K. Van Benthem. 1986. *Essays in logical semantics*. Number 29. Reidel.

Carl M Vogel. 1995. *Inheritance reasoning: Psychological plausibility, proof theory and semantics*. Ph.D. thesis, University of Edinburgh. College of Science and Engineering. School of Informatics.

Luke S Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of the 21st Conference on Uncertainty in AI*, pages 658–666.

Dependency Graph-to-String Translation

Liangyou Li Andy Way Qun Liu
ADAPT Centre, School of Computing
Dublin City University
{liangyouli,away,qliu}@computing.dcu.ie

Abstract

Compared to tree grammars, graph grammars have stronger generative capacity over structures. Based on an edge replacement grammar, in this paper we propose to use a synchronous graph-to-string grammar for statistical machine translation. The graph we use is directly converted from a dependency tree by labelling edges. We build our translation model in the log-linear framework with standard features. Large-scale experiments on Chinese–English and German–English tasks show that our model is significantly better than the state-of-the-art hierarchical phrase-based (HPB) model and a recently improved dependency tree-to-string model on BLEU, METEOR and TER scores. Experiments also suggest that our model has better capability to perform long-distance reordering and is more suitable for translating long sentences.

1 Introduction

Compared to trees, which have dominated the field of natural language processing (NLP) for decades, graphs are more general for modelling natural languages. The corresponding grammars for recognizing and producing graphs are more flexible and powerful than tree grammars. However, because of their high complexity, graph grammars have not been widely used in NLP.

Recently, along with progress on graph-based meaning representation, hyperedge replacement grammars (HRG) (Drewes et al., 1997) have been revisited, explored and used for semantic-based machine translation (Jones et al., 2012). However, the translation process is rather complex and the resources it relies on, namely abstract meaning corpora, are limited as well.

As most available syntactic resources and tools are tree-based, in this paper we propose to convert dependency trees, which are usually taken as a kind of shallow semantic representation, to dependency graphs by labelling edges. We then use a synchronous version of edge replacement grammar (ERG) (Section 2), a special case of HRG, to translate these graphs. The resulting translation model has the same order of magnitude in terms of time complexity with the hierarchical phrase-based model (HPB) (Chiang, 2005) under a certain restriction (Section 3).

Compared to dependency tree-to-string models, using ERG for graph-to-string translation brings some benefits (Section 3). Thanks to the stronger generative capacity of the grammar, our model can naturally translate siblings in a tree structure, which are usually treated as non-syntactic phrases and handled by other techniques (Huck et al., 2014; Xie et al., 2014). Furthermore, compared to the known treelet approach (Quirk et al., 2005) and Dep2Str (Xie et al., 2011), our method not only uses treelets but also has a full capacity of reordering.

We define our translation model (Section 4) in the log-linear framework (Och and Ney, 2002). Large-scale experiments (Section 5) on Chinese–English and German–English, two language pairs that have a high degree of syntactic reordering, show that our method significantly improves translation quality over both HPB and Dep2Str, as measured by BLEU (Papineni et al., 2002), TER (Snover et al., 2006) and METEOR (Denkowski and Lavie, 2011). We also find that the rules in our model are more suitable for long-distance reordering and translating long sentences.

2 Edge Replacement Grammar

As a special case of HRG, ERG is also a context-free rewriting grammar to recognize and produce graphs. Following HRG, the graph we use in this

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 33–43,
Lisbon, Portugal, 17-21 September 2015.

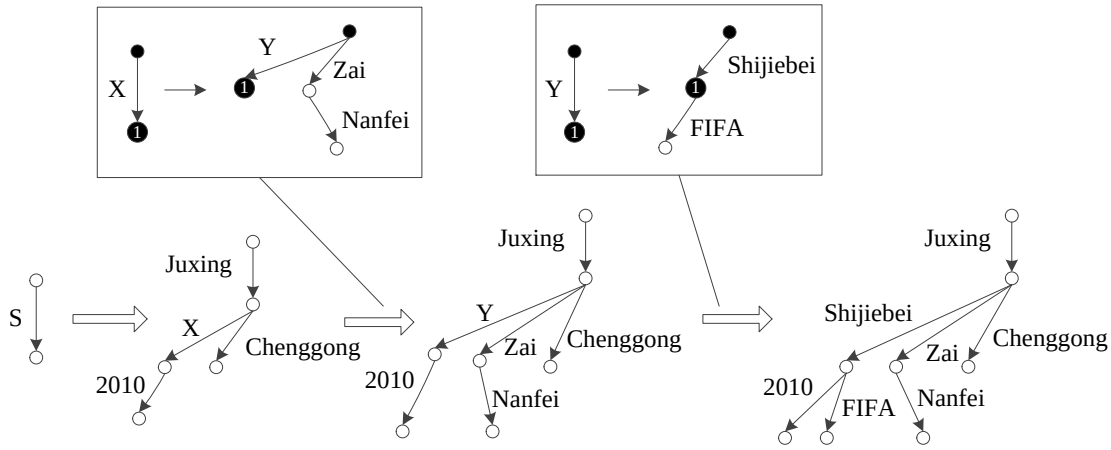


Figure 1: An example of a derivation in an ERG. Dark circles are external nodes.

paper is connected, nodes ordered, acyclic and has edge labels but no node labels (Chiang et al., 2013). We provide some formal definitions on ERG.

Definition 1. A *connected, edge-labeled, ordered graph* is a tuple $H = \langle V, E, \phi \rangle$, where

- V is a finite set of nodes.
- $E \subseteq V^2$ is a finite set of edges.
- $\phi : E \rightarrow C$ assigns a label (drawn from C) to each edge.

In ERG, the elementary unit is a graph fragment, which is also the right-hand side of a production in the grammar. Its definition is as follows.

Definition 2. A *graph fragment* is a tuple $H = \langle V, E, \phi, X \rangle$, where $\langle V, E, \phi \rangle$ is a graph and $X \in (V \cup V^2)$ is a list of distinct nodes. Following Chiang et al. (2013), we call these *external nodes*.

The external nodes indicate how to integrate a graph into another one during a derivation. Different to HRG, ERG limits the number of external nodes to 2 at most to make sure hyperedges do not exist during a derivation. Now we define the ERG.

Definition 3. An *edge replacement grammar* is a tuple $\langle N, T, P, S \rangle$, where

- N and T are disjoint finite sets of nonterminal symbols and terminal symbols, respectively.
- P is a finite set of productions of the form $A \rightarrow R$, where $A \in N$ and R is a graph fragment, where edge-labels are from $N \bigcup T$.
- $S \in N$ is the start symbol.

Figure 1 shows an example of a derivation in an ERG to produce a graph. Starting from the start symbol S, when a rule $(A \rightarrow R)$ is applied to an edge e, the edge is replaced by the graph fragment R. Just like in HRG, the ordering of nodes V_e in e and external nodes X_R in R implies the mapping from V_e to X_R (Chiang et al., 2013).

3 Graph-to-String Grammar

In SMT, we need a synchronous grammar to simultaneously parse an input graph and produce translations. The graph we use in this paper is from a dependency structure which is capable of modelling long-distance relations in a sentence.

3.1 The Grammar

Before defining the synchronous grammar, we firstly define a dependency graph which is a special case of a graph.

Definition 4. A *dependency graph* is a tuple $\langle V, E, \phi, \Delta \rangle$, where $\langle V, E, \phi \rangle$ is a graph and Δ is a restriction: edges are ordered.

A dependency graph is directly derived from a dependency tree by labeling edges with words, as shown in Figure 2. Although in general graph edges are unordered, in Definition 4 we keep word order by ordering edges, because the word order is an important piece of information for translation.

Similar to the graph fragment, a dependency-graph fragment is defined as below.

Definition 5. A *dependency-graph fragment* is a tuple $\langle V, E, \phi, \Delta, X \rangle$, where $\langle V, E, \phi, \Delta \rangle$ is a dependency graph, $X \in (V \cup V^2)$ is a list of external nodes.

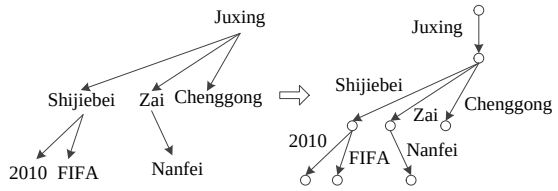


Figure 2: An example of deriving a dependency graph from a dependency tree by labelling edges with words.

In this paper, we define a synchronous ERG over dependency graphs as a dependency graph-to-string grammar, which can be used for MT.

Definition 6. A *dependency graph-to-string grammar* (DGSG) is a tuple $\langle N, T, T', P, S \rangle$, where

- N is a finite set of non-terminal symbols.
- T and T' are finite sets of terminal symbols.
- $S \in N$ is the start symbol.
- P is a finite set of productions of the form $\langle A \rightarrow R, A' \rightarrow R', \sim \rangle$, where $A, A' \in N$, R is a dependency-graph fragment over $N \bigcup T$ and R' is a string over $N \bigcup T'$. $\sim$ is a one-to-one mapping between non-terminal symbols in R and R'.

Figure 3 shows a derivation simultaneously producing a Chinese dependency graph and an English string using a DGSG. Each time a rule is applied, the dependency-graph fragment in the rule replaces an edge in the source graph, and the string in the rule replaces a non-terminal in the target string.

Proposition 1. DGSG has stronger generative capacity over graph-string pairs than both SCFG and synchronous tree substitution grammar (STSG).

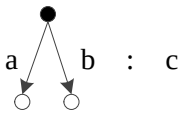
Proof. STSG has stronger generative capacity over structures than SCFG (Chiang, 2012).[1]

Any STSG can easily be converted into a DGSG by labelling edges in tree structures.

[1]The following STSG generates a trivial example of a tree-string pair that no SCFG can generate, as SCFG must always have an equal number of non-terminal symbols.

$$\begin{array}{c} X \\ X \quad | \\ | \ : X \\ \epsilon \quad | \\ \epsilon \end{array}$$

The following DGSG generates a trivial example of a graph-string pair, which no STSG can generate, as the left-head side has no head nodes while STSG always requires one to form a tree.



□

This proof is also verified in Figure 3 where the third rule is used to translate a non-syntactic phrase, which can be a problem for dependency tree-to-string methods. In addition, the second rule translates a treelet and the first rule encodes reordering information inside. All these three aspects are uniformly modeled in our grammar, which makes it more powerful than other methods, such as the treelet approach and the Dep2Str.

3.2 Time Complexity and a Restriction

Given a dependency graph, training and decoding time using DGSG depends on the number of dependency-graph fragments. For example, for a graph where the degree of a node is k, the number of all possible fragments starting from the node is $O(2^k)$. Therefore, the time complexity would be exponential if we consider them all.

It is easy to find that the high complexity of DGSG comes from the free combination of edges. That means that a dependency-graph fragment can cover discontinuous words of an input sentence. However, this is not the convention in the field of SMT.

For efficient training and decoding, we add a restriction to DGSG: each dependency-graph fragment covers a continuous span of the source sentence. This reduces the complexity from exponential time to cubic time.

3.3 Non-terminal Symbols

In this paper we build a dependency graph-to-string model, so we only use one non-terminal symbol X as in HPB on the target side. However, on the source side we define non-terminal symbols over Part-of-Speech (POS) tags, which can be easily obtained as a by-product of dependency parsing.

We define the *head* of a dependency-graph fragment H as a list of edges, the dependency head of each of which is not in this fragment. Then the

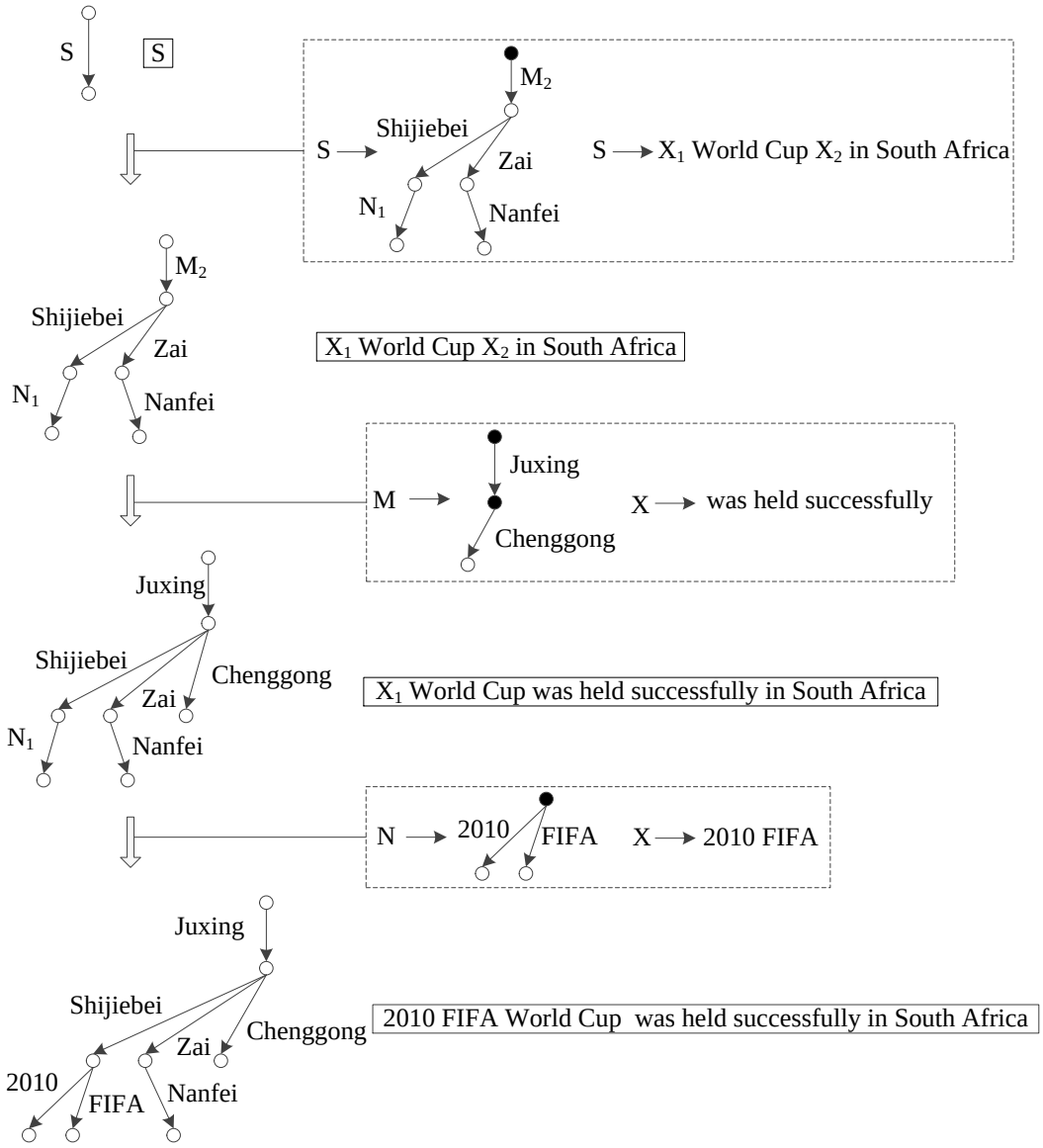


Figure 3: An example of a derivation in dependency graph-to-string grammar to produce a Chinese dependency graph and an English string. Rules are included in dashed rectangles. Target strings are in solid rectangles. External nodes are dark circles. This example is under the restriction in Section 3.2. In addition to the start symbol S, non-terminal symbols for the source side are M and N, while the target side only has one non-terminal X. The index in each non-terminal of a rule indicates the mapping.

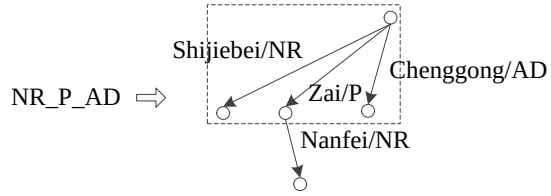


Figure 4: An example inducing a non-terminal symbol (left side) for a dependency-graph fragment (right side). Each edge is labeled by a word associated with its POS tag. The head of this fragment includes three edges which are in the rectangle.

non-terminal symbol for H is defined as the joining of POS tags of its head (Li et al., 2012). Figure 4 shows an example.

3.4 Rule Extraction

As well as the restriction defined in Section 3.2 making the grammar much smaller, it also results in a similar way of extracting rules as in HPB. Inspired by HPB, we define the rule set over *initial pairs*.

Given a word-aligned dependency graph-string pair $P = \langle G, e, \sim \rangle$, let G_i^j stand for the sub-graph (it may not be connected) covering words from position i to position j. Then a rule $\langle G_i^j, e_{i'}^{j'} \rangle$ is an initial pair of P, iff:

1. G_i^j is a dependency-graph fragment. That means it is a connected sub-graph and has at most two external nodes, nodes which connect with nodes outside or are the root.
2. It is consistent with the word alignment $\sim$ (Och and Ney, 2004).

The set of rules from P satisfies the following:

1. If $\langle G_i^j, e_{i'}^{j'} \rangle$ is an initial pair, then
$$\langle N(G_i^j) \rightarrow G_i^j, X \rightarrow e_{i'}^{j'} \rangle$$
is a rule, where $N(G)$ defines the non-terminal symbol for G.
2. If $\langle N(R) \rightarrow R, X \rightarrow R' \rangle$ is a rule of P and $\langle G_i^j, e_{i'}^{j'} \rangle$ is an initial pair such that G_i^j is a sub-graph of R and $R' = r_1 e_{i'}^{j'} r_2$, then
$$\langle N(R) \rightarrow R \backslash G_{i\,k}^j, X \rightarrow r_1 X_k r_2 \rangle$$
is a rule of P, where $\backslash$ means replacing G_i^j in R with an edge labelled with $N(G_i^j)$ and k is a unique index for a pair of non-terminal symbols.

As in HPB, in addition to rules extracted from the parallel corpus, we also use glue rules to combine fragments and translations when no matched rule can be found.

Furthermore, we can use the same rule extraction algorithm as that in HPB, except that we need to check if a span of a source sentence indicates a dependency-graph fragment, in which case we keep the dependency structure and induce a non-terminal for the fragment.

4 Model and Decoding

We define our model in the log-linear framework over a derivation d, as in Equation (1):

$$P(d) \propto \prod_i \phi_i(d)^{\lambda_i} \quad (1)$$

where ϕ_i are features defined on derivations and λ_i are feature weights. In our experiments, we use 9 features:

- translation probabilities $P(s|t)$ and $P(t|s)$, where s is the source graph fragment and t is the target string.
- lexical translation probabilities $P_{lex}(s|t)$ and $P_{lex}(t|s)$.
- language model $lm(e)$ over translation e.
- rule penalty $exp(-1)$.
- word penalty $exp(|e|)$.
- glue penalty $exp(-1)$.
- unknown words penalty $exp(u(g))$, where $u(g)$ is the number of unknown words in a source graph g.

Our decoder is based on the conventional chart parsing CYK algorithm (Kasami, 1965; Younger, 1967; Cocke and Schwartz, 1970). It searches for the best derivation d^* among all possible derivations D, as in Equation (2):

$$d^* = \underset{d \in D}{\operatorname{argmax}} P(d) \quad (2)$$

For each span of an input graph, the decoder checks if it is a dependency-graph fragment. Then

ZH–EN

corpus	#sent.	#words(ZH)	#words(EN)
train	1.5M+	38M+	∼45M
dev	878	22,655	26,905
MT04	1,597	43,719	52,705
MT05	1,082	29,880	35,326

DE–EN

corpus	#sent.	#words(DE)	#words(EN)
train	2M+	52M+	55M+
dev	3,003	72,661	74,753
WMT12	3,003	72,603	72,988
WMT13	3,000	63,412	64,810

Table 1: Chinese–English (ZH–EN) and German–English (DE–EN) corpora. For the English side of dev and test sets, words counts are averaged across all references.

for each fragment, the decoder finds rules to translate it. The translation of a large span can be obtained by combining translations from its sub-span using rules which have non-terminals. Finally, glue rules are used to make sure that at least one translation is produced.

5 Experiment

We conduct experiments on Chinese–English and German–English translation tasks.

5.1 Datasets

The Chinese–English training corpus is from LDC, including LDC2002E18, LDC2003E07, LDC2003E14, LDC2004T07, the Hansards portion of LDC2004T08 and LDC2005T06. NIST 2002 is taken as a development set to tune weights, and NIST 2004 (MT04) and NIST 2005 (MT05) are two test sets to evaluate systems. Table 1 provides a summary of this corpus. The Stanford Chinese word segmenter (Chang et al., 2008) is used to segment Chinese sentences. The Stanford dependency parser (Chang et al., 2009) parses a Chinese sentence into a projective dependency tree which is then converted to a dependency graph in our model.

The German–English training corpus is from WMT 2014, including Europarl V7 and News Commentary. News-test 2011 is taken as a development set, while News-test 2012 (WMT12) and News-test 2013 (WMT13) are our test sets. Table 1 provides a summary of this corpus. We use mate-tools[2] to perform morphological analysis and parse German sentences (Bohnet, 2010). Then MaltParser[3] converts a parse result into a projective dependency tree (Nivre and Nilsson, 2005).

5.2 Settings

In this paper, we mainly compare our system (**DGST**) with HPB in Moses (Koehn et al., 2007). We implement our model in Moses and take the same settings as Moses HPB in all experiments. In addition, translation results from a recently open-source dependency tree-to-string system, Dep2Str[4] (Li et al., 2014), which is implemented in Moses and improves the dependency-based model in Xie et al. (2011), are also reported. All systems use the same sets of features defined in Section 4.

In all experiments, word alignment is performed by GIZA++ (Och and Ney, 2003) with the heuristic function *grow-diag-final-and*. We use SRILM (Stolcke, 2002) to train a 5-gram language model on the Xinhua portion of the English Gigaword corpus 5th edition with modified Kneser-Ney discounting (Chen and Goodman, 1996). Minimum Error Rate Training (MERT) (Och, 2003) is used to tune weights.

To obtain more reliable results, in each experiment, we run MERT three times and report average scores. These scores are calculated by three widely used automatic metrics in case-insensitive mode: BLEU, METEOR and TER.

5.3 Results

Table 2 shows the scores of all three metrics on all systems. Similar to Li et al. (2014), in our experiments Dep2Str has on average a comparable result with Moses HPB in terms of BLEU and METEOR scores. However, it obtains a significantly higher (i.e. worse) TER score on the Chinese–English task. This may suggest that translations produced by Dep2Str need more post-editing effort (He et al., 2010).

By contrast, on all test sets, measured by all metrics, our system is significantly better than Moses HPB. On the Chinese–English task, our system achieves an average gain of 1.25 (absolute, 3.6% relative) BLEU score and 0.55 (absolute, 1.7% relative) METEOR score while also ob-

[2] http://code.google.com/p/mate-tools/
[3] http://www.maltparser.org/
[4] http://computing.dcu.ie/~liangyouli/dep2str.zip

Metric	System	ZH–EN		DE–EN	
		MT04	MT05	WMT12	WMT13
BLEU ↑	Moses HPB	35.6	33.8	20.2	22.7
	Dep2Str	35.4	33.9	20.3	22.8
	DGST	**36.6**	**35.3**	**20.7**	**23.3**
METEOR ↑	Moses HPB	31.6	31.9	28.6	29.7
	Dep2Str	**31.8**	31.9	28.5	29.5*
	DGST	**32.1**	**32.5**	**28.7**	**29.8**
TER ↓	Moses HPB	57.0	58.3	63.2	59.5
	Dep2Str	58.2*	59.6*	63.1	59.6
	DGST	**56.1**	**57.0**	**62.6**	**59.0**

Table 2: Metric scores for all systems on Chinese–English (ZH–EN) and German–English (DE–EN) corpus. Each score is the average score over three MERT runs. Bold figures mean a system is significantly better than Moses HPB at $p \leq 0.01$. Moses HPB is significantly better than systems with * at $p \leq 0.01$.

Length	Percentage			
	MT04	MT05	WMT12	WMT13
(0, 10]	7.6%	8.6%	15.0%	19.2%
(10, 20]	28.2%	26.0%	31.4%	37.2%
(20, 30]	28.2%	26.5%	26.3%	24.5%
(30, 40]	20.2%	23.8%	14.4%	12.0%
(40, ∞)	15.7%	15.2%	12.9%	7.2%

Table 3: Statistics of sentence length on four test sets.

taining a reduction of 1.1 (absolute, 1.91% relative) TER score on average.

On the German–English task, our system achieves an average gain of 0.55 (absolute, 2.56% relative) BLEU score and 0.1 (absolute, 0.35% relative) METEOR score and also obtains a reduction of 0.55 (absolute, 0.89% relative) TER score on average.

5.4 Analysis

As shown in Table 2, compared to Moses HPB and Dep2Str, our system achieves higher translation quality as measured by three automatic metrics. In this section, we investigate whether dependency structures bring benefits as expected on long-distance reordering. Table 3 provides the statistics on sentence length of our four test sets.

In both HPB and our model, the length range of a reordering performed on an input sentence is related to the use of glue grammars which bring two benefits during decoding. When no matched rule is found in the models, glue grammars are applied to make sure a translation is produced. In addition, because of the generalization capability of rules, which typically are learned under a length limitation, using them on long sentences could cause translation quality to deteriorate. Therefore, when the length of a phrase is greater than a certain value, glue grammars are also applied. Therefore, our experiment of analysis is based on the length limitation that a rule can cover (max. phrase length) during decoding.

We set this max. phrase length to different values, including 10, 20 (default), 30, 40 and 50. Figure 5 gives the BLEU scores on all test sets. We find that on all different values, our system achieves higher BLEU scores than Moses HPB. In addition, when the max. phrase length becomes larger, Moses HPB shows a declining trend in most cases, especially on the German–English task (WMT12 and WMT13). However, our system is less sensitive to this value. We hypothesize that this is because rules from dependency graphs have better generalization for translating longer phrases and are more suitable for translating long sentences.

5.5 Case Study

On a manual check, we find that translations produced by our system are more fluent than those of both Moses HPB and Dep2Str. Figure 6 gives an example comparing translations produced by three systems on the Chinese–English task.

We first find a case of long-distance relation, i.e. the subject-verb-object (SVO) structure in the source sentence. In this example, this relation implies a long-distance reordering, which moves the translation of the object to the front of its modifiers, as shown in the given reference. Com-

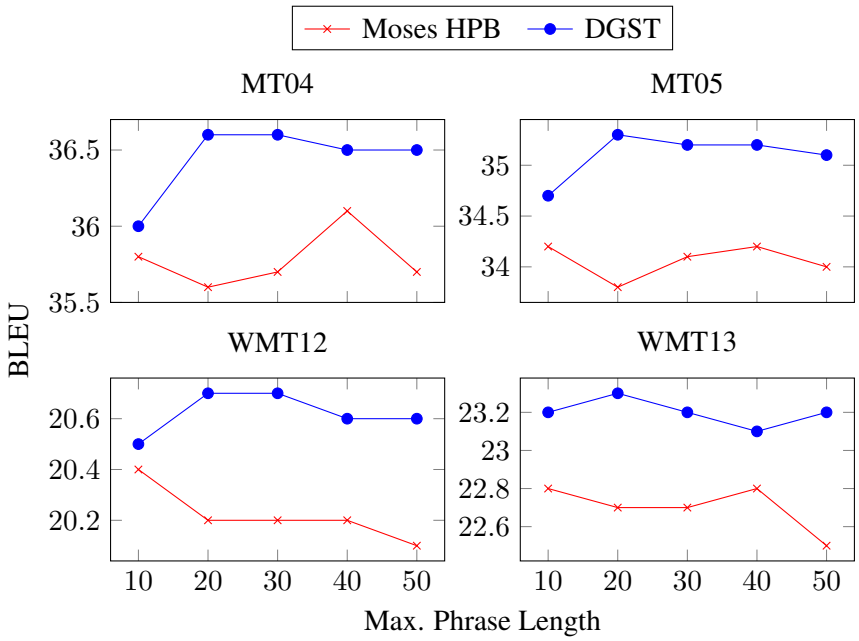
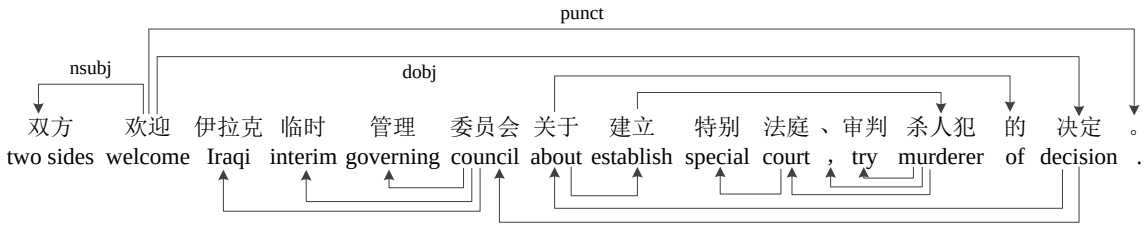


Figure 5: BLEU scores of Moses HPB and DGST (our system) when the length of maximum phrase that a rule can cover during decoding is set to different values.



Ref: The two sides welcomed the decision by the Iraqi Interim Governing Council to establish a special court to try the murderers.

HPB: the two sides welcomed the interim iraqi authority on establishing a special court, trial of the murderer.

Dep2Str: the two sides welcomed the decision on the Establishment of a special court, justice murderers of the provisional governing council of iraq.

DGST: the two sides welcomed the decision of the iraqi interim governing council on the establishment of a special court, justice murderers.

Figure 6: An example of comparing translations produced by three systems on the Chinese–English task. The source sentence is parsed into a dependency structure. Each source word is annotated by a corresponding English word (or phrase).

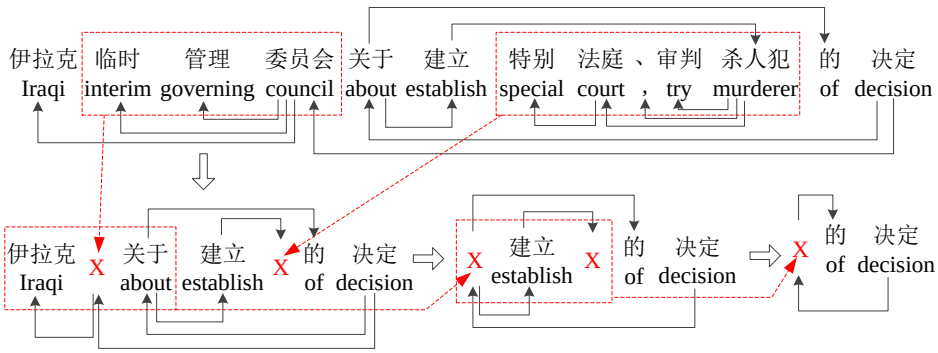


Figure 7: An example of inducing a dependency structure in Figure 6 to ”X 的(of) X” structure in our system by using treelets and non-syntactic phrases. ➪ denotes one or more steps. All non-terminals are simply represented by X.

pared to Moses HPB, both Dep2Str and our system, which rely on dependency structures, are capable of dealing with this. This also suggests that dependency structures are useful for long-distance reordering.

Furthermore, compared to Dep2Str, our system produces a better translation for the "X 的(of) X" expression, which is not explicitly represented in the dependency structure and thus results in a wrong translation in Dep2Str. After looking into the details of the translation process, we find that our system induces the dependency structure to the "X 的(of) X" structure by handling both treelets and non-syntactic phrases. Figure 7 shows the process of this induction.

6 Related Work

Dependency structures have been used in SMT for a few years. Because of its better inter-lingual phrasal cohesion properties (Fox, 2002), it is believed to be beneficial to translation.

Researchers have tried to use dependency structures on both target and source sides. Shen et al. (2010) propose a string-to-dependency model by using dependency fragments of neighbouring words on the target side, which makes the model easier to include a dependency-based language model.

Menezes and Quirk (2005) and Quirk et al. (2005) propose the treelet approach which uses dependency structures on the source side. Xiong et al. (2007) extend this approach by allowing gaps in rules. However, their methods need a separate reordering model to decide the position of translated words (insertion problem). To avoid this problem, Xie et al. (2011) propose to use full head-dependent structures of a dependency tree and build a new dependency-to-string model. However, this model has difficulties in handling non-syntactic phrasal rules and ignores treelets. Meng et al. (2013) and Xie et al. (2014) further augment this model by incorporating constituent phrases and integrating fix/float structures (Shen et al., 2010), respectively, to allow phrasal rules. Li et al. (2014) extend this model by decomposing head-dependent structures into treelets.

Different from these methods, by labelling edges and using the ERG, our model considers the three aspects in a unified way: treelet, reordering and non-syntactic phrase. In addition, the ERG also naturally provides a decision on what kind of treelets and phrases should be used.

7 Conclusion

In this paper, we present a dependency graph-to-string grammar based on a graph grammar, which we call edge replacement grammar. This grammar can simultaneously produce a pair of dependency graph and string. With a restriction of using contiguous edges, our translation model built using this grammar can decode an input dependency graph, which is directly converted from a dependency tree, in cubic time using the CYK algorithm.

Experiments on Chinese–English and German–English tasks show that our model is significantly better than the hierarchical phrase-based model and a recent dependency tree-to-string model (Dep2Str) in Moses. We also find that the rules used in our model are more suitable for long-distance reordering and translating long sentences.

Although experiments show significant improvements over baselines, our model has limitations that can be avenues for future work. The restriction used in this paper reduces the time complexity but at the same time reduces the generative capacity of graph grammars. Without allowing hyperedges or only using at most two external nodes reduces the phrase coverage in our model as well.

Acknowledgments

This research has received funding from the People Programme (Marie Curie Actions) of the European Union's Framework Programme (FP7/2007-2013) under REA grant agreement n° 317471. The ADAPT Centre for Digital Content Technology is funded under the SFI Research Centres Programme (Grant 13/RC/2106) and is co-funded under the European Regional Development Fund. We thank anonymous reviewers for their insightful comments and suggestions.

References

Bernd Bohnet. 2010. Very High Accuracy and Fast Dependency Parsing is Not a Contradiction. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 89–97, Beijing, China.

Pi-Chuan Chang, Michel Galley, and Christopher D. Manning. 2008. Optimizing Chinese Word Segmentation for Machine Translation Performance. In *Proceedings of the Third Workshop on Statistical*

Machine Translation, pages 224–232, Columbus, Ohio.

Pi-Chuan Chang, Huihsin Tseng, Dan Jurafsky, and Christopher D. Manning. 2009. Discriminative Reordering with Chinese Grammatical Relations Features. In *Proceedings of the Third Workshop on Syntax and Structure in Statistical Translation*, pages 51–59, Boulder, Colorado.

Stanley F. Chen and Joshua Goodman. 1996. An Empirical Study of Smoothing Techniques for Language Modeling. In *Proceedings of the 34th Annual Meeting on Association for Computational Linguistics*, ACL '96, pages 310–318, Santa Cruz, California.

David Chiang, Jacob Andreas, Daniel Bauer, Karl Moritz Hermann, Bevan Jones, and Kevin Knight. 2013. Parsing graphs with hyperedge replacement grammars. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 924–932, Sofia, Bulgaria, August.

David Chiang. 2005. A Hierarchical Phrase-based Model for Statistical Machine Translation. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 263–270, Ann Arbor, Michigan.

David Chiang. 2012. *Grammars for Language and Genes: Theoretical and Empirical Investigations*. Springer.

John Cocke and Jacob T. Schwartz. 1970. Programming Languages and Their Compilers: Preliminary Notes. Technical report, Courant Institute of Mathematical Sciences, New York University, New York, NY.

Michael Denkowski and Alon Lavie. 2011. Meteor 1.3: Automatic Metric for Reliable Optimization and Evaluation of Machine Translation Systems. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, WMT '11, pages 85–91, Edinburgh, Scotland.

Frank Drewes, Hans Jörg Kreowski, and Annegret Habel. 1997. Handbook of graph grammars and computing by graph transformation. chapter Hyperedge Replacement Graph Grammars, pages 95–162. World Scientific Publishing Co., Inc., River Edge, NJ, USA.

Heidi J. Fox. 2002. Phrasal Cohesion and Statistical Machine Translation. In *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing - Volume 10*, pages 304–3111, Philadelphia.

Yifan He, Yanjun Ma, Josef van Genabith, and Andy Way. 2010. Bridging SMT and TM with Translation Recommendation. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 622–630, Uppsala, Sweden, July.

Matthias Huck, Hieu Hoang, and Philipp Koehn. 2014. Augmenting String-to-Tree and Tree-to-String Translation with Non-Syntactic Phrases. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 486–498, Baltimore, Maryland, USA, June.

Bevan Jones, Jacob Andreas, Daniel Bauer, Karl Moritz Hermann, and Kevin Knight. 2012. Semantics-Based Machine Translation with Hyperedge Replacement Grammars. In *COLING 2012, 24th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers*, pages 1359–1376, Mumbai, India, December.

Tadao Kasami. 1965. An Efficient Recognition and Syntax-Analysis Algorithm for Context-Free Languages. Technical report, Air Force Cambridge Research Lab, Bedford, MA.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ond'ej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open Source Toolkit for Statistical Machine Translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, ACL '07, pages 177–180, Prague, Czech Republic.

Junhui Li, Zhaopeng Tu, Guodong Zhou, and Josef van Genabith. 2012. Head-Driven Hierarchical Phrase-based Translation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 33–37, Jeju Island, Korea, July.

Liangyou Li, Jun Xie, Andy Way, and Qun Liu. 2014. Transformation and Decomposition for Efficiently Implementing and Improving Dependency-to-String Model In Moses. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, October.

Arul Menezes and Chris Quirk. 2005. Dependency Treelet Translation: The Convergence of Statistical and Example-Based Machine-translation? In *Proceedings of the Workshop on Example-based Machine Translation at MT Summit X*, September.

Fandong Meng, Jun Xie, Linfeng Song, Yajuan L', and Qun Liu. 2013. Translation with Source Constituency and Dependency Trees. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1066–1076, Seattle, Washington, USA, October.

Joakim Nivre and Jens Nilsson. 2005. Pseudo-Projective Dependency Parsing. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 99–106, Ann Arbor, Michigan.

Franz Josef Och and Hermann Ney. 2002. Discriminative Training and Maximum Entropy Models for Statistical Machine Translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 295–302, Philadelphia, PA, USA.

Franz Josef Och and Hermann Ney. 2003. A Systematic Comparison of Various Statistical Alignment Models. *Computational Linguistics*, 29(1):19–51, March.

Franz Josef Och and Hermann Ney. 2004. The Alignment Template Approach to Statistical Machine Translation. *Computational Linguistics*, 30(4):417–449, December.

Franz Josef Och. 2003. Minimum Error Rate Training in Statistical Machine Translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, ACL '03, pages 160–167, Stroudsburg, PA, USA. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 311–318, Philadelphia, Pennsylvania.

Chris Quirk, Arul Menezes, and Colin Cherry. 2005. Dependency Treelet Translation: Syntactically Informed Phrasal SMT. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 271–279, Ann Arbor, Michigan, June.

Libin Shen, Jinxi Xu, and Ralph Weischedel. 2010. String-to-Dependency Statistical Machine Translation. *Computational Linguistics*, 36(4):649–671, December.

M. Snover, B. Dorr, R. Schwartz, L. Micciulla, and J. Makhoul. 2006. A Study of Translation Edit Rate with Targeted Human Annotation. In *Proceedings of Association for Machine Translation in the Americas*, pages 223–231, Cambridge, Massachusetts, USA, August.

Andreas Stolcke. 2002. SRILM '' An Extensible Language Modeling Toolkit. In *Proceedings of the International Conference Spoken Language Processing*, pages 901–904, Denver, CO.

Jun Xie, Haitao Mi, and Qun Liu. 2011. A Novel Dependency-to-string Model for Statistical Machine Translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 216–226, Edinburgh, United Kingdom.

Jun Xie, Jinan Xu, and Qun Liu. 2014. Augment Dependency-to-String Translation with Fixed and Floating Structures. In *Proceedings of the 25th International Conference on Computational Linguistics*, pages 2217–2226, Dublin, Ireland.

Deyi Xiong, Qun Liu, and Shouxun Lin. 2007. A Dependency Treelet String Correspondence Model for Statistical Machine Translation. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 40–47, Prague, June.

Daniel H. Younger. 1967. Recognition and Parsing of Context-Free Languages in Time n^3. *Information and Control*, 10(2):189–208.

Reordering Grammar Induction

Miloš Stanojević
ILLC
University of Amsterdam
m.stanojevic@uva.nl

Khalil Sima'an
ILLC
University of Amsterdam
k.simaan@uva.nl

Abstract

We present a novel approach for unsupervised induction of a Reordering Grammar using a modified form of permutation trees (Zhang and Gildea, 2007), which we apply to preordering in phrase-based machine translation. Unlike previous approaches, we induce in one step both the hierarchical structure and the transduction function over it from word-aligned parallel corpora. Furthermore, our model (1) handles non-ITG reordering patterns (up to 5-ary branching), (2) is learned from all derivations by treating not only labeling but also bracketing as latent variable, (3) is entirely unlexicalized at the level of reordering rules, and (4) requires no linguistic annotation.

Our model is evaluated both for accuracy in predicting target order, and for its impact on translation quality. We report significant performance gains over phrase reordering, and over two known preordering baselines for English-Japanese.

1 Introduction

Preordering (Collins et al., 2005) aims at permuting the words of a source sentence $\mathbf{s}$ into a new order $\acute{\mathbf{s}}$, hopefully close to a plausible target word order. Preordering is often used to bridge long distance reorderings (e.g., in Japanese- or German-English), before applying phrase-based models (Koehn et al., 2007). Preordering is often broken down into two steps: finding a suitable tree structure, and then finding a transduction function over it. A common approach is to use monolingual syntactic trees and focus on finding a transduction function of the sibling subtrees under the nodes (Lerner and Petrov, 2013; Xia and Mccord, 2004). The (direct correspondence) assumption underlying this approach is that permuting the siblings of nodes in a source syntactic tree can produce a plausible target order. An alternative approach creates reordering rules manually and then learns the right structure for applying these rules (Katz-Brown et al., 2011). Others attempt learning the transduction structure and the transduction function in two separate, consecutive steps (DeNero and Uszkoreit, 2011). Here we address the challenge of learning both the trees and the transduction functions jointly, in one fell swoop, from word-aligned parallel corpora.

Learning both trees and transductions jointly raises two questions. How to obtain suitable trees for the source sentence and how to learn a distribution over random variables specifically aimed at reordering in a hierarchical model? In this work we solve both challenges by using the factorizations of permutations into Permutation Trees (PETs) (Zhang and Gildea, 2007). As we explain next, PETs can be crucial for exposing the hierarchical reordering patterns found in word-alignments.

We obtain permutations in the training data by segmenting every word-aligned source-target pair into *minimal phrase pairs*; the resulting alignment between minimal phrases is written as a permutation (1:1 and onto) on the source side. Every permutation can be factorized into a *forest* of PETs (over the source sentences) which we use as a *latent treebank* for training a Probabilistic Context-Free Grammar (PCFG) tailor made for preordering as we explain next.

Figure 1 shows two alternative PETs for the same permutation over minimal phrases. The nodes have labels (like $P3142$) which stand for local permutations (called prime permutation) over the child nodes; for example, the root label $P3142$ stands for prime permutation $\langle 3, 1, 4, 2\rangle$, which says that the first child of the root becomes 3^{rd} on the target side, the second becomes 1^{st}, the third

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 44–54,
Lisbon, Portugal, 17-21 September 2015.

becomes 4^{th} and the fourth becomes 2^{nd}. The prime permutations are non-factorizable permutations like $\langle 1, 2\rangle$, $\langle 2, 1\rangle$ and $\langle 2, 4, 1, 3\rangle$.

We think PETs are suitable for learning preordering for two reasons. Firstly, PETs specify *exactly* the phrase pairs defined by the permutation. Secondly, every permutation is factorizable into prime permutations only (Albert and Atkinson, 2005). Therefore, PETs expose *maximal* sharing between different permutations in terms of both phrases and their reordering. We expect this to be advantageous for learning hierarchical reordering.

For learning preordering, we first extract an *initial* PCFG from the latent treebank of PETs over the source sentences only. We initialize the nonterminal set of this PCFG to the *prime permutations* decorating the PET nodes. Subsequently we split these coarse labels in the same way as latent variable splitting is learned for treebank parsing (Matsuzaki et al., 2005; Prescher, 2005; Petrov et al., 2006; Saluja et al., 2014). Unlike treebank parsing, however, our training treebank is latent because it consists of a whole forest of PETs per training instance ($\mathbf{s}$).

Learning the splits on a latent treebank of PETs results in a *Reordering PCFG* which we use to parse input source sentences into split-decorated trees, i.e., the labels are the splits of prime permutations. After parsing $\mathbf{s}$, we map the splits back on their initial prime permutations, and then retrieve a reordered version $\acute{\mathbf{s}}$ of $\mathbf{s}$. In this sense, our latent splits are *dedicated to reordering*.

We face two technical difficulties alien to work on latent PCFGs in treebank parsing. Firstly, as mentioned above, permutations may factorize into more than one PET (a forest) leading to a latent training treebank.[1] And secondly, after we parse a source string $\mathbf{s}$, we are interested in $\acute{\mathbf{s}}$, the permuted version of $\mathbf{s}$, not in the best derivation/PET. Exact computation is a known NP-Complete problem (Sima'an, 2002). We solve this by a new Minimum-Bayes Risk decoding approach using Kendall reordering score as loss function, which is an efficient measure over permutations (Birch and Osborne, 2011; Isozaki et al., 2010a).

In summary, this paper contributes:

- A novel latent hierarchical source reordering model working over all derivations of PETs
- A label splitting approach based on PCFGs over minimal phrases as terminals, learned from an ambiguous treebank, where the label splits start out from prime permutations.
- A fast Minimum Bayes Risk decoding over Kendall τ reordering score for selecting $\acute{\mathbf{s}}$.

We report results for extensive experiments on English-Japanese showing that our Reordering PCFG gives substantial improvements when used as preordering for phrase-based models, outperforming two existing baselines for this task.

2 PETs and the Hidden Treebank

We aim at learning a PCFG which we will use for parsing source sentences $\mathbf{s}$ into synchronous trees, from which we can obtain a reordered source version $\acute{\mathbf{s}}$. Since PCFGs are non-synchronous grammars, we will use the nonterminal labels to encode reordering transductions, i.e., this PCFG is implicitly an SCFG. We can do this because $\mathbf{s}$ and $\acute{\mathbf{s}}$ are over the same alphabet.

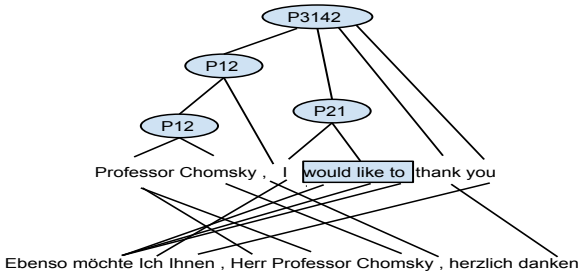
Here, we have access only to a word-aligned parallel corpus, not a treebank. The following steps summarize our approach for acquiring a latent treebank and how it is used for learning a Reordering PCFG:

1. Obtain a permutation over minimal phrases from every word-alignment.
2. Obtain a latent treebank of PETs by factorizing the permutations.
3. Extract a PCFG from the PETs with initial nonterminals taken from the PETs.
4. Learn to split the initial nonterminals and estimate rule probabilities.

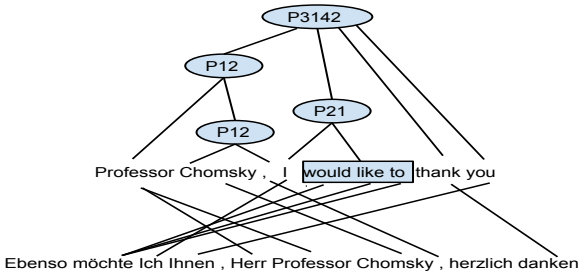
These steps are detailed in the next section, but we will start out with an intuitive exposition of PETs, the latent treebank and the Reordering Grammar.

Figure 1 shows examples of how PETs look like – see (Zhang and Gildea, 2007) for algorithmic details. Here we label the nodes with nonterminals which stand for *prime* permutations from the operators on the PETs. For example, nonterminals $P12$, $P21$ and $P3142$ correspond respectively to reordering transducers $\langle 1, 2\rangle$, $\langle 2, 1\rangle$ and $\langle 3, 1, 4, 2\rangle$. A prime permutation on a source node μ is a transduction dictating how the children of μ are reordered at the target side, e.g., $P21$ inverts the child order. We must stress that any similarity with ITG (Wu, 1997) is restricted to the fact that the straight and inverted operators of ITG are the binary case of prime permutations

[1] All PETs for the same permutation share the same set of prime permutations but differ only in bracketing structure (Zhang and Gildea, 2007).



(a) Canonical PET



(b) Alternative PET

Figure 1: Possible Permutation Trees (PETs) for one sentence pair

in PETs ($P12$ and $P21$). ITGs recognize only the binarizable permutations, which is a major restriction when used on the data: there are many non-binarizable permutations in actual data (Wellington et al., 2006). In contrast, our PETs are obtained by factorizing permutations obtained from the data, i.e., they exactly fit the range of prime permutations in the parallel corpus. In practice we limit them to maximum arity 5.

We can extract PCFG rules from the PETs, e.g., $P21 \rightarrow P12\ P2413$. However, these rules are decorated with too coarse labels. A similar problem was encountered in non-lexicalized monolingual parsing, and one solution was to lexicalize the productions (Collins, 2003) using *head words*. But linguistic heads do not make sense for PETs, so we opt for the alternative approach (Matsuzaki et al., 2005), which splits the nonterminals and softly percolates the splits through the trees gradually fitting them to the training data. Splitting has a shadow side, however, because it leads to combinatorial explosion in grammar size.

Suppose for example node $P21$ could split into $P21_1$ and $P21_2$ and similarly $P2413$ splits into $P2413_1$ and 2413_2. This means that rule $P21 \rightarrow P12\ P2413$ will form eight new rules:

$$P21_1 \rightarrow P12_1\ P2413_1 \quad P21_1 \rightarrow P12_1\ P2413_2$$
$$P21_1 \rightarrow P12_2\ P2413_1 \quad P21_1 \rightarrow P12_2\ P2413_2$$
$$P21_2 \rightarrow P12_1\ P2413_1 \quad P21_2 \rightarrow P12_1\ P2413_2$$
$$P21_2 \rightarrow P12_2\ P2413_1 \quad P21_2 \rightarrow P12_2\ P2413_2$$

Should we want to split each nonterminal into 30 subcategories, then an *n-ary* rule will split into 30^{n+1} new rules, which is prohibitively large. Here we use the "unary trick" as in Figure 2. The superscript on the nonterminals denotes the child position from left to right. For example $P21_1^2$ means that this node is a second child, and the mother nonterminal label is $P21_1$. For the running example rule, this gives the following rules:

$$P21_1 \rightarrow P21_1^1\ P21_1^2 \quad P21_2 \rightarrow P21_2^1\ P21_2^2$$
$$P21_1^1 \rightarrow P12_1 \quad P21_1^2 \rightarrow P2413_1$$
$$P21_1^1 \rightarrow P12_2 \quad P21_1^2 \rightarrow P2413_2$$
$$P21_2^1 \rightarrow P12_1 \quad P21_2^2 \rightarrow P2413_1$$
$$P21_2^1 \rightarrow P12_2 \quad P21_2^2 \rightarrow P2413_2$$

The unary trick leads to substantial reduction in grammar size, e.g., for arity 5 rules and 30 splits we could have had $30^6 = 729000000$ split-rules, but with the unary trick we only have $30{+}30^2{*}5 = 4530$ split rules. The unary trick was used in early lexicalized parsing work (Carroll and Rooth, 1998).[2] This split PCFG constitutes a *latent PCFG* because the splits cannot be read of a treebank. It must be learned from the latent treebank of PETs, as described next.

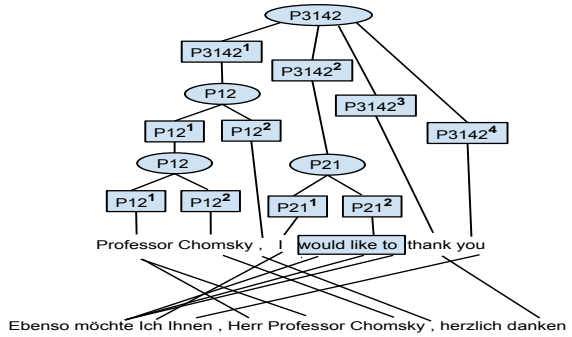


Figure 2: Permutation Tree with unary trick

3 Details of Latent Reordering PCFG

Obtaining permutations Given a source sentence $\mathbf{s}$ and its alignment $\mathbf{a}$ to a target sentence

[2] After applying the unary trick, we add a constraint on splitting: all nonterminals on an n-ary branching rule must be split simultaneously.

$\mathbf{t}$ in the training corpus, we segment $\langle \mathbf{s}, \mathbf{a}, \mathbf{t} \rangle$ into a sequence of *minimal phrases* $\mathbf{s}_m$ (maximal sequence) such that the reordering between these minimal phrases constitutes a permutation π_m. We do not extract non-contiguous or non-minimal phrases because reordering them often involves complicated transductions which could hamper the performance of our learning algorithm.[3]

Unaligned words Next we describe the use of the factorization of permutations into PET forests for training a PCFG model. But first we need to extend the PETs to allow for unaligned words. An unaligned word is joined with a neighboring phrase to the left or the right, depending on the source language properties (e.g., whether the language is head-initial or -final (Chomsky, 1970)). Our experiments use English as source language (head-initial), so the unaligned words are joined to phrases to their right. This modifies a PET by adding a new binary branching node μ (dominating the unaligned word and the phrase it is joined to) which is labeled with a dedicated nonterminal: $P01$ if the unaligned word joins to the right and $P10$ if it joins to the left.

3.1 Probability model

We decompose the permutation π_m into a forest of permutation trees $PEF(\pi_m)$ in $O(n^3)$, following algorithms in (Zhang et al., 2008; Zhang and Gildea, 2007) with trivial modifications. Each PET $\mathbf{\Delta} \in PEF(\pi_m)$ is a different bracketing (differing in binary branching structure only). We consider the bracketing hidden in the latent treebank, and apply unsupervised learning to induce a distribution over possible bracketings. Our probability model starts from the joint probability of a sequence of minimal phrases $\mathbf{s}_m$ and a permutation π_m over it. This demands summing over all PETs $\mathbf{\Delta}$ in the forest $PEF(\pi_m)$, and for every PET also over all its label splits, which are given by the grammar derivations $\mathbf{d}$:

$$P(\mathbf{s}_m, \pi_m) = \sum_{\mathbf{\Delta} \in PEF(\pi_m)} \sum_{\mathbf{d} \in \mathbf{\Delta}} P(\mathbf{d}, \mathbf{s}_m) \qquad (1)$$

The probability of a derivation $\mathbf{d}$ is a product of probabilities of all the rules $\mathbf{r}$ that build it:

$$P(\mathbf{s}_m, \pi_m) = \sum_{\mathbf{\Delta} \in PEF(\pi_m)} \sum_{\mathbf{d} \in \mathbf{\Delta}} \prod_{\mathbf{r} \in \mathbf{d}} P(\mathbf{r}) \qquad (2)$$

[3] Which differs from (Quirk and Menezes, 2006).

As usual, the parameters of this model are the PCFG rule probabilities which are estimated from the latent treebank using EM as explained next.

3.2 Learning Splits on Latent Treebank

For training the latent PCFG over the latent treebank, we resort to EM (Dempster et al., 1977) which estimates PCFG rule probabilities to maximize the likelihood of the parallel corpus instances. Computing expectations for EM is done efficiently using Inside-Outside (Lari and Young, 1990). As in other state splitting models (Matsuzaki et al., 2005), after splitting the non-terminals, we distribute the probability uniformly over the new rules, and we add to each new rule some random noise to break the symmetry. We split the non-terminals only once as in (Matsuzaki et al., 2005) (unlike (Petrov et al., 2006)). For estimating the distribution for unknown words we replace all words that appear ≤ 3 times with the "UNKNOWN" token.

3.3 Inference

We use CKY+ (Chappelier and Rajman, 1998) to parse a source sentence $\mathbf{s}$ into a forest using the learned split PCFG. Unfortunately, computing the most-likely permutation (or alternatively $\acute{\mathbf{s}}$) as in

$$\operatorname*{argmax}_{\pi \in \Pi} \sum_{\mathbf{\Delta} \in PEF(\pi)} \sum_{\mathbf{d} \in \mathbf{\Delta}} P(\mathbf{d}, \pi_m)$$

from a lattice of permutations Π using a PCFG is NP-complete (Sima'an, 2002). Existing techniques, like variational decoding or Minimum-Bayes Risk (MBR), used for minimizing loss over trees as in (Petrov and Klein, 2007), are not directly applicable here. Hence, we opt for minimizing the risk of making an error under a loss function over permutations using the MBR decision rule (Kumar and Byrne, 2004):

$$\hat{\pi} = \operatorname*{argmin}_{\pi} \sum_{\pi_r} Loss(\pi, \pi_r) P(\pi_r) \qquad (3)$$

The loss function we minimize is Kendall τ (Birch and Osborne, 2011; Isozaki et al., 2010a) which is a ratio of wrongly ordered pairs of words (including gapped pairs) to the total number of pairs. We do Monte Carlo sampling of 10000 derivations from the chart of the $\mathbf{s}$ and then find the least risky permutation in terms of this loss. We sample from the true distribution by sampling edges recursively

using their inside probabilities. An empirical distribution over permutations $P(\pi)$ is given by the relative frequency of π in the sample.

With large samples it is hard to efficiently compute expected Kendall τ loss for each sampled hypothesis. For sentence of length k and sample of size n the complexity of a naive algorithm is $O(n^2k^2)$. Computing Kendall τ alone takes $O(k^2)$. We use the fact that Kendall τ decomposes as a linear function over all skip-bigrams b that could be built for any permutation of length k:

$$Kendall(\pi, \pi_r) = \sum_b \frac{1 - \delta(\pi, b)}{\frac{k(k-1)}{2}} \delta(\pi_r, b) \quad (4)$$

Here δ returns 1 if permutation π contains the skip bigram b, otherwise it returns 0. With this decomposition we can use the method from (DeNero et al., 2009) to efficiently compute the MBR hypothesis. Combining Equations 3 and 4 we get:

$$\hat{\pi} = \operatorname*{argmin}_{\pi} \sum_{\pi_r} \sum_b \frac{1 - \delta(\pi, b)}{\frac{k(k-1)}{2}} \delta(\pi_r, b) P(\pi_r) \quad (5)$$

We can move the summation inside and reformulate the expected Kendall τ loss as expectation over the skip-bigrams of the permutation.

$$= \operatorname*{argmin}_{\pi} \sum_b (1 - \delta(\pi, b)) \Big[\textstyle\sum_{\pi_r} \delta(\pi_r, b) P(\pi_r) \Big] \quad (6)$$

$$= \operatorname*{argmin}_{\pi} \sum_b (1 - \delta(\pi, b)) \mathbb{E}_{P(\pi_r)} \delta(\pi_r, b) \quad (7)$$

$$= \operatorname*{argmax}_{\pi} \sum_b \delta(\pi, b) \mathbb{E}_{P(\pi_r)} \delta(\pi_r, b) \quad (8)$$

This means we need to pass through the sampled list only twice: (1) to compute expectations over skip bigrams and (2) to compute expected loss of each sampled permutation. The time complexity is $O(nk^2)$ which is quite fast in practice.

4 Experiments

We conduct experiments with three baselines:

- **Baseline A**: No preordering.
- **Baseline B**: Rule based preordering (Isozaki et al., 2010b), which first obtains an HPSG parse tree using Enju parser [4] and after that swaps the children by moving the syntactic head to the final position to account for different head orientation in English and Japanese.
- **Baseline C**: LADER (Neubig et al., 2012): latent variable preordering that is based on ITG and large-margin training with latent variables. We used LADER in standard settings without any linguistic features (POS tags or syntactic trees).

And we test four variants of our model:

- **RG$_{\text{left}}$** - only canonical left branching PET
- **RG$_{\text{right}}$** - only canonical right branching PET
- **RG$_{\text{ITG-forest}}$** - all PETs that are binary (ITG)
- **RG$_{\text{PET-forest}}$** - all PETs.

We test these models on English-Japanese NTCIR-8 Patent Translation (PATMT) Task. For tuning we use all NTCIR-7 dev sets and for testing the test set from NTCIR-9 from both directions. All used data was tokenized (English with Moses tokenizer and Japanese with KyTea [5]) and filtered for sentences between 4 and 50 words. A subset of this data is used for training the Reordering Grammar, obtained by filtering out sentences that have prime permutations of arity > 5, and for the ITG version arity > 2. Baseline C was trained on 600 sentences because training is prohibitively slow. Table 1 shows the sizes of data used.

corpus	#sents	#words source	#words target
train RG$_{\text{PET}}$	786k	21M	–
train RG$_{\text{ITG}}$	783k	21M	–
train LADER	600	15k	–
train translation	950k	25M	30M
tune translation	2k	55K	66K
test translation	3k	78K	93K

Table 1: Data stats

The Reordering Grammar was trained for 10 iterations of EM on *train RG* data. We use 30 splits for binary non-terminals and 3 for non-binary. Training on this dataset takes 2 days and parsing tuning and testing set without any pruning takes 11 and 18 hours respectively.

4.1 Intrinsic evaluation

We test how well our model predicts gold reorderings before translation by training the alignment model using MGIZA++ [6] on the training corpus and using it to align the test corpus. Gold reorderings for the test corpus are obtained by sorting words by their average target position and (unaligned words follow their right neighboring

[4] http://www.nactem.ac.uk/enju/

[5] http://www.phontron.com/kytea/

[6] http://www.kyloo.net/software/doku.php/mgiza:overview

word). We use Kendall τ score for evaluation (note the difference with Section 3.3 where we defined it as a loss function).

Table 2 shows that our models outperform all baselines on this task. The only strange result here is that rule-based preordering obtains a lower score than no preordering, which might be an artifact of the Enju parser changing the tokenization of its input, so the Kendall τ of this system might not really reflect the real quality of the preordering. All other systems use the same tokenization.

	Kendall τ
$A^{No\ preordering}$	0.7655
$B^{Rule\ based}$	0.7567
C^{LADER}	0.8176
$RG_{\text{left-branching}}$	0.8201
$RG_{\text{right-branching}}$	0.8246
$RG_{\text{ITG-forest}}$	0.823
$RG_{\text{PET-forest}}$	0.8255

Table 2: Reordering prediction

4.2 Extrinsic evaluation in MT

The reordered output of all the mentioned baselines and versions of our model are translated with phrase-based MT system (Koehn et al., 2007) (distortion limit set to 6 with distance based reordering model) that is trained on **gold preordering** of the training data [7] $\acute{\mathbf{s}} - \mathbf{t}$. The only exception is Baseline A which is trained on original $\mathbf{s} - \mathbf{t}$.

We use a 5-gram language model trained with KenLM [8], tune 3 times with kb-mira (Cherry and Foster, 2012) to account for tuner instability and evaluated using Multeval [9] for statistical significance on 3 metrics: BLEU (Papineni et al., 2002), METEOR (Denkowski and Lavie, 2014) and TER (Snover et al., 2006). We additionally report RIBES score (Isozaki et al., 2010a) that concentrates on word order more than other metrics.

Single or all PETs? In Table 3 we see that using *all PETs* during training makes a big impact on performance. Only the *all PETs* variants

[7] Earlier work on preordering applies the preordering model to the training data to obtain a parallel corpus of *guessed* $\acute{\mathbf{s}} - \mathbf{t}$ pairs, which are the word re-aligned and then used for training the back-end MT system (Khalilov and Sima'an, 2011). We skip this, we take the risk of mismatch between the preordering and the back-end system, but this simplifies training and saves a good amount of training time.

[8] http://kheafield.com/code/kenlm/

[9] https://github.com/jhclark/multeval

System	BLEU ↑	METEOR ↑	TER ↓	RIBES ↑
$A^{No\ preord.}$	27.8	48.9	59.2	68.29
$B^{Rule\ based}$	29.6	48.7	59.2	71.12
C^{LADER}	31.1	50.5	56.0	74.29
RG_{left}	31.2^{AB}	50.5^{AB}	56.3^{AB}_{C}	**74.45**
RG_{right}	31.4^{AB}	50.5^{AB}	56.3^{AB}_{C}	**75.29**
$RG_{\text{ITG-forest}}$	$\mathbf{31.6^{ABC}}$	$\mathbf{50.8^{ABC}}$	$\mathbf{55.7^{ABC}}$	**75.29**
$RG_{\text{PET-forest}}$	$\mathbf{32.0^{ABC}}$	$\mathbf{51.0^{ABC}}$	$\mathbf{55.7^{ABC}}$	**75.62**

Table 3: Comparison of different preordering models. Superscripts A, B and C signify if the system is significantly better ($p < 0.05$) than the respective baseline or significantly worse (in which case it is a subscript). Significance tests were not computed for RIBES. Score is bold if the system is significantly better than all the baselines.

($RG_{\text{ITG-forest}}$ and $RG_{\text{PET-forest}}$) significantly outperform all baselines. If we are to choose a *single PET* per training instance, then learning RG from only left-branching PETs (the one usually chosen in other work, e.g. (Saluja et al., 2014)) performs slightly worse than the right-branching PET. This is possibly because English is mostly right-branching. So even though both PETs describe the same reordering, RG_{right} captures reordering over English input better than RG_{left}.

All PETs or binary only? *$RG_{\text{PET-forest}}$ performs significantly better than $RG_{\text{ITG-forest}}$* ($p < 0.05$). Non-ITG reordering operators are predicted rarely (in only 99 sentences of the test set), but they make a difference, because these operators often appear high in the predicted PET. Furthermore, having these operators during training might allow for better fit to the data.

How much reordering is resolved by the Reordering Grammar? Obviously, completely factorizing out the reordering from the translation process is impossible because reordering depends to a certain degree on target lexical choice. To quantify the contribution of Reordering Grammar, we tested decoding with different distortion limit values in the SMT system. We compare the phrase-based (PB) system with distance based cost function for reordering (Koehn et al., 2007) with and without preordering.

Figure 3 shows that Reordering Grammar gives substantial performance improvements at all distortion limits (both BLEU and RIBES). $RG_{\text{PET-forest}}$ is less sensitive to changes in decoder distortion limit than standard PBSMT. The perfor-

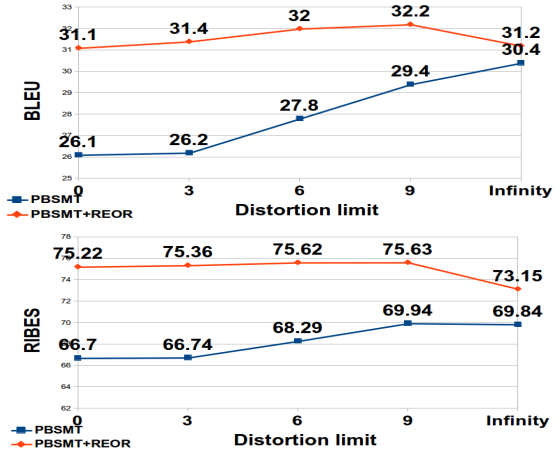


Figure 3: Distortion effect on BLEU and RIBES

mance of $RG_{PET\text{-}forest}$ varies only by 1.1 BLEU points while standard PBSMT by 4.3 BLEU points. Some local reordering in the decoder seems to help $RG_{PET\text{-}forest}$ but **large distortion limits seem to degrade the preordering choice**. This shows also that the improved performance of $RG_{PET\text{-}forest}$ is not only a result of efficiently exploring the full space of permutations, but also a result of improved scoring of permutations.

System	BLEU ↑	METEOR ↑	TER ↓	RIBES ↑
D^{PBMSD}	29.6	50.1	58.0	68.97
E^{Hiero}	32.6	52.1	54.5	74.12
$RG_{PET\text{-}forest}$+MSD	32.4^{D}	51.3^{D}_{E}	55.3^{D}_{E}	**75.72**

Table 4: Comparison to MSD and Hiero

Does the improvement remain for a decoder with MSD reordering model? We compare the $RG_{PET\text{-}forest}$ preordered model against a decoder that uses the strong MSD model (Tillmann, 2004; Koehn et al., 2007). Table 4 shows that using Reordering Grammar as front-end to MSD reordering (full Moses) improves performance by 2.8 BLEU points. The improvement is confirmed by METEOR, TER and RIBES. Our preordering model and MSD are complementary – the Reordering Grammar captures long distance reordering, while MSD possibly does better local reorderings, especially reorderings conditioned on the lexical part of translation units.

Interestingly, the MSD model (BLEU 29.6) improves over distance-based reordering (BLEU 27.8) by (BLEU 1.8), whereas the difference between these systems as back-ends to Reordering Grammar (respectively BLEU 32.4 and 32.0) is far smaller (0.4 BLEU). This suggests that a major share of reorderings can be handled well by preordering without conditioning on target lexical choice. Furthermore, this shows that $RG_{PET\text{-}forest}$ preordering is not very sensitive to the decoder's reordering model.

Comparison to a Hierarchical model (Hiero). Hierarchical preordering is not intended for a hierarchical model as Hiero (Chiang, 2005). Yet, here we compare our preordering system (PB MSD+RG) to Hiero for completeness, while we should keep in mind that Hiero's reordering model has access to much richer training data. We will discuss these differences shortly.

Table 4 shows that the difference in BLEU is not statistically significant, but there is more difference in METEOR and TER. RIBES, which concentrates more on reordering, prefers Reordering Grammar over Hiero. It is somewhat surprising that a preordering model combined with a phrase-based model succeeds to rival Hiero's performance on English-Japanese. Especially when looking at the differences between the two:

1. Reordering Grammar uses only minimal phrases, while Hiero uses composite (longer) phrases which encapsulate internal reorderings, but also non-contiguous phrases.
2. Hiero conditions its reordering on the lexical target side, whereas the Reordering Grammar does not (by definition).
3. Hiero uses a range of features, e.g., a language model, while Reordering Grammar is a mere generative PCFG.

The advantages of Hiero can be brought to bear upon Reordering Grammar by reformulating it as a discriminative model.

Which structure is learned? Figure 4 shows an example PET output showing how our model learns: (1) that the article "the" has no equivalent in Japanese, (2) that verbs go after their object, (3) to use postpositions instead of prepositions, and (4) to correctly group certain syntactic units, e.g. NPs and VPs.

5 Related work

The majority of work on preordering is based on syntactic parse trees, e.g., (Lerner and Petrov, 2013; Khalilov and Sima'an, 2011; Xia and Mccord, 2004). Here we concentrate on work that has common aspects with this work. Neubig et

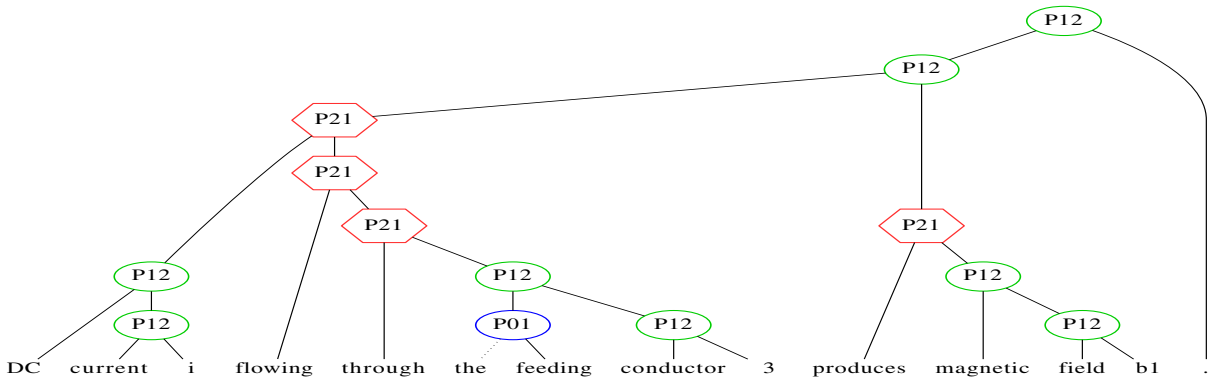


Figure 4: Example parse of English sentence that predicts reordering for English-Japanese

al (2012) trains a latent non-probabilistic discriminative model for preordering as an ITG-like grammar limited to binarizable permutations. Tromble and Eisner (2009) use ITG but do not train the grammar. They only use it to constrain the local search. DeNero and Uszkoreit (2011) present two separate consecutive steps for unsupervised induction of hierarchical structure (ITG) and the induction of a reordering function over it. In contrast, here we learn both the structure and the reordering function simultaneously. Furthermore, at test time, our inference with MBR over a measure of permutation (Kendall) allows exploiting both structure and reordering weights for inference, whereas test-time inference in (DeNero and Uszkoreit, 2011) is also a two step process – the parser forwards to the next stage the best parse.

Dyer and Resnik (2010) treat reordering as a latent variable and try to sum over all derivations that lead not only to the same reordering but also to the same translation. In their work they consider all permutations allowed by a given syntactic tree.

Saers et al (2012) induce synchronous grammar for translation by splitting the non-terminals, but unlike our approach they split generic non-terminals and not operators. Their most expressive grammar covers only binarizable permutations. The decoder that uses this model does not try to sum over many derivations that have the same yield. They do not make independence assumption like our "unary trick" which is probably the reason they do not split more than 8 times. They do not compare their results to any other SMT system and test on a very small dataset.

Saluja et al (2014) attempts inducing a refined Hiero grammar (latent synchronous CFG) from Normalized Decomposition Trees (NDT) (Zhang et al., 2008). While there are similarities with the present work, there are major differences. On the similarity side, NDTs are decomposing alignments in ways similar to PETs, and both Saluja's and our models refine the labels on the nodes of these decompositions. However, there are major differences between the two:

- Our model is completely monolingual and unlexicalized (does not condition its reordering on the translation) in contrast with the Latent SCFG used in (Saluja et al., 2014),
- Our Latent PCFG label splits are defined as refinements of prime permutations, i.e., specifically designed for learning reordering, whereas (Saluja et al., 2014) aims at learning label splitting that helps predicting NDTs from source sentences,
- Our model exploits **all PETs and all derivations**, both during training (latent treebank) and during inferences. In (Saluja et al., 2014) only left branching NDT derivations are used for learning the model.
- The training data used by (Saluja et al., 2014) is about 60 times smaller in number of words than the data used here; the test set of (Saluja et al., 2014) also consists of far shorter sentences where reordering could be less crucial.

A related work with a similar intuition is presented in (Maillette de Buy Wenniger and Sima'an, 2014), where nodes of a tree structure similar to PETs are labeled with reordering patterns obtained by factorizing word alignments into Hierarchical Alignment Trees. These patterns are used for labeling the standard Hiero grammar. Unlike this work, the labels extracted by (Maillette de Buy Wenniger and Sima'an, 2014) are clustered manually into less than a dozen labels without the possibility of fitting the labels to the training data.

6 Conclusion

We present a generative Reordering PCFG model learned from latent treebanks over PETs obtained by factorizing permutations over minimal phrase pairs. Our Reordering PCFG handles non-ITG reordering patterns (up to 5-ary branching) and it works with all PETs that factorize a permutation (rather than a single PET). To the best of our knowledge this is the first time both extensions are shown to improve performance. The empirical results on English-Japanese show that (1) when used for preordering, the Reordering PCFG helps particularly with relieving the phrase-based model from long range reorderings, (2) combined with a state-of-the-art phrase model, Reordering PCFG shows performance not too different from Hiero, supporting the common wisdom of factorizing long range reordering outside the decoder, (3) Reordering PCFG generates derivations that seem to coincide well with linguistically-motivated reordering patterns for English-Japanese. There are various direction we would like to explore, the most obvious of which are integrating the learned reordering with other feature functions in a discriminative setting, and extending the model to deal with non-contiguous minimal phrases.

Acknowledgments

This work is supported by STW grant nr. 12271 and NWO VICI grant nr. 277-89-002. We thank Wilker Aziz for comments on earlier version of the paper and discussions about MBR and sampling.

References

Michael H. Albert and Mike D. Atkinson. 2005. Simple Permutations and Pattern Restricted Permutations. *Discrete Mathematics*, 300(1-3):1–15.

Alexandra Birch and Miles Osborne. 2011. Reordering Metrics for MT. In *Proceedings of the Association for Computational Linguistics*, Portland, Oregon, USA. Association for Computational Linguistics.

Glenn Carroll and Mats Rooth. 1998. Valence Induction with a Head-Lexicalized PCFG. In *In Proceedings of Third Conference on Empirical Methods in Natural Language Processing*.

Jean-Cédric Chappelier and Martin Rajman. 1998. A Generalized CYK Algorithm for Parsing Stochastic CFG. In *Proceedings of the First Workshop on Tabulation in Parsing and Deduction*, pages 133–137.

Colin Cherry and George Foster. 2012. Batch Tuning Strategies for Statistical Machine Translation. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL HLT '12, pages 427–436.

David Chiang. 2005. A Hierarchical Phrase-Based Model for Statistical Machine Translation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 263–270, Ann Arbor, Michigan, June.

Noam Chomsky. 1970. Remarks on Nominalization. In Roderick A. Jacobs and Peter S. Rosenbaum, editors, *Readings in English Transformational Grammar*, pages 184–221. Ginn, Boston.

Michael Collins, Philipp Koehn, and Ivona Kučerová. 2005. Clause Restructuring for Statistical Machine Translation. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 531–540.

Michael Collins. 2003. Head-Driven Statistical Models for Natural Language Parsing. *Comput. Linguist.*, 29(4):589–637, December.

A. P. Dempster, N. M. Laird, and D. B. Rubin. 1977. Maximum Likelihood from Incomplete Data via the EM Algorithm. *JOURNAL OF THE ROYAL STATISTICAL SOCIETY, SERIES B*, 39(1):1–38.

John DeNero and Jakob Uszkoreit. 2011. Inducing Sentence Structure from Parallel Corpora for Reordering. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 193–203.

John DeNero, David Chiang, and Kevin Knight. 2009. Fast Consensus Decoding over Translation Forests. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2 - Volume 2*, ACL '09, pages 567–575.

Michael Denkowski and Alon Lavie. 2014. Meteor Universal: Language Specific Translation Evaluation for Any Target Language. In *Proceedings of the EACL 2014 Workshop on Statistical Machine Translation*.

Chris Dyer and Philip Resnik. 2010. Context-free Reordering, Finite-state Translation. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 858–866.

Hideki Isozaki, Tsutomu Hirao, Kevin Duh, Katsuhito Sudoh, and Hajime Tsukada. 2010a. Automatic Evaluation of Translation Quality for Distant Language Pairs. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 944–952.

Hideki Isozaki, Katsuhito Sudoh, Hajime Tsukada, and Kevin Duh. 2010b. Head Finalization: A Simple Reordering Rule for SOV Languages. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, WMT '10, pages 244–251.

Jason Katz-Brown, Slav Petrov, Ryan McDonald, Franz Och, David Talbot, Hiroshi Ichikawa, Masakazu Seno, and Hideto Kazawa. 2011. Training a Parser for Machine Translation Reordering. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 183–192, Edinburgh, Scotland, UK., July. Association for Computational Linguistics.

Maxim Khalilov and Khalil Sima'an. 2011. Context-Sensitive Syntactic Source-Reordering by Statistical Transduction. In *IJCNLP*, pages 38–46.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open Source Toolkit for Statistical Machine Translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, ACL '07, pages 177–180.

Shankar Kumar and William Byrne. 2004. Minimum Bayes-Risk Decoding for Statistical Machine Translation. In *Proceedings of the Joint Conference on Human Language Technologies and the Annual Meeting of the North American Chapter of the Association of Computational Linguistics (HLT-NAACL)*.

K. Lari and S. J. Young. 1990. The Estimation of Stochastic Context-Free Grammars using the Inside-Outside Algorithm. *Computer Speech and Language*, 4:35–56.

Uri Lerner and Slav Petrov. 2013. Source-Side Classifier Preordering for Machine Translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 513–523. Association for Computational Linguistics.

Gideon Maillette de Buy Wenniger and Khalil Sima'an. 2014. Bilingual Markov Reordering Labels for Hierarchical SMT. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 138–147, Doha, Qatar, October.

Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. 2005. Probabilistic CFG with Latent Annotations. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 75–82.

Graham Neubig, Taro Watanabe, and Shinsuke Mori. 2012. Inducing a Discriminative Parser to Optimize Machine Translation Reordering. In *Conference on Empirical Methods in Natural Language Processing and Natural Language Learning (EMNLP-CoNLL)*, pages 843–853, Jeju, Korea, July.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 311–318.

Slav Petrov and Dan Klein. 2007. Improved Inference for Unlexicalized Parsing. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 404–411, Rochester, New York, April.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning Accurate, Compact, and Interpretable Tree Annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 433–440, Sydney, Australia, July. Association for Computational Linguistics.

Detlef Prescher. 2005. Inducing Head-Driven PCFGs with Latent Heads: Refining a Tree-Bank Grammar for Parsing. In *In ECML05*.

Chris Quirk and Arul Menezes. 2006. Do we need phrases? Challenging the conventional wisdom in Statistical Machine Translation. In *Proceedings of HLT-NAACL 2006*. ACL/SIGPARSE, May.

Markus Saers, Karteek Addanki, and Dekai Wu. 2012. From Finite-State to Inversion Transductions: Toward Unsupervised Bilingual Grammar Induction. In *COLING 2012, 24th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, 8-15 December 2012, Mumbai, India*, pages 2325–2340.

A. Saluja, C. Dyer, and S. B. Cohen. 2014. Latent-Variable Synchronous CFGs for Hierarchical Translation. *Proceedings of EMNLP*.

Khalil Sima'an. 2002. Computational Complexity of Probabilistic Disambiguation. *Grammars*, 5(2):125–151.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *In Proceedings of Association for Machine Translation in the Americas*, pages 223–231.

Christoph Tillmann. 2004. A Unigram Orientation Model for Statistical Machine Translation. In *Proceedings of HLT-NAACL 2004: Short Papers*, HLT-NAACL-Short '04, pages 101–104.

Roy Tromble and Jason Eisner. 2009. Learning Linear Ordering Problems for Better Translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1007–1016, Singapore, August.

Benjamin Wellington, Sonjia Waxmonsky, and I. Dan Melamed. 2006. Empirical lower bounds on the complexity of translational equivalence. In *In Proceedings of ACL 2006*, pages 977–984.

Dekai Wu. 1997. Stochastic Inversion Transduction Grammars and Bilingual Parsing of Parallel Corpora. *Comput. Linguist.*, 23(3):377–403, September.

Fei Xia and Michael Mccord. 2004. Improving a Statistical MT System with Automatically Learned Rewrite Patterns. In *Proceedings of Coling 2004*, pages 508–514, Geneva, Switzerland, August. COLING.

Hao Zhang and Daniel Gildea. 2007. Factorization of Synchronous Context-Free Grammars in Linear Time. In *NAACL Workshop on Syntax and Structure in Statistical Translation (SSST)*, pages 25–32.

Hao Zhang, Daniel Gildea, and David Chiang. 2008. Extracting Synchronous Grammar Rules From Word-Level Alignments in Linear Time. In *Proceedings of the 22nd International Conference on Computational Linguistics (COLING-08)*, pages 1081–1088, Manchester, UK.

Syntax-based Rewriting for Simultaneous Machine Translation

He He
Computer Science
University of Maryland
hhe@cs.umd.edu

Alvin Grissom II, Jordan Boyd-Graber
Computer Science
University of Colorado
{Alvin.Grissom, Jordan.Boyd.Graber}@colorado.edu

Hal Daumé III
Computer Science and UMIACS
University of Maryland
hal@cs.umd.edu

Abstract

Divergent word order between languages causes delay in simultaneous machine translation. We present a sentence rewriting method that generates more monotonic translations to improve the speed-accuracy tradeoff. We design grammaticality and meaning-preserving syntactic transformation rules that operate on constituent parse trees. We apply the rules to reference translations to make their word order closer to the source language word order. On Japanese-English translation (two languages with substantially different structure), incorporating the rewritten, more monotonic reference translation into a phrase-based machine translation system enables better translations faster than a baseline system that only uses gold reference translations.

1 Introduction

Simultaneous interpretation is challenging because it demands both quality and speed. Conventional **batch** translation waits until the entire sentence is completed before starting to translate. This merely optimizes translation quality and often introduces undesirable lag between the speaker and the audience. Simultaneous interpretation instead requires a tradeoff between quality and speed. A common strategy is to translate independently translatable segments as soon as possible. Various segmentation methods (Fujita et al., 2013; Oda et al., 2014) reduce translation delay; they are limited, however, by the unavoidable word reordering between languages with drastically different word orders. We show an example of Japanese-English translation in Figure 1. Consider the **batch** translation: in English, the verb *change* comes immediately after the subject *We*, whereas in Japanese it comes at the end of the sentence; therefore, to produce an intelligible English sentence, we must translate the object after the final verb is observed, resulting in one large and painfully delayed segment.

To reduce structural discrepancy, we can apply syntactic transformations to make the word order of one language closer to the other. Consider the **monotone** translation in Figure 1. By passivizing the English sentence, we can cache the subject and begin translating before observing the final verb. Furthermore, by using the English possessive, we mimic the order of the Japanese genitive construction. These transformations enable us to divide the input into shorter segments, thus reducing translation delay.

To produce such monotone translations, a straightforward approach is to incorporate interpretation data into the learning of a machine translation (MT) system, because human interpreters use a variety of strategies (Shimizu et al., 2014; Camayd-Freixas, 2011; Tohyama and Matsubara, 2006) to fine-tune the word order. Shimizu et al. (2013) shows that this approach improves the speed-accuracy tradeoff. However, existing parallel simultaneous interpretation corpora (Shimizu et al., 2014; Matsubara et al., 2002; Bendazzoli and Sandrelli, 2005) are often small, and collecting new data is expensive due to the inherent costs of recording and transcribing speeches (Paulik and Waibel, 2010). In addition, due to the intense time pressure during interpretation, human interpretation has the disadvantage of simpler, less precise diction (Camayd-Freixas, 2011; Al-Khanji et al., 2000) compared to human translations done at the translator's leisure, allowing for more introspection and precise word choice.

We aim to address the data scarcity problem and combine translators' lexical precision and interpreters' syntactic flexibility. We propose to **rewrite the reference translation** in a way that uses the original lexicon, obeys standard grammar rules of

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 55–64,
Lisbon, Portugal, 17-21 September 2015.

Source:	我々は 政府の 構造 や 組織を 変更 すべき だ
	We-TOP **government**-GEN structure and composition-ACC **change** should COP
Batch:	我々は ‖ 政府の構造や組織を変更すべき だ
	We should **change** the structure and composition **of the government**
Monotone:	我々は ‖ 政府の ‖ 構造や組織を ‖ 変更すべきだ
	the government's structure and composition should **be changed** by us

Figure 1: Divergent word order between language pairs can cause long delays in simultaneous translation: Segments (||) mark the portions of the sentence that can be translated together. (Case markers: topic (TOP), genitive (GEN), accusative (ACC), copula (COP).)

the target language, preserves the original semantics, and yields more monotonic translations. We then train the MT system with the rewritten references so that it learns how to produce low-latency translations from the data. A data-driven approach to learning these rewriting rules is hampered by the dearth of parallel data: we have few examples of text that have been both interpreted and translated. Therefore, we design syntactic transformation rules based on linguistic analysis of the source and the target languages. We apply these rules to parsed text and decide whether to accept the rewritten sentence based on the amount of delay reduction. In this work, we focus on Japanese to English translation, because (i) Japanese and English have significantly different word orders (SOV vs. SVO); and consequently, (ii) the syntactic constituents required earlier by an English sentence often come late in the corresponding Japanese sentence.

We evaluate our approach using standard machine translation data (the Reuters newsfeed Japanese-English corpus) in a simultaneous translation setting. Our experimental results show that including the rewritten references into the learning of a phrase-based MT system results in a better speed-accuracy tradeoff against both the original and the rewritten reference translations.

2 The Problem of Delay Reduction

Simultaneous interpretation has two goals: producing good translations *and* producing them promptly. However, most existing parallel corpora and MT systems do not address the issue of **delay** during translation. We explicitly adapt the training data by rewriting rules to reduce delay. We first define translation delay and describe—in general terms—our rewriting rules. In the next section, we describe the rules in more detail.

While we are motivated by real-time interpretation, to simplify our problem, we assume that we have perfect text input. Given this constraint, a typical simultaneous interpretation system (Sridhar et al., 2013; Fujita et al., 2013; Oda et al., 2014) produces partial translations of consecutive segments in the source sentence and concatenates them to produce a complete translation. We define the *translation delay* of a sentence as the average number of tokens the system has to observe between translation of two consecutive segments (denoted by # words/seg).[1] For instance, the minimum delay of 1 word/seg is achieved when we translate immediately upon hearing a word. At test time, when the input is segmented, the delay is the average segment length. During the data preprocessing step of rewriting, we calculate delay from word alignments (Section 4).

Given a reference batch translation x, we apply a set of rewriting rules $\mathcal{R}$ to x to minimize its delay. A rewriting rule $r \in \mathcal{R}$ is a mapping that takes the constituent parse tree of x as input and outputs a modified parse tree, which specifies a rewritten sentence x'. The tree-editing operation includes node deletion, insertion, and swapping, as well as induced changes of word form and node label. A valid transformation rule should rearrange constituents in x to follow the word order of the input sentence as closely as possible, subject to grammatical constraints and preservation of the original meaning.

[1] Ideally, delay should be based on time lapse. However, timestamping is not applicable to typical MT corpus, therefore we approximate it by number of tokens and ignore decoding time.

3 Transformation Rules

We design a variety of syntactic transformation rules for Japanese-English translation motivated by their structural differences. Our rules cover verb, noun, and clause reordering. While we specifically focus on Japanese to English, many rules are broadly applicable to SOV to SVO languages.

3.1 Verb Phrases

The most significant difference between Japanese and English is that the head of a verb phrase comes at the end of Japanese sentences. In English, it occupies one of the initial positions. We now introduce rules that can postpone a head verb.

Passivization and Activization In Japanese, the standard structure of a sentence is $\texttt{NP}_1$ $\texttt{NP}_2$ verb, where case markers following the verb indicate the voice of the sentence. However, in English, we have $\texttt{NP}_1$ verb $\texttt{NP}_2$, where the form of the verb indicates its voice. Changing the voice is particularly useful when $\texttt{NP}_2$ (object in an active-voice sentence and subject in a passive-voice sentence) is long. By reversing positions of verb and $\texttt{NP}_2$, we are not held back by the upcoming verb and can start to translate $\texttt{NP}_2$ immediately. Figure 1 shows an example in which passive voice can help make the target and source word orders more compatible, but it is not the case that passivizing every sentence would be a good idea; sometimes making a passive sentence active makes the word orders more compatible if the objects are relatively short:

O: The talk was **denied** by the boycott group spokesman.
R: The boycott group spokesman **denied** the talk.

Quotative Verbs *Quotative* verbs are verbs that, syntactically and semantically, resemble *said* and often start an independent clause. Such verbs are frequent, especially in news, and can be moved to the end of a sentence:

O: **They announced** that the president will restructure the division.
R: The president will restructure the division, **they announced**.

In addition to quotative verbs, candidates typically include factive (e.g., *know*, *realize*, *observe*), factive-like (e.g., *announce*, *determine*), belief (e.g., *believe*, *think*, *suspect*), and antifactive (e.g., *doubt*, *deny*) verbs. When these verbs are followed by a clause (S or SBAR), we move the verb and its subject to the end of the clause.

While some exploratory work automatically extracts factive verbs, to our knowledge, an exhaustive list does not exist. To obtain a list with reasonable coverage, we exploit the fact that Japanese has an unambiguous quotative particle, *to*, that precedes such verbs.[2] We identify all of the verbs in the Kyoto corpus (Neubig, 2011) marked by the quotative particle and translate them into English. We then use these as our quotative verbs.[3]

3.2 Noun Phrases

Another difference between Japanese and English lies in the order of adjectives and the nouns they modify. We identify two situations where we can take advantage of the flexibility of English grammar to favor sentence structures consistent with positions of nouns in Japanese.

Genitive Reordering In Japanese, genitive constructions always occur in the form of X *no* Y, where Y belongs to X. In English, however, the order may be reversed through the *of* construction. Therefore, we transform constructions $\texttt{NP}_1$ of $\texttt{NP}_2$ to possessives using the apostrophe-s, $\texttt{NP}_2$'(s) $\texttt{NP}_1$ (Figure 1). We use simple heuristics to decide if such a transformation is valid. For example, when X / Y contains proper nouns (e.g., *the City of New York*), numbers (e.g., *seven pounds of sugar*), or pronouns (e.g., *most of them*), changing them to the possessive case is not legal.

***that* Clause** In English, clauses are often modified through a pleonastic pronoun. E.g., *It is* ADJP *to/that* SBAR/S. In Japanese, however, the subject (clause) is usually put at the beginning. To be consistent with the Japanese word order, we move the modified clause to the start of the sentence: *To* S/SBAR *is* ADJP. The rewritten English sentence is still grammatical, although its structure is less frequent in common English usage. For example,

O: **It is important** to remain watchful.
R: To remain watchful **is important**.

[2] We use a morphological analyzer to distinguish between the conjunction and quotative particles. Examples of words marked by this particle include 見られる (*expect*), 言う (*say*), 思われる (*seem*), する (*assume*), 信じる (*believe*) and so on.

[3] We also include the phrase *It looks like*.

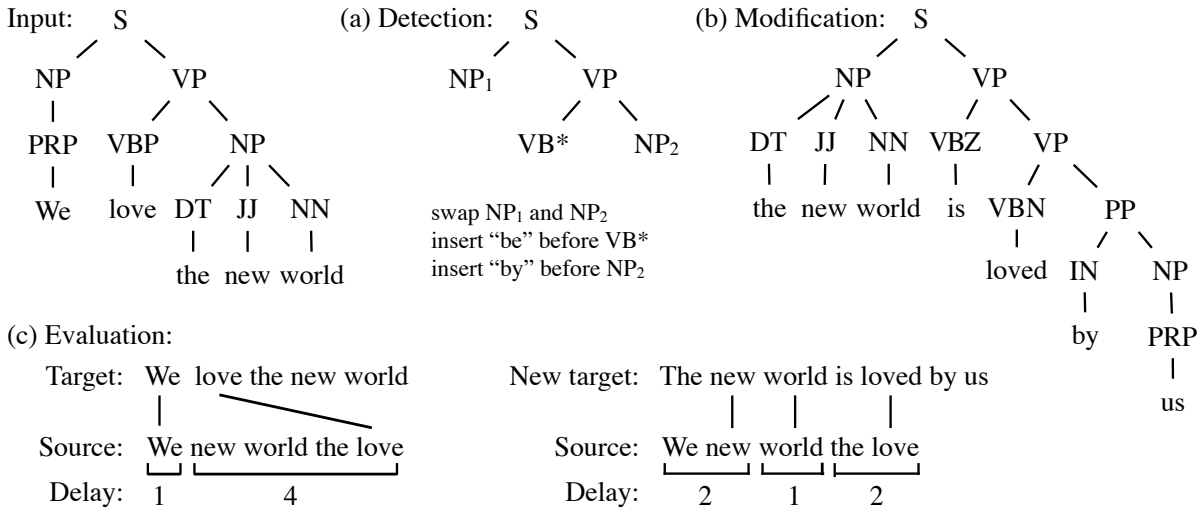


Figure 2: An example of applying the passivization rule to create a translation reference that is more monotonic.

3.3 Conjunction Clause

In Japanese, clausal conjunctions are often marked at the end of the initial clause of a compound sentence. In English, however, the order of clauses is more flexible. We can therefore reduce delay by reordering the English clauses to mirror how they typically appear in Japanese. Below we describe rules reversing the order of clauses connected by these conjunctions:

- Clausal conjunctions: *because (of)*, *in order to*
- Contrastive conjunctions: *despite*, *even though*, *although*
- Conditionals: *(even) if*, *as a result (of)*
- Misc: *according to*

In standard Japanese, such conjunctions include *no de*, *kara*, *de mo* and so on. The sentence often appears in the form of $\texttt{S}_2$ `conj`, $\texttt{S}_1$. In English, however, two common constructions are

> $\texttt{S}_1$ `conj` $\texttt{S}_2$: We should march **because** winter is coming.
> `conj` $\texttt{S}_2$, $\texttt{S}_1$: **Because** winter is coming, we should march.

To follow the Japanese clause order, we adapt the above two constructions to

> $\texttt{S}_2$, `conj'` $\texttt{S}_1$: Winter is coming, **because of this**, we should march.

Here `conj'` represents the original conjunction word appended with simple pronouns/phrases to refer to $\texttt{S}_2$. For example, *because* → *because of this*, *even if* → *even if this is the case*.

4 Sentence Rewriting Process

We now turn our attention to the implementation of the syntactic transformation rules described above. Applying a transformation consists of three steps:

1. **Detection:** Identify nodes in the parse tree for which the transformation is applicable;
2. **Modification:** Transform nodes and labels;
3. **Evaluation:** Compute delay reduction, and decide whether to accept the rewritten sentence.

Figure 2 illustrates the process using passivization as an example. In the detection step, we find the subtree that satisfies the condition of applying a rule. In this case, we look for an `S` node whose children include an `NP` (denoted by $\texttt{NP}_1$), the subject, and a `VP` to its right, such that the `VP` node has a leaf `VB*`, the main verb,[4] followed by another `NP` (denoted by $\texttt{NP}_2$), the object. We allow the parent nodes (`S` and `VP`) to have additional children besides the matched ones. They are not affected during the transformation. In the modification step, we swap the subject node and object node; we add the verb *be* in its correct form by checking the tense of the verb and the form of $\texttt{NP}_2$;[5]and we add the preposition *by* before the subject. The process is executed recursively throughout the parse tree.

[4]The main verb excludes *be* and *have* when it indicates tense (e.g., *have done*).

[5]We use the Nodebox linguistic library (`https://www.nodebox.net/code`) to detect and modify word forms.

Although our rules are designed to minimize long range reordering, there are exceptions.[6] Thus applying a rule does not always reduce delay. In the evaluation step, we compare translation delay before and after applying the rule. We accept a rewritten sentence if its delay is reduced; otherwise, we revert to the input sentence. Since we do not segment sentences during rewriting, we must estimate the delay.

To estimate the delay, we use word alignments. Figure 2c shows the source Japanese sentence in its word-for-word English translation and alignments from the target words to the source words. The first English word, *We*, is aligned to the first Japanese word; it can thus be treated as an independent segment and translated immediately. The second English word, *love*, is aligned to the last Japanese word, which means the system cannot start to translate until four more Japanese words are revealed. This alignment therefore forms a segment with delay of four words/seg. Alignments of the following words come before the source word aligned to *love*; hence, they are already translated in the previous segment and we do not double count their delay. In this example, the delay of the original sentence is 2.5 word/seg; after rewriting, it is reduced to 1.7 word/seg. Therefore, we accept the rewritten sentence. However, when the subject phrase is long and the object phrase is short, a swap may not reduce delay.

We can now formally define the **delay**. Let e_i be the ith target word in the input sentence x and a_i be the maximum index among indices of source words that e_i aligned to. We define the delay of e_i as $d_i = \max(0, a_i - \max_{j<i} a_j)$. The delay of x is then $\sum_{i=1}^{N} d_i/N$, where the sum is over all aligned words except punctuation and stopwords.

Given a set of rules, we need to decide which rules to apply and in what order. Fortunately, our rules have little interaction with each other, and the order of application has a negligible effect. We apply the rules, roughly, sequentially in order of complexity: if the output of current rule is not accepted, the sentence is reverted to the last accepted version.

	Train	Tune	Test
Ja	21.3M	30.2k	23.3k
En-GD	16.8M	23.8k	18.5k
En-RW	16.8M	24.1k	18.7k

Table 1: Number of words in the training, tuning, and test datasets. En-GD and En-RW represent the gold reference set and the rewritten reference set.

5 Experiments

We evaluate our method on the Reuters Japanese-English corpus of news articles (Utiyama and Isahara, 2003). For training the MT system, we also include the EIJIRO dictionary entries and the accompanying example sentences.[7] Statistics of the dataset are shown in Table 1. The rewritten translation is generally slightly longer than the gold translation because our rewriting often involves inserting pronouns (e.g. *it*, *this*) for antecedents.

We use the `TreebankWordTokenizer` from NLTK (Bird et al., 2009) to tokenize English sentences and Kuromoji Japanese morphological analyzer[8] to tokenize Japanese sentences. Our phrase-based MT system is trained by Moses (Koehn et al., 2003) with standard parameters settings. We use GIZA++ (Och and Ney, 2003) for word alignment and k-best batch MIRA (Cherry and Foster, 2012) for tuning. The translation quality is evaluated by BLEU (Papineni et al., 2002) and RIBES (Isozaki et al., 2010).[9] To obtain the parse trees for English sentences, we use the Stanford Parser (Klein and Manning, 2003) and the included English model.

5.1 Quality of Rewritten Translations

After applying the rewriting rules (Section 4), Table 2 shows the percentage of sentences that are candidates and how many rewrites are accepted. The most generalizable rules are passivization and delaying quotative verbs. We rewrite 32.2% of sentences, reducing the delay from 9.9 words/seg to 6.3 words/seg per segment for rewritten sentences and from 7.8 words/seg to 6.7 words/seg overall.

[6] For example, in clause transformation, the Japanese conjunction *moshi*, which is clause initial, may appear at the beginning of a sentence to emphasize conditionals, although its appearance is relatively rare.

[7] Available at `http://eijiro.jp`

[8] Available at `http://www.atilika.org/`

[9] In contrast to BLEU, RIBES is an order-sensitive metric commonly used for translation between Japanese and English.

	verb	voice	noun	conj.
Applicable %	39.9	50.0	26.4	4.8
Accepted %	22.5	24.0	51.2	38.4

Table 2: Percentage of sentences that each rule category can be applied to (Applicable) and the percentage of sentences for which the rule results in a more monotonic sentence (Accepted).

We evaluate the quality of our rewritten sentences from two perspectives: grammaticality and preserved semantics. To examine how close the rewritten sentences are to standard English, we train a 5-gram language model using the English data from the Europarl corpus, consisting of 46 million words, and use it to compute perplexity. Rewriting references increases the perplexity under the language model only slightly: from 332.0 to 335.4. To ensure that rewrites leave meaning unchanged, we use the SEMAFOR semantic role labeler (Das et al., 2014) on the original and modified sentence; for each role-labeled token in the reference sentence, we examine its corresponding role in the rewritten sentence and calculate the average accuracy acrosss all sentences. Even ignoring benign lexical changes—for example, *he* becoming *him* in a passivized sentence—95.5% of the words retain their semantic roles in the rewritten sentences.

Although our rules are conservative to minimize corruption, some errors are unavoidable propagation of parser errors. For example, the sentence *the London Stock Exchange closes at 1230 GMT today* is parsed as:[10]

(S (NP *the London Stock Exchange*)
(VP (VBZ *closes*)
(PP *at 1230*)
(NP *GMT today*)))

GMT today is separated from the PP as an NP and is mistaken as the object. The passive version is then *GMT today is closed at 1230 by the London Stock Exchange*. Such errors could be reduced by skipping nodes with low inside/outside scores given by the parser, or skipping low-frequency patterns. However, we leave this for future work.

5.2 Segmentation

At test time, we use right probability (Fujita et al., 2013, RP) to decide when to start translating a sentence. As we read in the source Japanese sentence, if the input segment matches an entry in the learned phrase table, we query the RP of the Japanese/English phrase pair. A higher RP indicates that the English translation of this Japanese phrase will likely be followed by the translation of the next Japanese phrase. In other words, translation of the two consecutive Japanese phrases is monotonic, thus, we can begin translating immediately. Following (Fujita et al., 2013), if the RP of the current phrase is lower than a fixed threshold, we cache the current phrase and wait for more words from the source sentence; otherwise, we translate all cached phrases. Finally, translations of segments are concatenated to form a complete translation of the input sentence.

5.3 Speed/Accuracy Trade-off

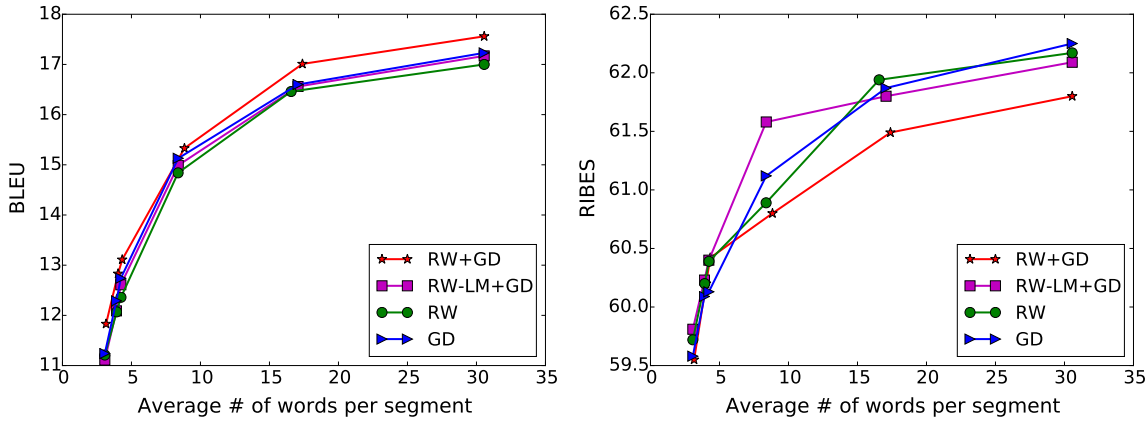
To show the effect of rewritten references, we compare the following MT systems:

- **GD:** only gold reference translations;
- **RW:** only rewritten reference translations;
- **RW+GD:** both gold and the rewritten references; and
- **RW-LM+GD:** using gold reference translations but using the rewritten references for training the LM and for tuning.

For RW+GD and RW-LM+GD, we interpolate the language models of GD and RW. The interpolating weight is tuned with the rewritten sentences. For RW+GD, we combine the translation models (phrase tables and reordering tables) of RW and GD by fill-up combination (Bisazza et al., 2011), where all entries in the tables of RW are preserved and entries from the tables of GD are added if new.

Increasing the RP threshold increases interpretation delay but improves the quality of the translation. We set the RP threshold at 0.0, 0.2, 0.4, 0.8 and finally 1.0 (equivalent to batch translation). Figure 3 shows the BLEU/RIBES scores vs. the number of words per segement as we increase the threshold. Rewritten sentences alone do not significantly improve over the baseline. We suspect this is because the transformation rules sometimes generate ungrammatical sentences due to parsing errors, which impairs learning. However, combining RW and GD results in a better speed-accuracy tradeoff: the RW+GD curve completely dominates other curves in Figure 3a, 3c. Thus, using more monotone translations improves simultaneous machine translation, and because RW-LM+GD is about

[10] For simplicity we show the shallow parse only.



(a) BLEU w.r.t. gold ref

(b) RIBES w.r.t. gold ref

(c) BLEU w.r.t. rewritten ref

(d) RIBES w.r.t. rewritten ref

Figure 3: Speed/accuracy tradeoff curves: BLEU (left) / RIBES (right) versus translation delay (average number of words per segment).

the same as GD, the major improvement likely comes from the translation model from rewritten sentences.

The right two plots recapitulate the evaluation with the RIBES metric. This result is less clear, as MT systems are optimized for BLEU and RIBES penalizes word reordering, making it difficult to compare systems that intentionally change word order. Nevertheless, RW is comparable to GD on gold references and superior to the baseline on rewritten references.

5.4 Effect on Verbs

Rewriting training data not only creates lower latency simultaneous translations, but it also improves *batch* translation. One reason is that SOV to SVO translation often drops the verb because of long range reordering. (We see this for Japanese here, but this is also true for German.) Similar word orders in the source and target results in less reordering and improves phrase-based MT (Collins et al., 2005; Xu et al., 2009). Table 3 shows the number of verbs in the translations of the test sentences produced by GD, RW, RW+GD, as well as the number in the gold reference translation. Both RW and RW+GD produce more verbs (a statistically significant result), although RW+GD captures the most verbs.

	Translation			
	GD	RW	RW+GD	Gold ref
# of verbs	1971	2050	**2224**	2731

Table 3: Number of verbs in the test set translation produced by different models and the gold reference translation. Boldface indicates the number is significantly larger than others (excluding the gold ref) according to two-sample t-tests with $p < 0.001$.

Ref	**he also said** that the real dangers for the euro lay in the potential for divergences in the domestic policy needs among the various participating nations of the single currency.
GD	**he** also for the euro, is a real danger to launch a single currency in many different countries and domestic policies on the need for the possibility of a difference.
RW	**he** also for the euro is a real danger to launch a single currency in many different countries and domestic policies to the needs of the possibility of a difference, **he said**.

Table 4: Example of translation produced by GD and RW.

5.5 Error Analysis

Table 4 compares translations by GD and RW. RW correctly puts the verb *said* at the end, while GD drops the final verb. However, RW still produces *he* at the beginning (also the first word in the Japanese source sentence). This is because our current segmentation strategy do not preserve words for *later* translation—a note-taking strategy used by human interpreters.

6 Related Work

Previous approaches to simultaneous machine translation have employed explicit interpretation strategies for coping with delay. Two major approaches are segmentation and prediction.

Most segmentation strategies are based on heuristics, such as pauses in speech (Fügen et al., 2007; Bangalore et al., 2009), comma prediction (Sridhar et al., 2013) and phrase reordering probability (Fujita et al., 2013). Learning-based methods have also been proposed. Oda et al. (2014) find segmentations that maximize the BLEU score of the final concatenated translation by dynamic programming. Grissom II et al. (2014) formulate simultaneous translation as a sequential decision making problem and uses reinforcement learning to decide when to translate. One limitation of these methods is that when learning with standard batch MT corpus, their gain can be restricted by natural word reordering between the source and the target sentences, as explained in Section 1.

In an SOV-SVO context, methods to predict unseen words are proposed to alleviate the above restriction. Matsubara et al. (1999) predict the English verb in the target sentence and integrates it syntactically. Grissom II et al. (2014) predict the final verb in the source sentence and decide when to use the predicted verb with reinforcement learning. Nevertheless, unless the predictor considers contextual and background information, which human interpreters often rely on for prediction (Hönig, 1997; Camayd-Freixas, 2011), such a prediction task is inherently hard.

Unlike previous approaches to simultaneous translation, we directly adapt the training data and transform a translated sentence to an "interpreted" one. We can, therefore, take advantage of the abundance of parallel batch-translated corpora for training a simultaneous MT system. In addition, as a data preprocessing step, our approach is orthogonal to the others, with which it can be easily combined.

This work is also related to preprocessing reordering approaches (Xu et al., 2009; Collins et al., 2005; Galley and Manning, 2008; Hoshino et al., 2013; Hoshino et al., 2014) in batch MT for language pairs with substantially different word orders. However, our problem is different in several ways. First, while the approaches resemble each other, our motivation is to reduce translation delay. Second, they reorder the source sentence, which is nontrivial and time-consuming when the sentence is incrementally revealed. Third, rewriting the target sentence requires the output to be grammatical (for it to be used as reference translation), which is not a concern when rewriting source sentences.

7 Conclusion

Training MT systems with more monotonic (interpretation-like) sentences improves the speed-accuracy tradeoff for simultaneous machine translation. By designing syntactic transformations and rewriting batch translations into more monotonic translations, we reduce the translation delay. MT systems trained on the rewritten reference translations learn interpretation strategies implicitly from the data.

Our rewrites are based on linguistic knowledge and inspired by techniques used by human interpreters. They cover a wide range of reordering phenomena between Japanese and English, and more generally, between SOV and SVO languages. A natural extension is to automatically extract such rules from parallel corpora. While there exist approaches that extract syntactic tree transformation rules automatically, one of the difficulties is that most parallel corpora is dominated by lexical paraphrasing instead of syntactic paraphrasing.

Acknowledgments

This work was supported by NSF grant IIS-1320538. Boyd-Graber is also partially supported by NSF grants CCF-1409287 and NCSE-1422492. Daumé III and He are also partially supported by NSF grant IIS-0964681. Any opinions, findings, conclusions, or recommendations expressed here are those of the authors and do not necessarily reflect the view of the sponsor.

References

Raja Al-Khanji, Said El-Shiyab, and Riyadh Hussein. 2000. On the use of compensatory strategies in simultaneous interpretation. *Journal des Traducteurs*, 45(3):548–577.

Srinivas Bangalore, Vivek Kumar Rangarajan Sridhar, Prakash Kolan, Ladan Golipour, and Aura Jimene. 2009. Real-time incremental speech-to-speech translation of dialogs. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Claudio Bendazzoli and Annalisa Sandrelli. 2005. An approach to corpus-based interpreting studies: Developing EPIC (european parliament interpreting corpus). In *Proceedings of Challenges of Multidimensional Translation*.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python*. O'Reilly Media.

Arianna Bisazza, Nick Ruiz, and Marcello Federico. 2011. Fill-up versus interpolation methods for phrase-based SMT adaptation. In *Proceedings of International Workshop on Spoken Language Translation (IWSLT)*.

Erik Camayd-Freixas. 2011. Cognitive theory of simultaneous interpreting and training. In *Proceedings of the 52nd Conference of the American Translators Association*.

Colin Cherry and George Foster. 2012. Batch tuning strategies for statistical machine translation. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Michael Collins, Philipp Koehn, and Ivona Kučerová. 2005. Clause restructuring for statistical machine translation. In *Proceedings of the annual meeting of the Association for Computational Linguistics (ACL)*.

Dipanjan Das, Desai Chen, André F. T. Martins, Nathan Schneider, , and Noah A. Smith. 2014. Frame-semantic parsing. *Computational Linguistics*, 40(1).

Christian Fügen, Alex Waibel, , and Muntsin Kolss. 2007. Simultaneous translation of lectures and speeches. *Machine Translation*, 21(4):209–252.

Tomoki Fujita, Graham Neubig, Sakriani Sakti, Tomoki Toda, and Satoshi Nakamura. 2013. Simple, lexicalized choice of translation timing for simultaneous speech translation. In *Proceedings of Interspeech*.

Michel Galley and Christopher D. Manning. 2008. A simple and effective hierarchical phrase reordering model. In *Proceedings of Empirical Methods in Natural Language Processing (EMNLP)*.

Alvin C. Grissom II, He He, Jordan Boyd-Graber, John Morgan, and Hal Daumé III. 2014. Don't until the final verb wait: Reinforcement learning for simultaneous machine translation. In *Proceedings of Empirical Methods in Natural Language Processing (EMNLP)*.

Hans G. Hönig. 1997. Using text mappings in teaching consecutive interpreting. *Translation and Translation Theory*, pages 19–34.

Sho Hoshino, Yusuke Miyao, Katsuhito Sudoh, and Masaaki Nagata. 2013. Two-stage pre-ordering for Japanese-to-English statistical machine translation. In *International Joint Conference on Artificial Intelligence (IJCNLP)*.

Sho Hoshino, Hubert Soyer, Yusuke Miyao, and Akiko Aizawa. 2014. Japanese to english machine translation using preordering and compositional distributed semantics. In *Workshop on Asian Translation*.

Hideki Isozaki, Tsutomu Hirao, Kevin Duh, Katsuhito Sudoh, and Hajime Tsukada. 2010. Automatic evaluation of translation quality for distant language pairs. In *Proceedings of Empirical Methods in Natural Language Processing (EMNLP)*.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the annual meeting of the Association for Computational Linguistics (ACL)*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Brooke Cowan Nicola Bertoldi, Wade Shen, Richard Zens Christine Moran, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbs. 2003. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the annual meeting of the Association for Computational Linguistics (ACL)*.

Shigeki Matsubara, Katsuhiko Toyama, and Yasuyoshi Inagaki. 1999. Sync/trans: Simultaneous machine interpretation between English and Japanese. In *Advanced Topics in Artificial Intelligence*, pages 134–143. Springer.

Shigeki Matsubara, Akira Takagi, Nobuo Kawaguchi, and Yasuyoshi Inagaki. 2002. Bilingual spoken

monologue corpus for simultaneous machine interpretation research. In *Proceedings of the Language Resources and Evaluation Conference (LREC)*.

Graham Neubig. 2011. The Kyoto free translation task. *Available online at http://www. phontron. com/kftt*.

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

Yusuke Oda, Graham Neubig, Sakriani Sakti, Tomoki Toda, and Satoshi Nakamura. 2014. Optimizing segmentation strategies for simultaneous speech translation. In *Proceedings of the Association for Computational Linguistics (ACL)*, June.

Kishore Papineni, Salim Roukos, Todd Ward, , and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the annual meeting of the Association for Computational Linguistics (ACL)*.

Matthias Paulik and Alex Waibel. 2010. Spoken language translation from parallel speech audio: Simultaneous interpretation as slt training data. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*.

Hiroaki Shimizu, Graham Neubig, Sakriani Sakti, Tomoki Toda, and Satoshi Nakamura. 2013. Constructing a speech translation system using simultaneous interpretation data. In *Proceedings of International Workshop on Spoken Language Translation (IWSLT)*.

Hiroaki Shimizu, Graham Neubig, Sakriani Sakti, Tomoki Toda, and Satoshi Nakamura. 2014. Collection of a simultaneous translation corpus for comparative analysis. In *International Language Resources and Evaluation (LREC)*.

Vivek Kumar Rangarajan Sridhar, John Chen, Srinivas Bangalore Andrej Ljolje, and Rathinavelu Chengalvarayan. 2013. Segmentation strategies for streaming speech translation. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Hitomi Tohyama and Shigeki Matsubara. 2006. Collection of simultaneous interpreting patterns by using bilingual spoken monologue corpus. In *Proceedings of the Language Resources and Evaluation Conference (LREC)*.

Masao Utiyama and Hitoshi Isahara. 2003. Reliable measures for aligning japanese-english news articles and sentences. In *Proceedings of the annual meeting of the Association for Computational Linguistics (ACL)*.

Peng Xu, Jaeho Kang, Michael Ringgaard, , and Franz Och. 2009. Using a dependency parser to improve SMT for subject-object-verb language. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Identifying Political Sentiment between Nation States with Social Media

Nathanael Chambers, Victor Bowen, Ethan Genco, Xisen Tian,
Eric Young, Ganesh Harihara, Eugene Yang
Department of Computer Science
United States Naval Academy
nchamber@usna.edu

Abstract

This paper describes an approach to large-scale modeling of sentiment analysis for the social sciences. The goal is to model relations between nation states through social media. Many cross-disciplinary applications of NLP involve making predictions (such as predicting political elections), but this paper instead focuses on a model that is applicable to broader analysis. Do citizens express opinions in line with their home country's formal relations? When opinions diverge over time, what is the cause and can social media serve to detect these changes? We describe several learning algorithms to study how the populace of a country discusses foreign nations on Twitter, ranging from state-of-the-art contextual sentiment analysis to some required practical learners that filter irrelevant tweets. We evaluate on standard sentiment evaluations, but we also show strong correlations with two public opinion polls and current international alliance relationships. We conclude with some political science use cases.

1 Introduction

The volume of text available on social media provides a new opportunity for public policy and political science. Specifically in the area of international relations, advances in natural language understanding and sentiment analysis may offer new insights into the sentiment of one nation toward another. This paper processes 17 months of Twitter data to identify discussions about sovereign states, and it aggregates opinions toward these states from foreign nations. We present a novel application of contextual sentiment with this task, and identify several semi-supervised learning algorithms that are needed to address the reference resolution challenge inherent to country names. We present intrinsic evaluations of our learners on labeled datasets as well as four extrinsic political science evaluations that show strong alignment with our large-scale sentiment extraction.

An open question for international policy makers is the extent to which public opinion drives decision making. How do military conflicts affect a neutral nation's relationship? Does public opinion shift toward a country after a formal alliance is created, or must popular opinion shift first? These questions are difficult to address due to the lack of measurable data. While polling data can be collected, collection beyond a handful of countries is cost prohibitive. This paper hypothesizes that sentiment analysis can be used as a proxy to track international relations between nation states. We describe the largest attempt (over 2 billion tweets) to measure nation state sentiment across hundreds of country pairs.

The core challenge to measuring public opinion between countries is an accurate algorithm to judge the sentiment of a text toward another nation. Unlike traditional sentiment analysis, the general sentiment of the text is not adequate. Let the following serve as an example.

> *I miss **Pakistan**. I am in full sad mode right about now.* @RachOrange (California)

This tweet is a *positive* example from the USA toward Pakistan. However, a typical sentiment classifier misclassifies this as negative because *miss* and *sad* express sadness. A *contextual* sentiment classification is needed to identify that the predicate *miss* is positive toward its argument. Several

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 65–75,
Lisbon, Portugal, 17-21 September 2015.

recent competitions included contextual classification tasks, and this paper builds on the best of those algorithms for a unique nation-nation sentiment classifier. We describe a multi-classifier model that aggregates tweets into counts of positive and negative sentiment from one country toward another. Several unique filters are required to resolve textual references toward country names.

We first present standard NLP sentiment experiments that show the classifiers achieve good performance on individual tweets. To evaluate the complete nation-nation system, we present four novel evaluations, including two public opinion polls. Correlation with the polls is high at $\rho = .8$, and our nation-nation sentiment is 84% accurate with NATO and EU relations. We then discuss the implications for both NLP as a technical science and political science as a social science.

2 Previous Work

Sentiment analysis is a large field applicable to many genres. This paper focuses on social media and contextual polarity, so we only address the closest work in those areas. For a broader perspective, several survey papers are available (Pang and Lee, 2008; Tang et al., 2009; Liu and Zhang, 2012; Tsytsarau and Palpanas, 2012).

Several sources for microblogs have been used to measure a large population's mood and opinion. O'Connor et al. (2010) used Twitter data to compute a ratio of positive and negative words to measure consumer confidence and presidential approval. Kramer (2010) counted lexicon words on Facebook for a general 'happiness' measure, and Thelwall (2011) built a general sentiment model on MySpace user comments. These are early general sentiment algorithms for social media.

Other microblog research focused on finding noisy training data with distant supervision. Many of these algorithms use emoticons as semantic indicators of polarity. For instance, a tweet that contains a sad face likely contains a negative polarity (Read, 2005; Go et al., 2009; Bifet and Frank, 2010; Pak and Paroubek, 2010; Davidov et al., 2010; Kouloumpis et al., 2011). In a similar vein, hashtags can serve as noisy labels (Davidov et al., 2010; Kouloumpis et al., 2011). Our bootstrap learner is similar in its selection of seed tokens.

Supervised learning for *contextual* polarity has received more attention recently. Jiang et al. (2011) is an early approach. Work on product reviews sought the sentiment toward particular product features. These systems used rule based models based on parts of speech and surface features (Nasukawa and Yi, 2003; Hu and Liu, 2004; Ding and Liu, 2007). Most notably, recent Semeval competitions addressed contextual polarity (Nakov et al., 2013; Rosenthal et al., 2014). The top performing systems learned their own lexicons custom to the domain (Mohammad et al., 2013; Zhu et al., 2014). Our proposed system includes many of their features, but several fail to help on nation-nation sentiment.

Early approaches to *topic detection* on social media were straightforward, selecting a keyword (e.g., "Obama") to represent the topic (e.g., "US President") and retrieving tweets containing the word (O'Connor et al., 2010; Tumasjan et al., 2010; Tan et al., 2011). These systems classify the polarity of *the entire tweet*, but ignore the question of polarity toward the particular topic. This paper focuses on identifying tweets with nation mentions, and identifying the sentiment toward the mention, not the overall sentiment of the text.

Event detection on Twitter is also relevant (Sakaki et al., 2010; Becker et al., 2011). In fact, O'Connor et al. (2013) modeled events to detect international relations, but our goal is to model long term relation trends, not isolated events.

Large-scale computational studies of social media are relatively new to the international relations community. Barbera and Rivero (2014) is a notable example for election analysis. Some studied online discussion about Palestine (Lynch, 2014) and the role of Twitter in the Arab Spring (Howard et al., 2011; Howard, 2013). However, they simply counted the volume of tweets containing keywords. This paper applies a deeper NLP analysis and we show that frequency alone fails at detecting nation-nation relations.

Most relevant to this paper is a study of Arabic tweets into anti-American sentiment. Jamal et al. (2015) used a supervised sentiment classifier on Arabic tweets to measure sentiment toward the USA. Our paper differs by taking a broader view. We investigate with state-of-the-art sentiment algorithms, and we we study practical problems that arise within when measuring nation-nation sentiment across *all* country pairs. To our knowledge, this paper is the largest computational approach (17 months with 2 billion tweets) to measuring international relations on social media.

3 Microblog Datasets

The main dataset for this study is 17 months of tweets obtained through the keyword Twitter API that mention one of 187 unique countries. The dataset spans from Sep. 3, 2013 to Jan 10, 2015 with 3-5 million tweets per day. Each tweet includes the profile location and geolocation data (if available) of the user who posted the tweet. Collection was not limited to a specific location in order to retrieve samples from across the world. This dataset is used in all political science experiments (Sections 6.2 and 6.3).

A smaller labeled dataset is used for supervised classification. We randomly sampled the data to create a dataset of 4250 tweets. The authors initially labeled each tweet with one of four sentiment labels: *positive*, *negative*, *objective*, or *irrelevant*. Text was only labeled as positive if it is positive toward the nation's mention. Text that contains a nation's mention, but does not contain sentiment toward the mention is labeled *objective*. Text with a mention that is *not* referent to a physical country is labeled *irrelevant* despite presence of sentiment. This irrelevant distinction is a departure from sentiment competitions. A second labeling added a fifth label to the *irrelevant* tweets to split off *dining* topics.

Usernames (e.g., @*user*) and URLs are replaced with placeholder tokens. Multiple whitespace characters are condensed and the text is split on it. Punctuation attached to tokens is removed (but saved) and used in later punctuation features. Punctuation is not treated as separate tokens in the n-gram features. We prepend occurrences of "not" to their subsequent tokens, merging the two into a new token (e.g., "not happy" becomes "not-happy"). Once the raw text of the tweet is tokenized as above, non-English tweets are filtered out. English filtering is performed by LingPipe[1]. We manually evaluated this filter and found it 86.2% accurate over 400 tweets. Accuracy is lost due to slang and the short nature of the text.

4 Classifying Nation-Nation Sentiment

Given a tweet containing a country's name, our goal is to identify the sentiment of the text toward that nation. Unlike most work on contextual polarity, this requires reference resolution of the target phrase (e.g., the country name). Previous Semeval competitions evaluate the sentiment of a text toward a phrase, but the semantics of the phrase is largely ignored. For instance, the following example would make an excellent Semeval test item, but its classification is irrelevant to the goal of measuring nation sentiment:

> *My daughter and I have been to Angelo's several times when in Little **Italy**. Love love it!*

The author is obviously positively inclined toward Little Italy, however, Little Italy does not refer to the country of Italy. We found that most tweets referring to dining or visiting foreign-themed restaurants are not relevant to determining nation to nation sentiment. It became necessary to research new classifiers that perform basic reference resolution.

4.1 Reference Resolution: Irrelevant Detection

This paper defines reference resolution in the traditional linguistic sense: determine the real-world referent of a text mention. Most NLP tasks use coreference resolution: determine the text antecedent of a text mention. This paper requires reference resolution because the target phrase often does not refer to an actual geolocated country. After collecting months of tweets that include country name mentions, data analysis revealed several types of these non-references. We treat reference resolution as a classification problem. Below are a variety of supervised and semi-supervised learners that identify different types of errant country references, and ultimately serve to filter out these irrelevant tweets.

4.1.1 Dining Classifier

One of our early observations was that mentions of nations are often in the context of eating and dining, as evidenced here:

> *This is the first **turkey** sandwich I've had in awhile... It's great **turkey**.*

> *Taste of **China** For chinese food lover's. For more info Please visit*

This class of tweet is problematic to our study of international politics. While microblogs about dining can contain heavy emotion, a link to the writer's opinion about the foreign nation itself is ambiguous. We thus filter out dining text through supervised classification. Using the labeled dataset in Section 3, we annotated a *dine* label for all dining tweets. Tweets without a *dine*

[1] *alias-i.com/lingpipe/#lingpipe*

	Dine	Rel
All unigrams in text	✓	✓
1-3grams that include the country	✓	✓
Bigram and Trigram country pattern	✓	✓
Four Square app pattern		✓
Named Entity 2-3grams w/ country		✓
Emoticon happy or sad		✓
Ending text punctuation		✓
Binary: contains exclamation point		✓

Table 1: Dining and Relevant features.

label are considered *not-dine*. We ran a logistic regression for two labels, *dine* and *not-dine*. Text features are shown in Table 1.

4.1.2 Irrelevancy Classifier

Beyond dining, a broader class of irrelevant tweets refer to non-nation entities. These microblogs contain country names, but the mentions do not reference the physical country. The following examples illustrate this class of *irrelevant* tweets (nation tokens in bold):

> *Yesterday was chilly out and now today's going to be 80. New **England** weather is so bipolar I hate it so much*

> *Bank Of **America** Upgrades ConocoPhillips On More Favorable Outlook*

Several types of irrelevancy can be found, but the most common is a non-nation geolocation like *New England*. Proper nouns like *Bank of America* are frequent as well. A named entity recognizer (NER) identified some of these, but we ultimately turned to supervised classification for better accuracy (space constraints prevent discussion of NER performance). We trained a logistic regression classifier on the **relevant** tweets in the Section 3 dataset, and mapped all other labels to **irrelevant**. Features used are shown in Table 1.

4.1.3 Bootstrap Learner

After filtering non-referent tweets, we observed that many *positive* and *negative* tweets reference countries in the context of sporting events and music/concerts. These are correctly labeled **relevant** by the above binary classifiers (and possibly annotated as positive or negative), but the topic (sports or music) does not contain a strong semantic connection to the author's actual opinion about the country. A couple of sport examples are given here:

> *@SpecialKBrook Wow - the Brittish judge scored the fight a draw - lucky **England**'s fighters are better than their judges.*

> ***Congo** LFC now someone give me that goalie's jersey :p*

The sport topic has a less diverse vocabulary than other topics. We hypothesized that a bootstrap learning framework (Riloff and Jones, 1999) could quickly learn its unique language without the need for supervised learning. Beginning with a short list of sport keywords (*football, basketball, baseball, cricket, soccer, golf, hockey, rugby, game, vs*), we ran two iterations of a bootstrapped learner. The first step retrieves tweets containing one of the keywords. The second counts token occurrences in this set and computes pointwise mutual information (PMI) scores for each unigram by comparing with the unigram counts over the entire corpus. The learner processed ˜190 million tweets (a couple months of data). The PMI scores from this process then form the basis of a simple topic classifier.

A tweet is classified as a topic (e.g., sports) if its average token PMI score is above a learned threshold for that topic:

$$score_T(text) = \frac{1}{N} \sum_{w \in text} pmi_T(w) \qquad (1)$$

where N is the number of tokens in the text and $T \in \{sports, concerts\}$. The text is classified as in topic if $score_T(text) > \lambda_t$. The threshold λ_T was determined by visual inspection of a held out 1000 tweets to maximize accuracy. The initial seed words and λ_T thresholds for each topic are given here:

Seed Words	λ
football, basketball, baseball, cricket, soccer, golf, hockey, rugby, game, vs	0.08
concert, music, album, song, playlist, stage, drum	0.15

4.2 Contextual Sentiment Analysis

The above classifiers identify *relevant* tweets with *references* to geolocated nations. Approximately 21% are filtered out, leaving 79% for the remaining component of this paper: contextual sentiment analysis. Contextual sentiment analysis focuses on the disposition of text toward a word or phrase (in this case, a country's name). Most data-driven approaches rely on labeled corpora to drive the learning process, and this paper is no different.

Assigning polarity to a word/phrase requires features that capture the surrounding context. The following tweets are examples of context with strong polarity toward the country in bold.

*RT @ChrissyCostanza: Happiest girl ever. I LOVE YOU **SINGAPORE***

*there's no **Singapore** Got Talent cus the only talent we have is stomping complaining & staring*

Singapore is the target country here. The first tweet is overtly positive toward it, but the second requires a more subtle interpretation. The negative context is toward *us*, referencing the people of the Singapore anaphor. It seems reasonable to infer that they are negative toward the country as a whole, but a deeper reasoning is required to make the connection. These difficult decisions require a wide-range of lexical features. We build on the top performing features from contextual polarity systems in Semeval 2013 and 2014 (Mohammad et al., 2013; Zhu et al., 2014). We used the following set of features to capture these different contexts:

Token Features: All unigrams and bigrams.

Target Patterns: This feature creates patterns from n-grams that include the target word. The target is replaced with a variable to capture generalized patterns. For instance, "to France last" becomes "to X last". Bigram and trigram patterns are created.

Punctuation: End of sentence punctuation and punctuation attached to target words. Prefix and postfix punctuation are separate features.

Emoticons: Two binary features for the presence/absence of smiley and sad face emoticons.

Hand-Built Dictionary: Two binary features, *postivemood* and *negativemood*, indicate if a token appears in a sentiment lexicon's positive or negative list. We use Bing Liu's Opinion Lexicon[2].

Nation-Nation Learned Dictionary: Following the success of Zhu et al. (2014), we learn a mood dictionary from our domain-specific nation dataset. We count unigrams (bigrams did not improve performance) in one year of unfiltered tweets with nation mentions that contain an emoticon. Using these counts, each unigram computes its PMI scores toward happy and sad contexts. We construct features based on these PMI scores: (1) the highest happy PMI score of all unigrams in a tweet, (2) the highest sad PMI score, (3) the number of positive tokens, (4) the number of negative tokens, and (5) the sum of the token PMI differences between happy-sad.

[2] *http://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html*

General Learned Dictionary: We computed the same features as in the above learned dictionary, but instead counted tokens in all tweets of the general emoticon corpus of Go et al. (2009).

The contextual sentiment learner is trained on the labeled dataset (Section 3). Only tweets with *positive*, *negative*, or *objective* labels are included (irrelevant and dining are ignored). Stanford's CoreNLP (nlp.stanford.edu/software) is used to train a MaxEnt classifier with its default settings.

5 Nation to Nation Pipeline

The complete system to determine the nation-nation sentiment of a tweet consists of 3 steps: (1) identify the country origin of the tweet, (2) filter out tweets without references to geolocated nations and filter out irrelevant topics, and (3) identify the sentiment toward the country. We processed 17 months of tweets (Section 3).

The first step identifies the origin of the tweet with either its GPS coordinates or the profile location of the Twitter user. Profile locations are mapped to countries with an exhaustive list of country names, major cities, and patterns that match US city/states (e.g., *Pensacola,FL* maps to *USA*). Non-english tweets are removed with lingpipe. The second step filters non-referent, irrelevant, dining, and concert tweets with the classifiers from Section 4.1 (about 21% of tweets at this stage). The final step is the contextual sentiment classifier (Section 4.2). Tweets that make it through receive 1 of 3 possible labels: positive, negative, objective.

The aggregate counts of the three labels are collected for each day. This was around 1.2 million nation labels per day over 17 months. The counts are used for evaluation in the experiments.

6 Experiments

Our goal is to first prove the accuracy of our sentiment classifiers, then show the broader pipeline's correlation with known nation-nation politics.We thus conducted three types of experiments. The first is an intrinsic evaluation of the classifiers with common frameworks from the NLP community. The second is an extrinsic evaluation from multiple political science datasets. The third is a set of use case proposals for application of this analysis.

Dining Classifier

Label	P	R	F1
dining	.76	.48	.59
not-dining	.96	.99	.98
Baseline Accuracy		93.1%	
Accuracy		95.3%	

Irrelevant Classifier

Label	Prec	Recall	F1
irrelevant	.84	.90	.87
relevant	.84	.75	.80
Baseline Accuracy		58.7%	
Accuracy		84.0%	

Sentiment Classifier

Label	Prec	Recall	F1
positive	.64	.47	.54
negative	.60	.33	.43
objective	.71	.87	.78
Baseline Accuracy		59.0%	
Accuracy		68.7%	

Table 2: Classifier performance. Precision/Recall is calculated for each label separately. Accuracy is over all labels: # correct/total.

6.1 Classifier Experiments

The dining, irrelevant, and sentiment classifiers are supervised systems trained on a labeled dataset of 4,250 tweets. We split the dataset into training, dev, and test sets. The dev set contains 200 tweets, the test set has 750 tweets, and the training set size varied based on the available labels. The features in this paper were developed solely on the training and dev datasets. Reported results are on the unseen test set of 750 tweets. The bootstrapped classifiers for sports and concerts were learned without labeled data, so we ran the sports and concerts classifiers on an unseen portion of our data, and manually evaluated the first 200 tweets that were labeled by each classifier.

Precision and recall are calculated individually for each class label: $P = \#correct/\#guessed$ and $R = \#correct/\#gold$. Where $\#guessed$ is how many times the classifier predicted the target label, and $\#gold$ is how many times the target label appears in the dataset. Accuracy is also shown, calculated as a single score over all labels together: $Accuracy = \#correct/N$. The first table in Table 2 shows the dining classifier's performance. The majority class baseline is high at 93% because only 7% of the data is about dining. The classifier achieves a 29% decrease in accuracy error (2% absolute increase). The second table shows the more general irrelevant classifier. The majority class baseline is much lower than dining at 58.7%. Many tweets that contain a country name are not relevant nor references to the geolocated country itself. Our trained classifier does well on this task achieving 84% accuracy, a 26% absolute increase over baseline. It is 84% precise with 90% recall on detecting irrelevant tweets. The third table in Table 2 shows sentiment classifier results. Accuracy is almost 10% absolute above the majority class.

Finally, the bootstrapped classifiers perform at 98% accuracy for sports and 90% for concerts.

Positive/Negative Ratios

Target	Ratio	Target	Ratio
US to Canada	11.9	US to Ireland	3.2
US to Italy	10.8	US to Spain	3.0
US to Japan	7.7	US to France	2.7
US to Australia	3.5	US to Jordan	2.1
US to UK	3.5	US to Mexico	1.9

Table 3: Positive/Negative ratios for the US toward its top 10 frequently mentioned nations.

6.2 Nation-Nation Sentiment Experiments

Nation opinions are represented as directed edges: each edge (X,Y) represents the opinion of nation X toward nation Y. The weight of an edge is the ratio of positive to negative counts:

$$R(X,Y) = C(X,Y,positive)/C(X,Y,negative)$$

where C(X,Y,L) is the number of tweets by nation X users about nation Y with sentiment L. Only tweets that make it through the Nation to Nation Pipeline of Section 5 receive sentiment labels. If a nation pair (X,Y) was observed less than 1000 times, it is not included in the evaluations. We provide experiments later to evaluate this cutoff's affect.

The dataset (Section 3) spans 17 months from 2013-2015. All tweets are classified or filtered out, and R(X,Y) is computed for all pairs. Table 3 shows the top 10 nation pair ratios (with over 500k tweets between them) for the U.S.

We present four formal evaluations to answer the central question of this paper: can sentiment from social media be used to help approximate international opinions? The first two experiments use public opinion polls of national sentiment toward other nations. The third uses military conflicts as a proxy for *negative* relations, and the fourth uses current formal alliances as a proxy for *positive* relations. None of these can provide a complete picture of the connection between popular sentiment and international relations, but the four together provide a strong case that sentiment contains a useful signal.

Correlation: Public Opinion Polls

Human Poll	Sentiment	Freq. Baseline
Germany	Canada	China
Canada	Japan	Israel
UK	EU	USA
Japan	France	Russia
France	UK	India
EU	Brazil	Japan
Brazil	USA	Canada
USA	India	UK
China	South Africa	Pakistan
South Korea	South Korea	France
South Africa	Germany	Iran
India	Russia	Brazil
Russia	China	Germany
Israel	Israel	North Korea
North Korea	North Korea	South Korea
Pakistan	Iran	South Africa
Iran	Pakistan	EU
Correlation	**0.80**	**-0.06**

Table 4: Polling Data: ranking of a nation's "positive contribution" to the world, compared to automatically identified nation-nation sentiment.

Each year, GlobeScan/PIPA releases polling data of 16 nations in a ranked ordering based on how 26,000 people view their "positive contribution" to the world[3]. This poll helps to determine whether or not this paper's sentiment pipeline matches human polling. We created our own ranking by assigning a world score to each nation n: the average sentiment ratio of all other nations toward n. Since the polling data also ranks the EU, we average the EU member nation world scores for an EU world score. Table 4 shows the PIPA poll (**Human Poll**) and our world ranking (**Sentiment**). Using Spearman's rank correlation coefficient to measure agreement, our ranking is strongly correlated at $\rho = 0.8$ (perfect is 1.0). The main mistake in our ranking is Germany. We also compare against a **Frequency Baseline** to eliminate the possibility that it's simply a matter of topic popularity. Poll rankings could simply be correlated with who people choose to discuss, or vice versa. The frequency baseline is the average number of twitter mentions per nation (i.e., the most discussed). This baseline shows no correlation at $\rho = -.06$.

We then evaluated against a US-centric polling agency, Gallup. They asked Americans to rate other nations as 'favorable' or 'unfavorable' in a 2014 poll[4]. The result is a ranking of favorability. In contrast to the PIPA poll which evaluates many nations looking in, Gallup evaluates a single nation looking out. Space constraints prevent us from visually showing the US ranking, but again the sentiment ratios have a strong correlation at $\rho = .81$. The frequency baseline is $\rho = .23$. The nation-nation sentiment extraction strongly correlates with both world views (PIPA) and US-specific views (Gallup).

The third evaluation uses a political science dataset from the Correlates of War Project, the **Militarized Interstate Disputes v4.01** (MID) (Ghosn et al., 2004). This dataset is used in the field of international relations, listing conflicts since 1816. We limit the evaluation to conflicts after 1990 to keep relations current. The dataset ends at 2001, so while not a completely current evaluation, it stands as a proxy for negative relations. Each conflict in MID is labeled with a conflict severity. We convert severity labels between nations to a pair score MID(X,Y):

$$MID(X,Y) = \sum_{d \in Disputes(X,Y)} score(d) \quad (2)$$

where Disputes(X,Y) is the set of conflicts between the two nations X and Y, and score(d) is a severity score for the type of dispute d. *War* is -5, *use of force* is -4, *displays of force* is -3, *threatening use of force* is -2, and *no militarized action* is -1. We take the sum of severity scores and save all nation pairs (X,Y) such that $MID(X,Y) < -10$. This score indicates multiple conflicts and are thus considered as nations with true negative relations.

We then compare our sentiment ratios R(X,Y) against these gold negative pairs. Each continuous R(X,Y) is discretized into sentiment categories for ease of comparison. Since the mean across all R(X,Y) is 1.25, we consider an interval around 1.25 as neutral and create positive and negative labels above and below that neutral center:

$$ratiolabel(Z) = \begin{cases} positive, & \text{if } Z \geq 2.4 \\ slightpos, & \text{if } 2.4 > Z \geq 1.4 \\ neutral, & \text{if } 1.4 > Z \geq 1.1 \\ slightneg, & \text{if } 1.1 > Z \geq 0.8 \\ negative, & \text{if } 0.8 > Z \end{cases}$$

The bottom table in Table 5 shows the number of nation pairs that align with the negative labels of the MID dataset. Only pairs that have at least 1000 tweets are evaluated. Of the resulting 90 pairs, 61 are correctly identified by our system as negative or slight negative for an **accuracy of 68%**. 19 *positive* pairs are incorrectly aligned with *MID-negative*. Error analysis shows that many incorrect

[3] http://www.worldpublicopinion.org/pipa/2013CountryRatingPoll.pdf
[4] http://www.gallup.com/poll/1624/perceptions-foreign-countries.aspx

Positive: Formal Alliances

	Pos	SlPos	N	SlNeg	Neg
# Nation Pairs	**341**	**65**	22	**26**	**28**

Negative: Military Disputes

	Pos	SlPos	N	SlNeg	Neg
MID-Negative	**12**	**7**	10	**15**	**46**

Table 5: Top: The number of NATO/EU nation pairs with automatic sentiment labels. Bottom: The number of pairs with military disputes (MID dataset) and automatic sentiment labels.

labels are between nations with a smaller Twitter presence, so performance likely suffers due to lack of data. For robustness testing, we shifted the thresholds that discretize the nation ratios and MID scores into postive and negative categories. The accuracy result shows little change. We also reran the experiment with a higher cutoff of 10,000 instead of 1,000. The negative disputes **accuracy increases from 68% to 81%**, but the recall obviously drops as less countries are included. This suggests the sentiment ratios might be used on a sliding confidence scale based on frequency of mention.

To evaluate positive relations, we use current alliances as a fourth evaluation. NATO and the EU are the main global alliances with elements of mutual defense. We do not include trade-only alliances as trade is not always an indication of allegiance and approval (Russia and Ukraine is a current example of this disparity). This evaluation considers pairs of nations within NATO and within the EU as gold positive relations. We compare our sentiment ratios to these pairs in the top of Table 5. This evaluation is broader than the conflict evaluation because NATO and EU nations have more of a Twitter presence. Of the 482 country pairs, our positive/slightpos accuracy is **84.2%**.

6.3 Application Experiments

We now briefly discuss how these positive results for nation-nation sentiment relates to political science analysis.

One core area of study is how national sentiment shifts over time, *and why*. Computing R(X,Y) on a bi-weekly basis, Figure 1 graphs the sentiment ratio from the USA toward India and Israel. The timeline shows significant favorability toward India during their extended election season, but afterward the opinion is similar to before the election. In contrast, the 2014 Israel-Gaza conflict shows a very different effect. US opinion of Israel is initially steady (slightly positive) until the conflict causes a large dip. Unlike India's spike, *US opinion stays depressed* even after the conflict concludes. It appears to have only risen to 'normal' levels months later. We do note that the water is slightly muddied because our algorithm may not distinguish well between sentiment toward the war, Israel itself, or even sympathy toward casualties. However, it's clear that nation-nation sentiment is captured, and future work is needed to identify finer grained sentiment as needed.

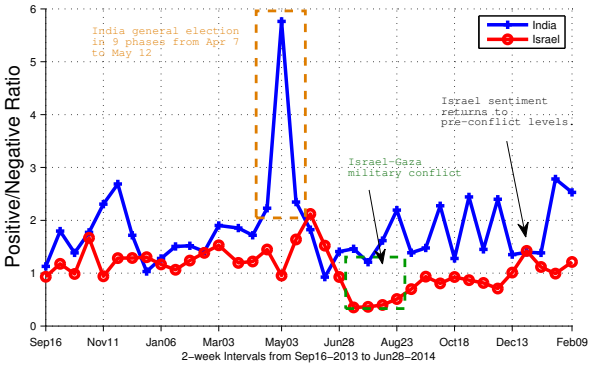


Figure 1: USA opinion of India/Israel over 2-week intervals from Sep-2013 to Feb-2015.

Another application is inter-alliance relations. For instance, Table 6 shows how NATO member nations view *other* alliances. The table shows the average of all R(X,Y) edges for each nation within an alliance to a nation in the other. According to our ratios, NATO countries have stark differences between how they view themselves versus how they view African Union/Arab League nations. Further, our pipeline enables analysis of outside nations looking in. For instance, the nations with the most positive view of the EU are Uruguay, Lithuania (EU member), Belarus, Moldova, and Slovakia (EU member). Almost all (not Uruguay) are eastern european nations. Moldova is currently seeking EU membership and Belarus had closer ties until recently. Our results might point to potential future alliances. Future work is needed to explore this implied connection.

Finally, the R(X,Y) ratios can also represent a nation's *opinion profile*. Represent each nation X by a vector of its R(X,Y) ratios. This represents its entire international view based on social me-

Inter-Alliance Opinion Ratios

Source	Target	Average R(X,Y)
NATO	African Union	0.45
NATO	Arab League	0.48
NATO	European Union	1.51
NATO	NATO	1.55

Table 6: Average pos/neg ratio of NATO nations toward the nations in other formal alliances.

MID Accuracy with Filters

Filters	Correct	Incorrect	Acc.
Dining+Sports	61	29	**68%**
Sports only	61	37	**62%**
None	56	51	**52%**

Table 7: Filtering effects on the MID results.

dia sentiment. Space prohibits more detail, but we clustered opinion profiles with k-means (k=12) and cosine similarity. Typical alliances, such as European and African clusters, are learned.

6.4 Ablation Tests

The sentiment pipeline includes two practical filters to remove tweets about **dining** and **sports**. We added these during training and developement solely based on our interpretation and analysis of the data. We did not evaluate on the test datasets until the very end. Table 7 shows results from the MID evaluation with the dining and sports filters removed in sequence.

The number of correctly identified negative nation pairs is mostly unchanged, but the number of incorrect decisions increases dramatically. This occurs because a greater number of tweets make it through the pipeline. Further, this shows that the filters effectively remove tweets that cause misclassification errors.

7 Discussion

This work is an important first step toward automatic means to broadly detect international relations from social media. We use sentiment analysis as a proxy for extracting at least one aspect of the large set of factors involved in such relations. This paper is the largest application of sentiment analysis across a diverse set of nation-nation pairs (hundreds of country pairs over 17 months), and we showed that this sentiment is strongly correlated ($\rho = 0.8$) with two independent public opinion polls. These correlations more importantly suggest that we are not simply identifying a binary positive or negative relation, but that the *relative sentiment scores* are useful. The failure of frequency baselines on this ranking further suggests that this is not a side effect of topic frequency.

One argument against using public opinion polls for an evaluation is that the same people who are polled by PIPA might be the same people who tend to voice opinions on Twitter. The Twitter dataset is not independent from the polls, so the strong correlation we found could simply be a matter of sampling the same population. This is not possible to know, but whether or not it is the case, this paper's pipeline could be quite valuable in automating expensive and time consuming human polls.

The results that focused on positive sentiment (polls and alliances) are quite high. Negative sentiment revealed a lower 68% accuracy on the MID dataset, but it is due to the fact that nation-nation conflicts often occur between smaller nations that are not represented well on Twitter. Requiring a higher observed count improves accuracy to 81%.

While we are cautious not to make broad claims about discovering international relations on Twitter, we are encouraged by the experimental alignment with current alliances and historical conflict data. The sentiment timeline for Israel and India (Figure 1) is also intriguing. Tracking nation relations over a longer time period presents an opportunity for future study. This continual tracking of sentiment is one of the most obvious benefits of an automated approach.

Finally, an interactive world map is available to browse this paper's data at *www.usna.edu/Users/cs/nchamber/nations*. Each nation can be selected to visually color the map with its positive/negative lens, and timelines showing sentiment shifts between nations are visible. All code, data, and results are also available on this page. We hope this work encourages even further connections between NLP and political science.

Acknowledgments

This work was supported in part by the DoD HPC Modernization Program. It also would not have been possible without the infrastructure support and help from the accommodating staff at the Maui HPCC. Special thanks to the reviewers and area chair for their parts in an exciting and fulfilling review process.

References

Pablo Barberá and Gonzalo Rivero. 2014. Understanding the political representativeness of twitter users. *Social Science Computer Review*, December.

Hila Becker, Mor Naaman, and Luis Gravano. 2011. Beyond trending topics: Real-world event identification on twitter. *ICWSM*, 11:438–441.

Albert Bifet and Eibe Frank. 2010. Sentiment knowledge discovery in twitter streaming data. In *Lecture Notes in Computer Science*, volume 6332, pages 1–15.

Dmitry Davidov, Oren Tsur, and Ari Rappoport. 2010. Enhanced sentiment learning using twitter hashtags and smileys. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING 2010)*.

Xiaowen Ding and Bing Liu. 2007. The utility of linguistic rules in opinion mining. In *Proceedings of SIGIR-2007*, pages 23–27.

Ghosn, Faten, Glenn Palmer, and Stuart Bremer. 2004. The mid3 data set, 1993-2001: Procedures, coding rules, and description. In *Conflict Management and Peace Science 21*, pages 133–154.

Alec Go, Richa Bhayani, and Lei Huang. 2009. Twitter sentiment classification using distant supervision. Technical report.

Philip N. Howard, Aiden Duffy, Deen Freelon, Muzammil Hussain, Will Mari, and Marwa Mazaid. 2011. Opening closed regimes: What was the role of social media during the arab spring? Technical Report working paper 2011.1, University of Washington, September.

Philip N. Howard. 2013. *Democracy's Fourth Wave? Digital Media and the Arab Spring*. Oxford University Press.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Amaney A. Jamal, Robert O. Keohane, David Romney, and Dustin Tingley. 2015. Anti-americanism and anti-interventionism in arabic twitter discourses. *Perspectives on Politics*, pages 55–73, March.

Long Jiang, Mo Yu, Ming Zhou, Xiaohua Liu, and Tiejun Zhao. 2011. Target-dependent twitter sentiment classification. In *Proceedings of the Association for Computational Linguistics (ACL-2011)*.

Efthymios Kouloumpis, Theresa Wilson, and Johanna Moore. 2011. Twitter sentiment analysis: The good the bad and the omg! In *Proceedings of the Fifth International AAAI Conference on Weblogs and Social Media*.

Adam D. I. Kramer. 2010. An unobtrusive behavioral model of 'gross national happiness'. In *Proceedings of the 28th International Conference on Human Factors in Computing Systems (CHI 2010)*.

Bing Liu and Lei Zhang. 2012. A survey of opinion mining and sentiment analysis. *Mining Text Data*, pages 415–463.

Marc Lynch. 2014. Arabs do care about gaza. www.washingtonpost.com/blogs/monkey-cage/wp/2014/07/14/arabs-do-care-about-gaza.

Saif M. Mohammad, Svetlana Kiritchenko, and Xiaodan Zhu. 2013. Nrc-canada: Building the state-of-the-art in sentiment analysis of tweets. In *Proceedings of the seventh international workshop on Semantic Evaluation Exercises (SemEval-2013)*.

Preslav Nakov, Zornitsa Kozareva, Alan Ritter, Sara Rosenthal, Veselin Stoyanov, and Theresa Wilson. 2013. Semeval-2013 task 2: Sentiment analysis in twitter. In *Proceedings of the 7th International Workshop on Semantic Evaluation*. Association for Computational Linguistics.

Tetsuya Nasukawa and Jeonghee Yi. 2003. Sentiment analysis: capturing favorability using natural language processing. In *Proceedings of K-CAP*.

Brendan O'Connor, Ramnath Balasubramanyan, Bryan R. Routledge, and Noah A. Smith. 2010. From tweets to polls: Linking text sentiment to public opinion time series. In *Proceedings of the AAAI Conference on Weblogs and Social Media*.

Brendan O'Connor, Brandon M Stewart, and Noah A Smith. 2013. Learning to extract international relations from political context. In *ACL (1)*, pages 1094–1104.

Alexander Pak and Patrick Paroubek. 2010. Twitter as a corpus for sentiment analysis and opinion mining. In *Proceedings of the Seventh International Conference On Language Resources and Evaluation (LREC)*.

B. Pang and L. Lee. 2008. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*.

Jonathon Read. 2005. Using emoticons to reduce dependency in machine learning techniques for sentiment classification. In *Proceedings of the ACL Student Research Workshop (ACL-2005)*.

Ellen Riloff and Rosie Jones. 1999. Learning dictionaries for information extraction by multi-level bootstrapping. In *Proceedings of the 16th National Conference on Artificial Intelligence (AAAI-99)*, pages 474–479.

Sara Rosenthal, Preslav Nakov, Alan Ritter, and Veselin Stoyanov. 2014. Semeval-2014 task 9: Sentiment analysis in twitter. Association for Computational Linguistics.

Takeshi Sakaki, Makoto Okazaki, and Yutaka Matsuo. 2010. Earthquake shakes twitter users: real-time event detection by social sensors. In *Proceedings of the 19th international conference on World wide web*, pages 851–860. ACM.

Chenhao Tan, Lillian Lee, Jie Tang, Long Jiang, Ming Zhou, and Ping Li. 2011. User-level sentiment analysis incorporating social networks. In *Proceedings of the 17th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*.

H. Tang, S. Tan, and X. Cheng. 2009. A survey on sentiment detection of reviews. *Expert Systems with Applications*.

Mike Thelwall, Kevan Buckley, and Georgios Paltoglou. 2011. Sentiment in twitter events. *Journal of the American Society for Information Science and Technology*, 62(2):406–418.

M. Tsytsarau and T. Palpanas. 2012. Survey on mining subjective data on the web. *Data Mining and Knowledge Discovery Journal*, 24(3):478–514.

Andranik Tumasjan, Timm O. Sprenger, Philipp G. Sandner, and Isabell M. Welpe. 2010. Election forecasts with twitter: How 140 characters reflect the political landscape. *Social Science Computer Review*.

Xiaodan Zhu, Svetlana Kiritchenko, and Saif M. Mohammad. 2014. Nrc-canada-2014: Recent improvements in the sentiment analysis of tweets. In *Proceedings of the seventh international workshop on Semantic Evaluation Exercises (SemEval-2013)*.

Open Extraction of Fine-Grained Political Statements

David Bamman
School of Information
University of California, Berkeley
Berkeley, CA 94720, USA
dbamman@berkeley.edu

Noah A. Smith
Computer Science & Engineering
University of Washington
Seattle, WA 98195, USA
nasmith@cs.washington.edu

Abstract

Text data has recently been used as evidence in estimating the political ideologies of individuals, including political elites and social media users. While inferences about people are often the intrinsic quantity of interest, we draw inspiration from open information extraction to identify a new task: inferring the political import of propositions like OBAMA IS A SOCIALIST. We present several models that exploit the structure that exists between people and the assertions they make to learn latent positions of people and propositions at the same time, and we evaluate them on a novel dataset of propositions judged on a political spectrum.

1 Introduction

Over the past few years, much work has focussed on inferring political preferences of people from their behavior, both in unsupervised and supervised settings. Classical ideal point models (Poole and Rosenthal, 1985; Martin and Quinn, 2002) estimate the political ideologies of legislators through their observed voting behavior, possibly paired with the textual content of bills (Gerrish and Blei, 2012) and debate text (Nguyen et al., 2015); other unsupervised models estimate ideologies of politicians from their speeches alone (Sim et al., 2013). Twitter users have also been modeled in a similar framework, using their observed following behavior of political elites as evidence to be explained (Barberá, 2015). Supervised models, likewise, have not only been used for assessing the political stance of sentences (Iyyer et al., 2014) but are also very popular for predicting the holistic ideologies of everyday users on Twitter (Rao et al., 2010; Pennacchiotti and Popescu, 2011; Al Zamal et al., 2012; Cohen and Ruths, 2013; Volkova et al., 2014), Facebook (Bond and Messing, 2015) and blogs (Jiang and Argamon, 2008), where training data is relatively easy to obtain—either from user self-declarations, political following behavior, or third-party categorizations.

Aside from their intrinsic value, estimates of users' political ideologies have been useful for quantifying the orientation of news media sources (Park et al., 2011; Zhou et al., 2011). We consider in this work a different task: estimating the political import of propositions like OBAMA IS A SOCIALIST.

In focusing on propositional statements, we draw on a parallel, but largely independent, strand of research in open information extraction. IE systems, from early slot-filling models with predetermined ontologies (Hobbs et al., 1993) to the large-scale open-vocabulary systems in use today (Fader et al., 2011; Mitchell et al., 2015) have worked toward learning type-level propositional information from text, such as BARACK OBAMA IS PRESIDENT. To a large extent, the ability to learn these facts from text is dependent on having data sources that are either relatively factual in their presentation (e.g., news articles and Wikipedia) or are sufficiently diverse to average over conflicting opinions (e.g., broad, random samples of the web).

Many of the propositional statements that individuals make online are, of course, not objective descriptions of reality at all, but rather reflect their own beliefs, opinions and other private mental states (Wiebe et al., 2005). While much work has investigated methods for establishing the truth content of individual sentences — whether from the perspective of veridicality (de Marneffe et al., 2012), fact assessment (Nakashole and Mitchell, 2014), or subjectivity analysis (Wiebe et al., 2003; Wilson, 2008) — the structure that exists between users and their assertions gives us an opportunity to situate them both in the same political space: in this work we operate at the level of subject-

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 76–85,
Lisbon, Portugal, 17-21 September 2015.

predicate propositions, and present models that capture not only the variation in what subjects (e.g., OBAMA, ABORTION, GUN CONTROL) that individual communities are more likely to discuss, but also the variation in what predicates different communities assert of the same subject (e.g., GLOBAL WARMING IS A HOAX vs. IS A FACT). The contributions of this work are as follows:

- We present a new evaluation dataset of 766 propositions judged according to their positions in a political spectrum.
- We present and evaluate several models for estimating the ideal points of subject-predicate propositions, and find that unsupervised methods perform best (on sufficiently partisan data).

2 Task and Data

The task that we propose in this work is assessing the political import of type-level propositions; on average, are liberals or conservatives more likely to claim that GLOBAL WARMING IS A HOAX? To support this task, we create a benchmark of political propositions, extracted from politically partisan data, paired with human judgments (details in §2.3). We define a **proposition** to be a tuple comprised of a subject and predicate, each consisting of one or more words, such as ⟨global warming, *is a hoax*⟩.[1] We adopt an open vocabulary approach where each unique predicate defines a unary relation.

2.1 Data

In order to extract propositions that are likely to be political in nature and exhibit variability according to ideology, we collect data from a politically volatile source: comments on partisan blogs.

We draw data from NPR,[2] Mother Jones[3] and Politico[4], all listed by Pew Research (Mitchell et al., 2014) as news sources most trusted by those with consistently liberal views; Breitbart,[5] most trusted by those with consistently conservative views; and the Daily Caller,[6] Young Conservatives[7] and the Independent Journal Review,[8] all popular among conservatives (Kaufman, 2014). All data comes from articles published between 2012–2015 and is centered on the US political landscape.

Source	Articles	Posts	Tokens	Users
Politico	10,305	9.8M	348.4M	173,519
Breitbart	46,068	8.8M	336.4M	165,607
Daily Caller	46,114	5.4M	240.4M	228,696
Mother Jones	16,830	1.9M	119.2M	138,995
NPR	14993	1.6M	82.6M	62,600
IJ Review	3,396	278K	13.1M	51,589
Young Cons.	4,948	222K	10.6M	34,434
Total	142,654	28.0M	1.15B	621,231

Table 1: Data.

We gather comments using the Disqus API;[9] as a comment hosting service, Disqus allows users to post to different blogs using a single identity. Table 1 lists the total number of articles, user comments, unique users and tokens extracted from each blog source. In total, we extract 28 million comments (1.2 billion tokens) posted by 621,231 unique users.[10]

2.2 Extracting Propositions

The blog comments in table 1 provide raw data from which to mine propositional assertions. In order to extract structured ⟨subject, *predicate*⟩ propositions from text, we first parse all comments using the collapsed dependencies (de Marneffe and Manning, 2008) of the Stanford parser (Manning et al., 2014), and identify all subjects as those that hold an `nsubj` or `nsubjpass` relation to their head. In order to balance the tradeoff between generality and specificity in the representation of assertions, we extract three representations of each predicate.

1. Exact strings, which capture verbatim the specific nuance of the assertion. This includes all subjects paired with their heads and all descendants of that head. Tense and number are preserved.

   Example: ⟨Reagan, *gave amnesty to 3 million undocumented immigrants*⟩

2. Reduced syntactic tuples, which provide a level of abstraction by lemmatizing word forms and including only specific syntactic relationships. This includes propositions de-

[1] We use these typographical conventions throughout: Subjects are in sans serif, predicates in *italics*.

[2] `http://www.npr.org`

[3] `http://www.motherjones.com`

[4] `http://www.politico.com`

[5] `http://www.breitbart.com`

[6] `http://dailycaller.com`

[7] `http://www.youngcons.com`

[8] `https://www.ijreview.com`

[9] `https://disqus.com/api/`

[10] While terms of service prohibit our release of this data, we will make available tools to allow others to collect similar data from Disqus for these blogs.

fined as nominal subjects paired with their heads and children of that head that are negators, modal auxiliaries (*can, may, might, shall, could, would*), particles and direct objects. All word forms are lemmatized, removing tense information on verbs and number on nouns.

Example: ⟨Reagan, *give amnesty*⟩

3. Subject-verb tuples, which provide a more general layer of abstraction by only encoding the relationship between a subject and its main action. In this case, a proposition is defined as the nominal subject and its lemmatized head.

Example: ⟨Reagan, *give*⟩

The human benchmark defined in §2.3 below considers only verbatim predicates, while all models proposed in §3 and all baselines in §4 include the union of all three representations as data.

Here, syntactic structure not only provides information in the representation of propositions, but also allows us to define criteria by which to exclude predicates — since we are looking to extract propositions that are directly asserted by an author of a blog comment (and not second-order reporting), we exclude all propositions dominated by an attitude predicate (*Republicans think that Obama should be impeached*) and all those contained within a conditional clause (*If Obama were impeached...*). We also exclude all assertions drawn from questions (i.e., sentences containing a question mark) and all assertions extracted from quoted text (i.e., surrounded by quotation marks).

In total, from all 28 million comments across all seven blogs, we extract all propositions defined by the criteria above, yielding a total of 61 million propositions (45 million unique).

2.3 Human Benchmark

From all propositions with a verbatim predicate extracted from the entire dataset, we rank the most frequent subjects and manually filter out non-content terms (like *that*, *one*, *someone*, *anyone*, etc.) to yield a set of 138 target topics, the most frequent of which are *obama*, *democrats*, *bush*, *hillary*, and *america*.

For each proposition containing one of these topics as its subject and mentioned by at least 5 different people across all blogs, we randomly sampled 1,000 in proportion to their frequency of use (so that sentences that appear more frequently in the data are more likely to be sampled); the sentences selected in this random way contain a variety of politically charged viewpoints. We then presented them to workers on Amazon Mechanical Turk for judgments on the extent to which they reflect a US liberal vs. conservative political worldview.

For each sentence, we paid 7 annotators in the United States to a.) confirm that the extracted sentence was a well-formed assertion and b.) to rate "the most likely political belief of the person who would say it" on a five-point scale: very conservative/Republican (-2), slightly conservative/Republican (-1), neutral (0), slightly liberal/Democrat (1), and very liberal/Democrat (2).

We keep all sentences that at least six annotators have marked as meaningful (those excluded by this criterion include sentence fragments like *bush wasn't* and those that are difficult to understand without context, such as *romney is obama*) and where the standard deviation of the responses is under 1 (which excludes sentences with flat distributions such as *government does nothing well* and those with bimodal distributions, such as *christie is done*). After this quality control, we average the responses to create a dataset of 766 propositions paired with their political judgments. Table 2 presents a random sample of annotations from this dataset.

proposition	mean	s.d.
obama lied and people died	-2.000	0.000
gay marriage is not a civil right	-1.857	0.350
obama can't be trusted	-1.714	0.452
hillary lied	-0.857	0.990
hillary won't run	-0.714	0.452
bush was just as bad	0.857	0.639
obama would win	1.429	0.495
rand paul is a phony	1.429	0.495
abortion is not murder	1.571	0.495
hillary will win in 2016	1.857	0.350

Table 2: Random sample of AMT annotations.

3 Models

The models we introduce to assess the political import of propositions are based on two fundamental ideas. First, users' latent political preferences, while unobserved, can provide an organizing principle for inference about propositions in an unsupervised setting. Second, by decoupling the variation in *subjects* discussed by different communities (e.g., liberals may talk more

about global warming while conservatives may talk more about gun rights) from variation in what statements are *predicated* of those subjects (e.g., liberals may assert that ⟨global warming, *is a fact*⟩ while conservatives may be more likely to assert that it *is a hoax*), we are able to have a more flexible and interpretable parameterization of observed textual behavior that allows us to directly measure both.

We present two models below: one that represents users and propositions as real-valued points, and another that represents each as categorical variables. For both models, the input is a set of users paired with a list of ⟨subject, *predicate*⟩ tuples they author; the variables of interest we seek are representations of those users, subjects, and predicates that explain the coupling between users and propositions we see.

3.1 Additive Model

The first model we present (fig. 1) represents each user, subject, and predicate as a real-valued point in K-dimensional space. In the experiments that follow, we consider the simple case where $K = 1$ but present the model in more general terms below.

In this model, we parameterize the generative probability of a subject (like Obama) as used by an individual u as the exponentiated sum of a background log frequency of that subject in the corpus overall (m_{sbj}) and K additive effects, normalized over the space of S possible subjects, as a real-valued analogue to the SAGE model of Eisenstein et al. (2011). While the background term controls the overall frequency of a subject in the corpus, $\beta \in \mathbb{R}^{K \times S}$ mediates the relative increase or decrease in probability of a subject for each latent dimension. Intuitively, when both $\eta_{u,k}$ and $\beta_{k,sbj}$ (for a given user u, dimension k, and subject sbj) are the same sign (either both positive or both negative), the probability of that subject under that user increases; when they differ, it decreases. $\beta_{\cdot,sbj}$ is a K-dimensional representation of subject sbj, and $\eta_{u,\cdot}$ is a K-dimensional representation of user u.

$$P(sbj \mid u, \eta, \beta, m_{sbj}) = \frac{\exp\left(m_{sbj} + \sum_{k=1}^{K} \eta_{u,k}\beta_{k,sbj}\right)}{\sum_{sbj'} \exp\left(m_{sbj'} + \sum_{k=1}^{K} \eta_{u,k}\beta_{k,sbj'}\right)} \quad (1)$$

Likewise, we parameterize the generative probability of a predicate (conditioned on a subject) in the same way; for S subjects, each of which contains (up to) P predicates, $\psi \in \mathbb{R}^{S \times K \times P}$ captures the relative increase or decrease in probability for a given predicate conditioned on its subject, relative to its background frequency in the corpus overall, $m_{pred|sbj}$.

$$P(pred \mid sbj, u, \eta, \psi, m_{pred|sbj}) = \frac{\exp\left(m_{pred|sbj} + \sum_{k=1}^{K} \eta_{u,k}\psi_{sbj,k,pred}\right)}{\sum_{pred'} \exp\left(m_{pred'|sbj} + \sum_{k=1}^{K} \eta_{u,k}\psi_{sbj,k,pred'}\right)} \quad (2)$$

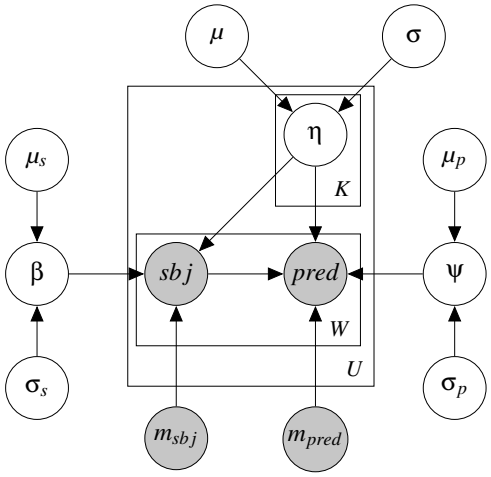


Figure 1: Additive model with decoupled subjects and predicates. η contains a K-dimensional representation of each user; β captures the variation in observed subjects, and ψ captures the variation in predicates for a fixed subject.

The full generative story for this model runs as follows. For a vocabulary of subjects of size S, where each subject s has P predicates:

- For each dimension k, draw subject coefficients $\beta_k \in \mathbb{R}^S \sim \text{Norm}(\mu_s, \sigma_s \mathbf{I})$
- For each subject s:
  - For each dimension k, draw subject-specific predicate coefficients $\psi_{s,k} \in \mathbb{R}^P \sim \text{Norm}(\mu_p, \sigma_p \mathbf{I})$
- For each user u:
  - Draw user representation $\eta \in \mathbb{R}^K \sim \text{Norm}(\mu, \sigma \mathbf{I})$
  - For each proposition $\langle sbj, pred \rangle$ made by u:
    - Draw sbj according to eq. 1
    - Draw $pred$ according to eq. 2

The unobserved quantities of interest in this model are η, β and ψ. In the experiments reported

below, we set the prior distributions on η, β and ψ to be standard normals ($\mu = 0, \sigma = 1$) and perform maximum *a posteriori* inference with respect to η, β and ψ in turn for a total of 25 iterations.

While β and ψ provide scores for the political import of subjects and of predicates conditioned on fixed subjects, respectively, we can recover a single ideological score for both a subject and its predicate by adding their effects together. In the evaluation given in §5, let the PREDICATE SCORE for $\langle$subject, *predicate*$\rangle$ be that given by $\psi_{\text{subject},\cdot,\textit{predicate}}$, and let the PROPOSITION SCORE be $\beta_{\cdot,\text{subject}} + \psi_{\text{subject},\cdot,\textit{predicate}}$.

3.2 Single Membership Model

While the additive model above represents each user and proposition as a real-valued point in K-dimensional space, we can also represent those values as categorical variables in an unsupervised naïve Bayes parameterization; in this case, a user is not defined as a mixture of different effects, but rather belongs to a single unique community. The generative story for this model (shown in fig. 2) is as follows:

- Draw population distribution over categories $\theta \sim \text{Dir}(\alpha)$
- For each category k, draw distribution over subjects $\phi_k \sim \text{Dir}(\gamma)$
- For each category k and subject s:
  - Draw distribution over subject-specific predicates $\xi_{k,s} \sim \text{Dir}(\gamma_s)$
- For each user u:
  - Draw user type index $z \sim \text{Cat}(\theta)$
  - For each proposition $\langle sbj, pred\rangle$ made by u:
    - Draw subject $sbj \sim \text{Cat}(\phi_z)$
    - Draw predicate $pred \sim \text{Cat}(\xi_{z,sbj})$

We set $K = 2$ in an attempt to recover a distinction between liberal and conservative users. For the experiments reported below, we run inference using collapsed Gibbs sampling (Griffiths and Steyvers, 2004) for 100 iterations, performing hyperparameter optimization on α, γ and γ_s (all asymmetric) every 10 using the fixed-point method of Minka (2003).

In order to compare the subject-specific predicate distributions across categories, we first calculate the posterior predictive distribution by taking a single sample of all latent variables z to estimate

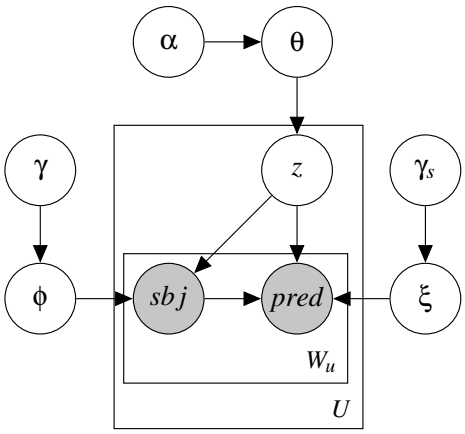


Figure 2: Single membership model with decoupled subjects and predicates. z is the latent category identity of a user (e.g., liberal or conservative); ϕ is a distribution over subjects for each category; and ξ is a distribution of predicates given subject s.

the following (Asuncion et al., 2009):

$$\hat{\zeta}_{z,v} = \frac{\mathbf{c}(z, v) + \gamma_v}{\sum_{v'} \mathbf{c}(z, v') + \gamma_{v'}} \tag{3}$$

Where $\hat{\zeta}_{z,v}$ is the vth element of the zth multinomial being estimated, $\mathbf{c}(z, v)$ is the count of element v associated with category z and γ_v is the associated Dirichlet hyperparameter for that element. Given this smoothed distribution, for each proposition we assign it a real valued score, the log-likelihood ratio between its value in these two distributions. In the evaluation that follows, let the PREDICATE SCORE for a given $\langle$subject, *predicate*$\rangle$ under this model be:

$$\log \left(\frac{\hat{\xi}_{0,\text{subject},\textit{predicate}}}{\hat{\xi}_{1,\text{subject},\textit{predicate}}} \right) \tag{4}$$

Let the PROPOSITION SCORE be:

$$\log \left(\frac{\hat{\phi}_{0,\text{subject}} \times \hat{\xi}_{0,\text{subject},\textit{predicate}}}{\hat{\phi}_{1,\text{subject}} \times \hat{\xi}_{1,\text{subject},\textit{predicate}}} \right) \tag{5}$$

4 Comparison

The two models described in §3 are unsupervised methods for estimating the latent political positions of users along with propositional assertions. We compare with three other models, a mixture of unsupervised, supervised, and semi-supervised methods. Unlike our models, these were not designed for the task described in §2.

4.1 Principal Component Analysis

To compare against another purely unsupervised model, we evaluate against principal component analysis (PCA), a latent linear model that minimizes the average reconstruction error between an original data matrix $\mathbf{X} \in \mathbb{R}^{n \times p}$ and a low-dimensional approximation $\mathbf{Z}\mathbf{W}^\top$, where $\mathbf{Z} \in \mathbb{R}^{n \times K}$ can be thought of as a K-dimensional latent representation of the input and $\mathbf{W} \in \mathbb{R}^{p \times K}$ contains the eigenvectors of the K largest eigenvalues of the covariance matrix $\mathbf{X}\mathbf{X}^\top$, providing a K-dimensional representation for each feature. We perform PCA with $K = 1$ on two representations of our data: a.) **counts**, where the input data matrix contains the counts for each proposition for each user, and b.) **frequencies**, where we normalize those counts for each user to unit length. While the input data is sparse, we must center each column to have a 0 mean (resulting in a dense matrix) and perform PCA through a singular value decomposition of that column-centered data using the method of Halko (2011); in using SVD for PCA, the right singular vectors correspond to the principal directions; from these we directly read off a $K = 1$ dimensional score for each proposition in our data.

4.2 ℓ_2-Regularized Logistic Regression

While unsupervised methods potentially allow us to learn interesting structure in data, they are often eclipsed in prediction tasks by the addition of any form of supervision. While purely supervised models give more control over the exact decision boundary being learned, they can suffer by learning from a much smaller training set than unsupervised methods have access to. To evaluate this tradeoff, we compare against a supervised model trained using naturally occurring data – users who self-declare themselves in their profiles to be *liberal*, *conservative*, *democrat*, or *republican*. We randomly sampled 150 users who self-identify as liberals and 150 who identify as conservatives. We do not expect these users to be a truly random sample of the population — those who self-declare their political affiliation are more likely to engage with political content differently from those who do not (Sandvig, 2015; Hargittai, 2015) — but is a method that has been used for political prediction tasks in the past (Cohen and Ruths, 2013).

We build a predictive model using two classes of features: a.) binary indicators of the most frequent 25,000 unigrams and multiword expressions[11] in the corpus overall; and b.) features derived from user posting activity to the seven blogs shown in table 1 (binary indicators of the blogs posted to, and the identity of the most frequent blog). In a tenfold cross-validation (using ℓ_2-regularized logistic regression), this classifier attains an accuracy rate of 76.7% (with a standard error of ± 1.7 across the ten folds).

In order to establish real-valued scores for propositions, we follow the same method as for the single membership model described above, using the log likelihood ratio of the probability of the proposition under each condition, where that probability is given as the count of the proposition among users classified as (e.g.) liberals (plus some small smoothing factor) divided by the total number of propositions used by them overall.

$$score(prop) = \log \frac{P(prop \mid z = conservative)}{P(prop \mid z = liberal)} \tag{6}$$

4.3 Co-Training

Since the features we use for the supervised model provide two roughly independent views of the data, we also evaluate against the semi-supervised method of co-training (Blum and Mitchell, 1998). Here, we train two different logistic regression classifiers, each with access to only the unigrams and multiword expressions employed by the user (h_{words}) or to binary indicators of the blogs posted to and the identity of the most frequent blog (h_{blogs}). For ten iterations, we pick a random sample U' of 1,000 data points from the full dataset U and classify each using the two classifiers; each classifier then adds up to 100 of the highest-confidence predictions to the training set, retaining the class distribution balance of the initial training set; after training, the final predictive probability for an item is the product of the two trained classifiers. In a tenfold cross-validation, co-training yielded a slightly higher (but not statistically significant) accuracy over pure supervision (77.0% ± 1.8). We calculate scores for propositions in the same way as for the fully supervised case above.

5 Evaluation

For the experiments that follow, we limit the input data available to all models to only those propo-

[11] Multiword expressions were found using the method of Justeson and Katz (1995).

sitions whose subject falls within the evaluation benchmark; and include only propositions used by at least five different users, and only users who make at least five different assertions, yielding a total dataset of 40,803 users and 1.9 million propositions (81,728 unique), containing the union of all three kinds of extracted propositions from §2.2.

Each of the automatic methods that we discuss above assigns a real-valued score to propositions like OBAMA IS A SOCIALIST. Our goal in evaluation is to judge how well those model scores recover those assigned by humans in our benchmark. Since each method may make different assumptions about the distribution of scores (and normalizing them may be sensitive to outliers), we do not attempt to model them directly, but rather use two nonparametric tests: Spearman's rank correlation coefficient and cluster purity.

Spearman's rank correlation coefficient. The set of scores in the human benchmark and as output by a model each defines a ranked list of propositions; Spearman's rank correlation coefficient (ρ) is a nonparametric test of the Pearson correlation coefficient measured over the ranks of items in two lists (rather than their values). We use the absolute value of ρ to compare the degree to which the ranked propositions of two lists are linearly correlated; a perfect correlation would have $\rho = 1.0$; no correlation would have $\rho = 0.0$.

Purity. While Spearman's rank correlation coefficient gives us a nonparametric estimate of the degree to which the exact order of two sequences are the same, we can also soften the exact ordering assumption and evaluate the degree to which a ranked proposition falls on the correct side of the political continuum (i.e., not considering whether OBAMA IS A SOCIALIST is more or less conservative than OBAMA IS A DICTATOR but rather that it is more conservative than liberal). For each ranked list, we form two clusters of propositions, split at the midpoint: all scores below the midpoint define one cluster, and all scores above or equal define a second. For $N = 766$ propositions, given gold clusters $\mathcal{G} = \{g_1, g_2\}$ and model clusters $\mathcal{C}_n = \{c_1, c_2\}$ (each containing 383 propositions), we calculate purity as the average overlap for the best alignment between the two gold clusters and their model counterparts.[12]

$$\text{Purity} = \frac{1}{N}\left(\max_j |g_1 \cap c_j| + \max_j |g_2 \cap c_j|\right) \tag{7}$$

A perfect purity score (in which all items from each cluster in $\mathcal{C}$ are matched to the same cluster in $\mathcal{G}$) is 1.0; given that all clusters are identically sized (being defined as the set falling on each half of a midpoint), a random assignment would yield a score of 0.50 in expectation.

Model	Purity	Spearman's ρ
Additive (PROP.)	0.757 $\pm$0.020	0.648 $\pm$0.017
Single mem. (PROP.)	0.754 $\pm$0.019	0.628 $\pm$0.017
Single mem. (PRED.)	0.702 $\pm$0.018	0.555 $\pm$0.015
Additive (PRED.)	0.705 $\pm$0.018	0.490 $\pm$0.013
Co-training	0.695 $\pm$0.018	0.450 $\pm$0.013
LR	0.619 $\pm$0.016	0.278 $\pm$0.010
PCA (frequency)	0.518 $\pm$0.014	0.098 $\pm$0.009
PCA (counts)	0.514 $\pm$0.014	0.066 $\pm$0.008

Table 3: Evaluation. Higher is better.

Table 3 presents the results of this evaluation. For both of the models described in §3, we present results for scoring a proposition like OBAMA IS A SOCIALIST based only on the conditional predicate score (PRED.) and on a score that includes variation in the subject as well (PROP.). Since both models are fit using approximate inference with a non-convex objective function, we run five models with different random initializations and present the average across all five.

We estimate confidence intervals using the block jackknife (Quenouille, 1956; Efron and Stein, 1981), calculating purity and Spearman's ρ over 76 resampled subsets of the full 766 elements, each leaving out 10.[13] For both metrics, the two best performing models show statistically significant improvement over all other models, but are not significantly different from each other.

We draw two messages from these results:

For heavily partisan data, unsupervised methods are sufficient. In drawing on comments on politically partisan blogs, we are able to match human judgments of the political import of propositions quite well (both of the unsupervised models

[12] In this case, with two clusters on each side, the best alignment in maximal in that $g_{n,i} \rightarrow c_{n,j} \Rightarrow g_{n,\neg i} \rightarrow c_{n,\neg j}$.

[13] As a clustering metric, purity has no closed-form expression for confidence sets, and since its evaluation requires its elements to be unique (in order to be matched across clusters), we cannot use common resampling-with-replacement techniques such as the bootstrap (Efron, 1979).

described in §3 outperform their supervised and semi-supervised counterparts by a large margin), which suggests that the easiest structure to find in this particular data is the affiliation of users with their political ideologies. Both unsupervised models are able to exploit the natural structure without being constrained by a small amount of training data that may be more biased (e.g., in its class balance) than helpful. The two generative models also widely outperform PCA, which may reflect a mismatch between its underlying assumptions and the textual data we observe; PCA treats data sparsity as structural zeros (not simply missing data) and so must model not only the variation that exists between users, but also the variation that exists in their frequency of use; other latent component models may be a better fit for this kind of data.

Joint information is important. For both models, including information about the full joint probability of a subject and predicate together yields substantial improvements for both purity and the Spearman correlation coefficient compared to scores calculated from variation in the conditional predicate alone. While we might have considered variation in the predicate to be sufficient in distinguishing between political parties, we see that this is simply not the case; variation in the subject may help anchor propositions in the spectrum relative to each other.

6 Convergent Validity

The primary quantity of interest that we are trying to estimate in the models described above is the political position of an *assertion*; a user's latent political affiliation is only a helpful auxiliary variable in reaching this goal. We can, however, also measure the correlation of those variables themselves with other variables of interest, such as users' self-declarations of political affiliation and audience participation on the different blogs. Both provide measures of convergent validity that confirm the distinction being made in our models is indeed one of political ideology.

6.1 Correlation with Self-declarations

One form of data not exploited by the unsupervised models described above are users' self-declarations; we omit these above in order to make the models as general as possible (requiring only text and not metadata), but they can provide an independent measure of the distinctions our unsupervised models are learning. (The supervised baselines in contrast are able to draw on this profile information for training data.)

Approximately 12% of the users in the data input to our models (4,718 of 40,804) have affiliated self-declared profile information; the most frequent of these include *retired*, *businessman*, *student*, and *patriot*. For all of these users, we regress binary indicators of the top 25,000 unigrams in their profiles against the MAP estimate of their political affiliation in the single-membership model. Across all 5 folds, the features with the highest predictive weights for one class were *patriot*, *conservative*, *obama*, and *god* while the highest predictive weights for the other are *progressive*, *voter*, *liberal*, and *science*.

6.2 Estimating Media Audience

We can also use users' latent political ideologies to estimate the overall ideological makeup of a blog's active audience. If we assign each post to our estimate of the political ideology of its author, we find that Mother Jones has the highest fraction of comments by estimated liberals at 80.4%, while Breitbart has the highest percentage of comments by conservatives (79.5%).

Blog	% Liberal by post
Mother Jones	80.4%
NPR	67.4%
Politico	51.6%
Young Conservatives	38.0%
Daily Caller	28.4%
IJ Review	28.0%
Breitbart	20.5%

Table 4: Media audience.

This broadly accords with Mitchell et al. (2014), which finds that among the blogs in our dataset, consistently liberal respondents trust NPR and Mother Jones most, while consistent conservatives trust Breitbart most and NPR and Mother Jones the least.

7 Conclusion

We introduce the task of estimating the political import of propositions such as OBAMA IS A SOCIALIST; while much work in open information extraction has focused on learning facts such as OBAMA IS PRESIDENT from text, we are able to exploit structure in the users and communities who make such assertions in order to align them all

within the same political space. Given sufficiently partisan data (here, comments on political blogs), we find that the unsupervised generative models presented here are able to outperform other models, including those given access to supervision.

One natural downstream application of this work is fine-grained opinion polling; while existing work has leveraged social media data on Twitter for uncovering correlations with consumer confidence, political polls (O'Connor et al., 2010), and flu trends (Paul and Dredze, 2011), our work points the way toward identifying fine-grained, interpretable propositions in public discourse and estimating latent aspects (such as political affiliation) of the communities who assert them. Data and code to support this work can be found at `http://people.ischool.berkeley.edu/~dbamman/emnlp2015/`.

8 Acknowledgments

We thank Jacob Eisenstein and our anonymous reviewers for their helpful comments. The research reported in this article was largely performed while both authors were at Carnegie Mellon University, and was supported by NSF grant IIS-1211277. This work was made possible through the use of computing resources made available by the Open Science Data Cloud (OSDC), an Open Cloud Consortium (OCC)-sponsored project.

References

Faiyaz Al Zamal, Wendy Liu, and Derek Ruths. 2012. Homophily and latent attribute inference: Inferring latent attributes of Twitter users from neighbors. In *Proc. of ICWSM*.

Arthur U. Asuncion, Max Welling, Padhraic Smyth, and Yee Whye Teh. 2009. On smoothing and inference for topic models. In *Proc. of UAI*.

Pablo Barberá. 2015. Birds of the same feather tweet together: Bayesian ideal point estimation using Twitter data. *Political Analysis*, 23(1):76–91.

Avrim Blum and Tom Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *Proc. of COLT*.

Robert Bond and Solomon Messing. 2015. Quantifying social media's political space: Estimating ideology from publicly revealed preferences on Facebook. *American Political Science Review*, 109(01):62–78.

Raviv Cohen and Derek Ruths. 2013. Classifying political orientation on Twitter: It's not easy! In *Proc. of ICWSM*.

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

Marie-Catherine de Marneffe, Christopher D. Manning, and Christopher Potts. 2012. Did it happen? the pragmatic complexity of veridicality assessment. *Computational Linguistics*, 38(2):301–333.

Bradley Efron and Charles Stein. 1981. The jackknife estimate of variance. *The Annals of Statistics*, 9(3):586–596.

Bradley Efron. 1979. Bootstrap methods: another look at the jackknife. *The Annals of Statistics*, 7(1):1–26.

Jacob Eisenstein, Amr Ahmed, and Eric P. Xing. 2011. Sparse additive generative models of text. In *Proc. of ICML*.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *Proc. of EMNLP*.

Sean Gerrish and David M. Blei. 2012. How they vote: Issue-adjusted models of legislative behavior. In *NIPS*.

Thomas L. Griffiths and Mark Steyvers. 2004. Finding scientific topics. *Proceedings of the National Academy of Sciences of the United States of America*, 101(Suppl. 1):5228–5235.

Nathan Halko, Per-Gunnar Martinsson, Yoel Shkolnisky, and Mark Tygert. 2011. An algorithm for the principal component analysis of large data sets. *SIAM Journal on Scientific Computing*, 33(5):2580–2594.

Eszter Hargittai. 2015. Why doesn't Science publish important methods info prominently? `http://goo.gl/wXUtys`, May.

Jerry R. Hobbs, Douglas Appelt, John Bear, David Israel, Megumi Kameyama, and Mabry Tyson. 1993. Fastus: A system for extracting information from text. In *Proc. of HLT*.

Mohit Iyyer, Peter Enns, Jordan Boyd-Graber, and Philip Resnik. 2014. Political ideology detection using recursive neural networks. In *Proc. of ACL*.

Maojin Jiang and Shlomo Argamon. 2008. Exploiting subjectivity analysis in blogs to improve political leaning categorization. In *Proc. of SIGIR*.

John S. Justeson and Slava M. Katz. 1995. Technical terminology: some linguistic properties and an algorithm for identification in text. *Natural language engineering*, 1(1):9–27.

Leslie Kaufman. 2014. Independent Journal Review website becomes a draw for conservatives. *New York Times*, Nov. 2, 2014.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proc. of ACL*.

Andrew D. Martin and Kevin M. Quinn. 2002. Dynamic ideal point estimation via Markov Chain Monte Carlo for the U.S. Supreme Court, 1953–1999. *Political Analysis*, 10(2):134–153.

Thomas P. Minka. 2003. Estimating a Dirichlet distribution. `http://research.microsoft.com/en-us/um/people/minka/papers/dirichlet/`.

Amy Mitchell, Jeffrey Gottfried, Jocelyn Kiley, and Katerina Eva Matsa. 2014. Political polarization and media habits: From Fox News to Facebook, how liberals and conservatives keep up with politics. Technical report, Pew Research Center.

T. Mitchell, W. Cohen, E. Hruschka, P. Talukdar, J. Betteridge, A. Carlson, B. Dalvi, M. Gardner, B. Kisiel, J. Krishnamurthy, N. Lao, K. Mazaitis, T. Mohamed, N. Nakashole, E. Platanios, A. Ritter, M. Samadi, B. Settles, R. Wang, D. Wijaya, A. Gupta, X. Chen, A. Saparov, M. Greaves, and J. Welling. 2015. Never-ending learning. In *Proc. of AAAI*.

Ndapandula Nakashole and Tom M. Mitchell. 2014. Language-aware truth assessment of fact candidates. In *ACL*.

Viet-An Nguyen, Jordan Boyd-Graber, Philip Resnik, and Kristina Miler. 2015. Tea party in the house: A hierarchical ideal point topic model and its application to Republican legislators in the 112th Congress. In *Proc. of ACL*.

Brendan O'Connor, Ramnath Balasubramanyan, Bryan R. Routledge, and Noah A. Smith. 2010. From tweets to polls: Linking text sentiment to public opinion time series. In *Proc. of ICWSM*.

Souneil Park, Minsam Ko, Jungwoo Kim, Ying Liu, and Junehwa Song. 2011. The politics of comments: Predicting political orientation of news stories with commenters' sentiment patterns. In *Proc. of CSCW*.

Michael J Paul and Mark Dredze. 2011. You are what you Tweet: Analyzing twitter for public health. In *Proc. of ICWSM*.

Marco Pennacchiotti and Ana-Maria Popescu. 2011. Democrats, Republicans and Starbucks afficionados: User classification in Twitter. In *Proc. of KDD*.

Keith T. Poole and Howard Rosenthal. 1985. A spatial model for legislative roll call analysis. *American Journal of Political Science*, 29(2):357–384.

Maurice H. Quenouille. 1956. Notes on bias in estimation. *Biometrika*, 43(3/4):353–360.

Delip Rao, David Yarowsky, Abhishek Shreevats, and Manaswi Gupta. 2010. Classifying latent user attributes in Twitter. In *Proc. of SMUC*.

Christian Sandvig. 2015. The Facebook "it's not our fault" study. `http://blogs.law.harvard.edu/niftyc/archives/1062`, May.

Yanchuan Sim, Brice D. L. Acree, Justin H. Gross, and Noah A. Smith. 2013. Measuring ideological proportions in political speeches. In *Proc. of EMNLP*.

Svitlana Volkova, Glen Coppersmith, and Benjamin Van Durme. 2014. Inferring user political preferences from streaming communications. In *Proc. of ACL*.

Janyce Wiebe, Eric Breck, Chris Buckley, Claire Cardie, Paul Davis, Bruce Fraser, Diane J. Litman, David R. Pierce, Ellen Riloff, and Theresa Wilson. 2003. Recognizing and organizing opinions expressed in the world press. In *Proceedings of the 2003 AAAI Spring Symposium on New Directions in Question Answering*.

Janyce Wiebe, Theresa Wilson, and Claire Cardie. 2005. Annotating expressions of opinions and emotions in language. *Language Resources and Evaluation*, 39(2-3):165–210.

Theresa Ann Wilson. 2008. *Fine-grained subjectivity and sentiment analysis: recognizing the intensity, polarity, and attitudes of private states*. Ph.D. thesis, University of Pittsburgh.

Daniel Xiaodan Zhou, Paul Resnick, and Qiaozhu Mei. 2011. Classifying the political leaning of news articles and users from user votes. In *Proc. of ICWSM*.

Using Personal Traits For Brand Preference Prediction

Chao Yang[1], Shimei Pan[2], Jalal Mahmud[3], Huahai Yang[4], and **Padmini Srinivasan[1]**
[1]Computer Science, The University of Iowa, Iowa City, IA, USA
{chao-yang, padmini-srinivasan}@uiowa.edu
[2]University of Maryland, Baltimore County, Baltimore, MD, USA
shimei@umbc.edu
[3]IBM Research Almaden, Almaden, CA, USA
jumahmud@us.ibm.com
[4]Juji Inc, Saratoga, CA, USA
hyang@juji-inc.com

Abstract

In this paper, we present a comprehensive study of the relationship between an individual's personal traits and his/her brand preferences. In our analysis, we included a large number of character traits such as personality, personal values and individual needs. These trait features were obtained from both a psychometric survey and automated social media analytics. We also included an extensive set of brand names from diverse product categories. From this analysis, we want to shed some light on (1) whether it is possible to use personal traits to infer an individual's brand preferences (2) whether the trait features automatically inferred from social media are good proxies for the ground truth character traits in brand preference prediction.

1 Introduction

Brand preference analysis is an important topic in marketing. To induce a desired brand choice, a marketer must understand the main factors that influence a consumer's brand preferences. This task is not easy since many factors may play a role in determining one's brand preferences such as a consumer's individual characteristics and preferences as well as the properties of a brand (e.g., its perceived quality). Among consumer related factors, demographics such as age, gender and income have been studied extensively in marketing research (Evans, 1959; Elliott, 1994; Lin, 2002). In this study, we focus on analyzing a set of consumer characteristics, which have received less attention but with these features, potentially we can build more precise and more accurate brand preference prediction models. Especially, we focus on three types of personal traits: *personality*, *personal values*, and *individual needs*. *Personality* is a combination of characteristics or qualities that form an individual's distinctive character; *Personal values* reflect what are important to different individuals and what motivate them in their decision making. Moreover, all people have certain *needs* that they want to satisfy. Thus, analyzing a comprehensive set of personal traits may help us understand the way we react to a particular brand.

Previously, the relationship between personal traits and brand preference/purchase decisions has drawn limited interest in marketing research due to the difficulty in obtaining consumer traits on a large scale. Among these efforts, Westfall found that differences exist between the personalities of the owners of convertible cars and those of standard & compact cars (Westfall, 1962). Similarly, the congruence of personal and brand personality was suggested to be a predictor of consumers' brand preferences (Jamal and Goode, 2001; Dikcius et al., 2013). However, Shank & Langmeyer found personal traits less useful in building a strategic marketing tool (Shank and Langmeyer, 1994).

Given limited and sometimes conflicting results in previous research, in this study, we want to systematically investigate the relationship between a comprehensive set of personal traits and brand preferences. Specifically, we want to shed some light on (1) whether it is possible to use personal traits to predict consumer's brand preferences? (2) whether it is feasible to use automatically inferred personal traits to build brand preference prediction systems that are scalable?

Our study offers several significant contributions to the field of brand preference analysis:

1. It is the first study that includes a comprehensive set of personal traits in brand preference analysis. Our current investigation includes *personality* (5 general categories and 30 sub-

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 86–96,
Lisbon, Portugal, 17-21 September 2015.

facets), *personal values* (5 dimensions) and *individual needs* (12 dimensions). In contrast, previous work typically only included a small number of personal traits (e.g., just 5 personality traits in (Hirsh et al., 2012)).

2. It is the first study that uses personal traits obtained from both psychometric evaluation and social media analytics. The traits scores derived from psychometric tests are more accurate, which allow us to focus on the relationship between personal traits and brand preferences without the distractions from the mistakes introduced by an automated trait inference system. However, since psychometric tests require users to answer a large number of survey questions, without sufficient incentives, it is difficult to perform psychometric evaluation for a large number of people. In contrast, automatically derived trait features based on social media analytics require no user effort, and can be applied to millions of social media users.

3. Our study involves diverse brand categories such as luxury car brands, retail brands, fast food brands, and household product brands (e.g., shampoo brands). With this data, we can investigate whether the relationship between personal traits and brand preferences varies across multiple product categories.

Since the current study focuses on a comprehensive set of consumer characteristics and preferences which does not include many important brand properties such as perceived quality, risk, price and market presence, the main goal of our investigation is not to build a highly accurate brand preference prediction system. Instead, we want to first establish the feasibility of using derived trait features in building large-scale brand preference prediction systems. In the following, we first summarize some prior work, then describe the details of our experiments.

2 Related Work

Predicting brand preference is a hard problem. A large number of factors may influence customers' choices. Table 1 summarizes the factors that have been explored in previous research. Due to the scope, so far, there isn't any prior investigation that is capable of incorporating all the factors in a single model. Our study is one of the most comprehensive analyses so far. We not only investigate the influence of a large number of personal traits but also combine them with other known consumer-related features such as demographics and personal interests. We however have not included any brand-related properties such as perceived quality, risk and market presence because we do not have access to these data.

Consumer Oriented Features	Demographic characteristics (Evans, 1959; Koponen, 1960; Elliott, 1994; Lin, 2002) including Age, Education, Gender, Family dimensions, Marital Status, Ethnic group, Geographic location, Social Class, Community
	Attitude (Bass and Talarzyk, 1972; Haley and Case, 1979), Personalities (Westfall, 1962; Shank and Langmeyer, 1994; Myszkowski and Storme, 2012), Needs (Evans, 1959), Trust (Chaudhuri and Holbrook, 2001), Customer Satisfaction (Bryant and Cha, 1996; Mittal and Kamakura, 2001; Olsen, 2002), Brand loyalty (Olsen, 2002), Group Influences (Witt and Bruce, 1972), Consumers' memory (Hutchinson et al., 1994), Aspirations (Truong et al., 2010), Purchase history (Dong and Stewart, 2012), Mental Accounting (Thaler, 1985), Involvement (Celsi and Olson, 1988), Social Influence (Wood and Hayes, 2012)
Brand-Related Factors	Price, Market presence, Market response in (Papadopoulos et al., 1990), Promotion (Graeff, 1996), Brand name (Zinkhan and Martin Jr, 1987; Klink, 2001), Quality (Dickerson, 1982; Olsen, 2002), Service quality, Equity, Value in (Hellier et al., 2003), Country of origin (Han and Terpstra, 1988; Tse and Gorn, 1993), Product image (Westfall, 1962), Brand personality (Aaker, 1997), Pioneering advantages (Carpenter and Nakamoto, 1989), Recallability (Costley and Brucks, 1992), Communication (advertising) (Nicholls et al., 2011; Liu and Tang, 2011), Social environment (Witt and Bruce, 1972), Perceived risk (Peter and Ryan, 1976; Campbell and Goodstein, 2001), Product attributes (Semeijn et al., 2004), Product visibility (Sutton and Riesz, 1979)

Table 1: Features explored in previous studies

In recent social media studies, Wang et al. utilized customer reviews to predict coffee brand satisfaction (Wang et al., 2013). Also, there is a large-body of work trying to predict brand preferences based on one's social media posts. Most of these work, however is performed in the context of sentiment analysis. In sentiment analysis, the main focus was to infer the sentiment associated with a post that mentions a particular brand/product. For example, Kim et al. collected 600,000 tweets that contain smartphone-related keywords and then performed sentiment analysis to infer whether a user's attitude toward a particular mobile phone is positive or negative (Kim et al., 2012). Similarly, Mostafa analyzed the sentiment associated with 3,500+ tweets, which showed a generally positive consumer sentiment towards several famous brands (Mostafa, 2013). In contrast, our trait-based analysis is more general since it does not require users to explicitly express their opinions about a specific brand. For example, to infer whether an individual likes BMW or not, with sentiment analysis, a user has to ex-

plicitly express his opinion towards BMW (e.g. Driving BMW is exciting!). In contrast, with our system, if we know that he likes to seek excitement (excitement, a needs dimension) and enjoys luxury products (Hedonism, a values dimension), we can guess he may like BMW although he has never explicitly mentioned BMW in his social media posts before. This difference is important since among the millions of products on social media, only a small number of products have been explicitly rated/mentioned by a particular user.

In summary, brand preferences may be influenced by many consumer and brand-related factors. Previous research has not paid sufficient attention to the influence of personal traits. In addition, most previous studies used psychometric surveys which are impractical in mass marketing since it is unlikely that a large number of customers would take the time to answer lengthy survey questions. In this study, we focus on investigating the feasibility of using automatically inferred personal traits in large-scale brand preference prediction. Next, we describe the dataset we collected to support this study.

3 Data Collection

To investigate how personal traits are related to an individual's brand preferences, we collected two datasets. In the first dataset, in addition to brand preferences, we also used standard psychometric tests to obtain clean and accurate personal trait measures. With this dataset, we can build and evaluate brand preference prediction models that use accurate personal traits. In contrast, the second dataset is used to build and evaluate brand preference prediction models that use trait features automatically inferred from social media. By comparing the models built from both datasets, we can answer questions such as: (1) whether personal traits are useful in predicting brand preferences (2) whether the traits automatically inferred from social media are useful in predicting brand preferences.

To collect these datasets, we designed two Amazon Mechanical Turk (MTurk) [1] tasks. All the MTurk participants are from the US since people outside the US may be unfamiliar with some of the brands. In the following, we describe the details of each MTurk task.

[1] http://mturk.com/

Category	Brand
Beverage (2)	Coca-Cola, Pepsi
Luxury Car (3)	BMW, Cadillac, Lexus
Fast Food (4)	Chipotle, McDonald's, Panera Bread (PB) , Subway
Retail (4)	Kohl's, Macy's, Nordstrom, Target
Shampoo (4)	Head & Shoulders (HS), Herbal Essences (HE), Pantene, Suave
Smart Phone (5)	HTC, iPhone, Samsung, SONY, Nokia

Table 2: Selected brand categories and brands

3.1 Task 1: PTBP Survey

To collect the first dataset, we conducted a Personal Traits & Brand Preferences (PTBP) survey. Our trait survey includes five parts designed to measure three types of personal traits: *personality*, *values* and *needs* plus demographics and personal interests. Specifically, since the Big-Five model of personality is the most popular model of personality traits among personality psychologists, we adopted a standard survey for Big 5 personality. Here to limit the time MTurkers need to spend on the survey, instead of the full 300-item personality test, we used the shorter 50-item IPIP survey (Goldberg, 1993) which will score a user along 5 general personality dimensions: openness, conscientiousness, extraversion, agreeableness and neuroticism. However, with the shorter survey, we can not obtain the scores for 30 additional personality facets. Similarly, we used the standard 21-item PVQ survey to obtain the *values* defined in Schwartz's theory of basic values (Schwartz, 2003). We also used the 35-item BNS survey to obtain the *needs* defined in Ford's needs model (Ford, 2005; Yang and Li, 2013). In addition, we also included survey questions about a user's demographics (e.g., gender, age, marital status, education and income) and personal interests (e.g,, automobile, sports, movies, travel) since they were used in some previous studies on brand preference (Pennacchiotti and Gurumurthy, 2011; Lin, 2002). Finally for each user, we collected her preferences for 22 brands in six categories. We have chosen well-known brands from diverse groups ranging from mobile phones to retail stores and fast food restaurants. The brands within each category are often competitors. For each brand in each category, we asked users to rate their preferences using a 5-point scale: *Love, Like, Neutral, Dislike* and *Hate*. A user can choose "n/a" if she has no knowledge of a particular brand. In total, we have collected the data from 1,207 MTurkers. To ensure the quality of the data collected from MTurk, we also included several validation ques-

tions. The validation questions are pairs of questions that are paraphrases of each other. If the answers to a pair of validation questions are significantly different, the user data are excluded from our analysis. Our final dataset has 1,017 valid responses. Table 2 lists all the brands used in our study. All the measures used in our PTBP survey are listed in Table 3.

Question Category	Features
Personalities (5)	Big-five personalities: Agreeableness, Conscientiousness, Extraversion, Neuroticism, Openness
Values (5)	Conservation, Hedonism, Openness to change, Self Enhancement, Self Transcendence
Needs (12)	Challenge, Closeness, Curiosity, Excitement, Harmony, Ideal, Liberty, Love, Practicality, Self-expression, Stability, Structure
Topics of Interest (20)	Politics, Business, Technology, Science, Health, Sports, Education, Books, Dance, Movie, Music, Television, Theater, Video games, Automobiles, Dining wine, Fashion style, Home garden, International news, US news
Location (2)	City, State
Demographics (6)	Gender, Age, Marital status, Have children, Education status, Income
Brand Preferences (22)	Ratings for all the 22 brands

Table 3: PTBP Survey Feature Summary

3.2 Task 2: TAE Survey

The data collected in the Text Analytics Evaluation (TAE) survey are used to study the correlation between the trait features inferred from a person's social media posts (e.g., tweets) and his brand preferences. Before the TAE survey, the participants were first asked to verify whether they had a Twitter account, if so, provide us their Twitter IDs. The users also agreed that we could access their tweets after the survey. Since our automated trait inference system relies on linguistic cues derived from a person's Twitter posts, to ensure we can have a stable and reliable reading of one's personal traits from his tweets, only active Twitter users with over 50 tweets (excluding retweets) can participate this survey. Since the majority of MTurkers are not active Twitter users, to increase the size of our data, in addition to MTurk, we also directly invited random Twitter users to participate in our TAE survey.

In addition to Twitter IDs, we also asked participants to provide their preferences for the same 22 brands as those used in the PTBP survey. Similarly, to filter out data by people who do not follow instructions, we also added two validation questions. In total, in the TAE survey, we have collected data from 659 participants, out of which 608 are valid. (550 valid ones are from MTurk, and 109 are from direct Twitter invitation).

3.3 Data Preparation

To obtain the trait scores for each user based on his answers in the PTBP survey, we first computed the raw trait scores based on the original survey guidelines. Since different surveys used different scales, we normalized the trait scores by using their rank percentile (e.g., top 1%, top 5%). As a result, all the normalized personal trait scores are between 0 and 1.

Moreover, for each of the 20 topics of interest, we created a binary variable, indicating whether a participant is interested in a specific topic. In addition, each demographics feature such as age, education, income, was first mapped to an integer and then normalized into a number between 0 and 1.

To derive the trait scores for a user in the TAE survey, we crawled all the tweets in his Twitter account. Since personal traits are inferred from the text authored by a user, we discarded all the retweets. Due to the restrictions of the Twitter API, we can only crawl a maximum of 3,200 tweets for each user[2].

Recent research in psycholinguistics has shown it is possible to automatically infer personal traits from one's linguistic footprints such as tweets and blogs (Yarkoni, 2010; Chen et al., 2014; Yang and Li, 2013). Here, we used a similar approach. Specifically, given input text authored by a user (e.g., tweets), our system computed the word counts of different psychologically-meaningful word categories defined in the Linguistic Inquiry and Word Count (LIWC) dictionary (Pennebaker et al., 2001). The LIWC counts were then used to build prediction models to correlate one's word usage with his ground truth personal traits obtained via a prior psychometric survey. Then the built models were used to automatically infer a user's personal traits. Based on a preliminary evaluation with 250 participants, more than 80 percent of them, scores for traits that were inferred for all three models correlated significantly with survey-based scores ($p<0.05$ and correlation coefficient between 0.05 and 0.8). Specifically, scores that were derived by our system correlated with survey-based scores for 80.8% of participants' $\widehat{\text{Big}}$ Five scores ($p<0.05$ and correlation coefficients between 0.05 and 0.75), for 86.6% of participants'

[2]https://dev.twitter.com

Needs scores ($p<0.05$ and correlation coefficient between 0.05 and 0.8), and for 98.21% of participants' *Values* scores ($p<0.05$ and correlation coefficients between 0.05 and 0.55). Moreover, the participants also rated on a five-point scale how well each derived characteristic matched their perceptions of themselves, and their ratings suggest that the inferred characteristics largely matched their self-perceptions. Specifically, means of all ratings were above 3 ("somewhat") out of 5 ("perfect"): 3.4 (with a std. of 1.14) for *Big Five*, 3.39 (with a std. of 1.34) for *Needs*, and 3.13 (with a std. of 1.17) for *Values*.

In addition to personal traits, we also included topics of interest in the TAE dataset. They were automatically inferred from tweets using Latent Dirichlet Allocation (LDA) (Blei et al., 2003). Since we need a large Twitter dataset to mine a list of general topics of interest, our current tweet collection is not sufficient. Therefore, we use a separate and much larger Twitter dataset from 10,000 randomly selected Twitter users. For each user, we crawled his tweets and then aggregated them into a big document, one for each Twitter user. As a result, we have 10,000 documents in our dataset. We then built an LDA topic model using this dataset. From the LDA inference results, we can infer a user's topics of interest. Basically, for a given user u, LDA outputs a per user topic distribution Θ_u, which is a T-dimensional vector where T is the number of topics. The value $\theta_{u,i}$ is an indication of how likely Topic i is mentioned in user u's tweets. The higher $\theta_{u,i}$ is, the more likely that user u is interested in topic i. Table 4 shows some of the topics automatically learned by LDA.

Manually Labeled Name	**Top 10 Topic keywords**
Mobile phone	google app apple data mobile iphone web android tech windows
Alcohol	drinking beer wine vegas earned badge tonight ale bar ipa
Travel	travel hotel cruise ttot trip family great world tips top
Driving	slow drive traffic lane car north south blvd lanes crash
Game	game app ipad video ve games gameinsight free android xbox

Table 4: Selected topics and top words from LDA

As a summary, table 5 shows all the features from the TAE survey, including those automatically inferred from tweets. For *personality*, following the same procedure defined in (Yarkoni, 2010), in addition to the Big five personality dimensions, our system is able to automatically extract 30 additional personality facets for the TAE dataset.

Question Category	**Features**
Survey Features	
Twitter ID	
Brand Preference	for 22 Brands
Derived Features	
Personalities (35)	Big-five personalities plus their sub-facets automatically inferred from tweets (Yarkoni, 2010)
Values (5)	Same as those in Table 3 but inferred from tweets
Needs (12)	Same as those in Table 3 but inferred from tweets
Topics of Interest (50)	Automatically inferred using a topic model
Location (2)	City, State inferred from IP address
Twitter Metadata (5)	Number of tweets, Number of followers, Number of friends, Favorite count, Listed count
Online Behavior (31)	Avg. number of tweets posted in each of the 7 days in a week, and each of the 24 hours in a day.

Table 5: TAE survey feature summary

In the following section, we explain two analyses we performed on these datasets.

4 Experiment 1

The main objective of this analysis is twofold: (1) to understand why people like or dislike a brand. (2) to build a computational model that automatically differentiates people who have *positive*, *negative*, or *neutral* opinions about a brand.

4.1 Definition and Statistics

For each brand in this study, we define people who have *positive* opinions as those who gave a *love* or *like* rating in their brand preference surveys. Similarly, people who have *negative* opinions are those who gave a *hate* or *dislike* rating. People who gave a *neutral* rating are in the *neutral* category. Table 6 shows the number of instances in each of the three categories for each brand.

4.2 Classification

In this experiment, we want to investigate whether it is possible to differentiate people who have (*Positive*, *Negative*, or *Neutral*) opinions towards different brands. For each brand, we built three-way classifiers using different classification algorithms including AdaBoost (Freund and Schapire, 1996), Decision Tree (C4.5) (Quinlan, 1993), Logistic Regression, Naive Bayes, Random Forest (Breiman, 2001), and SVM (Platt, 1999). In addition, for SVM, we have tested different kernels including polynomial kernel, pearson VII function-based universal kernel (Üstün et al., 2006), and the radial basis function kernel. They are all implemented in the Weka machine learning toolkit (Hall

Brands	PTBP Survey			TAE Survey		
	Positive	Negative	Neutral	Positive	Negative	Neutral
Coca-Cola	519	190	115	363	73	67
Pepsi	360	315	153	261	132	110
BMW	382	85	113	280	34	183
Cadillac	239	162	144	212	61	226
Lexus	361	80	119	255	44	200
Chipotle	443	95	103	298	46	114
McDonald's	304	383	143	270	152	84
PB	437	99	135	316	36	119
Subway	534	176	123	371	75	61
Kohl's	402	154	202	298	58	147
Macy's	293	137	210	252	37	207
Nordstrom	227	196	176	177	39	256
Target	640	87	106	430	18	58
HS	314	260	166	246	68	175
HE	388	138	171	259	46	181
Pantene	377	165	173	259	52	177
Suave	362	225	173	238	85	175
HTC	280	111	165	180	46	249
iPhone	426	218	83	338	98	67
Samsung	564	62	104	344	28	128
SONY	114	222	181	61	82	316
Nokia	161	266	215	126	94	263

Table 6: Number of instances in each category

et al., 2009). Since our current goal is not to build the best brand preference prediction system, but to show the feasibility of building brand preference prediction systems that are scalable to millions of users, we ran all our classifiers using the default parameter settings from Weka (E.g. for Random Forest, we used 10 trees. The umber of features was set to log2(number of all features)+1.). We expect in the future, by optimizing model parameters, we can further improve the prediction power of each model.

The baseline classifiers classify every data instance into the majority class. Among all the classifiers we tested, we found that overall Naive Bayes has the best performance on both the PTBP and the TAE datasets. In the following, we report the average F-scores and AUC across 22 different brands using Naive Bayes with 10-fold cross validations. We created models that use all the user features and also those that use only trait features. Table 7 shows the results.

Best Classifier	PTBP		TAE	
	F	AUC	F	AUC
All features	0.483	0.569	0.501	0.547
Traits only	0.475	0.556	0.502	0.528
Baseline	0.396	0.493	0.444	0.490

Table 7: 3-Way Classification Results

Overall, all the classifiers performed significantly better than the baselines ($p<0.05$). Moreover, the models using all the features performed similarly to those using only trait features. The differences are not statistically significant. In addition, comparing the models trained on the PTBP data with those on the TAE dataset, their performances are very similar, although the exact numbers are not directly comparable since they are from two different datasets.

To break down the results by product category, in Table 8, we list the per-brand classification results using only the trait features. The numbers in the parentheses show the F-score percentage increase from the baselines. In general, models with trait features did much better than the baselines on both datasets. But their effectiveness varied from one brand to another. For example, the trait features were very effective in predicting user preferences for *Cadillac* (50.8% increase on the PTBP dataset and 68.5% increase on the TAE dataset). In contrast, there was barely any improvement for *Target*. After inspecting the data, it seems this may be caused by the distribution of the data. For instance, the Target TAE data was very skewed. There were 430 people who had positive opinions about Target versus 18 people who had negative opinions. Since the baseline predicts "all people like Target", which resulted in a pretty high F-score (0.781), any further improvement over this baseline became more difficult.

Brand	PTBP			TAE		
	BL F	Best F	↑%	BL F	Best F	↑%
Coca-Cola	0.487	0.503	3.3%	0.605	0.605	0.0%
Pepsi	0.263	0.427	62.4%	0.355	0.416	17.2%
BMW	0.522	0.562	7.7%	0.406	0.499	22.9%
Cadillac	0.266	0.401	50.8%	0.282	0.474	68.1%
Lexus	0.505	0.531	5.2%	0.346	0.5	44.5%
Chipotle	0.564	0.59	4.6%	0.513	0.53	3.3%
McDonald's	0.291	0.42	44.3%	0.371	0.459	23.7%
PB	0.513	0.528	2.9%	0.539	0.58	7.6%
Subway	0.501	0.525	4.8%	0.618	0.623	0.8%
Kohl's	0.368	0.466	26.6%	0.441	0.506	14.7%
Macy's	0.288	0.465	61.5%	0.342	0.5	46.2%
Nordstrom	0.207	0.421	103.4%	0.381	0.467	22.6%
Target	0.668	0.67	0.3%	0.781	0.781	0.0%
HS	0.253	0.399	57.7%	0.337	0.454	34.7%
HE	0.397	0.43	8.3%	0.371	0.494	33.2%
Pantene	0.363	0.452	24.5%	0.368	0.51	38.6%
Suave	0.307	0.375	22.2%	0.309	0.434	40.5%
HTC	0.337	0.411	22.0%	0.361	0.455	26.0%
iPhone	0.432	0.501	16.0%	0.54	0.557	3.2%
Samsung	0.673	0.673	0.0%	0.561	0.574	2.3%
SONY	0.258	0.407	57.8%	0.561	0.575	2.5%
Nokia	0.243	0.397	63.4%	0.384	0.432	12.5%

Table 8: Classification Results By Brand

In summary, for the task of differentiating people who have *positive*, *negative*, or *neutral* opinions towards different brands, automatically inferred traits can be a good proxy for the clean data derived from psychometric surveys. Models based on the trait features inferred from social media can perform similarly to those using a much larger set of clean features. This result is encouraging since

it implies that it is possible to build large-scale brand preference prediction systems that do not require costly psychometric surveys.

4.3 Top Features

In this study, we want to find out what are the most significant features that can be used to differentiate a brand's likers from dislikers. The feature selection was conducted using logistic regression in SPSS[3]. Due to the page limit, we cannot list all the significant features for all the 22 brands. Here we only show the most important features in predicting people who like and dislike luxury car brands based on the PTBP dataset (Table 9). Based on the regression analysis, all the features are significantly associated with brand preferences ($p < 0.05$). In this table, personal trait features are highlighted and followed by their types: P (*Personalities*), V (*Values*), and N (*Needs*). "+" or "-" means the features contribute positively or negatively to the model. As shown in the table, more than half of all the top features are trait features. For example, the No. one trait feature to differentiate BMW likers from dislikers is *ideal*, a trait associated with people who have a desire for perfection. For Cadillac, the top trait is *hedonism*, which is often associated with people who pursue pleasure and sensuous gratification in life. For Lexus, the most useful feature is *self-expression*, a trait often associated with people who have a desire to assert their own identifies. Other interesting findings include that females are less likely to be a fan of a luxury car brand than males. This is true across all three luxury car brands.

BMW		***Cadillac***		***Lexus***	
ideal (N)	+	have children(no)	-	sports	+
love (N)	+	television	+	**self expression (N)**	+
conscientiousness (P)	+	**hedonism (V)**	+	television	+
gender(female)	-	home garden	-	**self enhancement (V)**	+
us news	+	gender(female)	-	fashion style	+
health	+	**conservation (V)**	+	theater	-
hedonism (V)	-	**self enhancement (V)**	-	**agreeableness (P)**	+
challenge (N)	-	science	+	**openness to change (V)**	-
conservation (V)	-	**love (N)**	-	**curiosity (N)**	+
self enhancement (V)	+	theater	-	gender(female)	-

Table 9: Top 10 features for predicting opinions toward cars

5 Experiment 2

In the previous section, we demonstrated that given a particular brand such as Pepsi, it is possible to automatically differentiate the people who have positive, negative or neutral opinions. In this section, we try to answer a different question: given a list of competing brands in the same product category, can we automatically rank a user's preferences of these brands? For example, given popular beverage brands such as Pepsi and Coca-Cola, can we automatically predict whether a person will like Pepsi or Coca-Cola more?

5.1 Average Rank for Each Brand

For each user, we rank all the brands in each product category based on his preferences in the survey (e.g., 1 means most preferred brand). We aggregate the ranks from all the users and show the overall brand preference ranks for both datasets. As shown in table 10, the overall brand preference ranks for the PTBP and TAE surveys are highly correlated. Half of the product categories have exact the same preference ranks for all the products; The other half has only one slightly mis-matched rank in each product category. This suggests that the population participated in the PTBP and TAE survey has very similar brand preference distributions. In the future, it maybe interesting to investigate how this rank is related to different brands' market share.

	PTBP	**TAE**
Beverage	1. Coca-cola 2. Pepsi	1. Coca-cola 2. Pepsi
Car	1. BMW 2. Lexus 3. Cadillac	1. BMW 2. Lexus 3. Cadillac
Fast Food	1. Chipotle 2. Panera Bread 3. Subway 4. McDonald's	1. Panera Bread 2. Chipotle 3. Subway 4. McDonald's
Retail	1. Target 2. Macy's 3. Kohl's 4. Nordstrom	1. Target 2. Kohl's 3. Macy's 4. Nordstrom
Shampoo	1. Herbal Essences 2. Pantene 3. Suave 4. Head & Shoulders	1. Herbal Essences 2. Pantene 3. Head & Shoulders 4. Suave
Smart Phone	1. Samsung 2. iPhone 3. HTC 4. Nokia 5. SONY	1. Samsung 2. iPhone 3. HTC 4. Nokia 5. SONY

Table 10: Overall preference rank

5.2 Rank Correlation

To predict the rank of a product in each category, we trained a multi-class classifier to estimate how likely a user will like a brand. For example, for smart phone brands, since we have four competing brands, we train a 4-way classifier to estimate the likelihood a person likes iPhone, HTC, Nokia and Sony. We then output the preference rank based on the estimated likelihood. Higher likelihood means a stronger preference. We also built

[3]http://www-01.ibm.com/software/analytics/spss/

two types of models, one used all the user features, the other used traits only. We applied them to both the PTBP and the TAE datasets.

Since our model and the ground truth both produce a ranked list for each product category, here we used rank correlation analysis to evaluate the quality of the predicted ranks. For each user and each product category, we computed the Spearman's rank correlation coefficient ρ. If the coefficient ρ is 1, there is a perfect positive correlation between the predict rank and the ground truth (i.e. both produce identical ranks). If ρ is -1, there is a perfect negative correlation between the predicted rank and the ground truth (i.e., the rank predicted by the system is exactly the opposite of the ground truth). If ρ is 0 , then the predicted rank and the ground truth are randomly related. For each product category, we report the average ρ across all the users.

	PTBP		TAE	
	All Features	**Traits Only**	**All Features**	**Traits Only**
Brand	**avg. ρ**	**avg. ρ**	**avg. ρ**	**avg. ρ**
Beverage	0.264	0.301	0.234	0.372
Car	0.322	0.345	0.461	0.447
Fast Food	0.359	0.326	0.328	0.292
Retail	0.326	0.341	0.553	0.505
Shampoo	0.187	0.116	0.284	0.258
Smart Phone	0.414	0.403	0.497	0.545
All Avg.	0.312	0.305	0.393	0.403

Table 11: Evaluating predicted ranks

We use the overall rank data in Table 10 as our baseline. Specifically, for each product category, the baseline always ranks all its brands based on the average ranks defined in Table 10. For each user and each product category, we compute the ρ between the user's ground truth rank in the survey and the rank produced by the baseline. We compute the average ρ across all the users and all the product categories to represent the baseline performance. For the PTBP data, the average ρ for the baseline is 0.193. For the TAE data, the average ρ is 0.060.

There are several main findings from these results. First, for all the product categories, the predicted ranks are all significantly and positively correlated with the ground truth ($p < 0.05$). Also, our models perform significantly better than the non-personalized ranks produced by the baseline. This result is important because it shows that there is a stable and statistically significant agreement between the predicted ranks and the ground truth and the personalized models with additional trait features perform significantly better than the non-personalized baseline system (on PTBP, the average ρ of the model with personal traits is 0.305 versus 0.193 of the baseline. It is 0.403 versus 0.060 on the TAE dataset). Second, the performance on the TAE dataset is better than that on the PTBP dataset (e.g., the average ρ is 0.403 on TAE versus 0.305 on PTBP when only trait features were used). This may be due to the fact that in the TAE dataset, in addition to the Big 5 personality features, we also automatically extracted 30 personality sub-facets from tweets using the procedure described in (Yarkoni, 2010). These finer-grained personality features are not available in the PTBP dataset. This result is encouraging since it suggests that using automatically inferred traits can predict brand preferences as well as if not better than the clean trait features that can be obtained only through costly psychometric evaluations. Finally, for our models, since the overall correlation coefficients ρ are between 0.3 and 0.4, the strength of these correlations is moderate. Thus, it may not be sufficient to build an accurate brand preference prediction system with only user features. Other features especially brand-related features as well as features that capture the compatibility of a brand and a user are needed.

5.3 Top Features

We used multinomial logistic regression to find the most significant predicting features for each brand category. We show the feature ranks by significance for survey data in Table 12 and 13. Almost all of the top 10 features for each brand are significantly correlated with the ranks. Again, the personal traits features are highlighted and followed by their types: P (Personalities), V (Values), and N (Needs).

6 Conclusion and Future Direction

In this paper, we present a comprehensive analysis of the relationship between personal traits and brand preferences. Our study includes a large number of personal traits including personality, personal values and individual needs. We collect two datasets: one contains clean user features obtained from psychometric surveys; The other includes noisy users features derived automatically from social media posts. We investigate the influence of personal traits in two scenarios: (1) in differentiating people who have *positive*, *negative*, or *neutral* opinion about a brand, (2) in ranking

Beverage	*Car*	*Fast Food*
practicality (N)	**self_enhancement (V)**	education_status
us_news	**hedonism (V)**	gender
stability (N)	television	**conservation (V)**
science	sports	sports
curiosity (N)	**ideal (N)**	video_games
have_children	**agreeableness (P)**	marital_status
agreeableness (P)	**excitement (N)**	science
love (N)	gender	age
books	health	automobiles
marital_status	dining_wine	business
Smart Phone	***Retail***	***Shampoo***
books	**stability (N)**	gender
ideal (N)	**structure (N)**	age
dining_wine	automobiles	**openness (P)**
closeness (N)	health	movie
have_children	international_news	education_status
income	education	**structure (N)**
television	have_children	**curiosity (N)**
self_enhancement (V)	**practicality (N)**	**openness_to_change (V)**
marital_status	**conservation (V)**	theater
automobiles	gender	**self_transcendence (V)**

Table 12: Top 10 features for predicting rank correlation (PTBP)

Beverage	*Car*	*Fast Food*
activity_level (P)	**altruism (P)**	friend_count
immoderation (P)	**adventurousness (P)**	**sympathy (P)**
altruism (P)	**hedonism (V)**	**conservation (V)**
intellect (P)	**openness (P)**	all_tweet_count
cautiousness (P)	**trust (P)**	**self_efficacy (P)**
extraversion (P)	**artistic_interests (P)**	**stability (N)**
friendliness (P)	**sympathy (P)**	**altruism (P)**
self_discipline (P)	**morality (P)**	**depression (P)**
openness (P)	**liberalism (P)**	**liberty (N)**
closeness (N)	listed_count	**gregariousness (P)**
Smart Phone	***Retail***	***Shampoo***
neuroticism (P)	**openness_to_change (V)**	**cautiousness (P)**
openness (P)	**love (N)**	**cooperation (P)**
achievement_striving (P)	**immoderation (P)**	**intellect (P)**
altruism (P)	**sympathy (P)**	**self_consciousness (P)**
anger (P)	**hedonism (V)**	**morality (P)**
assertiveness (P)	all_tweet_count	**harmony (N)**
cautiousness (P)	**activity_level (P)**	**activity_level (P)**
depression (P)	**trust (P)**	**vulnerability (P)**
dutifulness (P)	**cautiousness (P)**	**immoderation (P)**
immoderation (P)	**liberty (N)**	**openness (P)**

Table 13: Top 10 features for predicting rank correlation (TAE)

a user's preference of competing brands within a product category. Our findings demonstrated that it is possible to use personal traits in predicting a user's brand preferences. Moreover, we have also shown that automatically inferred user features are good proxies for the clean trait features that can be acquired only from costly psychometric surveys. This work may have significant impact on the field of brand preference analysis since this suggests that it is possible for businesses to build scalable marketing tools to identify and target potential customers on social media.

Brand preference prediction is a hard problem. So far, we have focused primarily on user features. To further improve the prediction accuracy, in the future, we will extend our current study by incorporating new features such as the properties of a brand as well social influence from people in one's social network.

References

Jennifer L Aaker. 1997. Dimensions of brand personality. *Journal of Marketing Research*, pages 347–356.

Frank M Bass and W Wayne Talarzyk. 1972. An attitude model for the study of brand preference. *Journal of Marketing Research*, pages 93–96.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022.

Leo Breiman. 2001. Random forests. *Machine Learning*, 45(1):5–32.

Barbara Everitt Bryant and Jaesung Cha. 1996. Crossing the threshold. *Marketing Research*, 8(4):20–28.

Margaret C Campbell and Ronald C Goodstein. 2001. The moderating effect of perceived risk on consumers evaluations of product incongruity: Preference for the norm. *Journal of Consumer Research*, 28(3):439–449.

Gregory S Carpenter and Kent Nakamoto. 1989. Consumer preference formation and pioneering advantage. *Journal of Marketing Research*, pages 285–298.

Richard L Celsi and Jerry C Olson. 1988. The role of involvement in attention and comprehension processes. *Journal of Consumer Research*, pages 210–224.

Arjun Chaudhuri and Morris B Holbrook. 2001. The chain of effects from brand trust and brand affect to brand performance: the role of brand loyalty. *Journal of Marketing*, 65(2):81–93.

Jilin Chen, Gary Hsieh, Jalal U Mahmud, and Jeffrey Nichols. 2014. Understanding individuals' personal values from social media word use. In *Proceedings of the 17th ACM conference on Computer supported cooperative work & social computing*, pages 405–414. ACM.

Carolyn L Costley and Merrie Brucks. 1992. Selective recall and information use in consumer preferences. *Journal of Consumer Research*, pages 464–474.

Kitty G Dickerson. 1982. Imported versus us-produced apparel: Consumer views and buying patterns. *Home Economics Research Journal*, 10(3):241–252.

Vytautas Dikcius, Eleonora Seimiene, and Ermita Zaliene. 2013. Congruence between brand and consumer personalities. *Economic and Management*, 18(3):526–536.

Diansheng Dong and Hayden Stewart. 2012. Modeling a households choice among food store types. *American Journal of Agricultural Economics*, 94(3):702–717.

Richard Elliott. 1994. Exploring the symbolic meaning of brands. *British Journal of Management*, 5(s1):S13–S19.

Franklin B Evans. 1959. Psychological and objective factors in the prediction of brand choice ford versus chevrolet. *The Journal of Business*, 32(4):340–369.

J Kevin Ford. 2005. *Brands Laid Bare: Using Market Research for Evidence-based Brand Management*. John Wiley & Sons.

Yoav Freund and Robert E Schapire. 1996. Experiments with a new boosting algorithm. In *Thirteenth International Conference on Machine Learning*, volume 96, pages 148–156.

Lewis R Goldberg. 1993. The structure of phenotypic personality traits. *American Psychologist*, 48(1):26.

Timothy R Graeff. 1996. Using promotional messages to manage the effects of brand and self-image on brand evaluations. *Journal of Consumer Marketing*, 13(3):4–18.

Russell I Haley and Peter B Case. 1979. Testing thirteen attitude scales for agreement and brand discrimination. *The Journal of Marketing*, pages 20–32.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. 2009. The weka data mining software: an update. *ACM SIGKDD explorations newsletter*, 11(1):10–18.

C Min Han and Vern Terpstra. 1988. Country-of-origin effects for uni-national and bi-national products. *Journal of International Business Studies*, pages 235–255.

Phillip K Hellier, Gus M Geursen, Rodney A Carr, and John A Rickard. 2003. Customer repurchase intention: a general structural equation model. *European Journal of Marketing*, 37(11/12):1762–1800.

Jacob B Hirsh, Sonia K Kang, and Galen V Bodenhausen. 2012. Personalized persuasion tailoring persuasive appeals to recipients personality traits. *Psychological Science*, 23(6):578–581.

J Wesley Hutchinson, Kalyan Raman, and Murali K Mantrala. 1994. Finding choice alternatives in memory: Probability models of brand name recall. *Journal of Marketing Research*, pages 441–461.

Ahmad Jamal and Mark MH Goode. 2001. Consumers and brands: a study of the impact of self-image congruence on brand preference and satisfaction. *Marketing Intelligence & Planning*, 19(7):482–492.

Jong Soo Kim, Ming Hao Yang, Young Jin Hwang, Sang Hoon Jeon, KY Kim, IS Jung, Chi-Hawn Choi, Wan-Sup Cho, and JH Na. 2012. Customer preference analysis based on sns data. In *Cloud and Green Computing (CGC), 2012 Second International Conference on*, pages 609–613. IEEE.

Richard R Klink. 2001. Creating meaningful new brand names: A study of semantics and sound symbolism. *Journal of Marketing Theory and Practice*, pages 27–34.

Arthur Koponen. 1960. Personality characteristics of purchasers. *Journal of Advertising Research*.

Chin-Feng Lin. 2002. Segmenting customer brand preference: demographic or psychographic. *Journal of Product & Brand Management*, 11(4):249–268.

Kun Liu and Lei Tang. 2011. Large-scale behavioral targeting with a social twist. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 1815–1824. ACM.

Vikas Mittal and Wagner A Kamakura. 2001. Satisfaction, repurchase intent, and repurchase behavior: investigating the moderating effect of customer characteristics. *Journal of Marketing Research*, 38(1):131–142.

Mohamed M Mostafa. 2013. More than words: Social networks text mining for consumer brand sentiments. *Expert Systems with Applications*, 40(10):4241–4251.

Nils Myszkowski and Martin Storme. 2012. How personality traits predict design-driven consumer choices. *Europes Journal of Psychology*, 8(4):641–650.

JAF Nicholls, Sydney Roslow, and Henry A Laskey. 2011. Sports event sponsorship for brand promotion. *Journal of Applied Business Research (JABR)*, 10(4):35–40.

Svein Ottar Olsen. 2002. Comparative evaluation and the relationship between quality, satisfaction, and repurchase loyalty. *Journal of the Academy of Marketing Science*, 30(3):240–249.

Nicolas Papadopoulos, Louise A Heslop, and Gary Bamossy. 1990. A comparative image analysis of domestic versus imported products. *International Journal of Research in Marketing*, 7(4):283–294.

Marco Pennacchiotti and Siva Gurumurthy. 2011. Investigating topic models for social media user recommendation. In *Proceedings of the 20th international conference companion on World wide web*, pages 101–102. ACM.

James W Pennebaker, Martha E Francis, and Roger J Booth. 2001. Linguistic inquiry and word count: Liwc 2001. *Mahway: Lawrence Erlbaum Associates*, 71:2001.

J Paul Peter and Michael J Ryan. 1976. An investigation of perceived risk at the brand level. *Journal of Marketing Research*, pages 184–188.

John C Platt. 1999. Fast training of support vector machines using sequential minimal optimization. In *Advances in kernel methods*, pages 185–208. MIT press.

John Ross Quinlan. 1993. *C4. 5: programs for machine learning*, volume 1. Morgan kaufmann.

Shalom H Schwartz. 2003. A proposal for measuring value orientations across nations. *Questionnaire Package of the European Social Survey*, pages 259–290.

Janjaap Semeijn, Allard CR Van Riel, and A Beatriz Ambrosini. 2004. Consumer evaluations of store brands: effects of store image and product attributes. *Journal of Retailing and Consumer Services*, 11(4):247–258.

Matthew D Shank and Lynn Langmeyer. 1994. Does personality influence brand image? *The Journal of Psychology*, 128(2):157–164.

Robert J Sutton and Peter C Riesz. 1979. The effect of product visibility upon the relationship between price and quality. *Zeitschrift für Verbraucherpolitik*, 3(2):145–150.

Richard Thaler. 1985. Mental accounting and consumer choice. *Marketing Science*, 4(3):199–214.

Yann Truong, Rod McColl, and Philip J Kitchen. 2010. Uncovering the relationships between aspirations and luxury brand preference. *Journal of Product & Brand Management*, 19(5):346–355.

David K Tse and Gerald J Gorn. 1993. An experiment on the salience of country-of-origin in the era of global brands. *Journal of International Marketing*, pages 57–76.

Bülent Üstün, Willem J Melssen, and Lutgarde MC Buydens. 2006. Facilitating the application of support vector regression by using a universal pearson vii function based kernel. *Chemometrics and Intelligent Laboratory Systems*, 81(1):29–40.

William Yang Wang, Edward Lin, and John Kominek. 2013. This text has the scent of starbucks: A laplacian structured sparsity model for computational branding analytics. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP 2013), Seattle, WA, USA*.

Ralph Westfall. 1962. Psychological factors in predicting product choice. *The Journal of Marketing*, pages 34–40.

Robert E Witt and Grady D Bruce. 1972. Group influence and brand choice congruence. *Journal of Marketing Research*, pages 440–443.

Wendy Wood and Timothy Hayes. 2012. Social influence on consumer decisions: Motives, modes, and consequences. *Journal of Consumer Psychology*, 22(3):324–328.

Huahai Yang and Yunyao Li. 2013. Identifying user needs from social media. Technical report, IBM Tech Report. goo. gl/2XB7NY.

Tal Yarkoni. 2010. Personality in 100,000 words: A large-scale analysis of personality and word use among bloggers. *Journal of Research in Personality*, 44(3):363–373.

George M Zinkhan and Claude R Martin Jr. 1987. New brand names and inferential beliefs: Some insights on naming new products. *Journal of Business Research*, 15(2):157–172.

Semantic Annotation for Microblog Topics Using Wikipedia Temporal Information

Tuan Tran
L3S Research Center
Hannover, Germany
ttran@L3S.de

Nam Khanh Tran
L3S Research Center
Hannover, Germany
ntran@L3S.de

Asmelash Teka Hadgu
L3S Research Center
Hannover, Germany
teka@L3S.de

Robert Jäschke
L3S Research Center
Hannover, Germany
jaeschke@L3S.de

Abstract

Trending topics in microblogs such as Twitter are valuable resources to understand social aspects of real-world events. To enable deep analyses of such trends, semantic annotation is an effective approach; yet the problem of annotating microblog trending topics is largely unexplored by the research community. In this work, we tackle the problem of mapping trending Twitter topics to entities from Wikipedia. We propose a novel model that complements traditional text-based approaches by rewarding entities that exhibit a high temporal correlation with topics during their burst time period. By exploiting temporal information from the Wikipedia edit history and page view logs, we have improved the annotation performance by 17-28%, as compared to the competitive baselines.

1 Introduction

With the proliferation of microblogging and its wide influence on how information is shared and digested, the studying of microblog sites has gained interest in recent NLP research. Several approaches have been proposed to enable a deep understanding of information on Twitter. An emerging approach is to use semantic annotation techniques, for instance by mapping Twitter information snippets to canonical entities in a knowledge base or to Wikipedia (Meij et al., 2012; Guo et al., 2013), or by revisiting NLP tasks in the Twitter domain (Owoputi et al., 2013; Ritter et al., 2011). Much of the existing work focuses on annotating a single Twitter message (tweet). However, information in Twitter is rarely digested in isolation, but rather in a collective manner, with the adoption of special mechanisms such as hashtags. When put together, the unprecedentedly massive adoption of a hashtag within a short time period can lead to bursts and often reflect trending social attention. Understanding the meaning of trending hashtags offers a valuable opportunity for various applications and studies, such as viral marketing, social behavior analysis, recommendation, etc. Unfortunately, the task of hashtag annotation has been largely unexplored so far.

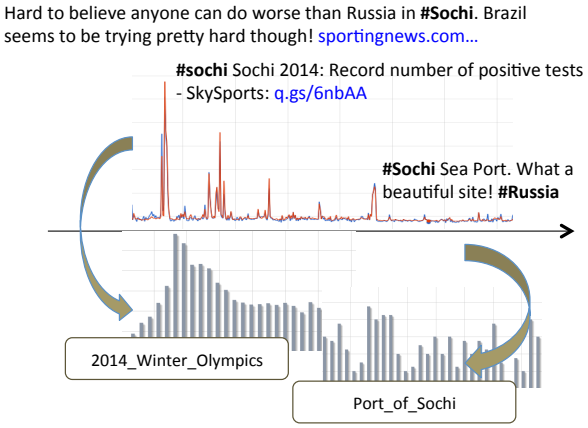


Figure 1: Example of trending hashtag annotation. During the *2014 Winter Olympics*, the hashtag '#sochi' had a different meaning.

In this paper, we study the problem of annotating trending hashtags on Twitter by entities derived from Wikipedia. Instead of establishing a static semantic connection between hashtags and entities, we are interested in *dynamically* linking the hashtags to entities that are closest to the underlying topics during burst time periods of the hashtags. For instance, while '#sochi' refers to a city in Russia, during February 2014, the hashtag was used to report the *2014 Winter Olympics* (cf. Figure 1). Hence, it should be linked more to Wikipedia pages related to the event than to the location.

Compared to traditional domains of text (e.g., news articles), annotating hashtags poses additional challenges. Hashtags' surface forms are

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 97–106,
Lisbon, Portugal, 17-21 September 2015.

very ad-hoc, as they are chosen not in favor of the text quality, but by the dynamics in attention of the large crowd. In addition, the evolution of the semantics of hashtags (e.g., in the case of '#sochi') makes them more ambiguous. Furthermore, a hashtag can encode multiple topics at once. For example, in March 2014, '#oscar' refers to the *86th Academy Awards*, but at the same time also to the *Trial of Oscar Pistorius*. Sometimes, it is difficult even for humans to understand a trending hashtag without knowledge about what was happening with the related entities in the real world.

In this work, we propose a novel solution to these challenges by leveraging temporal knowledge about entity dynamics derived from Wikipedia. We hypothesize that a trending hashtag is associated with an increase in public attention to certain entities, and this can also be observed on Wikipedia. As in Figure 1, we can identify *2014 Winter Olympics* as a prominent entity for '#sochi' during February 2014, by observing the change of user attention to the entity, for instance via the page view statistics of Wikipedia articles. We exploit both Wikipedia edits and page views for annotation. We also propose a novel learning method, inspired by the information spreading nature of social media such as Twitter, to suggest the optimal annotations without the need for human labeling. In summary:

- We are the first to combine the Wikipedia edit history and page view statistics to overcome the temporal ambiguity of Twitter hashtags.
- We propose a novel and efficient learning algorithm based on influence maximization to automatically annotate hashtags. The idea is generalizable to other social media sites that have a similar information spreading nature.
- We conduct thorough experiments on a real-world dataset and show that our system can outperform competitive baselines by 17-28%.

2 Related Work

Entity Linking in Microblogs The task of semantic annotation in microblogs has been recently tackled by different methods, which can be divided into two classes, i.e., content-based and graph-based methods. While the content-based methods (Meij et al., 2012; Guo et al., 2013; Fang and Chang, 2014) consider tweets independently, the graph-based methods (Cassidy et al., 2012; Liu et al., 2013) use all related tweets (e.g., posted by a user) together. However, most of them focus on entity mentions in tweets. In contrast, we take into account hashtags which reflect the topics discussed in tweets, and leverage external resources from Wikipedia (in particular, the edit history and page view logs) for semantic annotation.

Analysis of Twitter Hashtags In an attempt to understand the user interest dynamics on Twitter, a rich body of work analyzes the temporal patterns of popular hashtags (Lehmann et al., 2012; Naaman et al., 2011; Tsur and Rappoport, 2012). Few works have paid attention to the semantics of hashtags, i.e., to the underlying topics conveyed in the corresponding tweets. Recently, Bansal et al. (2015) attempt to segment a hashtag and link each of its tokens to a Wikipedia page. However, the authors only aim to retrieve entities directly mentioned within a hashtag, which are very few in practice. The external information derived from the tweets is largely ignored. In contrast, we exploit both context information from the microblog and Wikipedia resources.

Event Mining Using Wikipedia Recently some works exploit Wikipedia for detecting and analyzing events on Twitter (Osborne et al., 2012; Tolomei et al., 2013; Tran et al., 2014). However, most of the existing studies focus on the statistical signals of Wikipedia (such as the edit or page view volumes). We are the first to combine the content of the Wikipedia edit history and the magnitude of page views to handle trending topics on Twitter.

3 Framework

Preliminaries We refer to an *entity* (denoted by e) as any object described by a Wikipedia article (ignoring disambiguation, lists, and redirect pages). The number of times an entity's article has been requested is called the *entity view count*. The text content of the article is denoted by $C(e)$. In this work, we choose to study hashtags at the daily level, i.e., from the timestamps of tweets we only consider their creation day. A hashtag is called *trending* at a time point (a day) if the number of tweets where it appears is significantly higher than that on other days. There are many ways to detect such trendings. (Lappas et al., 2009; Lehmann et al., 2012). Each trending hashtag has one or multiple *burst time periods*, surrounding the trend-

ing day, where the users' interest in the underlying topic remains stronger than in other periods. We denote with $T(h)$ (or T for short) one hashtag burst time period, and with $D_T(h)$ the set of tweets containing the hashtag h created during T.

Task Definition Given a trending hashtag h and the burst time period T of h, identify the top-k most prominent entities to describe h during T.

It is worth noting that not all trending hashtags are mapable to Wikipedia entities, as the coverage of topics in Wikipedia is much lower than on Twitter. This is also the limitation of systems relying on Wikipedia such as entity disambiguation, which can only disambiguate popular entities and not the ones in the long tail. In this study, we focus on the precision and the popular trending hashtags, and leave the improvement of recall to future work.

Overview We approach the task in three steps. The first step is to identify all entity candidates by checking surface forms of the constituent tweets of the hashtag. In the second step, we compute different similarities between each candidate and the hashtag, based on different types of contexts, which are derived from either side (Wikipedia or Twitter). Finally, we learn a unified ranking function for each (hashtag, entity) pair and choose the top-k entities with the highest scores. The ranking function is learned through an unsupervised model and needs no human-defined labels.

3.1 Entity Linking

The most obvious resource to identify candidate entities for a hashtag is via its tweets. We follow common approaches that use a lexicon to match each textual phrase in a tweet to a potential entity set (Shen et al., 2013; Fang and Chang, 2014). Our lexicon is constructed from Wikipedia page titles, hyperlink anchors, redirects, and disambiguation pages, which are mapped to the corresponding entities. As for the tweet phrases, we extract all n-grams ($n \leq 5$) from the input tweets within T. We apply the longest-match heuristic (Meij et al., 2012): We start with the longest n-grams and stop as soon as the entity set is found, otherwise we continue with the smaller constituent n-grams.

Candidate Set Expansion While the lexicon-based linking works well for single tweets, applying it on the hashtag level has subtle implications. Processing a huge amount of text, especially during a hashtag burst time period, incurs expensive computational costs. Therefore, to guarantee a good recall in this step while still maintaining feasible computation, we apply entity linking only on a random sample of the complete tweet set. Then, for each candidate entity e, we include all entities whose Wikipedia article is linked with the article of e by an outgoing or incoming link.

3.2 Measuring Entity–Hashtag Similarities

To rank the entity by prominence, we measure the similarity between each candidate entity and the hashtag. We study three types of similarities:

Mention Similarity This measure relies on the explicit mentions of entities in tweets. It assumes that entities directly linked from more prominent anchors are more relevant to the hashtag. It is estimated using both statistics from Wikipedia and tweet phrases, and turns out to be surprisingly effective in practice (Fang and Chang, 2014).

Context Similarity For entities that are not directly linked to mentions (the mention similarity is zero) we exploit external resources instead. Their prominence is perceived by users via external sources, such as web pages linked from tweets, or entity home pages or Wikipedia pages. By exploiting the content of entities from these external sources, we can complement the explicit similarity metrics based on mentions.

Temporal Similarity The two measures above rely on the textual representation and are degraded by the linguistic difference between the two platforms. To overcome this drawback, we incorporate the temporal dynamics of hashtags and entities, which serve as a proxy to the change of user interests towards the underlying topics (Ciglan and Nørvåg, 2010). We employ the correlation between the times series of hashtag adoption and the entity view as the third similarity measure.

3.3 Ranking Entity Prominence

While each similarity measure captures one evidence of the entity prominence, we need to unify all scores to obtain a global ranking function. In this work, we propose to combine the individual similarities using a linear function:

$$f(e,h) = \alpha f_m(e,h) + \beta f_c(e,h) + \gamma f_t(e,h) \quad (1)$$

where α, β, γ are model weights and f_m, f_c, f_t are the similarity measures based on mentions, context, and temporal information, respectively, be-

tween the entity e and the hashtag h. We further constrain that $\alpha + \beta + \gamma = 1$, so that the ranking scores of entities are normalized between 0 and 1, and that our learning algorithm is more tractable. The algorithm, which automatically learns the parameters without the need of human-labeled data, is explained in detail in Section 5.

4 Similarity Measures

We now discuss in detail how the similarity measures between hashtags and entities are computed.

4.1 Link-based Mention Similarity

The similarity of an entity with one individual mention in a tweet can be interpreted as the probabilistic prior in mapping the mention to the entity via the lexicon. One common way to estimate the entity prior exploits the anchor statistics from Wikipedia links, and has been proven to work well in different domains of text. We follow this approach and define $LP(e|m) = \frac{|l_m(e)|}{\sum_{m'}|l_{m'}(e)|}$ as the link prior of the entity e given a mention m, where $l_m(e)$ is the set of links with anchor m that point to e. The mention similarity f_m is measured as the aggregation of link priors of the entity e over all mentions in all tweets with the hashtag h:

$$f_m(e,h) = \sum_m (LP(e|m) \cdot q(m)) \quad (2)$$

where $q(m)$ is the frequency of the mention m over all mentions of e in all tweets of h.

4.1.1 Context Similarity

To compute f_c, we first construct the contexts for hashtags and entities. The context of a hashtag is built by extracting all words from its tweets. We tokenize and parse the tweets' part-of-speech tags (Owoputi et al., 2013), and remove words of Twitter-specific tags (e.g., @-mentions, URLs, emoticons, etc.). Hashtags are normalized using the word breaking method by Wang et al. (2011).

The textual context of an entity is extracted from its Wikipedia article. One subtle aspect is that the articles are not created at once, but are incrementally updated over time in accordance with changing information about entities. Texts added in the same time period of a trending hashtag contribute more to the context similarity between the entity and the hashtag. Based on this observation, we use the Wikipedia revision history – an archive of all revisions of Wikipedia articles – to calculate the entity context. We collect the revisions of articles during the time period T, plus one day to acknowledge possible time lags. We compute the difference between two consecutive revisions, and extract only the added text snippets. These snippets are accumulated to form the *temporal context* of an entity e during T, denoted by $C_T(e)$. The distribution of a word w for the entity e is estimated by a mixture between the probability of generating w from the temporal context and from the general context $C(e)$ of the entity:

$$\hat{P}(w|e) = \lambda \hat{P}(w|M_{C_T(e)}) + (1-\lambda)\hat{P}(w|M_{C(e)})$$

where $M_{C_T(e)}$ and $M_{C(e)}$ are the language models of e based on $C_T(e)$ and $C(e)$, respectively. The probability $\hat{P}(w|M_{C(e)})$ can be regarded as corresponding to the background model, while $\hat{P}(w|M_{C_T(e)})$ corresponds to the foreground model in traditional language modeling settings. Here we use a simple maximum likelihood estimation to estimate these probabilities: $\hat{P}(w|M_{C(e)}) = \frac{tf_{w,c}}{|C(e)|}$ and $\hat{P}(w|M_{C_T(e)}) = \frac{tf_{w,c_T}}{|C_T(e)|}$, where $tf_{w,c}$ and tf_{w,c_T} are the term frequencies of w in the two text sources of $C(e)$ and $C_T(e)$, respectively, and $|C(e)|$ and $|C_T(e)|$ are the lengths of the two texts, respectively. We use the same estimation for tweets: $\hat{P}(w|h) = \frac{tf_{w,D(h)}}{|D(h)|}$, where $D(h)$ is the concatenated text of all tweets of h in T. We use and normalize the Kullback-Leibler divergence to compare the distributions over all words appearing both in the Wikipedia contexts and the tweets:

$$KL(e \parallel h) = \sum_w \hat{P}(w|e) \cdot \frac{\hat{P}(w|e)}{\hat{P}(w|h)}$$
$$f_c(e,h) = e^{-KL(e \parallel h)} \quad (3)$$

4.1.2 Temporal Similarity

The third similarity, f_t, is computed using temporal signals from both sources – Twitter and Wikipedia. For the hashtags, we build the time series based on the volume of tweets adopting the hashtag h on each day in T: $TS_h = [n_1, n_2, \ldots, n_{|T|}]$. Similarly for the entities, we build the time series of view counts for the entity e in T: $TS_e = [v_1, v_2, \ldots, v_{|T|}]$. A time series similarity metric is then used to compute f_t. Several metrics can be used, however most of them suffer from the time lag and scaling discrepancy, or incur expensive computational costs (Radinsky et al., 2011). In this work, we employ a simple yet

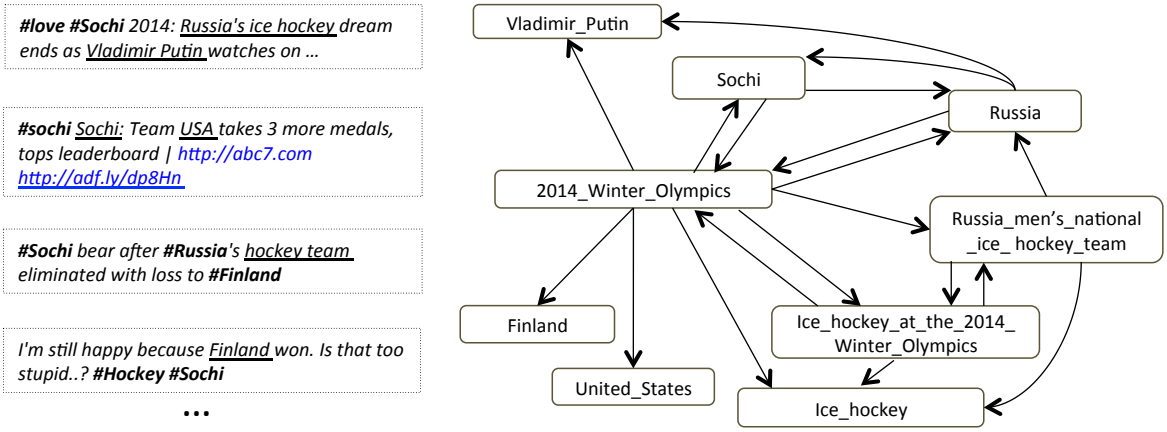


Figure 2: Excerpt of tweets about ice hockey results in the *2014 Winter Olympics* (left), and the observed linking process between time-aligned revisions of candidate Wikipedia entities (right). Links come more from prominent entities to marginal ones to provide background, or more context for the topics. Thus, starting from prominent entities, we can reach more entities in the graph of candidate entities

effective metric that is agnostic to the scaling and time lag of time series (Yang and Leskovec, 2011). It measures the distance between two time series by finding optimal shifting and scaling parameters to match the shape of two time series:

$$f_t(e,h) = \min_{q,\delta} \frac{\|TS_h - \delta d_q(TS_e)\|}{\|TS_h\|} \quad (4)$$

where $d_q(TS_e)$ is the time series derived from TS_e by shifting q time units, and $\|\cdot\|$ is the L_2 norm. It has been proven that Equation 4 has a closed-form solution for δ given fixed q, thus we can design an efficient gradient-based optimization algorithm to compute f_t (Yang and Leskovec, 2011).

5 Entity Prominence Ranking

5.1 Ranking Framework

To unify the individual similarities into one global metric (Equation 1), we need a guiding premise of what manifest the prominence of an entity to a hashtag. Such a premise can be instructed through manual assessment (Meij et al., 2012; Guo et al., 2013), but it requires human-labeled data and is biased from evaluator to evaluator. Other heuristics assume that entities close to the main topic of a text are also coherent to each other (Ratinov et al., 2011; Liu et al., 2013). Based on this, state-of-the-art methods in traditional disambiguation estimate entity prominence by optimizing the overall coherence of the entities' semantic relatedness. However, this coherence does not hold for topics in hashtags: Entities reported in a big topic such as the Olympics vary greatly with different sub-events. They are not always coherent to each other, as they are largely dependent on the users' diverse attention to each sub-event. This heterogeneity of hashtags calls for a different premise, abandoning the idea of coherence.

Influence Maximization (IM) We propose a new approach to find entities for a hashtag. We use an observed behavioral pattern in creating Wikipedia pages for guiding our approach to entity prominence: Wikipedia articles of entities that are prominent for a topic are quickly created or updated,[1] and subsequently enriched with links to related entities. This linking process signals the dynamics of editor attention and exposure to the event (Keegan et al., 2011). We argue that the process does not, or to a much lesser degree, happen to more marginal entities or to very general entities. As illustrated in Figure 2, the entities closer to the 2014 Olympics get more updates in the revisions of their Wikipedia articles, with subsequent links pointing to articles of more distant entities. The direction of the links influences the shifting attention of users (Keegan et al., 2011) as they follow the structure of articles in Wikipedia.

We assume that, similar to Wikipedia, the entity prominence also influences how users are exposed and spread the hashtag on Twitter. In particular, the initial spreading of a trending hashtag involves more entities in the focus of the topic. Subsequent exposure and spreading of the hashtag then include other related entities (e.g., discussing background or providing context), driven by interests in different parts of the topic. Based on this assumption,

[1] Osborne et al. (2012) suggested a time lag of 3 hours.

we propose to gauge the entity prominence as its potential in *maximizing the information spreading* within all entities present in the tweets of the hashtag. In other words, the problem of ranking the most prominent entities becomes identifying the set of entities that lead to the largest number of entities in the candidate set. This problem is known in social network research as *influence maximization* (Kempe et al., 2003).

Iterative Influence-Prominence Learning (IPL) IM itself is an NP-hard problem (Kempe et al., 2003). Therefore, we propose an approximation framework, which can *jointly* learn the influence scores of the entity and the entity prominence together. The framework (called IPL) contains several iterations, each consisting of two steps: (1) Pick up a model and use it to compute the entity influence score. (2) Based on the influence scores, update the entity prominence. In the sequel we detail our learning framework.

5.2 Entity Graph

Influence Graph To compute the entity influence scores, we first construct the entity *influence graph* as follows. For each hashtag h, we construct a directed graph $G_h = (E_h, V_h)$, where the nodes $E_h \subseteq E$ consist of all candidate entities (cf. Section 3.1), and an edge $(e_i, e_j) \in V_h$ indicates that there is a link from e_j's Wikipedia article to e_i's. Note that edges of the influence graph are inversed in direction to links in Wikipedia, as such a link gives an "influence endorsement" from the destination entity to the source entity.

Entity Relatedness In this work, we assume that an entity endorses more of its influence score to highly related entities than to lower related ones. We use a popular entity relatedness measure suggested by Milne and Witten (2008):

$$MW(e_1, e_2) = 1 - \frac{\log(\max(|I_1|,|I_2|) - \log(|I_1 \cap I_2|))}{\log(|E|) - \log(\min(|I_1|,|I_2|))}$$

where I_1 and I_2 are sets of entities having links to e_1 and e_2, respectively, and E is the set of all entities in Wikipedia. The influence transition from e_i to e_j is defined as the normalized value:

$$b_{i,j} = \frac{MW(e_i, e_j)}{\sum_{(e_i,e_k) \in V} MW(e_i, e_k)} \quad (5)$$

Influence Score Let $\mathbf{r}_h$ be the influence score vector of entities in G_h. We can estimate $\mathbf{r}_h$ efficiently using random walk models, similarly to the baseline method suggested by Liu et al. (2014):

$$\mathbf{r_h} \coloneqq \tau \mathbf{B} \mathbf{r_h} + (1 - \tau)\mathbf{s_h} \quad (6)$$

where $\mathbf{B}$ is the influence transition matrix, $\mathbf{s_h}$ are the initial influence scores that are based on the entity prominence model (Step 1 of IPL), and τ is the damping factor.

Algorithm 1: Entity Influence-Prominence Learning

Input : $h, T, D_T(h), \mathbf{B}, k$, learning rate μ, threshold ϵ
Output: ω, top-k most prominent entities.

Initialize: $\omega \coloneqq \omega^{(0)}$
Calculate $\mathbf{f}_m, \mathbf{f}_c, \mathbf{f}_t, \mathbf{f}_\omega \coloneqq \mathbf{f}_{\omega^{(0)}}$ using Eqs. 1, 2, 3, 4
while *true* **do**
  $\hat{\mathbf{f}}_\omega \coloneqq$ normalize $\mathbf{f}_\omega$
  Set $\mathbf{s_h} \coloneqq \hat{\mathbf{f}}_\omega$, calculate $\mathbf{r_h}$ using Eq. 6
  Sort $\mathbf{r_h}$, get the top-k entities $E(h,k)$
  if $\sum_{e \in E(h,k)} L(f(e,h), r(e,h)) < \epsilon$ **then**
    Stop
  end
  $\omega \coloneqq \omega - \mu \sum_{e \in E(h,k)} \nabla L(f(e,h), r(e,h))$
end
return $\omega, E(h,k)$

5.3 Learning Algorithm

Now we detail the IPL algorithm. The objective is to learn the model $\omega = (\alpha, \beta, \gamma)$ of the global function (Equation 1). The general idea is that we find an optimal ω such that the average error with respect to the top influencing entities is minimized

$$\omega = \arg\min \sum_{E(h,k)} L(f(e,h), r(e,h))$$

where $r(e,h)$ is the influence score of e and h, $E(h,k)$ is the set of top-k entities with highest $r(e,h)$, and L is the squared error loss function, $L(x,y) = \frac{(x-y)^2}{2}$.

The main steps are depicted in Algorithm 1. We start with an initial guess for ω, and compute the similarities for the candidate entities. Here $\mathbf{f}_m$, $\mathbf{f}_c$, $\mathbf{f}_t$, and $\mathbf{f}_\omega$ represent the similarity score vectors. We use matrix multiplication to calculate the similarities efficiently. In each iteration, we first normalize $\mathbf{f}_\omega$ such that the entity scores sum up to 1. A random walk is performed to calculate the influence score $\mathbf{r_h}$. Then we update ω using a batch gradient descent method on the top-k influencer entities. To derive the gradient of the loss function L, we first remark that our random walk Equation 6 is similar to context-sensitive PageRank (Haveliwala, 2002). Using the linearity property (Fogaras et al., 2005),

Total Tweets	500,551,041
Trending Hashtags	2,444
Test Hashtags	30
Test Tweets	352,394
Distinct Mentions	145,941
Test (Entity, Hashtag) pairs	6,965
Candidates per Hashtag (avg.)	50
Extended Candidates (avg.)	182

Table 1: Statistics of the dataset.

we can express $r(e,h)$ as the linear function of influence scores obtained by initializing with the individual similarities f_m, f_c, and f_t instead of f_ω. The derivative thus can be written as:

$$\nabla L(f(e,h), r(e,h)) = \alpha(r_m(e,h) - f_m(e,h)) + \beta(r_c(e,h) - f_c(e,h)) + \gamma(r_t(e,h) - f_t(e,h))$$

where $r_m(e,h), r_c(e,h), r_t(e,h)$ are the components of the three vector solutions of Equation 6, each having $\mathbf{s_h}$ replaced by $\mathbf{f}_m$, $\mathbf{f}_c$, $\mathbf{f}_t$ respectively.

Since both $\mathbf{B}$ and $\hat{\mathbf{f}_\omega}$ are normalized such that their column sums are equal to 1, Equation 6 is convergent (Haveliwala, 2002). Also, as discussed above, $\mathbf{r_h}$ is a linear combination of factors that are independent of ω, hence L is a convex function, and the batch gradient descent is also guaranteed to converge. In practice, we can utilize several indexing techniques to significantly speed up the similarity and influence scores calculation.

6 Experiments and Results

6.1 Setup

Dataset There is no standard benchmark for our problem, since available datasets on microblog annotation (such as the Microposts challenge (Basave et al., 2014)) do not have global statistics, so we cannot identify the trending hashtags. Therefore, we created our own dataset. We used the Twitter API to collect from the public stream a sample of $500,551,041$ tweets from January to April 2014. We removed hashtags that were adopted by less than 500 users, having no letters, or having characters repeated more than 4 times (e.g., '#oooommgg'). We identified trending hashtags by computing the daily time series of hashtag tweet counts, and removing those of which the time series' variance score is less than 900. To identify the hashtag burst time period T, we compute the *outlier fraction* (Lehmann et al., 2012) for each hashtag h and day t: $p_t(h) = \frac{|n_t - n_b|}{\max(n_b, n_{\min})}$, where n_t is the number of tweets containing h, n_b is the median value of n_t over all points in a 2-month time window centered on t, and $n_{\min} = 10$ is the threshold to filter low activity hashtags. The hashtag is skipped if its highest outlier fraction score is less than 15. Finally, we define the *burst time period* of a trending hashtag as the time window of size w, centered at day t_0 with the highest $p_{t_0}(h)$.

For the Wikipedia datasets we process the dump from 3rd May 2014, so as to cover all events in the Twitter dataset. We have developed Hedera (Tran and Nguyen, 2014), a scalable tool for processing the Wikipedia revision history dataset based on Map-Reduce paradigm. In addition, we download the Wikipedia page view dataset that stores how many times a Wikipedia article was requested on an hourly level. We process the dataset for the four months of our study and use Hedera to accumulate all view counts of redirects to the actual articles.

Sampling From the trending hashtags, we sample 30 distinct hashtags for evaluation. Since our study focuses on trending hashtags that are mapable to entities in Wikipedia, the sampling must cover a sufficient number of "popular" topics that are seen in Wikipedia, and at the same time cover rare topics in the long tail. To do this, we apply several heuristics in the sampling. First, we only consider hashtags where the lexicon-based linking (Section 3.1) results in at least 20 different entities. Second, we randomly choose hashtags to cover different types of topics (long-running events, breaking events, endogenous hashtags). Instead of inspecting all hashtags in our corpus, we follow Lehmann et al. (2012) and calculate the fraction of tweets published before, during and after the peak. The hashtags are then clustered in this 3-dimensional vector space. Each cluster suggests a group of hashtags with a distinct semantics (Lehmann et al., 2012). We then pick up hashtags randomly from each cluster, resulting in 200 hashtags in total. From this rough sample, three inspectors carefully checked the tweets and chose 30 hashtags where the meanings and hashtag types were certain to the knowledge of the inspectors.

Parameter Settings We initialize the similarity weights to $\frac{1}{3}$, the damping factor to $\tau = 0.85$, and the weight for the language model to $\lambda = 0.9$. The learning rate μ is empirically fixed to $\mu = 0.003$.

	Tagme	Wikiminer	Meij	Kauri	M	C	T	IPL
P@5	0.284	0.253	0.500	0.305	0.453	0.263	0.474	**0.642**
P@15	0.253	0.147	**0.670**	0.319	0.312	0.245	0.378	0.495
MAP	0.148	0.096	0.375	0.162	0.211	0.140	0.291	**0.439**

Table 2: Experimental results on the sampled trending hashtags.

Baseline We compare IPL with other entity annotation methods. Our first group of baselines includes entity linking systems in domains of general text, Wikiminer (Milne and Witten, 2008), and short text, Tagme (Ferragina and Scaiella, 2012). For each method, we use the default parameter settings, apply them for the individual tweets, and take the average of the annotation confidence scores as the prominence ranking function. The second group of baselines includes systems specifically designed for microblogs. For the content-based methods, we compare against Meij et al. (2012), which uses a supervised method to rank entities with respect to tweets. We train the model using the same training data as in the original paper. For the graph-based method, we compare against KAURI (Shen et al., 2013), a method which uses user interest propagation to optimize the entity linking scores. To tune the parameters, we pick up four hashtags from different clusters, randomly sample 50 tweets for each, and manually annotate the tweets. For all baselines, we obtained the implementation from the authors. The exception is Meij method, where we implemented ourselves, but we clarified with the authors via emails on several settings. In addition, we also compare three variants of our method, using only local functions for entity ranking (referred to as M, C, and T for *mention*, *context*, and *time*, respectively).

Evaluation In total, there are $6,965$ entity-hashtag pairs returned by all systems. We employ five volunteers to evaluate the pairs in the range from 0 to 2, where 0 means the entity is noisy or obviously unrelated, 2 means the entity is strongly tied to the topic of the hashtag, and 1 means that although the entity and hashtag might share some common contexts, they are not involved in a direct relationship (for instance, the entity is a too general concept such as *Ice hockey*, as in the case illustrated in Figure 2). The annotators were advised to use search engines, the Twitter search box or Wikipedia archives whenever applicable to get more background on the stories. Inter-annotator agreement under Fleiss score is 0.625.

6.2 Results and Discussion

Table 2 shows the performance comparison of the methods using the standard metrics for a ranking system (precision at 5 and 15 and MAP at 15). In general, all baselines perform worse than reported in the literature, confirming the higher complexity of the hashtag annotation task as compared to traditional tasks. Interestingly enough, using our local similarities already produces better results than Tagme and Wikiminer. The local model f_m significantly outperforms both the baselines in all metrics. Combining the similarities improves the performance even more significantly.[2] Compared to the baselines, IPL improves the performance by 17-28%. The time similarity achieves the highest result compared to other content-based mention and context similarities. This supports our assumption that lexical matching is not always the best strategy to link entities in tweets. The time series-based metric incurs lower cost than others, yet it produces a considerably good performance. Context similarity based on Wikipedia edits does not yield much improvement. This can be explained in two ways. First, information in Wikipedia is largely biased to popular entities, it fails to capture many entities in the long tail. Second, language models are dependent on direct word representations, which are different between Twitter and Wikipedia. This is another advantage of non-content measures such as f_t.

For the second group of baselines (Kauri and Meij), we also observe the reduction in precision, especially for Kauri. This is because the method relies on the coherence of user interests within a group of tweets to be able to perform well, which does not hold in the context of hashtags. One astonishing result is that Meij performs better than IPL in terms of P@15. However, it performs worse in terms of MAP and P@5, suggesting that most of the correctly identified entities are ranked lower in the list. This is reasonable, as Meij attempts to optimize (with human supervision effort) the se-

[2] All significance tests are done against both Tagme and Wikiminer, with a p-value < 0.01.

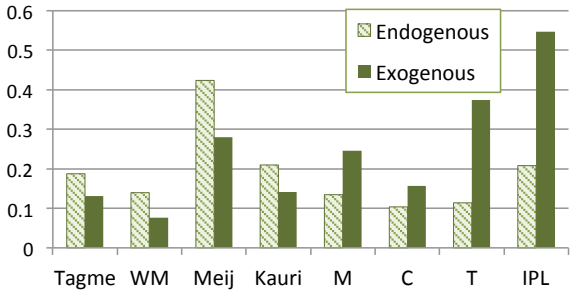


Figure 3: Performance of the methods for different types of trending hashtags.

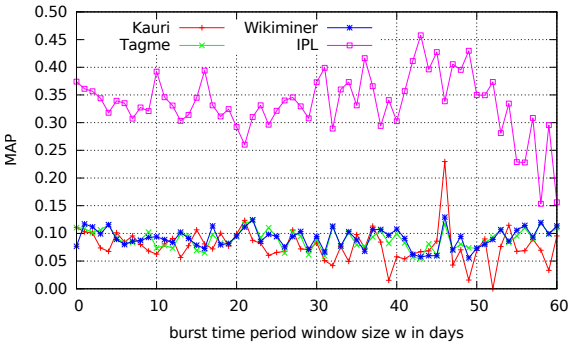


Figure 4: IPL compared to other baselines on different sizes of the burst time window T.

mantic agreement between entities and information found in the tweets, instead of ranking their prominence as in our work. To investigate this case further, we re-examined the hashtags and divided them by their semantics, as to whether the hashtags are spurious trends of memes inside social media (*endogenous*, e.g., "#stopasian2014"), or whether they reflect external events (*exogenous*, e.g., "#mh370"). The performance of the methods in terms of MAP scores is shown in Figure 3. It can be clearly seen that entity linking methods perform well in the endogenous group, but then deteriorate in the exogenous group. The explanation is that for endogenous hashtags, the topical consonance between tweets is very low, thus most of the assessments become just verifying general concepts (such as locations) In this case, topical annotation is trumped by conceptual annotation. However, whenever the hashtag evolves into a meaningful topic, a deeper annotation method will produce a significant improvement, as seen in Figure 3.

Finally, we study the impact of the burst time period on the annotation quality. For this, we expand the window size w (cf. Section 6.1) and examine how different methods perform. The result is depicted in Figure 4. It is obvious that within the window of 2 months (where the hashtag time series is constructed and a trending time is identified), our method is stable and always outperforms the baselines by a large margin. Even when the trending hashtag has been saturated, hence introduced more noise, our method is still able to identify the prominent entities with high quality.

7 Conclusion and Future Work

In this work, we address the new problem of topically annotating a trending hashtag using Wikipedia entities, which has many important applications in social media analysis. We study Wikipedia temporal resources and find that using efficient time series-based measures can complement content-based methods well in the domain of Twitter. We propose use similarity measures to model both the local mention-based, as well as the global context- and time-based prominence of entities. We propose a novel strategy of topical annotation of texts using and influence maximization approach and design an efficient learning algorithm to automatically unify the similarities without the need of human involvement. The experiments show that our method outperforms significantly the established baselines.

As future work, we aim to improve the efficiency of our entire workflow, such that the annotation can become an end-to-end service. We also aim to improve the context similarity between entities and the topic, for example by using a deeper distributional semantics-based method, instead of language models as in our current work. In addition, we plan to extend the annotation framework to other types of trending topics, by including the type of out-of-knowledge entities. Finally, we are investigating how to apply more advanced influence maximization methods. We believe that influence maximization has a great potential in NLP research, beyond the scope of annotation for microblogging topics.

Acknowledgments

This work was funded by the European Commission in the FP7 project ForgetIT (600826) and the ERC advanced grant ALEXANDRIA (339233), and by the German Federal Ministry of Education and Research for the project "Gute Arbeit" (01UG1249C). We thank the reviewers for the fruitful discussion and Claudia Niederee from L3S for suggestions on improving Section 5.

References

P. Bansal, R. Bansal, and V. Varma. 2015. Towards deep semantic analysis of hashtags. In *ECIR*, pages 453–464.

A. E. Cano Basave, G. Rizzo, A. Varga, M. Rowe, M. Stankovic, and A. Dadzie. 2014. Making sense of microposts (#microposts2014) named entity extraction & linking challenge. In *4th Workshop on Making Sense of Microposts*.

T. Cassidy, H. Ji, L.-A. Ratinov, A. Zubiaga, and H. Huang. 2012. Analysis and enhancement of wikification for microblogs with context expansion. In *COLING*, pages 441–456.

M. Ciglan and K. Nørvåg. 2010. WikiPop: personalized event detection system based on Wikipedia page view statistics. In *CIKM*, pages 1931–1932.

Y. Fang and M.-W. Chang. 2014. Entity linking on microblogs with spatial and temporal signals. *Trans. of the Assoc. for Comp. Linguistics*, 2:259–272.

P. Ferragina and U. Scaiella. 2012. Fast and accurate annotation of short texts with Wikipedia pages. *IEEE Softw.*, 29(1):70–75.

D. Fogaras, B. Rácz, K. Csalogány, and T. Sarlós. 2005. Towards scaling fully personalized PageRank: Algorithms, lower bounds, and experiments. *Internet Mathematics*, 2(3):333–358.

S. Guo, M.-W. Chang, and E. Kıcıman. 2013. To link or not to link? A study on end-to-end tweet entity linking. In *NAACL-HLT*, pages 1020–1030.

T. H. Haveliwala. 2002. Topic-sensitive PageRank. In *WWW*, pages 517–526.

Brian Keegan, Darren Gergle, and Noshir Contractor. 2011. Hot off the wiki: Dynamics, practices, and structures in wikipedia's coverage of the tohoku catastrophes. In *WikiSym*, pages 105–113.

D. Kempe, J. Kleinberg, and É. Tardos. 2003. Maximizing the spread of influence through a social network. In *KDD*, pages 137–146.

T. Lappas, B. Arai, M. Platakis, D. Kotsakos, and D. Gunopulos. 2009. On burstiness-aware search for document sequences. In *KDD*, pages 477–486.

J. Lehmann, B. Gonçalves, J. J. Ramasco, and C. Cattuto. 2012. Dynamical classes of collective attention in Twitter. In *WWW*, pages 251–260.

X. Liu, Y. Li, H. Wu, M. Zhou, F. Wei, and Y. Lu. 2013. Entity linking for tweets. In *ACL*, pages 1304–1311.

Q. Liu, B. Xiang, E. Chen, H. Xiong, F. Tang, and J. X. Yu. 2014. Influence maximization over large-scale social networks: A bounded linear approach. In *CIKM*, pages 171–180.

E. Meij, W. Weerkamp, and M. de Rijke. 2012. Adding semantics to microblog posts. In *WSDM*, pages 563–572.

D. Milne and I. H. Witten. 2008. Learning to link with Wikipedia. In *CIKM*, pages 509–518.

M. Naaman, H. Becker, and L. Gravano. 2011. Hip and trendy: Characterizing emerging trends on Twitter. *JASIST*, 62(5):902–918.

M. Osborne, S. Petrovic, R. McCreadie, C. Macdonald, and I. Ounis. 2012. Bieber no more: First story detection using Twitter and Wikipedia. In *Workshop on Time-aware Information Access*.

O. Owoputi, B. O'Connor, C. Dyer, K. Gimpel, N. Schneider, and N. A. Smith. 2013. Improved part-of-speech tagging for online conversational text with word clusters. In *NAACL-HLT*, pages 380–390.

K. Radinsky, E. Agichtein, E. Gabrilovich, and S. Markovitch. 2011. A word at a time: Computing word relatedness using temporal semantic analysis. In *WWW*, pages 337–346.

L. Ratinov, D. Roth, D. Downey, and M. Anderson. 2011. Local and Global Algorithms for Disambiguation to Wikipedia. In *ACL*, pages 1375–1384.

Alan Ritter, Sam Clark, Oren Etzioni, et al. 2011. Named entity recognition in tweets: an experimental study. In *EMNLP*, pages 1524–1534.

W. Shen, J. Wang, P. Luo, and M. Wang. 2013. Linking named entities in tweets with knowledge base via user interest modeling. In *WSDM*, pages 68–76.

G. Tolomei, S. Orlando, D. Ceccarelli, and C. Lucchese. 2013. Twitter anticipates bursts of requests for Wikipedia articles. In *Workshop on Data-driven User Behavioral Modelling and Mining from Social Media*, pages 5–8.

T. Tran and T. Ngoc Nguyen. 2014. Hedera: Scalable indexing, exploring entities in Wikipedia revision history. In *ISWC*, pages 297–300.

T. Tran, M. Georgescu, X. Zhu, and N. Kanhabua. 2014. Analysing the duration of trending topics in Twitter using Wikipedia. In *Conf. on Web Science*, pages 251–252.

O. Tsur and A. Rappoport. 2012. What's in a hashtag?: Content based prediction of the spread of ideas in microblogging communities. In *WSDM*, pages 643–652.

K. Wang, C. Thrasher, and B.-J. P. Hsu. 2011. Web scale NLP: a case study on URL word breaking. In *WWW*, pages 357–366.

J. Yang and J. Leskovec. 2011. Patterns of temporal variation in online media. In *WSDM*, pages 177–186.

System Combination for Multi-document Summarization

Kai Hong **Mitchell Marcus** **Ani Nenkova**
University of Pennsylvania
Philadelphia, PA, 19104
{hongkai1,mitch,nenkova}@seas.upenn.edu

Abstract

We present a novel framework of system combination for multi-document summarization. For each input set (input), we generate candidate summaries by combining whole sentences from the summaries generated by different systems. We show that the oracle among these candidates is much better than the summaries that we have combined. We then present a supervised model to select among the candidates. The model relies on a rich set of features that capture content importance from different perspectives. Our model performs better than the systems that we combined based on manual and automatic evaluations. We also achieve very competitive performance on six DUC/TAC datasets, comparable to the state-of-the-art on most datasets.

1 Introduction

Recent work shows that state-of-the-art summarization systems generate very different summaries, despite the fact that they have similar performance (Hong et al., 2014). This suggests that combining summaries from different systems might be helpful in improving content quality.

A handful of papers have studied system combination for summarization. Based on the ranks of the input sentences assigned by different systems (i.e., *basic systems*), methods have been proposed to re-rank these sentences (Wang and Li, 2012; Pei et al., 2012). However, these methods require the basic systems to assign importance scores to all input sentences. Thapar et al. (2006) combine the summaries from different systems, based on a graph-based measure that computes summary-input or summary-summary similarity. However, their method does not show an advantage over the basic systems. In summary, few prior papers have successfully generating better summaries by combining the summaries from different systems (i.e., *basic summaries*).

This paper focuses on *practical system combination*, where we combine the summaries generated by four portable unsupervised systems. We choose these systems, because: First, these systems are either off-the-shelf or easy-to-implement. Second, even though many systems have been proposed for multi-document summarization, the output of them are often available only on one dataset or even unavailable. Third, compared to more sophisticated supervised methods (Kulesza and Taskar, 2012; Cao et al., 2015a), simple unsupervised methods perform unexpectedly well. Many of them achieved the state-of-the-art performance when they were proposed (Erkan and Radev, 2004; Gillick et al., 2009) and still serve as competitive baselines (Hong et al., 2014).

After the summarizers have been chosen, we present a two-step pipeline that combines the basic summaries. In the first step, we generate combined candidate summaries (Section 4). We investigate two methods to do this: one uses entire basic summaries directly, the other combines these summaries on the sentence level. We show that the latter method has a much higher oracle performance. The second step includes a new supervised model that selects among the candidate summaries (Section 5).

Our contributions are:

- We show that by combining summaries on the sentence level, the best possible (oracle) performance is very high.
- In the second step of our pipeline, we propose a supervised model that includes a rich set of new features. These features capture content importance from different

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 107–117,
Lisbon, Portugal, 17-21 September 2015.

perspectives, based on different sources. We verify the effectiveness of these features.

- Our method outperforms the basic systems and several competitive baselines. Our model achieves competitive performance on six DUC/TAC datasets, which is on par with the state-of-the-art on most of these datasets.

- Our method can be used to combine summaries generated by any systems.

2 Related Work

System combination has enjoyed great success in many domains, such as automatic speech recognition (Fiscus, 1997; Mangu et al., 2000), machine translation (Frederking and Nirenburg, 1994; Bangalore et al., 2001) and parsing (Henderson and Brill, 1999; Sagae and Lavie, 2006). However, only a handful of papers have leveraged this idea for summarization. Mohamed and Rajasekaran (2005) present a method that relies on a document graph (DG), which includes concepts connected by relations. This method selects among the outputs of the basic systems, based on their overlaps with the input in terms of DG. Thapar et al. (2006) propose to iteratively include sentences, based on the overlap of DG between the current sentence and (1) the original input, or (2) the basic summaries. However, in both papers, the machine summaries are not compared against human references. Rather, their evaluations compare the summaries to the input based on the overlap of DG. Moreover, even when evaluated in this way, the combined system does not show an advantage over the best basic system.

System combination in summarization has also been regarded as rank aggregation, where the combined system re-ranks the input sentences based on the ranks of those sentences assigned by the basic systems. Wang and Li (2012) propose an unsupervised method to minimize the distance of the final ranking compared to the initial rankings. Pei et al. (2012) propose a supervised method which handles an issue in Wang and Li (2012) that all basic systems are regarded as equally important. Even though both methods show advantages over the basic systems, they have two limitations. Most importantly, only summarizers that assign importance scores to each sentence can be used as the input summarizers. Second, only the sentence scores (ranks) from the basic systems and system identity information is utilized during the re-ranking process. The signal from the original input is ignored. Our method handles these limitations.

Our method derives an overall informativeness score for each candidate summary, then selects the one with the highest score. This is related to the growing body of research in global optimization, which selects the most informative subset of sentences towards a global objective (McDonald, 2007; Gillick et al., 2009; Aker et al., 2010). Some work uses integer linear programming to find the exact solution (Gillick et al., 2009; Li et al., 2015), other work employs supervised methods to optimize the ROUGE scores of a summary (Lin and Bilmes, 2011; Kulesza and Taskar, 2012). Here we use the ROUGE scores of the candidate summaries as labels while training our model.

In our work, we propose novel features that encode the content quality of the entire summary. Though prior work has extensively investigated features that are indicative of important words (Yih et al., 2007; Hong and Nenkova, 2014) or sentences (Litvak et al., 2010; Ouyang et al., 2011), little work has focused on designing global features defined over the summary. Indeed, even for the papers that employ supervised methods to conduct global inference, the features are defined on the sentence level (Aker et al., 2010; Kulesza and Taskar, 2012). The most closely related papers are the ones that investigated automatic evaluation of summarization without human references (Louis and Nenkova, 2009; Saggion et al., 2010), where the effectiveness of several summary-input similarity metrics are examined. In our work, we propose a wide range of features. These features are derived not only based on the input, but also based on the basic summaries and the summary-input pairs from the New York Times (NYT) corpus (Sandhaus, 2008).

3 Data and Evaluation

We conduct a large scale experiment on six datasets from the Document Understanding Conference (DUC) and the Text Analysis Conference (TAC). The tasks include generic (DUC 2001–2004) and query-focused (TAC 2008, 2009) multi-document summarization. We evaluate on the task of generating 100-word summaries.

We use ROUGE (Lin, 2004) for automatic

evaluation, which compares the machine summaries to the human references. We report ROUGE-1 (unigram recall) and ROUGE-2 (bigram recall), with stemming and stopwords included.[1] Among automatic evaluation metrics, ROUGE-1 (R-1) can predict that one system performs significantly better than the other with the highest recall (Rankel et al., 2013). ROUGE-2 (R-2) provides the best agreement with manual evaluations (Owczarzak et al., 2012). R-1 and R-2 are the most widely used metrics in summarization literature.

4 Generating Candidate Summaries

We first introduce the four basic unsupervised systems, then describe our approach of generating candidate summaries. The four systems all perform extractive summarization, which directly selects *sentences* from the input. Among these systems, ICSISumm achieves the highest ROUGE-2 in the TAC 2008, 2009 workshops.[2] The other systems are often used as competitive baselines; we implement these ourselves. Table 1 shows their performances. The word overlap between summaries generated by these systems is low, which indicates high diversity.

The basic systems are used for both generic and query-focused summarization. For the latter task, we filter out the sentences that have no overlap with the query in terms of content words for the systems that we implemented.

4.1 Four Basic Unsupervised Systems

ICSISumm: This system (Gillick et al., 2009) optimizes the coverage of bigrams weighted by their document frequency within the input using Integer Linear Programming (ILP). Even though this problem is NP-hard, a standard ILP solver can find the exact solution fairly quickly in this case.

Greedy-KL: This system aims to minimize the Kullback-Leibler (KL) divergence between the word probability distribution of the summary and that of the input. Because finding the summary with the smallest KL divergence is intractable, we employ a greedy method that iteratively selects an additional sentence that minimizes the KL divergence (Haghighi and Vanderwende, 2009).

ProbSum: This system (Nenkova et al., 2006) scores a sentence by taking the average of word probabilities over the words in the sentence, with stopwords assigned zero weights. Compared to Nenkova et al. (2006), we slightly change the way of handling redundancy: we iteratively include a sentence into the summary if its cosine similarity with any sentence in the summary does not exceed 0.5.[3]

LLRSum: This system (Conroy et al., 2006) employs a log-likelihood ratio (LLR) test to select *topic words* of an input (Lin and Hovy, 2000). The LLR test compares the distribution of words in the input to a large background corpus. Similar to Conroy et al. (2006), we consider words as topic words if their χ-square statistic derived by LLR exceeds 10. The sentence importance score is equal to the number of topic words divided by the number of words in the sentence. Redundancy is handled in the same way as in ProbSum.

4.2 Generating Candidate Summaries

4.2.1 Selecting a Full Summary

There does not exist a system that always outperforms the others for all problems. Based on this fact, we directly use the summary outputs (i.e., basic summaries) as the candidate summaries.

4.2.2 Sentence Level Combination

Different systems provide different pieces of the correct answer. Based on this fact, the combined summary should include sentences that appear in the summaries produced by different systems. Here we exhaustively enumerate sentences so that to form the candidate summaries. A similar approach has been used to generate candidate summaries for single-document summarization (Ceylan et al., 2010).

Let $D = s_1, \ldots, s_n$ denote the sequence of unique sentences that appear in the basic summaries. We enumerate all subsequences $A_i = s_{i_1}, \ldots, s_{i_k}$ of D in lexicographical order. A_i can be used as a candidate summary iff $\sum_{j=1}^{k} l(s_{i_j}) \geq L$ and $\sum_{j=1}^{k-1} l(s_{i_j}) < L$, where $l(s)$ is the number of words in s and L is the predefined summary length. Table 2 shows the average number of (unique) sentences and summaries that are generated per input.

[1] ROUGE version 1.5.5 with arguments: -c 95 -r 1000 -n 2 -2 4 -u -m -a -l 100 -x

[2] We use the toolkit provided via this link directly: `https://code.google.com/p/icsisumm/`

[3] The threshold is determined on the development set.

	DUC 01		DUC 02		DUC 03		DUC 04		TAC 08		TAC 09	
	R-1	R-2	R-1	R-2	R-1	R-2	R-1	R-2	R-1	R-2	R-1	R-2
ICSISumm	0.342	0.079	0.373	0.095	0.381	0.103	0.384	0.098	0.388	0.119	0.393	0.121
Greedy-KL	0.331	0.067	0.358	0.075	0.383	0.086	0.383	0.090	0.372	0.094	0.384	0.099
ProbSum	0.303	0.056	0.326	0.071	0.360	0.088	0.354	0.082	0.350	0.087	0.357	0.094
LLRSum	0.318	0.067	0.329	0.068	0.354	0.085	0.359	0.081	0.372	0.096	0.364	0.097
SumOracle R-1	**0.361**	0.084	**0.391**	0.103	**0.407**	0.106	**0.403**	0.103	**0.408**	0.124	**0.417**	0.130
SumOracle R-2	0.349	**0.090**	0.385	**0.106**	0.398	**0.113**	0.394	**0.108**	0.403	**0.129**	0.411	**0.136**
SentOracle R-1	**0.400**	0.097	**0.439**	0.121	**0.442**	0.123	**0.437**	0.119	**0.448**	0.139	**0.453**	0.146
SentOracle R-2	0.368	**0.109**	0.416	**0.134**	0.422	**0.136**	0.420	**0.131**	0.430	**0.152**	0.437	**0.158**

Table 1: The performance of the basic systems and the performance of the oracle systems based on the methods described in Section 4.2.1 and Section 4.2.2. The evaluation metric that each oracle optimizes is shown in **Bold**.

Dataset	# sents	# unique	# summaries	# total
DUC 01	20.8	17.7	7498	224940
DUC 02	21.1	17.6	12048	710832
DUC 03	19.3	15.4	3448	103440
DUC 04	19.5	15.6	3270	163500
TAC 08	18.5	14.8	2436	107184
TAC 09	18.0	13.7	1328	63744

Table 2: Average number of sentences (# sents), unique sentences (# unique), candidate summaries per input (# summaries) and the total number of candidate summaries for each dataset (# total).

Note that we consider the order of sentences in A_i (generated from D) as a relatively unimportant factor. Though two summaries with the same set of sentences can have different ROUGE scores due to the truncation of the last sentence, because the majority of content covered is still the same, the difference in ROUGE score is relatively small. In order to generate other possible summaries, one needs to swap the last sentence. However, the total number of summaries per dataset is already huge (see Table 2). Therefore, we do not generate other candidate summaries, because it would cost much more additional space, while the difference in content is relatively small.

4.2.3 Comparison of the Oracle Systems

We examine the upper bounds of the two methods described in Section 4.2.1 and Section 4.2.2. For the first method, we design two oracle systems that pick the basic summary with the highest ROUGE-1 (R-1) and ROUGE-2 (R-2) (denoted as SumOracle R-1 and SumOracle R-2). For the second method, we design two oracle systems that pick the best summary in terms of R-1 and R-2 among the summary candidates (denoted as SentOracle R-1 and SentOracle R-2). As shown in Table 1, the advantage of the first two oracles over ICSISumm is limited: on average $0.021/0.006$ and $0.013/0.011$ (R-1/R-2). However, the advantage of the latter oracles over ICSISumm is much larger: on average $0.060/0.022$ and $0.039/0.034$ (R-1/R-2). Clearly, system combination is more promising if we combine the basic summaries at the sentence level. Therefore, we adopt the latter method to generate candidate summaries.

5 Features

We introduce the features used in our model that selects among the candidate summaries. Traditionally in summarization, features are derived based on the input (denoted as I). In our work, we propose a class of novel features that compares the candidate summary to the set of the basic summaries (denoted as H), where H can be regarded as a *hyper-summary* of I. This excels in the way that it takes advantage of the consensus between systems. Moreover, we propose system identity features, which capture the fact that content from a better system should have a higher chance to be selected.

Our model includes classical indicators of content importance (e.g., frequency, locations) and novel features that have been recently proposed for other tasks. For example, we design features that estimate the intrinsic importance of words from a large corpus (Hong and Nenkova, 2014). We also include features that compute the information density of the first sentence that each word appears in (Yang and Nenkova, 2014). These features are specifically tailored for our task (see Section 5.2).

We classify our features into summary level, word level and system identity features. Note that we do not consider stopwords and do not perform stemming. There are 360 features in our model.

5.1 Summary Level Features

Summary level features directly encode the informativeness of the entire summary. Some of them are initially proposed in Louis and Nenkova (2013) that evaluates the summary content without human models. Different from them, the features in our work use not only I, but also H as the "*input*" (except for the redundancy features). "*Input*" refers to I or H in the rest of Section 5.

Distributional Similarity: These features compute the distributional similarity (divergence) between the n-gram ($n = 1, 2$) probability distribution of the summary and that of the *input* (I or H). Good summaries tend to have high similarity and low divergence. We use three measures: Kullback-Leibler (KL) divergence, Jenson-Shannon (JS) divergence and cosine similarity.

Let P and Q denote the n-gram distribution of the summary and that of the *input* respectively. Let $p_\lambda(w)$ be the probability of n-gram w in distribution λ. The KL divergence $KL(P \parallel Q)$ and the JS divergence $JS(P \parallel Q)$ are defined as:

$$KL(P \parallel Q) = \sum_w p_P(w) \cdot \log \frac{p_P(w)}{p_Q(w)} \quad (1)$$

$$JS(P \parallel Q) = \frac{1}{2} KL(P \parallel A) + \frac{1}{2} KL(Q \parallel A) \quad (2)$$

where A is the average of P and Q. Noticing that KL divergence is not symmetric, both $KL(P \parallel Q)$ and $KL(Q \parallel P)$ are computed. In particular, smoothing is performed while computing $KL(Q \parallel P)$, where we use the same setting as in Louis and Nenkova (2013).

Topic words: Good summaries tend to include more topic words (TWs). We derive TWs using the method described in the LLRSum system in Section 4.1. For each summary S, we compute: (1) the ratio of the words that are TWs to all words in S; (2) the recall of TWs in S.

Sentence location: Sentences that appear at the beginning of an article are likely to be more critical. Greedy-based summarizers (ProbSum, LLRSum, GreedyKL) also select important sentences first. To capture these intuitions, we set features over the sentences in a summary (S) based on their locations. There are features that indicate whether a sentence in S has appeared as the first sentence in the *input*. We also set features to indicate the normalized position of a sentence in the documents of an *input*: by assigning 1 to the first sentence, 0 to the last sentence. When one sentence appears multiple times, the earliest position is used. Features are then set on the summary level, which equal to the mean of their corresponding features on the sentence level over all sentences in the summary S.

Redundancy: Redundancy correlates negatively with content quality (Pitler et al., 2010). To indicate redundancy, we compute the maximum and average cosine similarity of all pairs of sentences in the summaries. Summaries with higher redundancy are expected to score higher.

5.2 Word Level Features

Better summaries should include words or phrases that are of higher importance. Hence, we design features to encode the overall importance of unigrams and bigrams in a summary. We first generate features for the n-grams ($n = 1, 2$) in a summary S, then generate the feature vector $\mathbf{v_S}$ for S. The procedure is as follows:

Let t denote the unigram or bigram in a summary. For each t that includes content words, we form $\mathbf{v_t}$, where each component of $\mathbf{v_t}$ is an importance indicator of t. If t does not include any content words, we set $\mathbf{v_t} = \mathbf{0}$. Let S' denote the unique n-grams in S and let L denote the summary length. We compute two feature vectors: $\mathbf{v_{S_1}} = (\sum_{t \in S} \mathbf{v_t})/L$ and $\mathbf{v_{S_2}} = (\sum_{t \in S'} \mathbf{v_t})/L$, which are the coverage of n-grams by word token and word type, normalized by summary length. Finally, $\mathbf{v_S}$ is formed by concatenating $\mathbf{v_{S_1}}$ and $\mathbf{v_{S_2}}$ for unigrams and bigrams.

Below we describe the features in $\mathbf{v_t}$. Similar to Section 5.1, the features are computed based on both I and H. We also derive features based on summary-article pairs from the NYT corpus.

Frequency related features: For each n-gram t, we compute its probability, TF*IDF[4], document frequency (DF) and χ-square statistic from LLR test. Another feature is set to be equal to DF normalized by the number of input documents. A binary feature is set to determine whether DF is at least three, inspired by the observation that document specific words should not be regarded as informative (Mason and Charniak, 2011).

It has been shown that unimportant words of an input should not be considered while scoring the summary (Gupta et al., 2007; Mason and Charniak, 2011). The features below are designed

[4]IDF is computed using the news articles between year 2004 and 2007 of the New York Times corpus.

capture this. Let the binary function $b(t)$ denote whether or not t includes topic words (which approximate whether or not t is important), features are set to be equal to the product of the DF related features and $b(t)$.

Word locations: The words that appear close to the beginning of I or H are likely to be important. Here for each n-gram token, we compute its normalized locations in the documents. Then for each n-gram type t, we compute its *first*, *average*, *last* and *average first* location across its occurrences in all documents of an *input*. Features are also set to determine whether t has appeared in the first sentence and the number of times t appears in the first sentences of an *input*.

Information density of the first sentence: The first sentence of an article can be either informative or entertaining. Clearly, the words that appear in an informative first sentence should be assigned higher importance scores. To capture this, we compute the importance score (called information density in Yang and Nenkova (2014)) of the first sentence, that is defined as the number of TWs divided by the number of words in the sentence. For each t, we compute the maximal and average of importance scores over all first sentences that t appears in.

Global word importance: Some words are globally important (e.g., "war", "death") or unimportant (e.g., "Mr.", "a.m.") to humans, independent of a particular input. Hong and Nenkova (2014) proposed a class of methods to estimate the global importance of words, based on the change of word probabilities between the summary-article pairs from the NYT corpus. The importance are used as features for identifying words that are used in human summaries. Here we replicate the features used in that work, except that we perform more careful pre-processings. This class of features are set only for unigrams.

5.3 System Identity Features

For each basic system A_i, we compute the sentence and n-gram overlap between S and the summary from A_i (S_{A_i}). We hypothesize that the quality (i.e., ROUGE score) of a summary is positively (negatively) correlated to the overlap between this summary and a good (bad) basic summary of the same input. We design six sentence and two word overlap features for each system, which leads to a total of 32 features.

Sentence overlap: Let D_0, D_{A_i} denote the set of sentences in S and S_{A_i}, respectively. For each system A_i, we set a feature $|D_0 \bigcap D_{A_i}|/|D_0|$. We further consider sentence lengths. Let $l(D)$ denote the total length of sentences in set D, we set a feature $l(D_0 \bigcap D_{A_i})/l(D_0)$ for each system A_i. Lastly, we compute the binary version of $|D_0 \bigcap D_{A_i}|/|D_0|$.

Furthermore, we exclude the sentences that appear in multiple basic summaries from D_0, then compute the three features above for the new D_0. System identity features might be more helpful in selecting among the sentences that are generated by only one of the systems.

N-gram overlap: We compute the fraction of n-gram (n = $1, 2$) tokens in S that appears in S_{A_i}. The n-grams consisting of solely stopwords are removed before computation.

6 Baseline Approaches

We present three summary combination methods that are used as baselines:

Voting: We select sentences according to the total number of times that they appear in all basic summaries, from large to small. When there are ties, we randomly pick an unselected sentence. The procedure is repeated 100 times and the mean ROUGE score is reported.

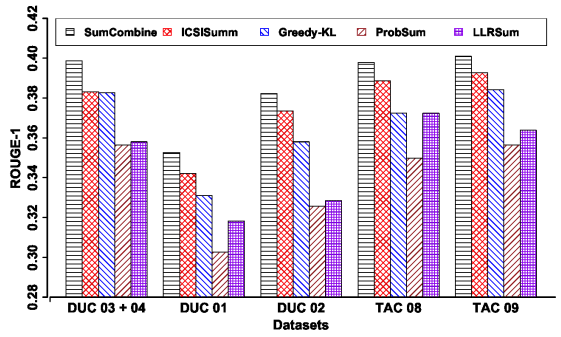
Summarization from Summaries: We directly run ICSISumm and Greedy-KL over the summaries from the basic systems.

Jensen-Shannon (JS) Divergence: We select among the pool of candidate summaries. The summary with the smallest JS divergence between the summary and (1) the input (JS-I), or (2) the hyper-summaries (JS-H) is selected. Summary-input JS divergence is the best metric to identify a better summarizer without human references (Louis and Nenkova, 2009).

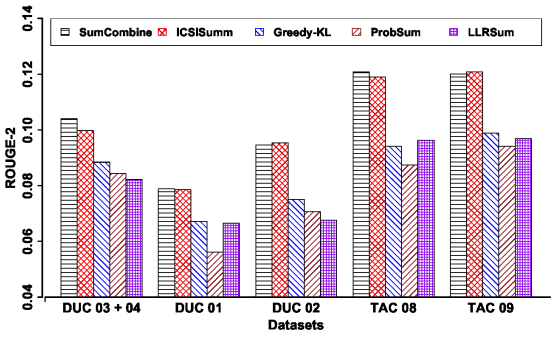
7 Experiments and Results

7.1 Experiment Settings

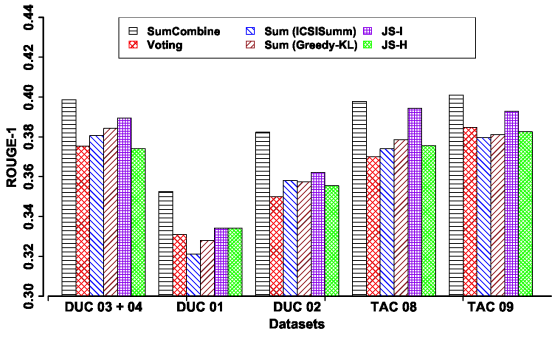
We use the DUC 03, 04 datasets as training and development sets. The candidate summaries of these two sets are used as training instances. There are 80 input sets; each input includes an average of 3336 candidate summaries. During development, we perform four-fold cross-validation. The DUC 01, 02 and TAC 08, 09 datasets are used as the held-out test sets. We use two-sided Wilcoxon test to compare the performance between two systems.



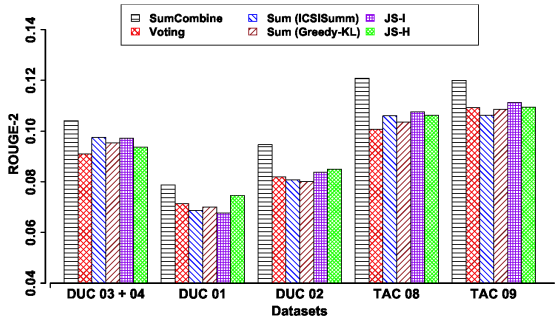
(a) ROUGE-1 of the proposed and the basic systems



(b) ROUGE-2 of the proposed and the basic systems



(c) ROUGE-1 of the proposed and baseline approaches



(d) ROUGE-2 of the proposed and baseline approaches

Figure 1: ROUGE scores of different systems on the DUC 2001–2004 and TAC 2008, 2009 datasets

We choose ROUGE-1 (R-1) as training labels, as it outperforms using ROUGE-2 (R-2) as labels (see Table 3). We suspect that the advantage of R-1 is because it has higher sensitivity in capturing the differences in content between summaries.[5]

In order to find a better learning method, we have experimented with support vector regression (SVR) (Drucker et al., 1997)[6] and SVM-Rank (Joachims, 1999).[7] SVR has been used for estimating sentence (Ouyang et al., 2011) or document (Aker et al., 2010) importance in summarization. SVM-Rank has been used for ranking summaries according to their linguistic qualities (Pitler et al., 2010). In SVM-Rank, only the relative ranks between training instances of an input are considered while learning the model. Our experiment shows that SVR outperforms SVM-Rank (see Table 3). This means that it is useful to compare the summaries across different input sets and leverage the actual ROUGE scores.

Settings	R-1	R-2
SVR + R-1	0.3986	0.1040
SVR + R-2	0.3890	0.1023
SVMRank + R-1	0.3932	0.0996
SVMRank + R-2	0.3854	0.0982

Table 3: Performance on the development set with different models and training labels.

7.2 Comparing with the Basic Systems and the Baseline Methods

We evaluate our model on the development set and the test sets. As shown in Figure 1 (a) and Table 4, our model performs consistently better than all basic systems on R-1. It performs similar to ICSISumm and better than the other basic systems on R-2 (see Figure 1 (b) and Table 4).

Apart from automatic evaluation, we also manually evaluate the summaries using the Pyramid method (Nenkova et al., 2007). This method solicits annotators to score a summary based on its coverage of summary content units, which are identified from human references. Here we evaluate the Pyramid scores of four systems: our system, two best basic systems and the oracle

[5]Recent methods that performs global optimization for summarization mostly use R-1 while training (Lin and Bilmes, 2011; Kulesza and Taskar, 2012; Sipos et al., 2012).

[6]We use the SVR model in SVMLight (Joachims, 1999) with linear kernel and default parameter settings when trained on R-1. When trained on R-2, we tune ϵ in loss function on the developmenet set, because the default setting assigns the same value to all data points.

[7]We use the SVM-Rank toolkit (Joachims, 2006) with default parameter settings.

Dataset	System	R-1	R-2
DUC 03	ICSISumm	0.3813	0.1028
	SumCombine	**0.3959**	0.1018
DUC 04	ICSISumm	0.3841	0.0978
	SumCombine	**0.3995**	**0.1048**
	DPP	0.3979	0.0962
	RegSum	0.3857	0.0975
DUC 01	ICSISumm	0.3421	0.0785
	SumCombine	0.3526†	0.0788
	R2N2_ILP	0.3691	0.0787
	PriorSum	0.3598	0.0789
DUC 02	ICSISumm	0.3733	0.0954
	SumCombine	**0.3823**	0.0946
	R2N2_ILP	0.3796	0.0888
	PriorSum	0.3663	0.0897
	ClusterCMRW	0.3855	0.0865
TAC 08	ICSISumm	0.3880	0.1186
	SumCombine	0.3978	0.1208
	Li et al. (2013)	n/a	0.1235
	A & M (2013)	n/a	0.1230
	Li et al. (2015)	n/a	0.1184
TAC 09	ICSISumm	0.3931	0.1211
	SumCombine	0.4009†	0.1200
	Li et al. (2015)	n/a	0.1277

Table 4: Performance comparison on six DUC and TAC datasets. **Bold** indicates statistical significant compared to ICSISumm ($p < 0.05$). † indicates the difference is close to significant compared to ICSISumm ($0.05 \leq p < 0.1$).

on the TAC 08 dataset. Our model (Combine) outperforms ICSISumm and Greedy-KL by 0.019 and 0.090, respectively (see Table 5).

	Oracle	Combine	ICSISumm	KL
Pyr. score	0.626	0.549	0.530	0.459

Table 5: The Pyramid score on the TAC 08 data.

Figure 1 (c), (d) compare our model with the baseline approaches proposed in Section 6. The baselines that only consider the consensus between different systems perform poorly (voting, summarization on summaries, JS-H). JS-I has the best ROUGE-1 among baselines, while it is still much inferior to our model. Therefore, effective system combination appears to be difficult using methods based on a single indicator.

7.3 Comparing with the State-of-the-art

Table 4 compares our model (SumCombine) with the state-of-the-art systems. On the DUC 03 and 04 data, ICSISumm is among one of the best systems. SumCombine performs significantly better compared to it on R-1. We also achieve a better performance compared to the other top performing extractive systems (DPP (Kulesza and Taskar, 2012), RegSum (Hong and Nenkova, 2014)) on the DUC 04 data.

On the DUC 01 and 02 data, the top performing systems we find are R2N2_ILP (Cao et al., 2015a) and PriorSum (Cao et al., 2015b); both of them utilize neural networks. Comparing to these two, SumCombine achieves a lower performance on the DUC 01 data and a higher performance on the DUC 02 data. It also has a slightly lower R-1 and a higher R-2 compared to ClusterCMRW (Wan and Yang, 2008), a graph-based system that achieves the highest R-1 on the DUC 02 data. On the TAC 08 data, the top performing systems (Li et al., 2013; Almeida and Martins, 2013) achieve the state-of-the-art performance by sentence compression. Our model performs extractive summarization, but still has similar R-2 compared to theirs.[8] On the TAC 09 data, the best system uses a supervised method that weighs bigrams in the ILP framework by leveraging external resources (Li et al., 2015). This system is better than ours on the TAC 09 data and is inferior to ours on the TAC 08 data.

Overall, our combination model achieves very competitive performance, comparable to the state-of-the-art on multiple benchmarks.

At last, we compare SumCombine to SSA (Pei et al., 2012) and WCS (Wang and Li, 2012), the models that perform system combination by rank aggregation. The systems are evaluated on the DUC 04 data. In order to compare with these two papers, we truncate our summaries to 665 bytes and report F_1-score. Pei et al. (2012) report the performance on 10 randomly selected input sets. In order to have the same size of training data with them, we conduct five-fold cross-validation.

System	R-1	R-2	R-SU4
SumCombine	0.3943	0.1015	0.1411
SSA (Pei et al., 2012)	0.3977	0.0953	0.1394
WCS (Wang and Li, 2012)	0.3987	0.0961	0.1353

Table 6: Comparison with other combination methods on the DUC 04 dataset.

As shown in Table 6, SumCombine performs better than SSA and WCS on R-2 and R-SU4, but not on R-1. It is worth noting that these three

[8]These papers report ROUGE-SU4 (R-SU4) (measures skip bigram with maximum gap of 4) instead of R-1. Our model has very similar R-SU4 (-0.0002/$+0.0007$) compared to them.

	Dev. Set		DUC 01		DUC 02		TUC 08		TAC 09		Average	
	R-1	R-2	R-1	R-2	R-1	R-2	R-1	R-2	R-1	R-2	R-1	R-2
All features	.3986	.1040	.3526	.0788	.3823	.0946	.3978	.1208	.4009	.1200	.3864	.1036
-summary	.3946	.1014	.3469	.0779	**.3760**	.0872	.3950	.1185	.3988	.1191	.3823	.1008
-word	**.3946**	.1002†	**.3429**	**.0733**	.3787	.0919	.3939	.1172	.3988	.1232	.3829	.1012
-system	.3964	.1022	.3483	.0776	.3772	.0895	.4009	.1193	.3936	**.1110**	.3833	.0999
-input	**.3822**	**.0956**	**.3433**	.0764	.3786	.0912	**.3858**	.1148	.3960	.1159	.3772	.0988
-hyper-sum	.3978	.1022	.3512	.0777	.3806	.0918	.3968	.1193	.3994	.1177	.3852	.1017
-global	.3948	.1021	.3457	.0760	.3821	.0954	.3959	.1136	.4010	.1215	.3839	.1017
summary	.3960	.1018	**.3344**	**.0701**	.3748	.0910	.3957	.1166	.4009	.1170	.3804	.0993
word	**.3919**	.1006	.3492	.0765	.3784	.0905	.3956	.1166	.3956	.1146	.3821	.0998
system	**.3881**	**.0958**	**.3430**	.0746	**.3689**	**.0868**	.3898	**.1096**	.3926	.1145	.3765	.0963
input	.3979	.1009	**.3410**	.0729†	.3764	.0904	.3907	.1129	.4015	.1189	.3815	.0992
hyper-sum	**.3852**	**.0952**	.3447	**.0725**	**.3665**	**.0823**	.3871†	**.1080**	.3906†	.1140	.3748	.0944

Table 7: Performance after ablating features (row 2–7) or using a single class of features (row 8–12). **Bold** and † represent statistical significant ($p < 0.05$) and close to significant ($0.05 \leq p < 0.1$) compared to using all features (two-sided Wilcoxon test).

systems cannot be directly compared, because different basic systems are used. In fact, compared to SumCombine, SSA and WCS achieve larger improvements over the basic systems that are used. This might be because ranker aggregation is a better strategy, or because combining weaker systems is easier to result in large improvements.

7.4 Effects of Features

We conduct two experiments to examine the effectiveness of features (see Table 7). First, we remove one class of feature at a time from the full feature set. Second, we show the performance of a single feature class. Apart from reporting the performance on the development and the test sets, we also show the macro average performance across the five sets.[9] This helps to understand the contribution of different features in general.

Summary level, word level and system identity features are all useful, with ablating them leads to an average of 0.0031 to 0.0041 decrease on R-1. Ablating summary and word level features can lead to a significant decrease in performance on some sets. If we use a single set of features, then the summary and word level features turn out to be more useful than the system identity features.

The word and summary level features compute the content importance based on three sources: the input, the basic summaries (hyper-sum) and the New York Times corpus (global). We ablate the features derived from these three sources respectively. The input-based features are the most important; removing them leads to a very large decrease in performance, especially on R-1. The features derived from the basic summaries are also effective; even though removing them only lead to a small decrease in performance, we can observe the decrease on all five sets. Ablating global indicators leads to an average decrease of about 0.002 on R-1 and R-2.

Interestingly, for the same feature class, the effectiveness vary to a great extent across different datasets. For example, ablating word level features decreases the R-2 significantly on the DUC 01 data, but increases the R-2 on the TAC 09 data. However, by looking at the average performance, it becomes clear that it is necessary to use all features. The features computed based on the input are identified as the most important.

8 Conclusion

In this paper, we present a pipeline that combines the summaries from four portable unsupervised summarizers. We show that system combination is very promising in improving content quality. We propose a supervised model to select among the candidate summaries. Experiments show that our model performs better than the systems that are combined, which is comparable to the state-of-the-art on multiple benchmarks.

Acknowledgements

We thank the reviewers for their insightful and constructive comments. Kai Hong would like to thank Yumeng Ou, Mukund Raghothaman and Chen Sun for providing feedback on earlier version of this paper. This work was funded by NSF CAREER award IIS 0953445.

[9] We do not compute the statistical significance for the average score.

References

Ahmet Aker, Trevor Cohn, and Robert Gaizauskas. 2010. Multi-document summarization using A* search and discriminative learning. In *Proceedings of EMNLP*, pages 482–491.

Miguel Almeida and André F.T. Martins. 2013. Fast and robust compressive summarization with dual decomposition and multi-task learning. In *Proceedings of ACL*, pages 196–206.

Srinivas Bangalore, German Bordel, and Giuseppe Riccardi. 2001. Computing consensus translation from multiple machine translation systems. In *Proceedings of ASRU*, pages 351–354.

Ziqiang Cao, Furu Wei, Li Dong, Sujian Li, and Ming Zhou. 2015a. Ranking with recursive neural networks and its application to multi-document summarization. In *Proceedings of AAAI*, pages 2153–2159.

Ziqiang Cao, Furu Wei, Sujian Li, Wenjie Li, Ming Zhou, and Houfeng Wang. 2015b. Learning summary prior representation for extractive summarization. In *Proceedings of ACL: Short Papers*, pages 829–833.

Hakan Ceylan, Rada Mihalcea, Umut Özertem, Elena Lloret, and Manuel Palomar. 2010. Quantifying the limits and success of extractive summarization systems across domains. In *Proceedings of ACL*, pages 903–911.

John M. Conroy, Judith D. Schlesinger, and Dianne P. O'Leary. 2006. Topic-focused multi-document summarization using an approximate oracle score. In *Proceedings of COLING/ACL*, pages 152–159.

Harris Drucker, Chris J.C. Burges, Linda Kaufman, Alex Smola, Vladimir Vapnik, et al. 1997. Support vector regression machines. In *Proceedings of NIPS*, volume 9, pages 155–161.

Gunes Erkan and Dragomir R. Radev. 2004. Lexrank: graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research*, 22(1):457–479.

Jonathan G. Fiscus. 1997. A post-processing system to yield reduced word error rates: Recognizer output voting error reduction (ROVER). In *Proceedings of ASRU*, pages 347–354.

Robert Frederking and Sergei Nirenburg. 1994. Three heads are better than one. In *Proceedings of ANLP*, pages 95–100.

Dan Gillick, Benoit Favre, Dilek Hakkani-Tur, Berndt Bohnet, Yang Liu, and Shasha Xie. 2009. The ICSI/UTD Summarization System at TAC 2009. In *Proceedings of TAC*.

Surabhi Gupta, Ani Nenkova, and Dan Jurafsky. 2007. Measuring importance and query relevance in topic-focused multi-document summarization. In *Proceedings of ACL*, pages 193–196.

Aria Haghighi and Lucy Vanderwende. 2009. Exploring content models for multi-document summarization. In *Proceedings of HLT-NAACL*, pages 362–370.

John C. Henderson and Eric Brill. 1999. Exploiting diversity for natural language processing: Combining parsers. In *Proceedings of EMNLP*, pages 187–194.

Kai Hong and Ani Nenkova. 2014. Improving the estimation of word importance for news multi-document summarization. In *Proceedings of EACL*, pages 712–721.

Kai Hong, John M. Conroy, Benoit Favre, Alex Kulesza, Hui Lin, and Ani Nenkova. 2014. A repositary of state of the art and competitive baseline summaries for generic news summarization. In *Proceedings of LREC*, pages 1608–1616.

Thorsten Joachims. 1999. Making large-scale SVM learning practical. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, chapter 11, pages 169–184. MIT Press, Cambridge, MA.

Thorsten Joachims. 2006. Training linear svms in linear time. In *Proceedings of KDD*, pages 217–226.

Alex Kulesza and Ben Taskar. 2012. Determinantal point processes for machine learning. *Foundations and Trends in Machine Learning*, 5(2–3).

Chen Li, Fei Liu, Fuliang Weng, and Yang Liu. 2013. Document summarization via guided sentence compression. In *Proceedings of EMNLP*, pages 490–500.

Chen Li, Yang Liu, and Lin Zhao. 2015. Using external resources and joint learning for bigram weighting in ilp-based multi-document summarization. In *Proceedings of NAACL-HLT*, pages 778–787.

Hui Lin and Jeff Bilmes. 2011. A class of submodular functions for document summarization. In *Proceedings of ACL*, pages 510–520.

Chin-Yew Lin and Eduard Hovy. 2000. The automated acquisition of topic signatures for text summarization. In *Proceedings of COLING*, pages 495–501.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, pages 74–81.

Marina Litvak, Mark Last, and Menahem Friedman. 2010. A new approach to improving multilingual summarization using a genetic algorithm. In *Proceedings of ACL*, pages 927–936.

Annie Louis and Ani Nenkova. 2009. Automatically evaluating content selection in summarization without human models. In *Proceedings of EMNLP*, pages 306–314.

Annie Louis and Ani Nenkova. 2013. Automatically assessing machine summary content without a gold standard. *Computational Linguistics*, 39(2):267–300.

Lidia Mangu, Eric Brill, and Andreas Stolcke. 2000. Finding consensus in speech recognition: word error minimization and other applications of confusion networks. *Computer Speech & Language*, 14(4):373–400.

Rebecca Mason and Eugene Charniak. 2011. Extractive multi-document summaries should explicitly not contain document-specific content. In *Proceedings of the Workshop on Automatic Summarization for Different Genres, Media, and Languages*, pages 49–54.

Ryan McDonald. 2007. A study of global inference algorithms in multi-document summarization. In *Proceedings of ECIR*, pages 557–564.

Ahmed A Mohamed and Sanguthevar Rajasekaran. 2005. A text summarizer based on meta-search. In *Proceedings of ISSPIT*, pages 670–674.

Ani Nenkova, Lucy Vanderwende, and Kathleen McKeown. 2006. A compositional context sensitive multi-document summarizer: exploring the factors that influence summarization. In *Proceedings of SIGIR*, pages 573–580.

Ani Nenkova, Rebecca Passonneau, and Kathleen McKeown. 2007. The pyramid method: Incorporating human content selection variation in summarization evaluation. *ACM Transactions on Speech and Language Processing (TSLP)*, 4(2):4.

You Ouyang, Wenjie Li, Sujian Li, and Qin Lu. 2011. Applying regression models to query-focused multi-document summarization. *Inf. Process. Manage.*, 47(2):227–237, March.

Karolina Owczarzak, John M. Conroy, Hoa Trang Dang, and Ani Nenkova. 2012. An assessment of the accuracy of automatic evaluation in summarization. In *Proceedings of NAACL-HLT 2012: Workshop on Evaluation Metrics and System Comparison for Automatic Summarization*, pages 1–9.

Yulong Pei, Wenpeng Yin, Qifeng Fan, and Lian'en Huang. 2012. A supervised aggregation framework for multi-document summarization. In *Proceedings of COLING*, pages 2225–2242.

Emily Pitler, Annie Louis, and Ani Nenkova. 2010. Automatic evaluation of linguistic quality in multi-document summarization. In *Proceedings of ACL*, pages 544–554.

Peter A. Rankel, John M. Conroy, Hoa Trang Dang, and Ani Nenkova. 2013. A decade of automatic content evaluation of news summaries: Reassessing the state of the art. In *Proceedings of ACL*, pages 131–136.

Kenji Sagae and Alon Lavie. 2006. Parser combination by reparsing. In *Proceedings of NAACL: Short Papers*, pages 129–132.

Horacio Saggion, Juan-Manuel Torres-Moreno, Iria da Cunha, and Eric SanJuan. 2010. Multilingual summarization evaluation without human models. In *Proceedings of COLING*, pages 1059–1067.

Evan Sandhaus. 2008. The new york times annotated corpus. *Linguistic Data Consortium, Philadelphia, PA*.

Ruben Sipos, Pannaga Shivaswamy, and Thorsten Joachims. 2012. Large-margin learning of submodular summarization models. In *Proceedings of EACL*, pages 224–233.

Vishal Thapar, Ahmed A Mohamed, and Sanguthevar Rajasekaran. 2006. Consensus text summarizer based on meta-search algorithms. In *Proceedings of ISSPIT*, pages 403–407.

Xiaojun Wan and Jianwu Yang. 2008. Multi-document summarization using cluster-based link analysis. In *Proceedings of SIGIR*, pages 299–306.

Dingding Wang and Tao Li. 2012. Weighted consensus multi-document summarization. *Information Processing & Management*, 48(3):513–523.

Yinfei Yang and Ani Nenkova. 2014. Detecting information-dense texts in multiple news domains. In *Proceedings of AAAI*, pages 1650–1656.

Wen-tau Yih, Joshua Goodman, Lucy Vanderwende, and Hisami Suzuki. 2007. Multi-document summarization by maximizing informative content-words. In *Proceedings of IJCAI*, pages 1776–1782.

Phrase-based Compressive Cross-Language Summarization

Jin-ge Yao **Xiaojun Wan** **Jianguo Xiao**
Institute of Computer Science and Technology, Peking University, Beijing 100871, China
Key Laboratory of Computational Linguistic (Peking University), MOE, China
{yaojinge, wanxiaojun, xiaojianguo}@pku.edu.cn

Abstract

The task of cross-language document summarization is to create a summary in a target language from documents in a different source language. Previous methods only involve direct extraction of automatically translated sentences from the original documents. Inspired by phrase-based machine translation, we propose a phrase-based model to simultaneously perform sentence scoring, extraction and compression. We design a greedy algorithm to approximately optimize the score function. Experimental results show that our methods outperform the state-of-the-art extractive systems while maintaining similar grammatical quality.

1 Introduction

The task of cross-language summarization is to produce a summary in a target language from documents written in a different source language. This task is particularly useful for readers to quickly get the main idea of documents written in a source language that they are not familiar with. Following Wan (2011), we focus on English-to-Chinese summarization in this work.

The simplest and the most straightforward way to perform cross-language summarization is pipelining general summarization and machine translation. Such systems either translate all the documents before running generic summarization algorithms on the translated documents, or summarize from the original documents and then only translate the produced summary into the target language. Wan (2011) show that such pipelining approaches are inferior to methods that utilize information from both sides. In that work, the author proposes graph-based models and achieves fair amount of improvement. However, to the best of our knowledge, no previous work of this task tries to focus on summarization beyond pure sentence extraction.

On the other hand, cross-language summarization can be seen as a special kind of machine translation: translating the original documents into a brief summary in a different language. Inspired by phrase-based machine translation models (Koehn et al., 2003), we propose a phrase-based scoring scheme for cross-language summarization in this work.

Since our framework is based on phrases, we are not limited to produce extractive summaries. We can use the scoring scheme to perform joint sentence selection and compression. Unlike typical sentence compression methods, our proposed algorithm does not require additional syntactic preprocessing such as part-of-speech tagging or syntactic parsing. We only utilize information from translated texts with phrase alignments. The scoring function consists of a submodular term of compressed sentences and a bounded distortion penalty term. We design a greedy procedure to efficiently get approximate solutions.

For experimental evaluation, we use the DUC2001 dataset with manually translated reference Chinese summaries. Results based on the ROUGE metrics show the effectiveness of our proposed methods. We also conduct manual evaluation and the results suggest that the linguistic quality of produced summaries is not decreased by too much, compared with extractive counterparts. In some cases, the grammatical smoothness can even be improved by compression.

The contributions of this paper include:

- Utilizing the phrase alignment information, we design a scoring scheme for the cross-language document summarization task.
- We design an efficient greedy algorithm to generate summaries. The greedy algorithm is

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 118–127,
Lisbon, Portugal, 17-21 September 2015.

partially submodular and has a provable constant approximation factor to the optimal solution up to a small constant.

- We achieve state-of-the-art results using the extractive counterpart of our compressive summarization framework. Performance in terms of ROUGE metrics can be significantly improved when simultaneously performing extraction and compression.

2 Background

Document summarization can be treated as a special kind of translation process: translating from a bunch of related source documents to a short target summary. This analogy also holds for cross-language document summarization, with the only difference that the languages of source documents and the target summary are different.

Our design of sentence scoring function for cross-language document summarization purpose is inspired by phrase-based machine translation models. Here we briefly describe the general idea of phrase-based translation. One may refer to Koehn (2009) for more detailed description.

2.1 Phrase-based Machine Translation

Phrase-based machine translation models are currently giving state-of-the-art translations for many pairs of languages and dominating modern statistical machine translation. Classical word-based IBM models cannot capture local contextual information and local reordering very well. Phrase-based translation models operate on lexical entries with more than one word on the source language and the target language. The allowance of multi-word expressions is believed to be the main reason for the improvements that phrase-based models give. Note that these multi-word expressions, typically addressed as phrases in machine translation literature, are essentially continuous n-grams and do not need to be linguistically integrate and meaningful constituents.

Define y as a phrase-based derivation, or more precisely a finite sequence of phrases $p_1, p_2, \ldots, p_L$. For any derivation y we use $e(y)$ to refer to the target-side translation text defined by y. This translation is derived by concatenating the strings $e(p_1), e(p_2), \ldots, e(p_L)$. The scoring scheme for a phrase-based derivation y from the source sentence to the target sentence $e(y)$ is:

$$\begin{aligned} f(y) &= \sum_{k=1}^{L} g(p_k) + LM(e(y)) \\ &\quad + \sum_{k=1}^{L-1} \eta |start(p_{k+1}) - 1 - end(p_k)| \end{aligned}$$

where $LM(\cdot)$ is the target-side language model score, $g(\cdot)$ is the score function of phrases, $\eta < 0$ is the distortion parameter for penalizing the distance between neighboring phrases in the derivation. Note that the phrases addressed here are typically continuous n-grams and need not to be grammatical linguistic phrasal units. Later we will directly use phrases provided by modern machine translation systems.

Searching for the best translation under this score definition is difficult in general. Thus approximate decoding algorithms such as beam search should be applied. Meanwhile, several constraints should be satisfied during the decoding process. The most important one is to set a constant limit of the distortion term $|start(p_{k+1}) - 1 - end(p_k)| \leq \delta$ to exhibit derivations with distant phrase translations.

3 Phrase-based Cross-Language Summarization

Inspired by the general idea of phrase-based machine translation, we describe our proposed phrase-based model for cross-language summarization in this section.

3.1 Phrase-based Sentence Scoring

In the context of cross-language summarization, here we assume that we can also have phrases in both source and target languages along with phrase alignments between the two sides. For summarization purposes, we may wish to select sentences containing more important phrases. Then it is plausible to measure the scores of these aligned phrases via importance weighing.

Inspired by phrase-based translation models, we can assign phrase-based scores to sentences from the translated documents for summarization purposes. We define our scoring function for each sentence s as:

$$\begin{aligned} F(s) &= \sum_{p \in s} d_0 g(p) + bg(s) \\ &\quad + \eta \, dist(y(s)) \end{aligned}$$

Here in the first term $g(\cdot)$ is the score of phrase p, which can be simply set to document frequency. The phrase score is penalized with a constant damping factor d_0 to decay scores for repeated phrases. The second term $bg(s)$ is the bigram score of sentence s. It is used here to simulate the effect of language models in phrase-based translation models. Denoting $y(s)$ as the phrase-based derivation (as mentioned earlier in the previous section) of sentence s, the last distortion term $dist(y(s)) = \sum_{k=1}^{L} |start(p_{k+1}) - 1 - end(p_k)|$ is exactly the same as the distortion penalty term in phrase-based translation models. This term can be used as a reflection of complexity of the translation. All the above terms can be derived from bilingual sentence pairs with phrase alignments.

Meanwhile, we may also wish to exclude unimportant phrases and badly translated phrases. Our definition can also be used to guide sentence compression by trying to remove redundant phrase.

Based on the definition over sentences, we define our summary scoring measure over a summary S:

$$F(S) = \sum_{p \in S} \sum_{i=1}^{count(p,S)} d^{i-1} g(p) + \sum_{s \in S} bg(s) + \eta \sum_{s \in S} dist(y(s))$$

where d is a predefined constant damping factor to penalize repeated occurrences of the same phrases, $count(p, S)$ is the number of occurrences in the summary S for phrase p. All other terms are inherited from the sentence score definition.

In the next section we describe our framework to efficiently utilize this scoring function for cross-language summarization.

3.2 A Greedy Algorithm for Compressed Sentence Selection

Utilizing the phrase-based score definition of sentences, we can use greedy algorithms to simultaneously perform sentence selection and sentence compression. Assuming that we have a predefined budget B (e.g. total number of Chinese characters allowed) to restrict the total length of a generated summary. We use $C(S)$ to denote the cost of a summary S, measured by the number of Chinese characters contained in total. The greedy algorithm we will use for our compressive summarization is listed in Algorithm 1.

Algorithm 1 A greedy algorithm for phrase-based summarization

1: $S \leftarrow \emptyset$
2: $i \leftarrow 1$
3: $single\_best = \text{argmax}_{s \in U, C(\{s\}) \leq B} F(\{s\})$
4: **while** $U \neq \emptyset$ **do**
5: $\quad s_i = \text{argmax}_{s \in U} \frac{F(S_{i-1} \cup \{s\}) - F(S_{i-1})}{C(\{s\})^r}$
6: $\quad$ **if** $C(S_{i-1} \cup \{s\}) \leq B$ **then**
7: $\quad\quad S_i \leftarrow S_{i-1} \cup \{s\}$
8: $\quad\quad i \leftarrow i + 1$
9: $\quad$ **end if**
10: $\quad U \leftarrow U \setminus \{s_i\}$
11: **end while**
12: **return** $S^* = \text{argmax}_{S \in \{single\_best, S_i\}} F(S)$

The space U denotes the set of all possible compressed sentences. In each iteration, the algorithm tries to find the compressed sentence with maximum gain-cost ratio (Line 5, where we will follow previous work to set $r = 1$), and merge it to the summary set at the current iteration (denoted as S_i). The target is to find the compression with maximum gain-cost ratio. This will be discussed in the next section. Note that the algorithm is also naturally applicable to extractive summarization. For extractive summarization, Line 5 corresponds to direct calculations of sentence scores based on our proposed phrase-based function and U will denote all full sentences from the original translated documents.

The outline of this algorithm is very similar to the greedy algorithm used by Morita et al. (2013) for subtree extraction, except that in our context the increase of cost function when adding a sentence is exactly the cost of that sentence.

When the distortion term is ignored ($\eta = 0$), the scoring function is clearly submodular [1] (Lin and Bilmes, 2010) in terms of the set of compressed sentences, since the score now only consists of functional gains of phrases along with bigrams of a compressed sentence. Morita et al. (2013) have proved that when $r = 1$, this greedy algorithm will achieve a constant approximation factor $\frac{1}{2}(1 - e^{-1})$ to the optimal solution. Note that this only gives us the worst case guarantee. What we can achieve in practice is usually far better.

On the other hand, setting $\eta < 0$ will not affect

[1] A set function $F : 2^U \rightarrow \mathbb{R}$ defined over subsets of a universe set U is said to be *submodular* iff it satisfies the *diminishing returns* property: $\forall S \subseteq T \subseteq U \setminus u$, we have $F(S \cup \{u\}) - F(S) \geq F(T \cup \{u\}) - F(T)$.

the performance guarantee too much. Intuitively this is because in most phrase-based translation models a distortion limit constraint $|start(p_{k+1}) - 1 - end(p_k)| \leq \delta$ will be applied on distortion terms, while performing sentence compression can never increase distortion. The main conclusion is formulated as:

Theorem 1. *If Algorithm 1 outputs S^{greedy} while the optimal solution is OPT, we have*

$$F(S^{greedy}) \geq \frac{1}{2}(1 - e^{-1})F(OPT) + \frac{1}{2}\eta\gamma.$$

Here $\gamma > 0$ is a constant controlled by distortion difference between sentences, which is relatively small in practice compared with phrase scores. $\eta < 0$ is the distortion parameter. Note that when η is set to be 0, the scoring function is submodular and then we recover the $\frac{1}{2}(1 - e^{-1})$ approximation factor as studied by Morita et al. (2013). We leave the proof of Theorem 1 to supplementary materials due to space limit. The submodularity term in the score plays an important role in the proof.

3.3 Finding the Maximum Density Compression

In Algorithm 1, the most important part is the greedy selection process (Line 5). The greedy selection criteria here is to maximize the gain-cost ratio. For compressive summarization, we are trying to compress each unselected sentence s to $\tilde{s}$, aiming at maximizing the gain-cost ratio, where the gain corresponds to

$$\begin{aligned} & F(S_{i-1} \cup \{s\}) - F(S_{i-1}) \\ = & \sum_{p \in s} \sum_{i=1}^{count(p,S)} d^{i-1} g(p) + bg(s) + \eta dist(s), \end{aligned}$$

and then add the compressed sentence $\tilde{s}$ with maximum gain-cost ratio to the summary. We will also address the compression process for each sentence as finding the maximum density compression. The whole framework forms a joint selection and compression process.

In our phrase-based scoring for sentences, although there exist no apparent optimal substructure available for exact dynamic programming due to nonlocal distortion penalty, we can have a tractable approximate procedure since the search space is only defined by local decisions on whether a phrase should be kept or dropped.

Our compression process for each sentence s is displayed in Algorithm 2. It gradually expands the set of phrases to be kept in the final compression, from the initial set of large density phrases (Line 4, assuming that phrases with large scores and small costs will always be kept), we can recover the compression with maximum density. The function $dist(\cdot, \cdot)$ is the unit distortion penalty defined as $dist(a, b) = |start(b) - 1 - end(a)|$. We define $p.score$ to be the sum of damped phrase score for phrase p, i.e. $p.score = \sum_{i=1}^{count(p,S_{i-1})} d^{i-1} g(p)$, when the current partial summary is S_{i-1}. Therefore during each iteration of the greedy selection process, the compression procedure will also be affected by sentences that have already been included. Define $p.cost$ as the number of words p contains.

Algorithm 2 A growing algorithm for finding the maximum density compressed sentence

```
1: function GET_MAX_DENSITY_COMPRESSION(s, S_{i-1})
2:     queue Q ← ∅, kept ← ∅
3:     for each phrase p in s.phrases do
4:         if p.score/p.cost > 1 then
5:             kept ← kept ∪{p}
6:             Q.enqueue(p)
7:         end if
8:     end for
9:     while Q ≠ ∅ do
10:        p ← Q.deque()
11:        ppv ← p.previous_phrase, pnx ← p.next_phrase
12:        if (ppv.score+bg(ppv,p)+ηdist(ppv,p)) / (ppv.cost+p.cost) > 1 then
13:            Q.enqueue(ppv), kept ← kept ∪{ppv}
14:        end if
15:        if (pnx.score+bg(pnx,p)+ηdist(p,pnx)) / (p.cost+pnx.cost) > 1 then
16:            Q.enqueue(pnx), kept ← kept ∪{pnx}
17:        end if
18:    end while
19: return s̃ = kept, ratio = (F(S_{i-1}∪{s̃}) − F(S_{i-1})) / s̃.cost
20: end function
```

Empirically we find this procedure gives almost the same results with exhaustive search while maintaining efficiency. Assuming that sentence length is no more than L, then the asymptotic complexity of Algorithm 2 will be $O(L)$ since the algorithm requires two passes of all phrases. Therefore the whole framework requires $O(kNL)$ time for a document cluster containing N sentences in total to generate a summary with k sentences.

In the final compressed sentence we just leave the selected phrases continuously as they are, relying on bigram scores to ensure local smoothness. The task is after all a summarization task, where bigram scores play a role of not only controlling grammaticality but keeping main information of

the original documents.

Later we will see that this compression process will not hurt grammatical fluency of translated sentences in general. In many cases it may even improve fluency by deleting redundant parentheses or removing incorrectly reordered (unimportant) phrases.

4 Experiments

4.1 Data

Currently there are not so many available datasets for our particular setting of the cross-language summarization task. Hence we only evaluate our method on the same dataset used by Wan (2011). The dataset is created by manually translating the reference summaries into Chinese from the original DUC 2001 dataset in English. We will refer to this dataset as the DUC 2001 dataset in this paper. There are 30 English document sets in the DUC 2001 dataset for multi-document summarization. Each set contains several documents related to the same topic. Three generic reference English summaries are provided by NIST annotators for each document set. All these English summaries have been translated to Chinese by native Chinese annotators.

All the English sentences in the original documents have been automatically translated into Chinese using Google Translate. We also collect the phrase alignment information from the responses of Google Translate (stored in JSON format) along with the translated texts. We use the Stanford Chinese Word Segmenter [2] for Chinese word segmentation.

The parameters in the algorithms are simply set to be $r = 1, d = 0.5, \eta = -0.5$.

4.2 Evaluation

We will report the performance of our compressive solution, denoted as PBCS (for Phrase-Based Compressive Summarization), with comparisons of the following systems:

- **PBES:** The acronym comes from Phrase-Based Extractive Summarization. It is the extractive counterpart of our solution without calling Algorithm 2.
- **Baseline (EN):** This baseline relies on merely the English-side information for English sentence ranking in the original documents. The scoring function is designed to be document frequencies of English bigrams, which is similar to the second term in our proposed sentence scoring function in Section 3.1 and is submodular. [3] The extracted English summary is finally automatically translated into the corresponding Chinese summary. This is also known as the summary translation scheme.
- **Baseline (CN):** This baseline relies on merely the Chinese-side information for Chinese sentence ranking. The scoring function is similarly defined by document frequency of Chinese bigrams. The Chinese summary sentences are then directly extracted from the translated Chinese documents. This is also known as the document translation scheme.
- **CoRank:** We reimplement the graph-based CoRank algorithm, which gives the state-of-the-art performance on the same DUC 2001 dataset for comparison.
- **Baseline (ENcomp):** This is a compressive baseline where the extracted English sentences in Baseline (EN) will be compressed before being translated to Chinese. The compression process follows from an integer linear program as described by Clarke and Lapata (2008). This baseline gives strong performance as we have found on English DUC 2001 dataset as well as other monolingual datasets.

We experiment with two kinds of summary budgets for comparative study. The first one is limiting the summary length to be no more than five sentences. The second one is limiting the total number of Chinese characters of each produced summary to be no more than 300. They will be addressed as Sentence Budgeting and Character Budgeting in the experimental results respectively.

Similar to traditional summarization tasks, we use the ROUGE metrics for automatic evaluation of all systems in comparison. The ROUGE metrics measure summary quality by counting overlapping word units (e.g. n-grams) between the candidate summary and the reference summary. Following previous work in the same

[2] http://nlp.stanford.edu/software/segmenter.shtml

[3] In our experiments this method gives similar performance compared with graph-based pipelining baselines implemented in previous work.

task, we report the following ROUGE F-measure scores: ROUGE-1 (unigrams), ROUGE-2 (bigrams), ROUGE-W (weighted longest common subsequence; weight=1.2), ROUGE-L (longest common subsequences), and ROUGE-SU4 (skip bigrams with a maximum distance of 4). Here we investigate two kinds of ROUGE metrics for Chinese: ROUGE metrics based on words (after Chinese word segmentation) and ROUGE metrics based on singleton Chinese characters. The latter metrics will not suffer from the problem of word segmentation inconsistency.

To compare our method with extractive baselines in terms of information loss and grammatical quality, we also ask three native Chinese students as annotators to carry out manual evaluation. The aspects considered during evaluation include Grammaticality (GR), Non-Redundancy (NR), Referential Clarity (RC), Topical Focus (TF) and Structural Coherence (SC). Each aspect is rated with scores from 1 (poor) to 5 (good) [4]. This evaluation is performed on the same random sample of 10 document sets from the DUC 2001 dataset. One group of the gold-standard summaries is left out for evaluation of human-level performance. The other two groups are shown to the annotators, giving them a sense of topics talked about in the document sets.

4.3 Results and Discussion

Table 1 and Table 2 display the ROUGE results for our proposed methods and the baseline methods, including both word-based and character-based evaluation. We also conduct pairwise t-test and find that almost all the differences between PBCS and other systems are statistically significant with $p \ll 0.01$ [5] except for the ROUGE-W metric.

We have the same observations with previous work on the inferiority of using information from only one-side, while using Chinese-side information only is more beneficial than English-side only. The CoRank algorithm utilizes both sides of information together and achieves significantly better performance over Baseline(EN) and Baseline(CN). Our compressive system outperforms the CoRank algorithm [6] in all metrics.

Also our system overperforms the compressive pipelining system (Baseline(ENcomp)) as well. Note that the latter only considers information from the source language side. Meanwhile sentence compression may sometimes causes worse translations compared with translating the full original sentence.

For manual evaluation, the average score and standard deviation for each metric is displayed in Table 3. From the comparison between compressive summarization and the extractive version, there exist slight improvements of non-redundancy. This exactly matches what we can expect from sentence compression that keeps only important part and drop redundancy. We also observe certain amount of improvements on referential clarity. This may be a result of deletions of some phrases containing pronouns, such as *he said*. Most of such phrases are semantically unimportant and will be dropped during the process of finding the maximum density compression.

Despite not directly using syntactic information, our compressive summaries do not suffer too much loss of grammaticality. This suggest that bigrams can be treated as good indicators of local grammatical smoothness. We reckon that sentences describing the same events may partially share descriptive bigram patterns, thus sentences selected by the algorithm will consist of mostly important patterns that appear repeatedly in the original document cluster. Only those words that are neither semantically important nor syntactically pivotal will be deleted.

Figure 1 lists the summaries for the first document set D04 in the DUC 2001 dataset produced by the proposed compressive system. The Chinese side sentences have been split with spaces according to phrase alignment results. Phrases that have been compressed are grayed out. We also include original English sentences for reference, with deletions according to word alignments from the Chinese sentences. We can observe that our compressive system tries to compress sentences by removing relatively unimportant phrases. The effect of translation errors (e.g. the word *watch* in *on storm watch* has been incorrectly translated in the example) can also be reduced since those incorrectly translated words will be dropped for having low information gains. In some cases the gram-

[4] Fractional numbers are allowed for cases where the annotators feel uncertain about.

[5] The significance level holds after Bonferroni adjustment, for the purpose of multiple testing.

[6] There exists ignorable difference between the results of our reimplemented version of CoRank and those reported by Wan (2011). We believe that this comes from different machine translation results output by Google Translate.

Sentence Budgeting	ROUGE-1	ROUGE-2	ROUGE-W	ROUGE-L	ROUGE-SU4
Baseline(EN)	0.23655	0.03550	0.05324	0.12559	0.06410
Baseline(CN)	0.23454	0.03858	0.05753	0.13120	0.06962
PBES	0.25313	0.04073	0.06103	0.13583	0.06970
CoRank (reported)	N/A	0.04282	0.06158	0.14521	0.07805
CoRank (reimplemented)	0.24257	0.04115	0.06076	0.13717	0.07453
Baseline(ENcomp)	0.24879	0.04441	0.05865	0.13233	0.07543
PBCS	**0.26872**	**0.04815**	**0.06425**	**0.14607**	**0.08065**

Character Budgeting	ROUGE-1	ROUGE-2	ROUGE-W	ROUGE-L	ROUGE-SU4
Baseline(EN)	0.21460	0.03494	0.05150	0.12343	0.06278
Baseline(CN)	0.21589	0.03732	0.05420	0.12867	0.06405
PBES	0.22825	0.04037	0.05527	0.12856	0.06894
CoRank (reimplemented)	0.22593	0.04069	0.05887	0.12818	0.07241
Baseline(ENcomp)	0.23663	0.04245	0.06134	0.13070	0.07365
PBCS	**0.24917**	**0.04632**	**0.06252**	**0.13591**	**0.07953**

Table 1: Results of word-based ROUGE evaluation

Sentence Budgeting	ROUGE-1	ROUGE-2	ROUGE-W	ROUGE-L	ROUGE-SU4
Baseline(EN)	0.34842	0.11823	0.05505	0.15665	0.12320
Baseline(CN)	0.34901	0.12015	0.05664	0.15942	0.12625
PBES	0.36618	0.12281	0.05913	0.16018	0.11317
CoRank (reimplemented)	0.37601	0.12570	0.06088	0.17350	0.13352
Baseline(ENcomp)	0.36982	0.13001	0.06906	0.16233	0.13543
PBCS	**0.37890**	**0.13549**	**0.07102**	**0.17632**	**0.14098**

Character Budgeting	ROUGE-1	ROUGE-2	ROUGE-W	ROUGE-L	ROUGE-SU4
Baseline(EN)	0.33602	0.10546	0.05263	0.15437	0.12161
Baseline(CN)	0.34075	0.12012	0.05678	0.15736	0.11981
PBES	0.35483	0.11902	0.05642	0.15899	0.11205
CoRank (reimplemented)	0.36147	0.12305	0.05847	**0.16962**	0.13364
Baseline(ENcomp)	0.36654	0.12960	0.06503	0.15987	0.13421
PBCS	**0.37842**	**0.13441**	**0.07005**	**0.16928**	**0.13985**

Table 2: Results of character-based ROUGE evaluation

System	GR	NR	RC	TF	SC
CoRank	3.00±0.75	3.35±0.57	3.55±0.82	3.90±0.79	3.55±0.74
PBES	2.90±0.89	3.25±0.70	3.50±0.87	3.96±0.80	3.45±0.50
PBCS	2.90±0.83	3.60±0.49	3.75±0.82	3.93±0.68	3.40±0.58
Human	4.60±0.49	4.15±0.73	4.35±0.73	4.93±0.25	3.90±0.94

Table 3: Manual evaluation results

matical fluency can even be improved from sentence compression, as redundant parentheses may sometimes be removed. We leave the output summaries from all systems for the same document set to supplementary materials.

In our experiments, we also study the influence of relevant parameter settings. Figure 2a depicts the variation of ROUGE-2 F-measure when changing the damping factor d from different values in $\{1, 2^{-1}, 3^{-1}, 4^{-1}, 5^{-1}\}$, while $\eta = -0.5$ being fixed. We can see that under proper range the value of d does not effect the result for too much. No damping or too much damping will severely decrease the performance. Figure 2b shows the performance change under different settings of the distortion parameter η taking values from $\{0, -0.2, -0.5, -1, -3\}$, while fixing $d = 0.5$. The results suggest that, for our purposes of summarization, the difference of considering distortion penalty or not is obvious. At certain level, the effect brought by different values distortion parameter becomes stable.

We also empirically study the effect of approximation. The compressive summarization framework proposed in this paper can be trivially cast into an integer linear program (ILP), with the number of variables being too large to make the problem tractable [7]. In this experiment, we use

[7] By casting decisions on whether to select a certain phrase or bigram as binary variables, with additional linear constraints on phrase/bigram selection consistency, we get an ILP with essentially the same objective function and a linear budget constraint. This is conceptually equivalent to solving

凯特 女士 硬朗 ， 紧急服务 在佛罗里达州 的 戴德 县， 承担了 风暴 的冲击 主任 估计， 安德鲁 已经 造成 150亿 美元 到 200亿 美元 的损害 （ 75亿 英镑 ， 100亿 英镑 ） 。
Ms Kate Hale, director of emergency services in Florida's Dade County, which bore the brunt of the storm, estimated that Andrew had already caused Dollars 15bn to Dollars 20bn (Pounds 7.5bn-Pounds 10bn) of damage.

雨果飓风 ， 袭击 东海岸 在 1989年9月 ， 花费了 保险业 约 42亿 美元 。
Hurricane Hugo, which hit the east coast in September 1989, cost the insurance industry about Dollars 4.2bn.

美国城市 沿 墨西哥湾的 阿拉巴马州 到得克萨斯州 东部 是 在 风暴 手表 昨晚 安德鲁 飓风 向西 横跨 佛罗里达州南部 席卷 后 ，造成 至少 八人死亡 和严重的 财产损失 。
US CITIES along the Gulf of Mexico from Alabama to eastern Texas were on storm watch last night as Hurricane Andrew headed west after sweeping across southern Florida, causing at least eight deaths and severe property damage.

过去的 严重 飓风 美国 ，雨果 ， 袭击 南卡罗来纳州 于1989年 ， 耗资 从 保险 损失 行业 42亿 美元 ，但 造成的 总伤害 的 估计 60亿 美元 和 100亿 美元 之间 不等 。
The last serious US hurricane, Hugo, which struck South Carolina in 1989, cost the industry Dollars 4.2bn from insured losses, though estimates of the total damage caused ranged between Dollars 6bn and Dollars 10bn.

最初的 报道称， 至少有一人 已经 死亡 ， 75 人受伤 ，数千 取得 沿着 路易斯安那州海岸 无家可归 ， 14 证实 在佛罗里达州和 死亡 三 巴哈马群岛 后 。
Initial reports said at least one person had died, 75 been injured and thousands made homeless along the Louisiana coast, after 14 confirmed deaths in Florida and three in the Bahamas.

Figure 1: Example compressive summary

lp_solve package [8] as the ILP solver to obtain an exact solution on the first document cluster (D04) in DUC 2001 dataset. In Figure 2c, we depict the objective value achieved by ILP as exact solution, comparing with results from sentences which are gradually selected and compressed by our greedy algorithm. We can see that the approximation is close.

5 Related Work

The task focused in this paper is cross-language document summarization. Several pilot studies have investigated this task. Before Wan (2011)'s work that explicitly utilizes bilingual information in a graph-based framework, earlier methods often use information only from one language (de Chalendar et al., 2005; Pingali et al., 2007; Orasan and Chiorean, 2008; Litvak et al., 2010).

This work is closely related to greedy algorithms for budgeted submodular maximization. Many studies have formalized text summarization tasks as submodular maximization problems (Lin and Bilmes, 2010; Lin and Bilmes, 2011; Morita et al., 2013). A more recent work (Dasgupta et al., 2013) discussed the problem of maximizing a function with a submodular part and a non-submodular dispersion term, which may appear to be closer to our scoring functions.

In recent years, some research has made progress beyond extractive summarization, especially in the context of compressive summarization. Zajic et al. (2006) tries a pipeline strategy with heuristics to generate multiple candidate compressions and extract from this compressed sentences. Berg-Kirkpatrick et al. (2011) create linear models of weights learned by structural SVMs for different components and tried to jointly formulate sentence selection and syntax tree trimming in integer linear programs. Woodsend and Lapata (2012) propose quasi tree substitution grammars for multiple rewriting operations. All these methods involve integer linear programming solvers to generate compressed summaries, which is time-consuming for multi-document summarization tasks. Almeida and Martins (2013) form the compressive summarization problem in a more efficient dual decomposition framework. Models for sentence compression and extractive summarization are trained by multi-task learning techniques. Wang et al. (2013) explore different types of compression on constituent parse trees for query-focused summarization. Li et al. (2013) propose a guided sentence compression model with ILP-based summary sentence selection. Their following work (Li et al., 2014) incorporate various constraints on constituent parse trees to improve the linguistic quality of the compressed sentences. In these studies, the best-performing systems require supervised learning for different subtasks. More recent work tries to formulate document summarization tasks as optimization problems and use their solutions to guide sentence compression(Li et al., 2015; Yao et al., 2015). Bing et al. (2015) employ integer linear programming for conducting phrase selection and merging simultaneously to form compressed sentences after phrase extraction.

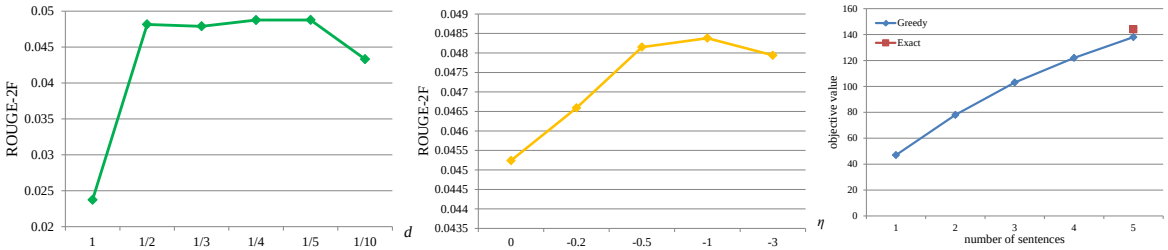
6 Conclusion and Future Work

In this paper we propose a phrase-based framework for the task of cross-language document summarization. The proposed scoring scheme can be naturally operated on compressive summarization. We use efficient greedy procedure to approximately optimize the scoring function. Experimental results show improvements of our compressive solution over state-of-the-art systems. Even though we do not explicitly use any syntactic information, the generated summaries of our system do not lose much grammaticality and fluency.

The scoring function in our framework is in-

the original maximization problem with pruned brute-force enumeration and therefore exactly optimal but too costly.

[8] http://lpsolve.sourceforge.net/



(a) Effect of damping factor d (b) Effect of distortion parameter η (c) Comparison with exact solution

Figure 2: Experimental analysis

spired by earlier phrase-based machine translation models. Our next step is to try more fine-grained scoring schemes using similar techniques from modern approaches of statistical machine translation. To further improve grammaticality of generated summaries, we may try to sacrifice the time efficiency for a little bit and use syntactic information provided by syntactic parsers.

Our framework currently uses only the single best translation. It will be more powerful to integrate machine translation and summarization, utilizing multiple possible translations.

Currently many successful statistical machine translation systems are phrase-based with alignment information provided and we utilize this fact in this work. It is interesting to explore how will the performance be affected if we are only provided with parallel sentences and then alignments can only be derived using an independent aligner.

Acknowledgments

We thank all the anonymous reviewers for helpful comments and suggestions. This work was supported by National Hi-Tech Research and Development Program (863 Program) of China (2015AA015403, 2014AA015102) and National Natural Science Foundation of China (61170166, 61331011). The contact author of this paper, according to the meaning given to this role by Peking University, is Xiaojun Wan.

References

Miguel Almeida and Andre Martins. 2013. Fast and robust compressive summarization with dual decomposition and multi-task learning. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 196–206, Sofia, Bulgaria, August. Association for Computational Linguistics.

Taylor Berg-Kirkpatrick, Dan Gillick, and Dan Klein. 2011. Jointly learning to extract and compress. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 481–490, Portland, Oregon, USA, June. Association for Computational Linguistics.

Lidong Bing, Piji Li, Yi Liao, Wai Lam, Weiwei Guo, and Rebecca Passonneau. 2015. Abstractive multi-document summarization via phrase selection and merging. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1587–1597, Beijing, China, July. Association for Computational Linguistics.

James Clarke and Mirella Lapata. 2008. Global inference for sentence compression: An integer linear programming approach. *Journal of Artificial Intelligence Research*, 31:273–381.

Anirban Dasgupta, Ravi Kumar, and Sujith Ravi. 2013. Summarization through submodularity and dispersion. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1014–1022, Sofia, Bulgaria, August. Association for Computational Linguistics.

Gaël de Chalendar, Romaric Besançon, Olivier Ferret, Gregory Grefenstette, and Olivier Mesnard. 2005. Crosslingual summarization with thematic extraction, syntactic sentence simplification, and bilingual generation. In *Workshop on Crossing Barriers in Text Summarization Research, 5th International Conference on Recent Advances in Natural Language Processing (RANLP2005)*.

Philipp Koehn, Franz Josef Och, and Marcu Daniel. 2003. Statistical phrase-based translation. In *Human Language Technologies: The 2003 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 48–54, Edmonton, May-June. Association for Computational Linguistics.

Philipp Koehn. 2009. *Statistical Machine Translation*. Cambridge University Press.

Chen Li, Fei Liu, Fuliang Weng, and Yang Liu. 2013. Document summarization via guided sentence compression. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 490–500, Seattle, Washington, USA, October. Association for Computational Linguistics.

Chen Li, Yang Liu, Fei Liu, Lin Zhao, and Fuliang Weng. 2014. Improving multi-documents summarization by sentence compression based on expanded constituent parse trees. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 691–701, Doha, Qatar, October. Association for Computational Linguistics.

Piji Li, Lidong Bing, Wai Lam, Hang Li, and Yi Liao. 2015. Reader-aware multi-document summarization via sparse coding. In *IJCAI*.

Hui Lin and Jeff Bilmes. 2010. Multi-document summarization via budgeted maximization of submodular functions. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 912–920, Los Angeles, California, June. Association for Computational Linguistics.

Hui Lin and Jeff Bilmes. 2011. A class of submodular functions for document summarization. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 510–520, Portland, Oregon, USA, June. Association for Computational Linguistics.

Marina Litvak, Mark Last, and Menahem Friedman. 2010. A new approach to improving multilingual summarization using a genetic algorithm. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 927–936, Uppsala, Sweden, July. Association for Computational Linguistics.

Hajime Morita, Ryohei Sasano, Hiroya Takamura, and Manabu Okumura. 2013. Subtree extractive summarization via submodular maximization. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1023–1032, Sofia, Bulgaria, August. Association for Computational Linguistics.

Constantin Orasan and Oana Andreea Chiorean. 2008. Evaluation of a cross-lingual romanian-english multi-document summariser. In *LREC*.

Prasad Pingali, Jagadeesh Jagarlamudi, and Vasudeva Varma. 2007. Experiments in cross language query focused multi-document summarization. In *Workshop on Cross Lingual Information Access Addressing the Information Need of Multilingual Societies in IJCAI2007*. Citeseer.

Xiaojun Wan. 2011. Using bilingual information for cross-language document summarization. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1546–1555, Portland, Oregon, USA, June. Association for Computational Linguistics.

Lu Wang, Hema Raghavan, Vittorio Castelli, Radu Florian, and Claire Cardie. 2013. A sentence compression based framework to query-focused multi-document summarization. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1384–1394, Sofia, Bulgaria, August. Association for Computational Linguistics.

Kristian Woodsend and Mirella Lapata. 2012. Multiple aspect summarization using integer linear programming. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 233–243. Association for Computational Linguistics.

Jin-ge Yao, Xiaojun Wan, and Jianguo Xiao. 2015. Compressive document summarization via sparse optimization. In *IJCAI*.

David M Zajic, Bonnie Dorr, Jimmy Lin, and Richard Schwartz. 2006. Sentence compression as a component of a multi-document summarization system. In *Proceedings of the 2006 Document Understanding Workshop, New York*.

Re-evaluating Automatic Summarization with BLEU and 192 Shades of ROUGE

Yvette Graham
ADAPT Centre
School of Computer Science and Statistics
Trinity College Dublin
graham.yvette@gmail.com

Abstract

We provide an analysis of current evaluation methodologies applied to summarization metrics and identify the following areas of concern: (1) movement away from evaluation by correlation with human assessment; (2) omission of important components of human assessment from evaluations, in addition to large numbers of metric variants; (3) absence of methods of significance testing improvements over a baseline. We outline an evaluation methodology that overcomes all such challenges, providing the first method of significance testing suitable for evaluation of summarization metrics. Our evaluation reveals for the first time which metric variants significantly outperform others, optimal metric variants distinct from current recommended best variants, as well as machine translation metric BLEU to have performance on-par with ROUGE for the purpose of evaluation of summarization systems. We subsequently replicate a recent large-scale evaluation that relied on, what we now know to be, suboptimal ROUGE variants revealing distinct conclusions about the relative performance of state-of-the-art summarization systems.

1 Introduction

Automatic metrics of summarization evaluation have their origins in machine translation (MT), with ROUGE (Lin and Hovy, 2003), the first and still most widely used automatic summarization metric, comprising an adaption of the BLEU score (Papineni et al., 2002). Automatic evaluation in MT and summarization have much in common, as both involve the automatic comparison of system-generated texts with one or more human-generated reference texts, contrasting either *system-output translations* or *peer summaries* with *human reference translations* or *model summaries*, depending on the task. In both MT and summarization evaluation, any newly proposed automatic metric must be assessed by the degree to which it provides a good substitute of human assessment, and although there are obvious parallels between evaluation of *systems* in the two areas, when it comes to evaluation of *metrics*, summarization has diverged considerably from methodologies applied to evaluation of metrics in MT.

Since the inception of BLEU, evaluation of automatic metrics in MT has been by correlation with human assessment. In contrast in summarization, over the years since the introduction of ROUGE, summarization evaluation has seen a variety of different methodologies applied to evaluation of its metrics. Evaluation of summarization metrics has included, for example, the ability of a metric/significance test combination to distinguish between sets of human and system-generated summaries (Rankel et al., 2011), or by accuracy of conclusions drawn from metrics when combined with a particular significance test, Wilcoxon rank-sum (Owczarzak et al., 2012).

Besides moving away from well-established methods such as correlation with human judgment, previous summarization metric evaluations have been additionally limited by inclusion of only a small proportion of possible metrics and variants. For example, although the most commonly used metric ROUGE has a very large number of possible variants, it is common to include only a small range of those in evaluations. This has the

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 128–137,
Lisbon, Portugal, 17-21 September 2015.

obvious disadvantage that superior variants may exist but remain unidentified due to their omission.

Despite such limitations, however, subsequent evaluations of state-of-the-art summarization systems operate under the assumption that recommended ROUGE variants are optimal and rely on this assumption to draw conclusions about the relative performance of systems (Hong et al., 2014). This forces us to raise some important questions. Firstly, to what degree was the divergence away from evaluation methodologies still applied to MT metrics today well-founded? For example, were the original methodology, by correlation with human assessment, to be applied, would a distinct variant of ROUGE emerge as superior and subsequently lead to distinct system rankings? Secondly, were *all variants* of ROUGE to be included in evaluations, would a variant originally omitted from the evaluation emerge as superior and lead to further differences in summarization system rankings? Furthermore, although methods of statistical significance testing are commonly applied to evaluation of summarization *systems*, attempts to identify significant differences in performance of *metrics* are extremely rare, and when they have been applied unfortunately have not used an appropriate test.

This motivates our review of past and current methodologies applied to the evaluation of summarization metrics. Since MT evaluation in general has its own imperfections, we do not attempt to indiscriminately impose all MT evaluation methodologies on summarization, but specifically revisit evaluation methodologies applied to one particular area of summarization, evaluation of metrics. Correlations with human assessment reveal an extremely wide range in performance among variants, highlighting the importance of an optimal choice of ROUGE variant in system evaluations. Since distinct variants of ROUGE achieve significantly stronger correlation with human assessment than previous recommended best variants, we subsequently replicate a recent evaluation of state-of-the-art summarization systems revealing distinct conclusions about the relative performance of systems. In addition, we include in the evaluation of metrics, an evaluation of BLEU for the purpose of summarization evaluation, and contrary to common belief, precision-based BLEU is on-par with recall-based ROUGE for evaluation of summarization systems.

2 Related Work

When ROUGE (Lin and Hovy, 2003) was first proposed, the methodology applied to its evaluation, in one respect, was similar to that applied to metrics in MT, as ROUGE variants were evaluated by correlation with a form of human assessment. Where the evaluation methodology diverged from MT, however, was with respect to the precise representation of human assessment that was employed. In MT evaluation of metrics, although experimentation has taken place with regards to methods of elicitation of assessments from human judges (Callison-Burch et al., 2008), human assessment is always aimed to encapsulate the overall quality of translations. In contrast in summarization, metrics are evaluated by the degree to which metric scores correlate with *human coverage scores* for summaries, a recall-based formulation of the number of peer summary units that a human assessor believed had the same meaning as model summaries. Substitution of overall quality assessments with a recall-based manual metric, unfortunately has the potential to introduce bias into the evaluation of metrics in favor of recall-based formulations.

One dimension of summary quality omitted from human coverage scores is, for example, the order in which the units of a summary are arranged within the summary. Despite unit order quite likely being something of importance to a human assessor, assessment of metrics by correlation with human coverage scores does not in any respect take into account the order in which the units of a summary appear, and evaluation by human coverage scores alone means that a summary with its units scrambled or even reversed in theory receives precisely the same metric score as the original. Given current evaluation methodologies for assessment of metrics, a metric that scores two such summaries *differently* would be unfairly penalized for it. Furthermore, when the *linguistic quality* of summaries has been assessed in parallel with annotations used to compute human coverage scores, it has been shown that the two dimensions of quality do not correlate with one another (no significant correlation) (Pitler et al., 2010), providing evidence that coverage scores alone do not fully represent human judgment of the overall quality of summaries.

Subsequent summarization metric evaluations depart from correlation with human judgment fur-

ther by evaluating metrics according to the ability of a metric/significance test combination to identify a significant difference between the quality of human and system-generated summaries (Rankel et al., 2011). Unfortunately, the evaluation of metrics with respect to how well they distinguish between *high-quality human summaries* and *all system-generated summaries*, does not provide insight into the task of metrics, to score *better quality system-generated summaries* higher than *worse quality system-generated summaries*, however. This is in contrast to evaluation of MT metrics by correlation with human judgment, where metrics *only* receive credit for their ability to appropriately score system-output documents relative to other system-output documents. Since differences in quality levels between pairs of system-generated summaries are likely to be far smaller than differences in system and human-generated summaries, the methodology unfortunately sets too low a bar for summarization metrics to meet.

Furthermore, the approach to metric evaluation unfortunately does not work in the long-term, as the performance of summarization systems improves and approaches or achieves the quality of a human, a metric that accurately identifies this achievement would be unfairly penalized for it. Separate from the evaluation of metrics, Rankel et al. (2011) make the highly important recommendation of *paired* tests for identification of significant differences in performance of summarization systems. Since data used in the evaluation of summarization systems is not independent, paired tests are more appropriate and more powerful.

Owczarzak et al. (2012) diverge further from correlation with human judgment for evaluation of metrics by assessing the accuracy of metrics to identify significant differences between pairs of systems when combined with a significance test. Although the approach to evaluation of metrics provides insight into the accuracy of conclusions drawn from metric/test combinations, the evaluation is limited by inclusion of only six variants of ROUGE, fewer than 4% of possible ROUGE variants. Despite such limitations, however, subsequent evaluations relied on recommended ROUGE variants to rank state-of-the-art systems (Hong et al., 2014).

Although methods of identifying significant differences in performance are commonly applied to the evaluation of *systems* in summarization, the application of significance tests to the evaluation of summarization *metrics* is extremely rare, and when attempts have been made, unfortunately appropriate tests have not been applied. Computation of confidence intervals for individual correlation with human coverage scores, for example, unfortunately does not provide insight into whether or not a *difference* in correlation with human coverage scores is significant.

3 Summarization Metric Evaluation

When large-scale human evaluation of summarization systems takes place, human evaluation commonly takes the form of annotation of whether or not system-generated summary units express the meaning of model summary units, annotations subsequently used to compute *human coverage scores*. In addition, an evaluation of the *linguistic quality* of summaries is commonly carried out. As described in Section 2, when used for the evaluation of metrics, linguistic quality is commonly omitted, however, with metrics only assessed by the degree to which they correlate with human coverage scores. In contrast, we include all available human assessment data for evaluating metrics.

3.1 Combining Quality and Coverage

In DUC-2004 (Over et al., 2007), human annotations used to compute summary *coverage* are carried out by identification of matching peer units (PUs), the units in a peer summary that express content of the corresponding model summary. In addition, an overall coverage estimate (E) is provided by the human annotator, the proportion of the corresponding model summary or collective model units (MUs) expressed overall by a given peer summary. Human coverage scores (CS) are computed by combining Matching PUs with coverage estimates as follows:

$$\text{CS} = \frac{|Matching\ PUs| \cdot E}{|MUs|} \qquad (1)$$

In addition to annotations used to compute human coverage scores, human assessors were asked to rate the *linguistic quality* of summaries under 7 different criteria, providing ratings from A to E, with A denoting highest and E least quality rating.

Figure 1 is a scatter-plot of human coverage scores and corresponding linguistic quality scores

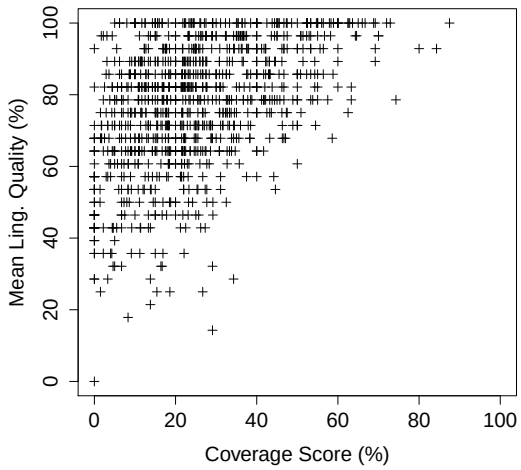


Figure 1: Scatter-plot of mean linguistic quality and coverage scores for human assessments of summaries in DUC-2004

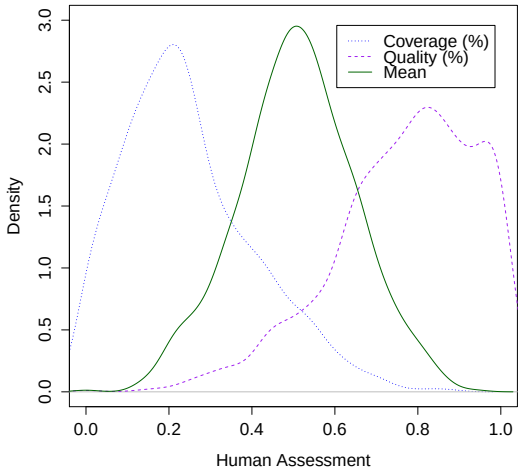


Figure 2: Combining linguistic quality and coverage scores provided by human assessors in DUC-2004

for all human-assessed summaries from DUC-2004, where, for the purpose of comparison, each of the 7 linguistic quality ratings are converted to a corresponding percentage quality (A= 100%; B= 75%; C= 50%; D= 25%; E= 0%). The location of all points almost exclusively within the upper left corner of the plot in Figure 1 indicates that the linguistic quality of almost all summaries reaches at least as high a level as its corresponding coverage score. This follows the intuition that a summary is unlikely to obtain high coverage without sufficient linguistic quality, while the same cannot be said for the converse, that a high level of linguistic quality necessarily leads to high coverage. More importantly, however, linguistic quality scores provide an additional dimension of human assessment, allowing greater discriminatory power between the quality of summaries than was possible with coverage scores alone.

Figure 2 includes linguistic quality and coverage score distributions from DUC-2004 human evaluation, where each distribution is skewed in opposing directions, in addition to the distribution of the average of the two scores for summaries.

For the purpose of metric evaluation, we combine human coverage and linguistic quality scores using the average of the two scores, and use this as a gold standard human score for evaluation of metrics:

$$\text{Human Assessment Score} = \frac{CS+MLQ}{2}$$

3.2 ROUGE

ROUGE includes a large number of distinct variants, including eight choices of n-gram counting method (ROUGE-1; 2; 3; 4; S4; SU4; W; L), binary settings such as word-stemming of summaries and an option to remove or retain stop-words. Additional configurations include the use of precision, recall or f-score to compute individual summary scores. Finally, options for computation of the overall score for a system is by computation of the mean or median of that system's summary score distribution. In total, therefore, when employing ROUGE for the evaluation of summarization systems, there are 192 (8 x 2 x 2 x 3 x 2) possible system-level variants to choose from.

The fact that final overall ROUGE scores for systems are comprised of the mean or median of ROUGE scores of individual summaries, is, again, a divergence from MT evaluation, as n-gram counts used to compute BLEU scores are computed at the document as opposed to sentence-level. However, in this respect, ROUGE has a distinct advantage over BLEU, as the fact that ROUGE comprises the mean or median score of individual summary scores makes possible the application of standard methods of significance testing differences in system-level ROUGE scores, while BLEU is restricted to the application of randomized methods (Koehn, 2004; Graham et al., 2014). For this purpose, differences in median ROUGE

scores can be tested for statistical significance using, for example, Wilcoxon signed-rank test, while paired t-test can be applied to difference of mean ROUGE scores for systems.

3.3 Metric Evaluation by Pearson's r

Moses (Koehn et al., 2007) multi-bleu[1] was used to compute BLEU (Papineni et al., 2002) scores for summaries and prepare4rouge[2] applied to summaries before running ROUGE (Lin and Hovy, 2003). Table 1 shows the Pearson correlation of each variant of ROUGE with human assessment, in addition to BLEU's correlation with the same human assessment of summaries from DUC-2004. Somewhat surprisingly, BLEU MT evaluation metric achieves strongest correlation with human assessment overall, $r = 0.797$, with performance of ROUGE variants ranging from $r = 0.786$, just below that of BLEU, to as low as $r = 0.293$. For many pairs of metrics, differences in correlation with human judgment are small, however, and prior to concluding superiority in performance of one metric over another, significance tests should be applied.

4 Metric Significance Testing

In MT, recent work has identified the suitability of Williams significance test (Williams, 1959) for evaluation of automatic MT metrics (Graham and Baldwin, 2014; Graham et al., 2015; Graham, 2015), and, for similar reasons, Williams test is suited to significance testing differences in performance of competing summarization metrics which we detail further below. Williams test has additionally been used in evaluation of systems that automatically assess spoken and written language quality (Yannakoudakis et al., 2011; Yannakoudakis and Briscoe, 2012; Evanini et al., 2013).

Evaluation of a given summarization metric, M_{new}, by Pearson correlation takes the form of quantifying the correlation, $r(M_{new}, H)$, that exists between metric scores for systems and corresponding human assessment scores, and contrasting this correlation with the correlation for some baseline metric, $r(M_{base}, H)$.

One approach to testing for significance that may seem reasonable is to apply a significance test separately to the correlation of each metric with human assessment, with the hope that the new metric will achieve a significant correlation where the baseline metric does not. The reasoning here is flawed however: the fact that one correlation is significantly higher than zero ($r(M_{new}, H)$) and that of another is not, does not necessarily mean that the *difference* between the two correlations is significant. Instead, a specific test should be applied to the difference in correlations. For this same reason, confidence intervals for individual correlations with human assessment are also not useful.

If samples that data are drawn from are independent, and differences in correlations are computed on independent data sets, the Fisher r to z transformation is applied to test for significant differences in correlations. Data used for the evaluation of summarization metrics are not independent, as evaluations comprise three sets of scores for precisely the same set of summaries (corresponding to variables X_1, X_2 and X_3 below), and subsequently three correlations: $r(M_{base}, H)$, $r(M_{new}, H)$ and $r(M_{new}, M_{base})$. If $r(M_{base}, H)$ and $r(M_{new}, H)$ are both > 0, then the third correlation, between metric scores themselves, $r(M_{base}, M_{new})$, must also be > 0. The strength of this correlation, directly between scores of pairs of summarization metrics, should be taken into account using a significance test of the difference in correlation between $r(M_{base}, H)$ and $r(M_{new}, H)$.

Williams test [3] (Williams, 1959) evaluates the significance of a difference in dependent correlations (Steiger, 1980). It is formulated as follows as a test of whether the population correlation between X_1 and X_3 equals the population correlation between X_2 and X_3:

$$t(n-3) = \frac{(r_{13} - r_{23})\sqrt{(n-1)(1+r_{12})}}{\sqrt{2K\frac{(n-1)}{(n-3)} + \frac{(r_{23}+r_{13})^2}{4}(1-r_{12})^3}},$$

where r_{ij} is the correlation between X_i and X_j, n is the size of the population, and:

$$K = 1 - {r_{12}}^2 - {r_{13}}^2 - {r_{23}}^2 + 2r_{12}r_{13}r_{23}$$

Since the power of Williams test increases when the third correlation, $r(M_{base}, M_{new})$, between metric scores is stronger, metrics should not be ranked by the number of competing metrics they

[1] https://github.com/moses-smt/mosesdecoder/commits/master/scripts/generic/multi-bleu.perl

[2] http://kavita-ganesan.com/content/prepare4rouge-script-prepare-rouge-evaluation

[3] Also known as Hotelling-Williams.

Metric	Stemming	RSW	Ave./Med	P/R/F	r	Metric	Stemming	RSW	Ave./Med	P/R/F	r	Metric	Stemming	RSW	Ave./Med	P/R/F	r
BLEU					0.797 •	R-2	Y	N	M	F	0.706	R-L	Y	Y	A	F	0.638
R-2	Y	Y	A	P	0.786 •	R-3	N	Y	M	P	0.704 •	R-1	N	N	A	F	0.637
R-3	N	N	A	F	0.785 •	R-1	N	Y	A	P	0.704 •	R-S4	Y	N	M	F	0.634
R-2	N	Y	A	P	0.783 •	R-4	N	N	M	R	0.703 •	R-4	Y	N	M	P	0.634
R-3	N	Y	A	P	0.781 •	R-L	N	Y	A	P	0.700 •	R-1	N	N	M	F	0.634
R-3	Y	N	A	F	0.779 •	R-W	Y	Y	A	P	0.700 •	R-SU4	N	Y	A	R	0.633
R-3	N	N	A	R	0.777 •	R-4	N	Y	A	R	0.700 •	R-L	Y	Y	M	P	0.633
R-4	N	N	A	F	0.771 •	R-1	Y	N	M	P	0.699 •	R-SU4	Y	Y	M	R	0.631
R-3	N	N	A	P	0.771 •	R-S4	N	Y	M	P	0.698	R-1	Y	N	A	F	0.630
R-3	Y	N	A	R	0.770 •	R-1	Y	Y	A	P	0.698 •	R-1	Y	Y	M	F	0.629
R-2	N	Y	A	F	0.769 •	R-3	N	Y	M	F	0.697 •	R-S4	Y	Y	M	R	0.626
R-4	N	N	A	R	0.768 •	R-W	N	N	A	P	0.696 •	R-S4	N	N	A	R	0.626
R-2	Y	Y	A	F	0.768 •	R-W	Y	N	A	P	0.695 •	R-SU4	Y	N	M	F	0.625
R-3	Y	N	A	P	0.767 •	R-4	N	N	M	F	0.695 •	R-S4	Y	N	A	R	0.624
R-3	N	N	M	F	0.766 •	R-S4	N	Y	M	F	0.693	R-L	N	Y	M	F	0.623
R-3	N	Y	A	F	0.764 •	R-S4	N	Y	A	F	0.691	R-SU4	Y	Y	A	R	0.622
R-3	Y	Y	A	P	0.764 •	R-SU4	N	Y	M	P	0.690	R-1	Y	N	M	F	0.617
R-4	Y	N	A	F	0.763 •	R-1	N	N	M	P	0.690 •	R-1	N	Y	M	R	0.615
R-4	N	N	A	P	0.762 •	R-2	N	N	M	R	0.689	R-W	N	Y	A	R	0.613
R-4	Y	N	A	R	0.761 •	R-L	Y	Y	A	P	0.688 •	R-S4	N	N	M	R	0.611
R-3	N	N	M	P	0.760 •	R-3	N	Y	M	R	0.687 •	R-L	N	Y	M	R	0.609
R-4	Y	Y	A	P	0.759 •	R-S4	N	N	M	P	0.687	R-1	N	Y	A	R	0.604
R-2	Y	N	A	P	0.759 •	R-S4	Y	N	A	F	0.687	R-L	N	Y	A	R	0.601
R-4	N	Y	A	P	0.758 •	R-S4	N	N	A	F	0.687	R-W	N	N	M	F	0.600
R-2	N	N	A	P	0.757 •	R-4	N	N	M	P	0.687 •	R-L	N	N	M	F	0.599
R-3	N	N	M	R	0.753 •	R-L	N	N	A	P	0.686 •	R-W	Y	Y	A	R	0.598
R-4	Y	N	A	P	0.752 •	R-SU4	N	N	M	P	0.686	R-W	N	Y	M	R	0.597
R-3	Y	Y	A	F	0.748 •	R-L	Y	N	A	P	0.683 •	R-1	Y	Y	A	R	0.595
R-2	N	N	A	F	0.747 •	R-W	N	N	M	P	0.682 •	R-1	Y	Y	M	R	0.591
R-2	Y	N	A	F	0.747 •	R-W	Y	N	M	P	0.680 •	R-L	N	N	A	F	0.586
R-3	N	Y	A	R	0.746 •	R-SU4	Y	N	M	P	0.678	R-W	Y	Y	M	F	0.586
R-3	Y	N	M	P	0.744 •	R-SU4	N	Y	A	F	0.678	R-W	Y	N	M	F	0.585
R-2	N	Y	M	P	0.743 •	R-S4	Y	Y	A	F	0.676	R-L	Y	Y	A	R	0.583
R-3	Y	N	M	F	0.743 •	R-SU4	N	Y	M	F	0.676	R-L	Y	Y	M	F	0.582
R-2	N	Y	A	R	0.742 •	R-SU4	N	N	A	F	0.673	R-L	Y	Y	M	R	0.579
R-2	Y	Y	M	P	0.741 •	R-1	N	Y	M	P	0.673	R-L	Y	N	A	F	0.579
R-2	N	Y	M	F	0.740 •	**R-2**	**Y**	**N**	**M**	**R**	0.672	R-W	N	N	A	F	0.579
R-3	Y	N	M	R	0.739 •	R-SU4	Y	N	A	F	0.671	R-SU4	N	N	M	R	0.576
R-2	Y	Y	A	R	0.737 •	R-S4	N	Y	M	R	0.670	R-W	Y	N	A	F	0.576
R-2	Y	Y	M	F	0.735 •	R-S4	Y	N	M	P	0.670	R-SU4	N	N	A	R	0.574
R-2	N	N	M	P	0.734 •	R-SU4	Y	Y	A	F	0.668	R-SU4	Y	N	A	R	0.571
R-3	Y	Y	M	P	0.733 •	R-S4	N	N	M	F	0.666	R-L	Y	N	M	F	0.569
R-3	Y	Y	A	R	0.730	R-W	N	Y	M	P	0.664	R-W	Y	Y	M	R	0.567
R-4	Y	Y	A	F	0.729 •	R-S4	Y	Y	M	P	0.664	R-S4	Y	N	M	R	0.566
R-3	Y	Y	M	F	0.726 •	R-SU4	Y	Y	M	P	0.663	R-SU4	Y	N	M	R	0.525
R-S4	Y	N	A	P	0.725 •	R-L	N	N	M	P	0.661 •	R-1	N	N	M	R	0.488
R-SU4	N	N	A	P	0.724 •	R-SU4	N	N	M	F	0.658	**R-1**	**Y**	**N**	**M**	**R**	0.477
R-2	Y	N	M	P	0.724	R-1	N	Y	A	F	0.656	R-W	Y	N	M	R	0.477
R-S4	N	Y	A	P	0.724	R-W	Y	Y	M	P	0.656	R-1	N	N	A	R	0.470
R-SU4	Y	N	A	P	0.723 •	R-S4	N	Y	A	R	0.656	R-W	N	N	M	R	0.470
R-S4	N	N	A	P	0.723 •	R-L	Y	N	M	P	0.656 •	R-L	N	N	M	R	0.470
R-2	N	Y	M	R	0.722 •	R-W	N	Y	A	F	0.655	R-1	Y	N	A	R	0.459
R-4	N	Y	A	F	0.721 •	R-1	N	Y	M	F	0.653	R-W	N	N	A	R	0.456
R-1	N	N	A	P	0.720 •	R-L	N	Y	A	F	0.652	R-W	Y	N	A	R	0.452
R-2	N	N	M	F	0.719 •	R-1	Y	Y	M	P	0.651	R-L	Y	N	M	R	0.423
R-SU4	N	Y	A	P	0.719	R-S4	Y	Y	M	F	0.649	R-L	N	N	A	R	0.416
R-1	Y	N	A	P	0.714 •	R-1	Y	Y	A	F	0.649	R-L	Y	N	A	R	0.406
R-2	Y	Y	M	R	0.714 •	R-SU4	Y	Y	M	F	0.649	R-4	Y	Y	M	P	0.307
R-3	Y	Y	M	R	0.713 •	R-SU4	N	Y	M	R	0.646	R-4	Y	Y	M	F	0.302
R-4	Y	Y	A	R	0.712 •	R-L	N	Y	M	P	0.645	R-4	N	Y	M	P	0.301
R-S4	Y	Y	A	P	0.711	R-W	N	Y	M	F	0.642	R-4	Y	Y	M	R	0.297
R-SU4	Y	Y	A	P	0.710	R-W	Y	Y	A	F	0.642	R-4	N	Y	M	F	0.296
R-2	N	N	A	R	0.710 •	**R-4**	**Y**	**N**	**M**	**R**	0.641	R-4	N	Y	M	R	0.293
R-W	N	Y	A	P	0.709 •	R-S4	Y	Y	A	R	0.641						
R-2	Y	N	A	R	0.707 •	R-4	Y	N	M	F	0.639						

Table 1: Pearson correlation (r) of BLEU and 192 variants of ROUGE (R-*) with human assessment in DUC-2004, with (Y) and without (N) stemming, with (Y) and without (N) removal of stop words (RSW), aggregated at the summary level using precision (P), recall (R) or f-score (F), aggregated at the system level by average (A) or median (M) summary score, correlations marked with • signify a metric/variant whose correlation with human assessment is not significantly weaker than that of any other metric/variant (an optimal variant) according to pairwise Williams significance tests, variants employed in Hong et al. (2014) are in bold.

outperform, as a metric that happens to correlate strongly with a higher number of competing metrics in a given competition would be at an unfair advantage. This increased power also means, somewhat counter-intuitively, it can happen for a pair of competing metrics for which the correlation between metric scores is strong, that a small difference in competing correlations with human assessment is significant, while, for a different pair of metrics with a larger difference in correlation, the difference is not significant, because $r(M_{base}, M_{new})$ is weak. For example, in Table 1 the difference in correlation with human assessment of BLEU and that of median ROUGE-L precision with stemming and stop-words retained, 0.141 (0.797 − 0.656), is not significant, while the smaller difference in correlation with human assessment between correlations of BLEU and average ROUGE-3 recall with stemming and stop-words removed, 0.067 (0.797 − 0.73) is significant, since scores of the latter pair of metrics correlate with one another with more strength.

As part of this research, we have made available an open-source implementation of statistical tests for evaluation of summarization metrics, at `https://github.com/ygraham/nlp-williams`.

4.1 Significance Test Results

In Table 1, • identifies variants of ROUGE not significantly outperformed by any other variant. Figure 3 shows pairwise Williams significance test outcomes for BLEU, the top ten ROUGE variants, as well as current recommended ROUGE variants (Owczarzak et al. (2012)) used to compare systems in Hong et al. (2014). Current recommended best variants of ROUGE are shown to be *significantly outperformed* by several other ROUGE variants.

Although BLEU achieves strongest correlation with human assessment overall, Figure 3 reveals the difference between BLEU's correlation with human assessment and that of the best-performing ROUGE variant as *not statistically significant*, and since ROUGE holds the distinct advantage over BLEU of facilitating standard methods of significance testing differences in scores for systems, for this reason alone we recommend the use of the best-performing ROUGE variant over BLEU, average ROUGE-2 precision with stemming and stop-words removed.

Table 2 shows proportions of optimal ROUGE

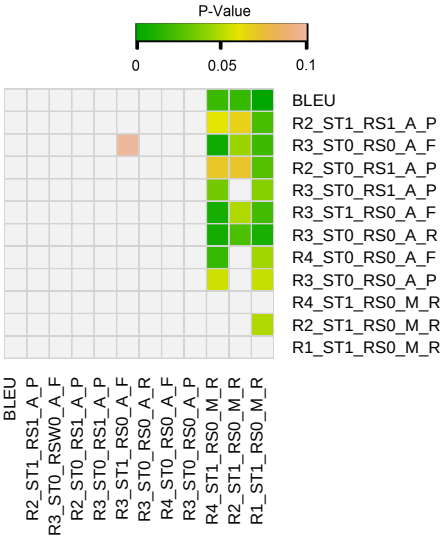


Figure 3: Pairwise significance test outcomes for BLEU, best-performing ROUGE (rows 2-9), and ROUGE applied in Hong et al. (2014) (bottom 3 rows), with (ST1) and without (ST0) stemming, with (RS1) and without (RS0) removal of stop words, for average (A) or median (M) ROUGE precision (P), recall (R) or f-score (F), colored cells denote significant win for row i metric over column j metric with Williams test.

variants that can be attributed to each of ROUGE's configuration options. Contrary to prior belief, the vast majority of optimal ROUGE variants are precision-based, showing that the assumption that recall-based metrics are superior for evaluation of summarization systems to be inaccurate, and a likely presence of bias in favor of recall-based metrics in evaluations by correlation with human coverage scores alone. Furthermore, since there exists a vast number of possible formulations that could potentially be applied to evaluation of summaries that are neither purely precision nor recall-based, evaluation methodologies should avoid reliance on assumptions that either precision or recall is superior and instead base conclusions on empirical evidence where possible.

5 Summarization System Evaluation

Since we have established that the variants of ROUGE used to rank state-of-the-art and baseline summarization systems in Hong et al. (2014) have significantly weaker correlations with human assessment than several other ROUGE variants, this motivates our replication of the evaluation. We evaluate systems using the variant of ROUGE that

N-gram Count	
R-3	28.7
R-2	25.0
R-4	18.8
R-1	7.5
R-L	7.5
R-W	7.5
R-S4	2.5
R-SU4	2.5

Stemming	
Not Stemmed	53.8
Stemmed	46.2

Stop-words	
Not Rem.	56.2
Removed	43.8

Summary-level Agg.	
Prec.	52.5
F-score	25.0
Recall	22.5

System-level Agg.	
Average	63.7
Median	36.3

Table 2: Proportions of optimal ROUGE variants attributed to each ROUGE configuration option (%).

System	ROUGE Best	ROUGE Original
DPP	8.498	9.62
ICSISumm	8.317	9.78
RegSum	8.187	9.75
Submodular	8.047	9.35
CLASSY11	7.717	9.20
CLASSY04	7.690	8.96
OCCAMS_V	7.643	9.76
GreedyKL	6.918	8.53
FreqSum	6.838	8.11
TsSum	6.671	8.15
Centroid	6.660	7.97
LexRank	6.655	7.47

Table 3: Summarization systems originally included in Hong et al. (2014) evaluated with the best-performing ROUGE variant (Best): average ROUGE-2 precision with stemming and stop words removed; and evaluated with original suboptimal variant (median ROUGE-2 recall with stemming and without removal of stop-words)

achieves strongest correlation with human assessment, average ROUGE-2 precision with stemming and stop-words removed.

Table 3 shows ROUGE scores for summarization systems originally presented in Hong et al. (2014). System rankings diverge considerably from those of the original evaluation. Notably, the system now taking first place had originally ranked in fourth position.

Since the best variant of ROUGE is based on average ROUGE scores as opposed to median ROUGE scores, a difference of means significance test is appropriate provided the normality assumption of score distributions for systems is not violated. In addition, since data used to evaluate systems are not independent, *paired* tests are also appropriate (Rankel et al., 2011). ROUGE score distributions for systems were tested for normality using the Shapiro-Wilk test (Royston, 1982) where score distributions for none of the included systems were shown to be significantly non-normal.

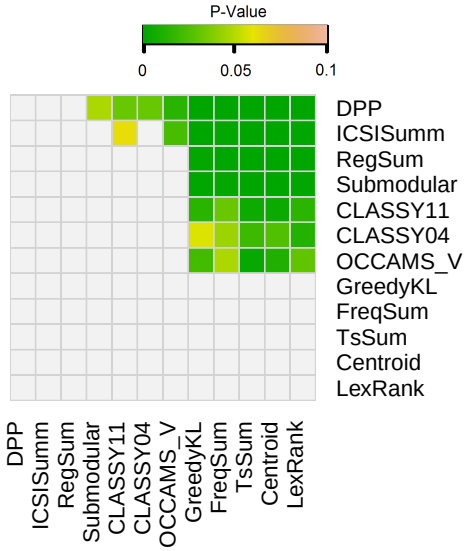


Figure 4: Summarization system pairwise significance test outcomes (paired t-test) for state-of-the-art (top 7 rows) and baseline systems (bottom 5 rows) of Hong et al. (2014) evaluated with best-performing ROUGE variant: average ROUGE-2 precision with stemming and stop words removed, colored cells denote a significant greater mean score for row i system over column j system according to paired t-test.

Figure 4 shows outcomes of paired t-tests for summary score distributions of each pair of systems, revealing three summarization systems not significantly outperformed by any other as DPP, ICSISUMM and REGSUM. In addition, as expected, all state-of-the-art systems significantly outperform all baseline systems.

6 Human Assessment Combinations

In order to evaluate metrics by correlation with human assessment, it is necessary to obtain a single human evaluation score per system. For example, in the evaluation of metrics in Section 3, we combined linguistic quality and coverage into a single score using the mean of the two scores. Other combinations are of course possible, but without any additional human evaluation data, it is challenging to identify the combination that best rep-

Metric	Stem.	RSW	Ave/Med	P/R/F	Mean	Geometric Mean	Harmonic Mean	Coverage Only	Ling. Qual. Only
BLEU					**0.797•**	**0.901•**	**0.936•**	**0.944•**	0.642•
ROUGE-2	Y	Y	A	P	0.786•	0.870•	0.887•	0.878	0.660•
ROUGE-3	N	N	A	F	0.785•	0.869•	0.893	0.894	0.650•
ROUGE-2	N	Y	A	P	0.783•	0.868•	0.885•	0.876	0.658•
ROUGE-3	N	Y	A	P	0.781•	0.836•	0.840	0.826	**0.682•**
ROUGE-3	Y	N	A	F	0.779•	0.866•	0.891	0.893	0.643•
ROUGE-3	N	N	A	R	0.777•	0.871•	0.901	0.907	0.632•
ROUGE-4	N	N	A	F	0.771•	0.843•	0.863	0.866	0.645•
ROUGE-3	N	N	A	P	0.771•	0.837•	0.849	0.843	0.658•
ROUGE-3	Y	N	A	R	0.770•	0.867•	0.899	0.905	0.624•
ROUGE-2	N	Y	A	F	0.769•	0.877•	0.909•	0.910•	0.619•
ROUGE-2	Y	Y	A	F	0.768•	0.875•	0.908•	0.908•	0.618•
ROUGE-3	Y	N	A	P	0.767•	0.835•	0.849	0.843	0.652•
ROUGE-3	Y	Y	A	P	0.764•	0.825•	0.832	0.821	0.660•
ROUGE-4	N	N	A	P	0.762•	0.815•	0.824	0.819	0.657•
ROUGE-4	Y	Y	A	P	0.759•	0.794	0.790	0.774	0.678•
ROUGE-4	N	Y	A	P	0.758•	0.793	0.789	0.772	0.678•
ROUGE-4	Y	N	A	P	0.752•	0.809	0.819	0.815	0.646•
ROUGE-2	N	N	A	F	0.747•	0.867•	0.907•	0.910•	0.587•
ROUGE-2	Y	N	A	F	0.747•	0.868•	0.908•	0.912•	0.586•
ROUGE-2	N	Y	A	R	0.742•	0.862•	0.904•	0.912•	0.578•
ROUGE-2	N	Y	M	F	0.740•	0.855•	0.894•	0.898•	0.584•
ROUGE-2	Y	Y	A	R	0.737•	0.858•	0.900•	0.908•	0.575•
ROUGE-2	N	Y	M	R	0.722•	0.848•	0.895•	0.905•	0.553•
ROUGE-2	N	N	M	R	0.689	0.828	0.884•	0.901•	0.508

Table 4: Correlation of top-ten metric variants for each alternate combination of linguistic quality and coverage, • denotes a metric not significantly outperformed by any other under that particular human evaluation combination, highest correlations highlighted in bold font.

resents an overall human assessment for a given summary. One possibility would be to search for optimal weights for combining quality and coverage, but there is a risk with this approach that we will not find the most representative combination but simply the combination that best describes the metrics.

An additional variation of human assessment scores is by combining coverage and quality with a variant of the arithmetic mean, such as the harmonic or geometric mean. Table 4 shows correlations of BLEU and the top ten performing variants of ROUGE when evaluated against the arithmetic (mean), harmonic and geometric mean of quality and coverage scores for summaries. In addition, Table 4 includes correlations of metric scores with coverage alone, as well as linguistic quality scores alone to provide additional insight, although linguistic quality scores alone do not provide a sufficient evaluation of metrics – since it is possible to generate summaries with perfect linguistic quality without inclusion of any relevant content whatsoever.

BLEU MT metric achieves highest correlation across all human evaluation combinations and highest again when evaluated against human coverage scores alone, and BLEU's brevity penalty, that like recall penalizes a system for too short output, is a probable cause of the metric overcoming the recall-based bias of an evaluation based on coverage scores alone. In addition, our recommended variant, ave. ROUGE-2 prec. with stemming and stop words removed is not significantly outperformed by BLEU or any other variant of ROUGE for any of the three combined mean human assessment scores.

7 Conclusions

An analysis of evaluation of summarization metrics was provided with an evaluation of BLEU and 192 variants of ROUGE. Detail of the first suitable summarization metric significance test, Williams test, was provided. Results reveal superior variants of metrics distinct from previously best recommendations. Replication of a recent evaluation of state-of-the-art summarization systems also revealed contrasting conclusions about the relative performance of systems. In addition, BLEU achieves strongest correlation with human assessment overall, but does not significantly outperform the best-performing ROUGE variant.

Acknowledgements

We wish to thank the anonymous reviewers. This research is supported by Science Foundation Ireland through the CNGL Programme (Grant 12/CE/I2267) in the ADAPT Centre (www.adaptcentre.ie) at Trinity College Dublin.

References

Chris Callison-Burch, Cameron Fordyce, Philipp Koehn, Christof Monz, and Josh Schroeder. 2008. Further meta-evaluation of machine translation. In *Proc. 3rd Wkshp. Statistical Machine Translation*, pages 70–106, Columbus, Ohio. Association for Computational Linguistics.

Keelan Evanini, Shasha Xie, and Klaus Zechner. 2013. Prompt-based content scoring for automated spoken language assessment. In *Proc. 2013 Conf. North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 157–162, Atlanta, GA. Association for Computational Linguistics.

Yvette Graham and Timothy Baldwin. 2014. Testing for significance of increased correlation with human judgment. In *Proc. 2014 Conf. Empirical Methods in Natural Language Processing*, pages 172–176, Doha, Qatar. Association for Computational Linguistics.

Yvette Graham, Nitika Mathur, and Timothy Baldwin. 2014. Randomized significance tests in machine translation. In *Proc. 9th Wkshp. Statistical Machine Translation*, pages 266–274, Baltimore, MD. Association for Computational Linguistics.

Yvette Graham, Nitika Mathur, and Timothy Baldwin. 2015. Accurate evaluation of segment-level machine translation metrics. In *Proc. 2015 Conf. North American Chapter of the Association for Computational Linguistics - Human Language Technologies*, pages 1183–1191, Denver, CO. Association for Computational Linguistics.

Yvette Graham. 2015. Improving evaluation of machine translation quality estimation. In *Proc. Fifty-Third Annual Meeting of the Association for Computational Linguistics*, pages 1804–1813, Beijing, China. Association for Computational Linguistics.

Kai Hong, John M Conroy, Benoit Favre, Alex Kulesza, Hui Lin, and Ani Nenkova. 2014. A repository of state of the art and competitive baseline summaries for generic news summarization. In *Proc. 9th edition of the Language Resources and Evaluation Conference*, pages 1608–1616, Reykjavik, Iceland.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proc. 45th Annual Meeting of the Association for Computational Linguistics*, pages 177–180, Prague, Czech Republic. Association for Computational Linguistics.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proc. 2004 Conf. Empirical Methods in Natural Language Processing*, pages 388–395, Barcelona, Spain. Association for Computational Linguistics.

Chin-Yew Lin and Eduard Hovy. 2003. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Proc. 2003 Conf. North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 71–78, Edmonton, Canada. Association for Computational Linguistics.

Paul Over, Hoa Dang, and Donna Harman. 2007. Duc in context. *Information Processing & Management*, 43(6):1506–1520.

Karolina Owczarzak, John M. Conroy, Hoa Trang Dang, and Ani Nenkova. 2012. An assessment of the accuracy of automatic evaluation in summarization. In *Proc. Wkshp. on Evaluation Metrics and System Comparison for Automatic Summarization*, pages 1–9, Quebec, Canada. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *Proc. 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, PA. Association for Computational Linguistics.

Emily Pitler, Annie Louis, and Ani Nenkova. 2010. Automatic evaluation of linguistic quality in multi-document summarization. In *Proc. 48th Annual Meeting of the Association for Computational Linguistics*, pages 544–554, Uppsala, Sweden. Association for Computational Linguistics.

Peter Rankel, John M. Conroy, Eric V. Slud, and Dianne P. O'Leary. 2011. Ranking human and machine summarization systems. In *Proc. 2011 Conf. Empirical Methods in Natural Language Processing*, pages 467–473, Edinburgh, Scotland. Association for Computational Linguistics.

Patrick Royston. 1982. Algorithm as 181: The W test for normality. *Applied Statistics*, 31:176–180.

James H. Steiger. 1980. Tests for comparing elements of a correlation matrix. *Psychological Bulletin*, 87(2):245–251.

Evan J. Williams. 1959. *Regression analysis*, volume 14. Wiley New York.

Helen Yannakoudakis and Ted Briscoe. 2012. Modeling coherence in ESOL learner texts. In *Proc. Seventh Wkshp. on Building Educational Applications Using NLP*, pages 33–43, Montreal, Canada. Association for Computational Linguistics.

Helen Yannakoudakis, Ted Briscoe, and Ben Medlock. 2011. A new dataset and method for automatically grading ESOL texts. In *Proc. 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 180–189, Portland, OR. Association for Computational Linguistics.

Indicative Tweet Generation: An Extractive Summarization Problem?

Priya Sidhaye
School of Computer Science
McGill University
Montreal, QC, Canada, H3A 0E9
priya.sidhaye@mail.mcgill.ca

Jackie Chi Kit Cheung
School of Computer Science
McGill University
Montreal, QC, Canada, H3A 0E9
jcheung@cs.mcgill.ca

Abstract

Social media such as Twitter have become an important method of communication, with potential opportunities for NLG to facilitate the generation of social media content. We focus on the generation of *indicative tweets* that contain a link to an external web page. While it is natural and tempting to view the linked web page as the source text from which the tweet is generated in an extractive summarization setting, it is unclear to what extent actual indicative tweets behave like extractive summaries. We collect a corpus of indicative tweets with their associated articles and investigate to what extent they can be derived from the articles using extractive methods. We also consider the impact of the formality and genre of the article. Our results demonstrate the limits of viewing indicative tweet generation as extractive summarization, and point to the need for the development of a methodology for tweet generation that is sensitive to genre-specific issues.

1 Introduction

With the rise in popularity of social media, message broadcasting sites such as Twitter and other microblogging services have become an important means of communication, with an estimated 500 million tweets being written every day[1]. In addition to individual users, various organizations and public figures such as newspapers, government officials and entertainers have established themselves on social media in order to disseminate information or promote their products.

While there has been recent progress in the development of Twitter-specific POS taggers, parsers, and other tweet understanding tools (Owoputi et al., 2013; Kong et al., 2014), there has been little work on methods for generating tweets, despite the utility this would have for users and organizations.

In this paper, we study the generation of the particular class of tweets that contain a link to an external web page that is composed primarily of text. Given the short length of a tweet, the presence of a URL in the tweet is a strong signal that the tweet is functioning to help Twitter users decide whether to read the full article. This class of tweets, which we call *indicative tweets*, represents a large subset of tweets overall, constituting more than half of the tweets in our data set. Indicative tweets would appear to be the easiest to handle using current methods in text summarization, because there is a clear source of input from which a tweet could be generated. In effect, the tweet would be acting as an indicative summary of the article it is being linked to, and it would seem that existing methods in summarization can be applied. It should be noted that a tweet being indicative does not preclude it from also providing a critical evaluation of the linked article.

There has in fact been some work along these lines, within the framework of extractive summarization. Lofi and Krestel (2012) describe a system to generate tweets from local government records through keyphrase extraction. Lloret and Palomar (2013) compares various extractive summarization algorithms applied on Twitter data to generate tweets from documents.

Lofi and Krestel do not provide a formal evaluation of their model, while Lloret and Palomar compared overlap between system-generated and user-generated tweets using ROUGE (Lin, 2004). Unfortunately, they also show that there is little correlation between ROUGE scores and the perceived quality of the tweets when rated by human users for indicativeness and interest. More scrutiny is

[1] https://about.twitter.com/company

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 138–147,
Lisbon, Portugal, 17-21 September 2015.

required to determine whether the wholesale adoption of methods and evaluation schemes from extractive summarization is justified.

Beyond issues of evaluation measures, it is also unclear whether extraction is the strategy employed by human tweeters. One of the original motivations behind extractive summarization was the observation that human summary writers tended to extract snippets of key phrases from the source text (Mani, 2001). And while it may be true that an automatic tweet generation system need not necessarily follow the same approach to writing as human tweeters, it is still necessary to know what proportion of tweets could be accounted for in an extractive summarization paradigm.

With indicative tweets, an additional issue arises in that the genre of the source text is not constrained; for example it may be a news article or an informal blog post. This may be vastly different from the desired formality of tweet itself, and thus, a genre-appropriate extract may not be available.

We begin to address the above issues through a study that examines to what extent tweet generation can be viewed as an extractive summarization problem. We extracted a dataset of indicative tweets containing a link to an external article, including the documents linked to by the tweets. We used this data and applied unigram, bigram and LCS (longest common subsequence) matching techniques inspired by ROUGE to determine what proportion of tweets can be found in the linked article. Even with the permissive unigram match measure, we find that well under half of the tweet can be found in the linked article. We also use stylistic analysis on the articles to examine the role that genre differences between the source text and the target tweet play, and find that it is easier to extract tweets from more formal articles than less formal ones.

Our results point to the need for the development of a methodology for indicative tweet generation, rather than to expropriate the extractive summarization paradigm that was developed mostly on news text. Such a methodology will ideally be sensitive to stylistic factors as well as the underlying intent of the tweet.

2 Background and Related Work

There have been studies on a number of different issues related to Twitter data, including classifying tweets and sentiment analysis of tweets. Ghosh et al. (2011) classified the retweeting activity of users based on time intervals between retweets of a single user and frequency of retweets from unique users. 'Retweet' here means the occurrence of the same URL in a different tweet. The study was able to classify the retweeting as automatic or robotic retweeting, campaigns, news, blogs and so on, based on the time-interval and user-frequency distributions. In another study, Chen et al. (2012) were able to extract sentiment expressions from a corpus of tweets including both formal words and informal slang that bear sentiment.

Other studies using Twitter data include O'Connor et al. (2010), who use topic summarization for a given search for better browsing. Chakrabarti and Punera (2011) generate an event summary by learning about the event using a Hidden Markov Model over the tweets describing it. Wang et al. (2014) generate a coherent event summary by treating summarization as an optimization problem for topic cohesion. Inouye and Kalita (2011) compare multiple summarization techniques to generate a summary of multi-post blogs on Twitter. Wei and Gao (2014) use tweets to help in generating better summaries of news articles.

As described in Section 1, we analyze tweet generation using measures inspired by extractive summarization evaluation. There has been one study comparing different text summarization techniques for tweet generation by Lloret and Palomar (2013). Summarization systems were used to generate sentences lesser than 140 characters in length by summarizing documents, which could then be taken to be tweets. The system-generated tweets were evaluated using ROUGE measures (Lin, 2004). The ROUGE-1, ROUGE-2 and ROUGE-L measures were used, and a human-written reference tweet was taken to be the gold standard.

The limits of extractive summarization have been studied by He et al. (2000). They compare user preferences for various types of summaries of an audio-visual presentation. They demonstrate that the most preferred method of summarization is highlights and notes provided by the author, rather than transcripts or slides from the presentation. Conroy et al. (2006) computed an oracle ROUGE score to investigate the same issue of the limits of extraction for news text.

These studies show that extractive summarization algorithms may not generate good quality summaries despite giving high ROUGE evaluation scores. Cheung and Penn (2013) show that for the news genre, extractive summarization systems that are optimized for centrality—that is, getting the core parts of the text into the summary—cannot perform well when compared to model summaries, since the model summaries are abstracted from the document to a large extent.

3 Data Extraction and Preprocessing

3.1 Using Twitter for Data Extraction

As mentioned earlier, there have been numerous studies that used data from the public Twitter feeds. However, none of the datasets in those studies focused on tweets and related articles linked to these tweets. The dataset of Lloret and Palomar (2013) is an exception, as it contains tweets and the news articles they link to, but it only contains 200 English tweet-article pairs. Wei and Gao (2014) also constructed a dataset that contains both tweets and articles linked through them, but this data only deals with news text, and does not contain the variety of topics we wanted in the data. We therefore chose to build our own dataset. This section describes extraction, cleaning and other preprocessing of the data.

3.2 Extracting Data

Data was extracted from Twitter using the Twitter REST API using 51 search terms, or hashtags. These hashtags were chosen from a range of topics including pop culture, international summit meetings discussing political issues, lawsuits and trials, social issues and health care issues. All these hashtags were trending (being tweeted about at a high rate) at the time of extraction of the data. To get a broader sample, the data was extracted over the course of 15 days in November, 2014, which gave us multiple news stories to choose from for the search terms. The search terms were chosen so that there would be broad representation in terms of various stylistic properties of text like formality, subjectivity, etc. For example, searches related to politics would be more formal, while those related to films would be informal, and would also have a lot more opinion pieces about them. A few examples of the search terms and their distribution in genre are shown in Table 1.

We extracted about 30,000 tweets, of which more than half, or around 16,000, contained URLs to an external news article, photo on photo sharing sites, or video.

Politics	Science & Technology
#apec2014	#rosetta
#G20	#lollipop
#oscarpistorius	#mangalayan
Events	Films and Pop culture
#haiyan	#TaylorSwift
#memorialday	#theforceawakens
#ottawashootings	#johnoliver
International	Sports
#berlinwall	#ausvssa
#ebola	#playingitmyway
#erdogan	#nycmarathon

Table 1: Examples of the hashtags used for extraction, grouped into various categories.

The data from the tweets was cleaned by removing the tweets that were not in English as well as the retweets; i.e., re-publications of a tweet by a different user.

We deduplicated the 16,000 extracted URLs into 6,003 unique addressed, then extracted and preprocessed their contents. The `newspaper` package[2] was used to extract article text and the title from the web page. Since we are interested in text articles that can serve as the source text for summarization algorithms, we needed to remove photos and video links such as those from Instagram and YouTube. To do so, we removed those links that contained fewer than a threshold of 150 words. After this preprocessing, the number of useful articles was reduced from 6003 to 3066. There were some further tweet-article pairs where the text of the tweets was identical, these were removed by further preprocessing and the number of unique tweet-article pairs came down to 2471.

The final version of the data consists of tweets along with other information about the tweet, such as links to articles, hashtags, time of publication, etc. We also retain the linked article text and preprocessed it using the CoreNLP toolkit (Manning et al., 2014). This includes the URL itself and the text extracted from the article, as well as some extracted information such as sentence boundaries, POS tags for tokens, parse trees and dependency trees. These annotations are used later during our

[2]https://pypi.python.org/pypi/newspaper

analysis in Section 4. Table 2 shows an example of an entry in the dataset.

Tweet	'#RiggsReport: #CA as the #Election-Night exception. Voters rewarded #GOP nationally, but not in the #GoldenState. http://t.co/K542wvSNVz'
Title	'The Riggs Report: California as the Election Night exception'
Text	'When the dust settled on Election Night last week...'

Table 2: Example of a tweet, title of the article and the text.

4 Analysis

We now describe the analyses we performed on the data. Our goal is to investigate what proportion of the indicative tweets that we extracted can be found in the articles that they link to, in order to determine whether indicative tweet generation can be viewed as an extractive summarization problem. Table 3 gives an example of data where the tweet that was shared about the article does not come directly from the article text, while Table 4 shows a tweet that was almost entirely extracted from the text of the article, but changed a little for the purpose of readability.

Tweet	Are #Airlines doing enough with #Ebola? http://t.co/XExWwxmjnk #travel
Title	Could shortsighted airline refund policies lead to an outbreak?
Text	The deadly Ebola virus has arrived in the United States just in time for the holiday travel season, carrying fear and uncertainty with it...

Table 3: Example of a tweet, title of the article and the text when tweet cannot be extracted from text.

We first compute the proportion of tweets that can be recovered directly from the article in its entirety (Section 4.1). Then, we calculate the degree of overlap in terms of unigrams and bigrams between the tweet and the text of the document (Sections 4.2, 4.3).

In addition, we consider locality within the article when computing the overlap. For the unigram analysis, we performed a variant of the analysis,

Tweet	Officer **Wilson will be returned to active duty if no indictment**, says #Ferguson Police **Chief** http://t.co/zrRIBxMUYJ
Title	Jackson clarifies comments on Wilson's future status
Text	...**Chief** Jackson said if the grand jury does **not indict Wilson**, he **will** immediately **return to active duty**....

Table 4: Example of a tweet, title of the article and the text when tweet can be extracted from text. The matched portions of the tweet and article are in bold.

in which we computed the overlap within three-sentence windows in the source article (Section 4.4). We also compute the least common subsequences between the tweet and the document (Section 4.5). This was done to investigate whether sentence compression techniques could be applied to local context windows to generate the tweet.

These calculations are analogous to the ROUGE-1, -2 and -L style calculations. These results give an indication of the degree to which the tweet is extracted from the document text.

For all these analyses, the stop words have been eliminated from the tweet as well as the document, so that only the informative words are taken into consideration. The comparisons were made without lemmatization or stemming, to adhere closely to existing work in extractive summarization, where the only modifications to the source text are removing discourse cue words or removing words by sentence compression techniques. The hashtags, references (@) and URLs from the tweets were all removed for analysis.

4.1 Exact Match Calculations

We first checked for a complete substring match of the tweet in the text. Out of the 2471 unique instances of tweet and article pairs, a complete match was found only 23 times. In 9 cases out of these, the tweet text matched the title of the article, which our preprocessing tool did not correctly separate from the body of the article. In the other cases, the text of the tweet appears in its entirety inside the body of the article. This suggests that the user chose the sentence that either seemed to be the most conclusive contribution of the article, or expressed the opinion of the user to be tweeted.

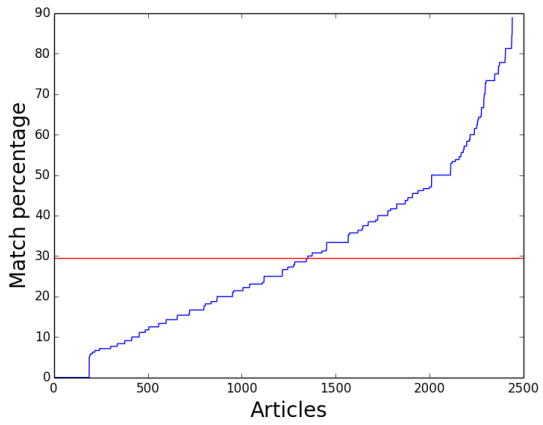


Figure 1: Distribution of unigram match percentage over unique tweets and articles. The mean is 29.53%, indicated by the red horizontal line, with a standard deviation of 20.2%

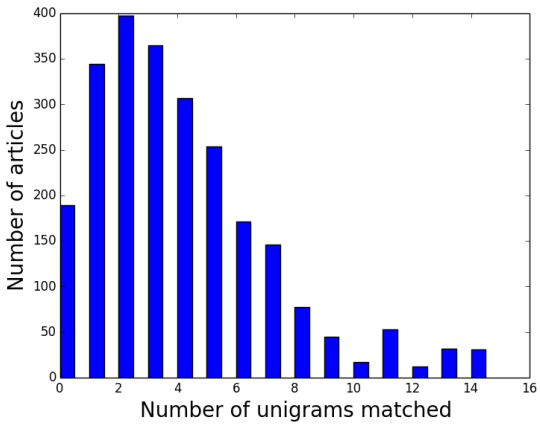


Figure 2: Histogram of number of unique tweet-article pairs vs number of unigrams matched. The mean number of unigrams matched per tweet-article pair is 3.9.

An example for this is detailed in Table 5.

Apart from the 9 times where the tweet was matched with title in the article, we also checked to see if the tweet text matched with the article titles that were separately extracted by the `newspaper` package in order to determine if tweets could be generated using the headline generation methods. We found that it did not match with the titles. However, even though there are no exact matches, there might still be matches where the tweet is a slight modification of the headline of the article, and can be measured using a partial match measure.

Tweet	@PNHP: **6. Renounce punitive and counterproductive measures such as sealing the borders,** http://t.co/LRLS2MhPRE #Ebola
Title	Physicians for a National Health Program
Text	As health professionals and trainees, we call on President Obama to take the following immediate steps to address the Ebola crisis... **6. Renounce punitive and counterproductive measures such as sealing the borders,** and take steps to address the...

Table 5: Example where tweet is extracted as is from the text, matched portion in bold.

4.2 Percentage Match for Unigrams

Next, we did a percentage match with the text of the article. This was a bag-of-words check using unigram overlap between the tweet and the document. Let $unigrams(x)$ be the set of unigrams for some text x, then u, the percentage of matching unigrams found between a given tweet, t and a given article, a, can be defined as

$$u = \frac{|unigrams(t) \cap unigrams(a)|}{|unigrams(t)|} * 100 \quad (1)$$

Figure 1 shows the percentage of matches in the tweet and the article text as compared to the number of unigrams in the tweet. The mean match percentage is 29.53% and standard deviation is 20.2%. The mean of this distribution shows that the number of matched unigrams from a tweet in the article are fairly low. Figure 2 shows the number of articles with a certain number of matching unigrams. The graph shows that the most common number of unigrams matched was 2. The number of articles with higher unigrams matched goes on decreasing. The slight rise at the end - more than 10 matched unigrams - is accounted for by the completely matched tweets described above.

4.3 Percentage Match for Bigrams

Similar to the unigram matching techniques, the bigram percentage matching was also calculated. The text of the tweet was converted into bigrams and we then looked for those bigrams in the article text. The percentage was calculated similar

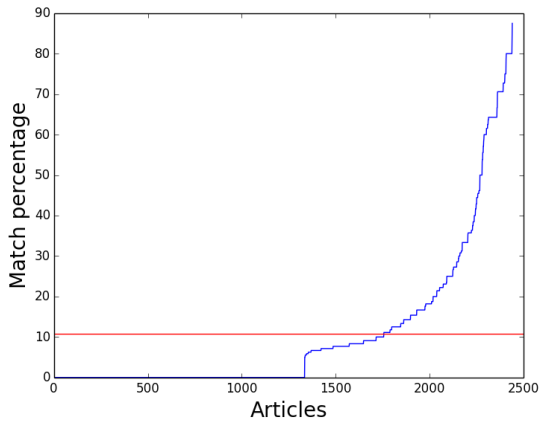


Figure 3: Distribution of bigram match percentage over the tweet-article pair. The mean here is 10.73% shown by the red horizontal line, with a standard deviation of 18.5%

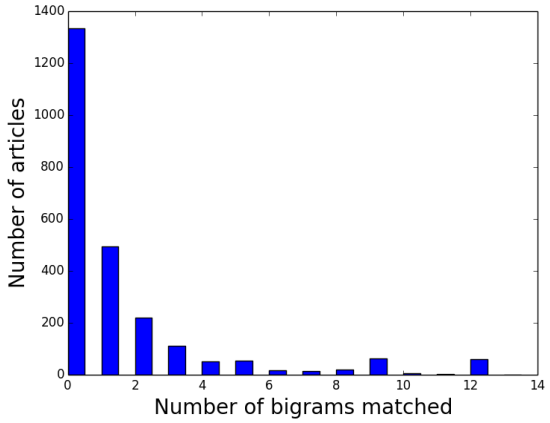


Figure 4: Histogram of number of unique tweet-article pairs vs number of bigrams matched. The mean number of bigrams matched per article is 1.9.

to the unigram matching done earlier. For the set of bigrams for a text x, $bigrams(x)$, percentage of matching bigrams b for the tweet t and article a is:

$$b = \frac{|bigrams(t) \cap bigrams(a)|}{|bigrams(t)|} * 100 \quad (2)$$

Figure 3 shows the percentages of matched bigrams found. The mean is 10.73 with a standard deviation of 18.5. As seen in the figure, most of the tweet-article pairs have no matched bigrams. The percentage then increases to reflect the complete matches found above.

Figure 4 shows frequency of the number of tweet-article pairs for the number of bigrams matched. There are no matched bigrams for most of the pairs. A smaller number of articles had one matched bigram, and the number decreased until the end, where it increases a little at more than 10 matched bigrams because of exact tweet matches.

4.4 Percentage Match Inside a Window in the Article Text

The next analysis checks for a significant word matching inside a three-sentence window inside the article text. We used a three sentence long window using the sentence boundary information obtained during preprocessing. A window of three sentences was chosen to give a smaller context for the tweet to be extracted from than the entire article. The number was chosen as a moderate context window size as not too small to reduce it to a sentence level, and not too big for the context to be diluted. This was done to investigate whether a pseudo-extractive multi-sentence compression approach could convert a small number of sentences into a tweet.

After the text of the window was extracted, we performed a similar analysis as the last one, except on a smaller set of sentences. The matching percentages from all three-sentence windows in the articles were computed and the maximum out of these was taken for the final results. Let a sentence window w_i be the set of three consecutive sentences starting from the sentence number i. For this window, the unigram match in the tweet t, and the window is the unigram match u calculated above. Then, the maximum match from all the windows, uw is

$$uw = max_{w_i \in S} u(t, w_i) \quad (3)$$

The result from this experiment is shown in Figure 5. Here, the mean of the values is 26.6% and deviation 17%. Again this shows that only a small proportion of tweets can be generated even with an approach that combines unigrams from multiple sentences in the article.

4.5 Longest Common Subsequence Match Inside a Window for the Text

The percentage match analyses were a bag-of-words approach that disregarded the order of the words inside the texts and tweets. To respect the order of the words in the sentence of the tweet, we also used the least common subsequence algorithm between the tweet text and the document

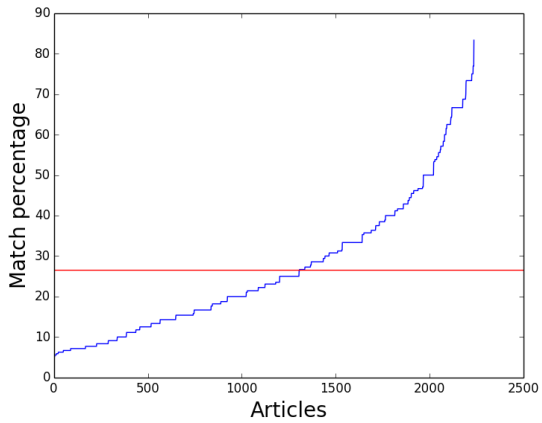


Figure 5: Percentages of common words in tweet and a three sentence window in the article. The maximum match from all percentages is chosen for an article. The red horizontal line is the mean is 26.6%, and standard deviation 17%.

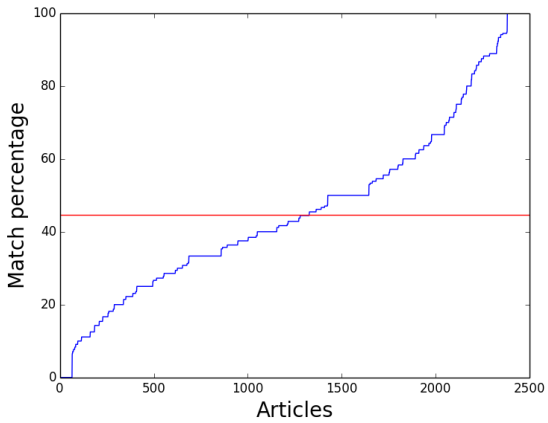


Figure 6: Percentages of words matching in tweet and document text using an LCS algorithm. Mean is 44.6%, which is shown by the red horizontal line, and standard deviation is 22.7%.

text. This subsequence matching was done inside a sentence window of 5 sentences. Again, the final result for the article was the window in which the maximum percentage was recorded among all windows. The percentage match was calculated using the number of words in the tweet as the denominator.

If *lcs(t, a)* is the longest common subsequence between the tweet t and article a, $\textit{unigrams}(x)$ is the set of unigrams for a text x, then the percentage of match for the lcs as compared to the tweet, l is

$$l = \frac{|\textit{lcs}(t,a)|}{|\textit{unigrams}(t)|} * 100 \quad (4)$$

These numbers are shown in Figure 6. The mean here is 44.6% and the standard deviation is 22.7%.

5 Interaction with Formality

As seen in the results of the analyses performed in Section 4, the tweets have little in common with the articles they are linked to. This shows that extractive summarization algorithms can only recover a small proportion of the indicative tweets.

To tie in the results of the findings above with some intuitive notions about the text and see how formality interacts with the results, we also calculated the formality of the articles. This formality score was correlated with the longest common subsequence measure that we defined above.

We assume that the formality of an article can be estimated by the formality of the words and phrases in the article. We used the formality lexicon of Brooke and Hirst (2013). They calculate formality scores for words and sentences by training a model on a large corpus based on the appearance of words in specific documents. Their model represents words as vectors and the formal and informal seeds appear in opposite halves of the graphs, suggesting that we can use these seeds to determine if an article is formal or informal. The lexicon consists of words and phrases and their degree of formality. Thus, more formal words are marked on a positive scale and informal words like those occurring in colloquial language are marked on a negative scale.

Let the set of formality expressions from the lexicon be L, and the formality score for an expression e be $\textit{score}(e)$. Let the set of all substrings from the article $\textit{substrings}(a)$ be S. Then, the formality score f for an article a is the number of formal expressions per 10 words in article is

$$f = \frac{\sum_{e \in L \& e \in S} \textit{score}(e)}{|\textit{unigrams}(a)|} * 10 \quad (5)$$

The formality lexicon gave positive weights for formal expressions and negative for informal expressions. When we computed f using both formal and informal expressions, we found that the informal words predominated and "swamped" the signal of the formal words, leading to incomprehensible results. Thus, we discarded the informal words and used only the weights from the formal words in our final calculations. To check that these

formality scores made sense intuitively, we calculated the average formality score for the articles belonging to each hashtag and ordered them, as shown in Table 7.

Lowest	Highest
#theforceawakens	#KevinVickers
#TaylorSwift	#erdogan
#winteriscoming	#apec

Table 6: Table of hashtags (broadly, topics) with highest and lowest formality according to the lexicon.

This formality score for each article was correlated with the percentage of match obtained using the longest common subsequence algorithm. The Pearson correlation value was 0.41, with a p-value of 7.08e-66, indicating that the interaction between formality and overlap was highly significant. Hence, we can say that the more formal the subject or the article, the better the tweet can be extracted from the article. Table 7 gives an example of the formality of the article, which is a low 4.2 formality words per 10 words, where the tweet is not extracted from the article, but rephrased from the article instead.

Tweet	@globetoronto: Why Buffalo got clobbered with snow and Toronto did not. #weather #snowstorm http://t.co/gcwwoDPZmX... http://t.co/BXY7EH6F3u"
Title	What caused Buffalos massive snow and why Toronto got lucky
Text	Torontonians have long been the butt of jokes about calling in the army every time a few snow flurries whip by...

Table 7: Example of a tweet, title of the article where the formality of the article is over the mean, and the tweet is extracted from the article.

We speculate that tweets associated with less formal articles may contain more abbreviations and non-standard words or spellings, which decreases the amount of overlap. We plan to experiment with tweet normalization systems to account for this factor.

6 Discussions

Having presented the above statistics showing that only a small portion of indicative tweets can be recovered from the article they link to if viewed as an extractive summarization problem, the question then becomes, how should we view the process of tweet generation?

We think that one promising direction is to model more explicitly the intent, or the purpose of the tweets. There have been several studies on classifying intents in tweets, but in many cases the intents are general, high-level intents of the tweets, more akin to classifying the topic or genre of the tweet than the intent. Wang et al. (2015) classify intents as food and drink, travel, career and so on, ones that can directly be used as intents for purchasing and can be utilized for advertisements. They also focus on finding tweets with intent and then classifying those. Banerjee et al. (2012) analyze real time data to detect presence of intents in tweets. Gómez-Adorno et al. (2014) use features from text and stylistics to determine user intentions, which are classified as news report, opinion, publicity and so on. Mohammad et al. (2013) study the classification of user intents specifically for tweets related to elections. They study one election and classify tweets as ones that agree or disagree with the candidate, ones that are meant for humour, support and so on.

These definitions of intent, while a promising start, will not be sufficient for tweet generation. For this purpose, intent would be the reason the user chose to share the article with that particular text. This would include reasons like support some cause, promote a product or an article, agree or disagree with an event, or express an opinion about it. Identifying these intents will help provide parameters for generating tweets.

7 Conclusion

We have described a study that investigates whether indicative tweet generation can be viewed as an extractive summarization problem. By analyzing a collection of indicative tweets that we collected according to measures inspired by extractive summarization evaluation measures, we find that most tweets cannot be recovered from the article that they link to, demonstrating a limit to the effectiveness of extractive methods.

We further performed an analysis to determine the role of formality differences between the source article and the Twitter genre. We find evidence that formality is an important factor, as the less formal the source article is, the less extrac-

tive the tweets seem to be. Future methods that can change the level of formality of a piece of text without changing the contents will be needed, as will those that explicitly consider the intended use of the tweet.

Acknowledgments

We would like to thank the anonymous reviewers for their comments and suggestions, and Julian Brooke for the formality lexicon used in a part of this study.

References

Nilanjan Banerjee, Dipanjan Chakraborty, Anupam Joshi, Sumit Mittal, Angshu Rai, and Balaraman Ravindran. 2012. Towards analyzing micro-blogs for detection and classification of real-time intentions. In *ICWSM*, pages 391–394.

Julian Brooke and Graeme Hirst. 2013. A multi-dimensional bayesian approach to lexical style. In *HLT-NAACL*, pages 673–679.

Deepayan Chakrabarti and Kunal Punera. 2011. Event summarization using tweets. *ICWSM*, 11:66–73.

Lu Chen, Wenbo Wang, Meenakshi Nagarajan, Shaojun Wang, and Amit P Sheth. 2012. Extracting diverse sentiment expressions with target-dependent polarity from twitter. In *ICWSM*, pages 50–57.

Jackie Chi Kit Cheung and Gerald Penn. 2013. Towards robust abstractive multi-document summarization: A caseframe analysis of centrality and domain. In *ACL (1)*, pages 1233–1242.

John M Conroy, Judith D Schlesinger, and Dianne P O'Leary. 2006. Topic-focused multi-document summarization using an approximate oracle score. In *Proceedings of the COLING/ACL on Main conference poster sessions*, pages 152–159. Association for Computational Linguistics.

Rumi Ghosh, Tawan Surachawala, and Kristina Lerman. 2011. Entropy-based classification of'retweeting'activity on twitter. *arXiv preprint arXiv:1106.0346*.

Helena Gómez-Adorno, David Pinto, Manuel Montes, Grigori Sidorov, and Rodrigo Alfaro. 2014. Content and style features for automatic detection of users intentions in tweets. In *Advances in Artificial Intelligence–IBERAMIA 2014*, pages 120–128. Springer.

Liwei He, Elizabeth Sanocki, Anoop Gupta, and Jonathan Grudin. 2000. Comparing presentation summaries: slides vs. reading vs. listening. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, pages 177–184. ACM.

David Inouye and Jugal K Kalita. 2011. Comparing twitter summarization algorithms for multiple post summaries. In *Privacy, Security, Risk and Trust (PASSAT) and 2011 IEEE Third Inernational Conference on Social Computing (SocialCom), 2011 IEEE Third International Conference on*, pages 298–306. IEEE.

Lingpeng Kong, Nathan Schneider, Swabha Swayamdipta, Archna Bhatia, Chris Dyer, and Noah A. Smith. 2014. A dependency parser for tweets. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1001–1012, Doha, Qatar, October. Association for Computational Linguistics.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, pages 74–81.

Elena Lloret and Manuel Palomar. 2013. Towards automatic tweet generation: A comparative study from the text summarization perspective in the journalism genre. *Expert Systems with Applications*, 40(16):6624–6630.

Christoph Lofi and Ralf Krestel. 2012. iparticipate: Automatic tweet generation from local government data. In *Database Systems for Advanced Applications*, pages 295–298. Springer.

Inderjeet Mani. 2001. *Automatic summarization*, volume 3. John Benjamins Publishing.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60.

Saif M Mohammad, Svetlana Kiritchenko, and Joel Martin. 2013. Identifying purpose behind electoral tweets. In *Proceedings of the Second International Workshop on Issues of Sentiment Discovery and Opinion Mining*, pages 1–9. ACM.

Brendan O'Connor, Michel Krieger, and David Ahn. 2010. Tweetmotif: Exploratory search and topic summarization for Twitter. In *ICWSM*, pages 384–385.

Olutobi Owoputi, Brendan O'Connor, Chris Dyer, Kevin Gimpel, Nathan Schneider, and Noah A. Smith. 2013. Improved part-of-speech tagging for online conversational text with word clusters. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 380–390, Atlanta, Georgia, June. Association for Computational Linguistics.

Lu Wang, Claire Cardie, and Galen Marchetti. 2014. Socially-informed timeline generation for complex events. In *Proceedings of Human Language Technologies: The 2015 Annual Conference of the North American Chapter of the ACL*, pages 1055–1065.

Jinpeng Wang, Gao Cong, Xin Wayne Zhao, and Xiaoming Li. 2015. Mining user intents in twitter: A semi-supervised approach to inferring intent categories for tweets. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, pages 339–345.

Zhongyu Wei and Wei Gao. 2014. Utilizing microblogs for automatic news highlights extraction. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 872–883.

Visual Bilingual Lexicon Induction with Transferred ConvNet Features

Douwe Kiela
Computer Laboratory
University of Cambridge
douwe.kiela@cl.cam.ac.uk

Ivan Vulić
Department of Computer Science
KU Leuven
ivan.vulic@cs.kuleuven.be

Stephen Clark
Computer Laboratory
University of Cambridge
stephen.clark@cl.cam.ac.uk

Abstract

This paper is concerned with the task of bilingual lexicon induction using image-based features. By applying features from a convolutional neural network (CNN), we obtain state-of-the-art performance on a standard dataset, obtaining a 79% relative improvement over previous work which uses bags of visual words based on SIFT features. The CNN image-based approach is also compared with state-of-the-art linguistic approaches to bilingual lexicon induction, even outperforming these for one of three language pairs on another standard dataset. Furthermore, we shed new light on the type of visual similarity metric to use for genuine similarity versus relatedness tasks, and experiment with using multiple layers from the same network in an attempt to improve performance.

1 Introduction

Bilingual lexicon induction is the task of finding words that share a common meaning across different languages. It plays an important role in a variety of tasks in information retrieval and natural language processing, including cross-lingual information retrieval (Lavrenko et al., 2002; Levow et al., 2005) and statistical machine translation (Och and Ney, 2003). Although parallel corpora have been used successfully for inducing bilingual lexicons for some languages (Och and Ney, 2003), these corpora are either too small or unavailable for many language pairs. Consequently, mono-lingual approaches that rely on comparable instead of parallel corpora have been developed (Fung and Yee, 1998; Koehn and Knight, 2002). These approaches work by mapping language pairs to a shared bilingual space and extracting lexical items from that space. Bergsma and Van Durme (2011) showed that this bilingual space need not be linguistic in nature: they used labeled images from the Web to obtain bilingual lexical translation pairs based on the visual features of corresponding images. Local features are computed using SIFT (Lowe, 2004) and color histograms (Deselaers et al., 2008) and aggregated as bags of visual words (BOVW) (Sivic and Zisserman, 2003) to get bilingual representations in a shared visual space. Their highest performance is obtained by combining these visual features with normalized edit distance, an orthographic similarity metric (Navarro, 2001).

There are several advantages to having a visual rather than a linguistic intermediate bilingual space: First, while labeled images are readily available for many languages through resources such as Google Images, language pairs that have sizeable comparable, let alone parallel, corpora are relatively scarce. Second, it has been found that meaning is often grounded in the perceptual system, and that the quality of semantic representations improves significantly when they are grounded in the visual modality (Silberer and Lapata, 2012; Bruni et al., 2014). Having an intermediate visual space means that words in different languages can be grounded in the same space. Third, it is natural to use vision as an intermediate: when we communicate with someone who does not speak our language, we often communicate by directly referring to our surroundings. Languages that are linguistically far apart will, by cognitive necessity, still refer to objects in the same visual space. While some approaches to bilingual lexicon induction rely on orthographic properties (Haghighi et al., 2008; Koehn and Knight, 2002) or properties of frequency distributions (Schafer and Yarowsky, 2002) that will work only for

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 148–158,
Lisbon, Portugal, 17-21 September 2015.

closely related languages, a visual space can work for any language, whether it's English or Chinese, Arabic or Icelandic, or all Greek to you.

It has recently been shown, however, that much better performance can be achieved on semantic similarity and relatedness tasks by using visual representations from deep convolutional neural networks (CNNs) instead of BOVW features (Kiela and Bottou, 2014). In this paper we apply such CNN-derived visual features to the task of bilingual lexicon induction. To obtain a translation of a word in a source language, we find the nearest neighbours from words in the target language, where words in both languages reside in a shared visual space made up of CNN-based features. Nearest neighbours are found by applying similarity metrics from both Kiela and Bottou (2014) and Bergsma and Van Durme (2011). In summary, the contributions of this paper are:

- We obtain a relative improvement of 79% over Bergsma and Van Durme (2011) on a standard dataset based on fifteen language pairs.
- We shed new light on the question of whether genuine similarity versus semantic relatedness tasks require different similarity metrics for optimal performance (Kiela and Bottou, 2014).
- We experiment with using different layers of the CNN and find that performance is not affected significantly in either case, obtaining a slight improvement for the relatedness task but no improvement for genuine similarity.
- Finally, we show that the visual approach outperforms the linguistic approaches on one of the three language pairs on a standard dataset. To our knowledge this is the first work to provide a comparison of visual and state-of-the-art linguistic approaches to bilingual lexicon induction.

2 Related Work

2.1 Bilingual Lexicon Learning

Bilingual lexicon learning is the task of automatically inducing word translations from raw data, and is an attractive alternative to the time-consuming and expensive process of manually building high-quality resources for a wide variety of language pairs and domains. Early approaches relied on limited and domain-restricted parallel data, and the induced lexicons were typically a by-product of word alignment models (Och and Ney, 2003). To alleviate the issue of low coverage, a large body of work has been dedicated to lexicon learning from more abundant and less restricted comparable data, e.g., (Fung and Yee, 1998; Rapp, 1999; Gaussier et al., 2004; Shezaf and Rappoport, 2010; Tamura et al., 2012). However, these models typically rely on the availability of bilingual seed lexicons to produce shared bilingual spaces, as well as large repositories of comparable data. Therefore, several approaches attempt to learn lexicons from large monolingual data sets in two languages (Koehn and Knight, 2002; Haghighi et al., 2008), but their performance again relies on language pair-dependent clues such as orthographic similarity. A further approach removed the requirement of seed lexicons, and induced lexicons using bilingual spaces spanned by multilingual probabilistic topic models (Vulić et al., 2011; Liu et al., 2013; Vulić and Moens, 2013b). However, these models require document alignments as initial bilingual signals.

In this work, following recent research in multi-modal semantics and image representation learning—in particular deep learning and convolutional neural networks—we test the ability of purely visual data to induce shared bilingual spaces and to consequently learn bilingual word correspondences in these spaces. By compiling images related to linguistic concepts given in different languages, the potentially prohibitive data requirements and language pair-dependence from prior work is removed.

2.2 Deep Convolutional Neural Networks

Deep convolutional neural networks (CNNs) have become extremely popular in the computer vision community. These networks currently provide state-of-the-art performance for a variety of key computer vision tasks such as object recognition (Razavian et al., 2014). They tend to be relatively deep, consisting of a number of rectified linear unit layers (Nair and Hinton, 2010) and a series of convolutional layers (Krizhevsky et al., 2012). Recently, such layers have been used in transfer learning techniques, where they are used as mid-level features in other computer vision tasks (Oquab et al., 2014). Although the idea of transferring CNN features is not new (Driancourt and Bottou, 1990), the simultaneous availability of

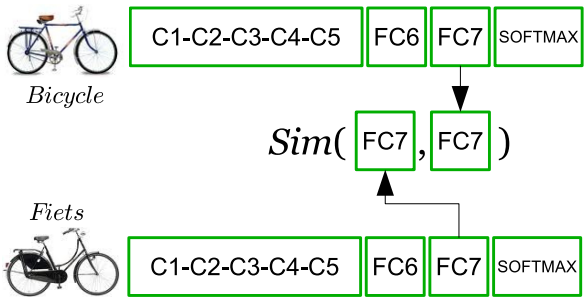


Figure 1: Illustration of calculating similarity between images from different languages.

massive amounts of data and cheap GPUs has led to considerable advances in computer vision, similar in scale to those witnessed with SIFT and HOG descriptors a decade ago (Razavian et al., 2014).

2.3 Multi-Modal Semantics

Multi-modal semantics is motivated by parallels with human concept acquisition. It has been found that semantic knowledge, from a very early age, relies heavily on perceptual information (Louwerse, 2008), and there exists substantial evidence that many concepts are grounded in the perceptual system (Barsalou, 2008). One way to accomplish such grounding is by combining linguistic representations with information from a perceptual modality, obtained from, e.g., property norming experiments (Silberer and Lapata, 2012; Silberer et al., 2013; Roller and Schulte im Walde, 2013; Hill and Korhonen, 2014) or extracting features from raw image data (Feng and Lapata, 2010; Leong and Mihalcea, 2011; Bruni et al., 2014; Kiela et al., 2014). Such multi-modal visual approaches often rely on local descriptors, such as SIFT (Lowe, 2004), SURF (Bay et al., 2008), or HOG (Dalal and Triggs, 2005), as well as pyramidal variants of these descriptors such as PHOW (Bosch et al., 2007). However, deep CNN features have recently been successfully transferred to multi-modal semantics (Kiela and Bottou, 2014; Shen et al., 2014). Deep learning techniques have also been successfully employed in cross-modal tasks (Frome et al., 2013; Socher et al., 2014; Lazaridou et al., 2014; Kiros et al., 2014). Other examples of multi-modal deep learning use restricted Boltzmann machines (Srivastava and Salakhutdinov, 2014) or auto-encoders (Wu et al., 2013; Silberer and Lapata, 2014).

3 A Purely Visual Approach to Bilingual Lexicon Learning

We assume that the best translation, or matching lexical item, of a word w_s (in the source language) is the word w_t (in the target language) that is the nearest cross-lingual neighbour to w_s in the bilingual visual space. Hence a similarity (or distance) score between lexical items from different languages is required. In this section, we describe: one, how to build image representations from sets of images associated with each lexical item, i.e. how to induce a shared bilingual visual space in which all lexical items are represented; and two, how to compute the similarity between lexical items using their visual representations in the shared bilingual space. We also describe the evaluation datasets and metrics we use.

To facilitate further research, we will make our code and data publicly available. Please see the following webpage: `http://www.cl.cam.ac.uk/~dk427/bli.html`.

3.1 Image Representations

We use Google Images to extract the top n ranked images for each lexical item in the evaluation datasets. It has been shown that images from Google yield higher quality representations than comparable sources such as Flickr (Bergsma and Goebel, 2011) and that Google-derived datasets are competitive with "hand prepared datasets" (Fergus et al., 2005). Google Images also has the advantage that it has full coverage and is multi-lingual, as opposed to other potential image sources such as ImageNet (Deng et al., 2009) or the ESP Game Dataset (von Ahn and Dabbish, 2004). For each Google search we specify the target language corresponding to the lexical item's language. Figure 2 gives some example images retrieved using the same query terms in different languages. For each image, we extract the pre-softmax layer of an AlexNet (Krizhevsky et al., 2012). The network contains a number of layers, starting with five convolutional layers, two fully connected layers and finally a softmax, and has been pre-trained on the ImageNet classification task using Caffe (Jia et al., 2014). See Figure 1 for a simple diagram illustrating the approach.

3.2 Visual Similarity

Suppose that, as part of the evaluation, the similarity between *bicycle* and *fiets* is required. Each of

the two words has n images associated with it – the top n as returned by Google image search, using *bicycle* and *fiets* as separate query terms. Hence to calculate the similarity, a measure is required which takes two sets of images as input. The standard approach in multi-modal semantics is to derive a single image representation for each word, e.g., by averaging the n images. An alternative is to take the pointwise maximum across the n image vector representations, also producing a single vector (Kiela and Bottou, 2014). Kiela and Bottou call these combined representations CNN-MEAN and CNN-MAX, respectively. Cosine is then used to calculate the similarity between the resulting pair of image vectors.

An alternative strategy, however, is to consider the similarities between individual images instead of their aggregated representations. Bergsma and Van Durme (2011) propose two similarity metrics based on this principle: taking the average of the maximum similarity scores (AVGMAX), or the maximum of the maximum similarity scores (MAXMAX) between associated images. Continuing with our example, for each of the n images for *bicycle*, the maximum similarity is found by searching over the n images for *fiets*. AVGMAX then takes the average of those n maximum similarites; MAXMAX takes the maximum. To avoid confusion, we will refer to the CNN-based models that use these metrics as CNN-AVGMAX and CNN-MAXMAX. Formally, these metrics are defined as in Table 1. We experiment with both kinds of MAX and find that they optimize for different kinds of similarity.

AVGMAX	$\frac{1}{n}\sum_{i_s \in \mathcal{I}(w_s)} \max_{i_t \in \mathcal{I}(w_t)} sim(i_s, i_t)$
MAXMAX	$\max_{i_s \in \mathcal{I}(w_s)} \max_{i_t \in \mathcal{I}(w_t)} sim(i_s, i_t)$
CNN-MEAN	$sim(\frac{1}{n}\sum_{i_s \in \mathcal{I}(w_s)} i_s, \frac{1}{n}\sum_{i_t \in \mathcal{I}(w_t)} i_t)$
CNN-MAX	$sim(\max' \mathcal{I}(w_s), \max' \mathcal{I}(w_t))$

Table 1: Visual similarity metrics between two sets of n images. $\mathcal{I}(w_s)$ represents the set of images for a given source word w_s, $\mathcal{I}(w_t)$ the set of images for a given target word w_t; $\max'$ takes a set of vectors and returns the single element-wise maximum vector.

3.3 Evaluations

Test Sets. Bergsma and Van Durme's primary evaluation dataset consists of a set of five hundred matching lexical items for fifteen language pairs, based on six languages. (The fifteen pairs results from all ways of pairing six languages). The data is publicly available online.[1] In order to get the five hundred lexical items, they first rank nouns by the conditional probability of them occurring in the pattern "{*image,photo,photograph,picture*} *of* {*a,an*} _____" in the web-scale Google N-gram corpus (Lin et al., 2010), and take the top five hundred words as their English lexicon. For each item in the English lexicon, they obtain corresponding items in the other languages—Spanish, Italian, French, German and Dutch—through Google Translate. We call this dataset BERGSMA500.

In addition to that dataset, we evaluate on a dataset constructed to measure the general performance of bilingual lexicon learning models from comparable Wikipedia data (Vulić and Moens, 2013a). The dataset comprises 1,000 nouns in three languages: Spanish (ES), Italian (IT), and Dutch (NL), along with their one-to-one gold-standard word translations in English (EN) compiled semi-automatically using Google Translate and manual annotators for each language. We call this dataset VULIC1000[2]. The test set is accompanied with comparable data for training, for the three language pairs ES/IT/NL-EN on which text-based models for bilingual lexicon induction were trained (Vulić and Moens, 2013a).

Given the way that the BERGSMA500 dataset was created, in particular the use of the pattern described above, it contains largely concrete linguistic concepts (since, eg, *image of a democracy* is unlikely to have a high corpus frequency). In contrast, VULIC1000 was designed to capture general bilingual word correspondences, and contains several highly abstract test examples, such as *entendimiento (understanding)* and *desigualdad (inequality)* in Spanish, or *scoperta (discovery)* and *cambiamento (change)* in Italian. Using the two evaluation datasets can potentially provide

[1] http://www.clsp.jhu.edu/~sbergsma/LexImg/

[2] http://people.cs.kuleuven.be/~ivan.vulic/software/

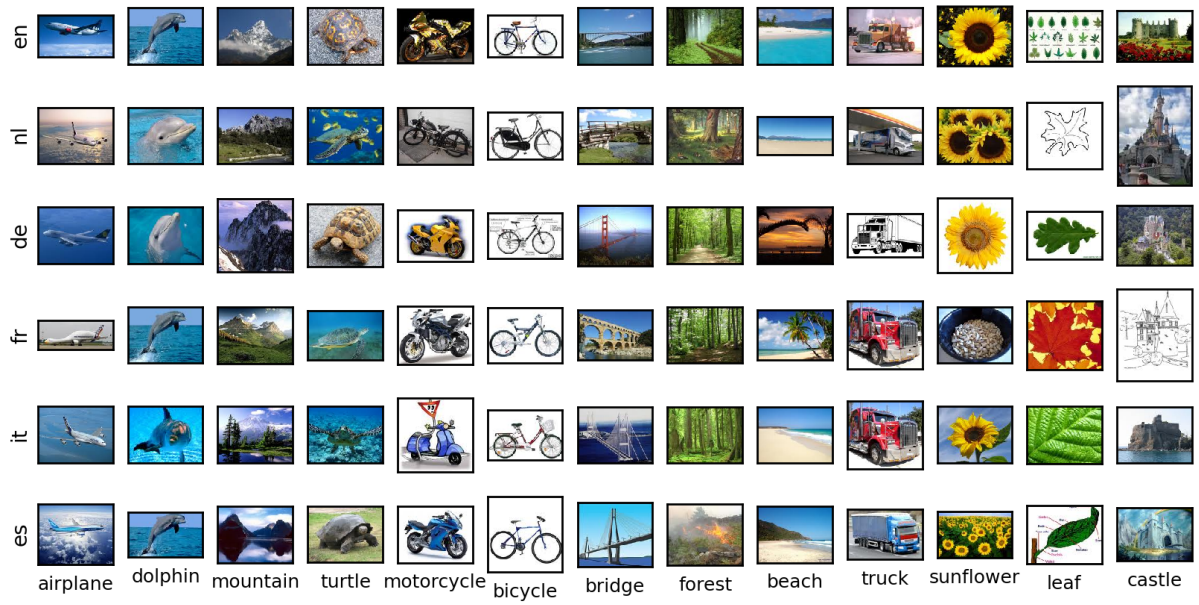


Figure 2: Example images for the languages in the Bergsma and Van Durme dataset.

Method	P@1	P@5	P@20	MRR
B&VD Visual-Only	31.1	41.4	53.7	0.367
B&VD Visual + NED	48.0	59.5	68.7	0.536
CNN-AVGMAX	**56.7**	**69.2**	**77.4**	**0.658**
CNN-MAXMAX	42.8	60.0	64.5	0.529
CNN-MEAN	50.5	62.7	71.1	0.586
CNN-MAX	51.4	64.9	74.8	0.608

Table 2: Performance on BERGSMA500 compared to Bergsma and Van Durme (B&VD).

some insight into how purely visual models for bilingual lexicon induction behave with respect to both abstract and concrete concepts.

Evaluation Metrics. We measure performance in a standard way using mean-reciprocal rank:

$$MRR = \frac{1}{M}\sum_{i=1}^{M}\frac{1}{rank(w_s, w_t)} \quad (1)$$

where $rank(w_s, w_t)$ denotes the rank of the correct translation w_t (as provided in the gold standard) in the ranked list of translation candidates for w_s, and M is the number of test cases. We also use *precision at N* (P@N) (Gaussier et al., 2004; Tamura et al., 2012; Vulić and Moens, 2013a), which measures the proportion of test instances where the correct translation is within the top N highest ranked translations.

4 Results

We evaluate the four similarity metrics on the BERGSMA500 dataset and compare the results to the systems of Bergsma and Van Durme, who report results for the AVGMAX function, having concluded that it performs better than MAXMAX on English-Spanish translations. We report their best-performing visual-only system, which combines SIFT-based descriptors with color histograms, as well as their best-performing overall system, which combines the visual approach with normalized edit distance (NED). Results are averaged over fifteen language pairs.

The results can be seen in Table 2. Each of the CNN-based methods outperforms the B&VD systems. The best performing method overall, CNN-AVGMAX, provides a 79% relative improvement over the B&VD visual-only system on the MRR measure, and a 23% relative improvement over their best-performing approach, which includes non-visual information in the form of orthographic similarity. Moreover, their methods include a tuning parameter λ that governs the contributions of SIFT-based, color histogram and normalized edit distance similarity scores, whilst our approach does not require any parameter tuning.

4.1 Similarity and Relatedness

The results in Table 2 indicate that the per-image CNN-AVGMAX metric outperforms the

Language Pair	Method	P@1	P@5	P@10	P@20	MRR
ES ⇒ EN	BOOTSTRAP	57.7	74.7	80.9	84.8	0.652
	CNN-AVGMAX	41.9	54.6	59.1	65.6	0.485
	CNN-MAXMAX	34.9	47.4	53.7	58.5	0.414
	CNN-MEAN	35.4	48.5	51.7	55.8	0.416
	CNN-MAX	33.3	46.3	50.3	54.5	0.395
IT ⇒ EN	BOOTSTRAP	64.7	80.6	85.6	89.7	0.716
	CNN-AVGMAX	28.3	40.6	44.8	50.9	0.343
	CNN-MAXMAX	22.6	33.5	38.6	44.4	0.282
	CNN-MEAN	22.7	33.2	37.9	42.6	0.281
	CNN-MAX	21.3	32.7	36.8	41.5	0.269
NL ⇒ EN	BOOTSTRAP	20.6	35.7	43.4	51.3	0.277
	CNN-AVGMAX	38.4	48.5	53.7	58.6	0.435
	CNN-MAXMAX	30.8	42.6	47.8	52.9	0.367
	CNN-MEAN	32.3	42.3	46.5	50.1	0.373
	CNN-MAX	30.4	41.0	44.3	49.3	0.356

Table 4: Performance on VULIC1000 compared to the linguistic bootstrapping method of Vulić and Moens (2013b).

Method	MEN	SimLex-999
CNN-AVGMAX	0.56	0.34
CNN-MAXMAX	0.55	**0.36**
CNN-MEAN	**0.61**	0.32
CNN-MAX	0.60	0.27

Table 3: Spearman ρ_s correlation for the visual similarity metrics on a relatedness (MEN) and a genuine similarity (SimLex-999) dataset.

aggregated visual representation-based metrics of CNN-MEAN and CNN-MAX, despite the fact that Kiela and Bottou (2014) achieved optimal performance using the latter metrics on a well-known conceptual relatedness dataset. It has been noted before that there is a clear distinction between similarity and relatedness. This is one of the reasons that, for example, WordSim353 (Finkelstein et al., 2002) has been criticized: it gives high similarity scores to cases of genuine similarity as well as relatedness (Agirre et al., 2009; Hill et al., 2014). The MEN dataset (Bruni et al., 2014) that Kiela and Bottou (2014) evaluate on explicitly measures word relatedness. In contrast, the current lexicon learning task seems to require something else than relatedness: whilst a *chair* and *table* are semantically related, a translation for *chair* is not a good translation for *table*. For example, we want to make sure we translate *chair* to *stuhl* in German, and not to *tisch*. In other words, what we are interested in for this particular task is genuine similarity, rather than relatedness.

Thus, we can evaluate the quality of our similarity metrics by comparing their performance on similarity and relatedness tasks: if a metric performs well at measuring genuine similarity, this is indicative of its performance in the bilingual lexicon induction task. In order to examine this question further, we evaluate performance on the MEN dataset, which measures relatedness (Bruni et al., 2014), and the nouns-subset of the SimLex-999 dataset, which measures genuine similarity (Hill et al., 2014). For each pair in the dataset, we calculate the similarity score and report the Spearman ρ_s correlation, which measures how well the ranking of pairs given by the automatic system matches that according to the gold-standard human similarity scores. The results are reported in Table 3.

It is clear that the per-image similarity metrics perform better on genuine similarity, as measured by SimLex-999, than on relatedness, as measured by MEN. In fact, the "aggressive" CNN-MAXMAX method, which picks out a single pair of images to represent a linguistic pair, works best for SimLex-999, indicating how stringently it focuses on genuine similarity. For the aggregated visual representation-based metrics, we see the opposite effect: they perform better on the relatedness task. This sheds light on a question raised by Kiela and Bottou (2014), where they speculate

that certain errors are a result of whether their visual similarity metric measures genuine similarity on the one hand or relatedness on the other: we are better off using per-image visual metrics for genuine similarity, while aggregated visual representation-based metrics yield better performance on relatedness tasks.

4.2 Results on VULIC1000

This section compares our visual-only approach to linguistic approaches for bilingual lexicon induction. Since BERGSMA500 has not been evaluated with such methods, we evaluate on the VULIC1000 dataset (Vulić and Moens, 2013a). This dataset has been used to test the ability of bilingual lexicon induction models to learn translations from comparable data (see sect. 3.3). We do not necessarily expect visual methods to outperform linguistic ones, but it is instructive to see the comparison.

We compare our visual models against the current state-of-the-art lexicon induction model using comparable data (Vulić and Moens, 2013b). This model induces translations from comparable Wikipedia data in two steps: (1) It learns a set of highly reliable one-to-one translation pairs using a shared bilingual space obtained by applying the multilingual probabilistic topic modeling (MuPTM) framework (Mimno et al., 2009). (2) These highly reliable one-to-one translation pairs serve as dimensions of a word-based bilingual semantic space (Gaussier et al., 2004; Tamura et al., 2012). The model then bootstraps from the high-precision seed lexicon of translations and learns new dimensions of the bilingual space until convergence. This model, which we call BOOTSTRAP, obtains the current best results on the evaluation dataset. For more details about the bootstrapping model and its comparison against other approaches, we refer to Vulić and Moens (2013b).

Table 4 shows the results for the language pairs in the VULIC1000 dataset. Of the four similarity metrics, CNN-AVGMAX again performs best, as it did for BERGSMA500. The linguistic BOOTSTRAP method outperforms our visual approach for two of the three language pairs, but, for the NL-EN language pair, the visual methods in fact perform better. This can be explained by the observation that Vulić and Moens's NL-EN training data for the BOOTSTRAP model is less abundant (2-3 times fewer Wikipedia articles) and of lower quality than the data for their ES-EN and IT-EN models. We view these results as highly encouraging: while purely visual methods cannot yet reach the peak performance of linguistic approaches that are trained on sufficient amounts of high-quality text data, they outperform linguistic state-of-the-art methods when there is less or lower quality text data available —which one might reasonably expect to be the default scenario.

Method	**FC7**	**FC6+FC7**	**POOL5+ FC6+FC7**
MEN			
CNN-AVGMAX	0.56	0.57	0.57
CNN-MAXMAX	0.55	0.55	0.56
CNN-MEAN	0.61	0.61	0.61
CNN-MAX	0.60	0.62	0.61
SimLex-999			
CNN-AVGMAX	0.34	0.33	0.31
CNN-MAXMAX	0.36	0.35	0.34
CNN-MEAN	0.32	0.32	0.31
CNN-MAX	0.27	0.26	0.26

Table 5: Spearman ρ_s correlation for the visual similarity metrics on a relatedness (MEN) and a genuine similarity (SimLex-999) dataset using more than one layer from the CNN.

4.3 Adding CNN Layers

The AlexNet (Krizhevsky et al., 2012) from which our image representations are extracted contains a number of layers. Kiela and Bottou (2014) only use the fully connected pre-softmax layer (which we call FC7) for their image representations. It has been found, however, that other layers in the network, especially the preceding fully connected (FC6) and fifth convolutional max pooling (POOL5) layers, also have good properties for usage in transfer learning (Girshick et al., 2014; Yosinski et al., 2014). Hence we performed a (very) preliminary investigation of whether performance increases with the use of additional layers.

In light of our findings concerning the difference between genuine similarity and relatedness, this also gives rise to the question of whether the additional layers might be useful for similarity or relatedness, or both. We hypothesize that the nature of the task matters here: if we are only concerned with genuine similarity, layer FC7 is likely to contain all the necessary information to judge whether two images are similar or not, since

Dataset	Language	Image dispersion
BERGSMA500	EN	0.640 (σ=0.074)
	ES	0.639 (σ=0.072)
	IT	0.646 (σ=0.071)
	FR	0.647 (σ=0.072)
	DE	0.642 (σ=0.072)
	NL	0.645 (σ=0.074)
VULIC1000	EN	0.705 (σ=0.095)
	ES	0.694 (σ=0.092)
	IT	0.725 (σ=0.078)
	NL	0.716 (σ=0.080)

Table 6: Average image dispersion for the datasets, by language.

the network has been trained for object recognition. If, however, we are interested in relatedness, related properties may just as well be encoded deeper in the network, so in the layers preceding FC7 rather than in FC7 itself.

We combined CNN layers with each other by concatenating the normalized layers. For the bilingual lexicon induction tasks, we found that performance did not signficantly increase, which is consistent with our hypothesis (since bilingual lexicon induction requires genuine similarity rather than relatedness, and so only requires FC7). We then tested on the MEN dataset (Bruni et al., 2014) for relatedness and the nouns subset of the SimLex-999 dataset (Hill et al., 2014) for genuine similarity. The results can be found in Table 5.

The results appear to indicate that adding such additional information does not have a clear effect for genuine similarity, but may lead to a small performance increase for relatedness. This could explain why we did not see increased performance on the bilingual lexicon induction task with additional layers. However, the increase in performance on the relatedness task is relatively minor, and further investigation is required into the utility of the additional layers for relatedness tasks.

5 Discussion

A possible explanation for the difference in performance between languages and datasets is that some words are more concrete than others: a visual representation for *elephant* is likely to be of higher quality than one for *happiness*. Visual representations in multi-modal models have been found to perform much better for concrete than abstract concepts (Kiela et al., 2014).

Although concreteness ratings are available for (some) English words, this is not the case for other languages, so in order to examine the concreteness of the datasets we use a substitute method that has been shown to closely mirror how abstract a concept is: image dispersion (Kiela et al., 2014). The image dispersion d of a concept word w is defined as the average pairwise cosine distance between all the image representations $\{i_1 \dots i_n\}$ in the set of images for a given word:

$$d(w) = \frac{2}{n(n-1)} \sum_{i<j\leq n} 1 - \frac{i_j \cdot i_k}{|i_j||i_k|} \quad (2)$$

The average image dispersions for the two datasets, broken down by language, are shown in Table 6. BERGSMA500 has a lower average image dispersion score in general, and thus is more concrete than VULIC1000. It also has less variance. This may explain why we score higher, in absolute terms, on that dataset than on the more abstract one.

When examining individual languages in the datasets, we note that the worst performing language on VULIC1000 is Italian, which is also the most abstract dataset, with the highest average image dispersion score and the lowest variance.

There is some evidence that abstract concepts are also perceptually grounded (Lakoff and Johnson, 1999), but in a more complex way, since abstract concepts express more varied situations (Barsalou and Wiemer-Hastings, 2005). Using an image resource like Google Images that has full coverage for almost any word, means that we can retrieve what we might call "associated" images (such as images of voters for words like *democracy*) as opposed to "extensional" images (such as images of cats for *cat*). This explains why we still obtain good performance on the more abstract VULIC1000 dataset, in some cases outperforming linguistic methods: even abstract concepts can have a clear visual representation, albeit of the associated rather than extensional kind.

However, abstract concepts are overall more likely to yield noisier image sets. Thus, one way to improve results would be to take a multi-modal approach, where we also include linguistic information, if available, especially for abstract concepts.

6 Conclusions and Future Work

We have presented a novel approach to bilingual lexicon induction that uses convolutional neural

network-derived visual features. Using only such visual features, we outperform existing visual and orthographic systems, and even a state-of-the-art linguistic approach for one language, on standard bilingual lexicon induction tasks. In doing so, we have shed new light on which visual similarity metric to use for similarity or relatedness tasks, and have experimented with using multiple layers from a CNN. The beauty of the current approach is that it is completely language agnostic and closely mirrors how humans would perform bilingual lexicon induction: by referring to the external world.

Acknowledgments

DK is supported by EPSRC grant EP/I037512/1. IV is supported by the PARIS project (IWT-SBO 110067) and the PDM Kort postdoctoral fellowship from KU Leuven. SC is supported by ERC Starting Grant DisCoTex (306920) and EPSRC grant EP/I037512/1. We thank Marco Baroni for useful feedback and the anonymous reviewers for their helpful comments.

References

Eneko Agirre, Enrique Alfonseca, Keith B. Hall, Jana Kravalova, Marius Pasca, and Aitor Soroa. 2009. A study on similarity and relatedness using distributional and WordNet-based approaches. In *NAACL*, pages 19–27.

Lawrence W. Barsalou and Katja Wiemer-Hastings. 2005. Situating abstract concepts. In *Grounding cognition: The role of perception and action in memory, language, and thought*, pages 129–163.

Lawrence W. Barsalou. 2008. Grounded cognition. *Annual Review of Psychology*, 59(1):617–645.

Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc J. Van Gool. 2008. Speeded-up robust features (SURF). *Computer Vision and Image Understanding*, 110(3):346–359.

Shane Bergsma and Randy Goebel. 2011. Using visual information to predict lexical preference. In *RANLP*, pages 399–405.

Shane Bergsma and Benjamin Van Durme. 2011. Learning bilingual lexicons using the visual similarity of labeled web images. In *IJCAI*, pages 1764–1769.

Anna Bosch, Andrew Zisserman, and Xavier Muñoz. 2007. Image classification using random forests and ferns. In *ICCV*, pages 1–8.

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

Navneet Dalal and Bill Triggs. 2005. Histograms of oriented gradients for human detection. In *CVPR*, pages 886–893.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Fei-Fei Li. 2009. ImageNet: A large-scale hierarchical image database. In *CVPR*, pages 248–255.

Thomas Deselaers, Daniel Keysers, and Hermann Ney. 2008. Features for image retrieval: An experimental comparison. *Information Retrieval*, 11(2):77–107.

Xavier Driancourt and Léon Bottou. 1990. TDNN-extracted features. In *Neuro Nimes 90*.

Yansong Feng and Mirella Lapata. 2010. Visual information in semantic representation. In *NAACL*, pages 91–99.

Robert Fergus, Fei-Fei Li, Pietro Perona, and Andrew Zisserman. 2005. Learning object categories from Google's image search. In *ICCV*, pages 1816–1823.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2002. Placing Search in Context: The Concept Revisited. *ACM Transactions on Information Systems*, 20(1):116—131.

Andrea Frome, Gregory S. Corrado, Jonathon Shlens, Samy Bengio, Jeffrey Dean, Marc'Aurelio Ranzato, and Tomas Mikolov. 2013. Devise: A deep visual-semantic embedding model. In *NIPS*, pages 2121–2129.

Pascale Fung and Lo Yuen Yee. 1998. An IR approach for translating new words from nonparallel, comparable texts. In *ACL*, pages 414–420.

Éric Gaussier, Jean-Michel Renders, Irina Matveeva, Cyril Goutte, and Hervé Déjean. 2004. A geometric view on bilingual lexicon extraction from comparable corpora. In *ACL*, pages 526–533.

Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*.

Aria Haghighi, Percy Liang, Taylor Berg-Kirkpatrick, and Dan Klein. 2008. Learning bilingual lexicons from monolingual corpora. In *ACL*, pages 771–779.

Felix Hill and Anna Korhonen. 2014. Learning abstract concept embeddings from multi-modal data: Since you probably can't see what I mean. In *EMNLP*, pages 255–265.

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

Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross B. Girshick, Sergio Guadarrama, and Trevor Darrell. 2014. Caffe: Convolutional architecture for fast feature embedding. In *ACM Multimedia*, pages 675–678.

Douwe Kiela and Léon Bottou. 2014. Learning image embeddings using convolutional neural networks for improved multi-modal semantics. In *EMNLP*, pages 36–45.

Douwe Kiela, Felix Hill, Anna Korhonen, and Stephen Clark. 2014. Improving multi-modal representations using image dispersion: Why less is sometimes more. In *ACL*, pages 835–841.

Ryan Kiros, Ruslan Salakhutdinov, and Richard S. Zemel. 2014. Multimodal neural language models. In *ICML*, pages 595–603.

Philipp Koehn and Kevin Knight. 2002. Learning a translation lexicon from monolingual corpora. In *ULA'02 Workshop*, pages 9–16.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. ImageNet classification with deep convolutional neural networks. In *NIPS*, pages 1106–1114.

George Lakoff and Mark Johnson. 1999. *Philosophy in the flesh: The embodied mind and its challenge to Western thought*.

Victor Lavrenko, Martin Choquette, and W. Bruce Croft. 2002. Cross-lingual relevance models. In *SIGIR*, pages 175–182.

Angeliki Lazaridou, Elia Bruni, and Marco Baroni. 2014. Is this a wampimuk? Cross-modal mapping between distributional semantics and the visual world. In *ACL*, pages 1403–1414.

Chee Wee Leong and Rada Mihalcea. 2011. Going beyond text: A hybrid image-text approach for measuring word relatedness. In *IJCNLP*, pages 1403–1407.

Gina-Anne Levow, Douglas Oard, and Philip Resnik. 2005. Dictionary-based techniques for cross-language information retrieval. *Information Processing & Management*, 41:523 – 547, 2005/05//.

Dekang Lin, Kenneth Ward Church, Heng Ji, Satoshi Sekine, David Yarowsky, Shane Bergsma, Kailash Patil, Emily Pitler, Rachel Lathbury, Vikram Rao, Kapil Dalwani, and Sushant Narsale. 2010. New tools for Web-scale N-grams. In *LREC*, pages 2221–2227.

Xiaodong Liu, Kevin Duh, and Yuji Matsumoto. 2013. Topic models + word alignment = A flexible framework for extracting bilingual dictionary from comparable corpus. In *CoNLL*, pages 212–221.

Max M. Louwerse. 2008. Symbol interdependency in symbolic and embodied cognition. *Topics in Cognitive Science*, 59(1):617–645.

David G. Lowe. 2004. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110.

David M. Mimno, Hanna M. Wallach, Jason Naradowsky, David A. Smith, and Andrew McCallum. 2009. Polylingual topic models. In *EMNLP*, pages 880–889.

Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *ICML*, pages 807–814.

Gonzalo Navarro. 2001. A guided tour to approximate string matching. *ACM Computing Surveys*, 33(1):31–88.

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

Maxime Oquab, Léon Bottou, Ivan Laptev, and Josef Sivic. 2014. Learning and transferring mid-level image representations using convolutional neural networks. In *CVPR*, pages 1717–1724.

Reinhard Rapp. 1999. Automatic identification of word translations from unrelated English and German corpora. In *ACL*.

Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. 2014. CNN features off-the-shelf: an astounding baseline for recognition. *CoRR*, abs/1403.6382.

Stephen Roller and Sabine Schulte im Walde. 2013. A multimodal LDA model integrating textual, cognitive and visual modalities. In *EMNLP*, pages 1146–1157.

Charles Schafer and David Yarowsky. 2002. Inducing translation lexicons via diverse similarity measures and bridge languages. In *CoNLL*, pages 1–7.

Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. 2014. Learning semantic representations using convolutional neural networks for Web search. In *WWW*, pages 373–374.

Daphna Shezaf and Ari Rappoport. 2010. Bilingual lexicon generation using non-aligned signatures. In *ACL*, pages 98–107.

Carina Silberer and Mirella Lapata. 2012. Grounded models of semantic representation. In *EMNLP*, pages 1423–1433.

Carina Silberer and Mirella Lapata. 2014. Learning grounded meaning representations with autoencoders. In *ACL*, pages 721–732.

Carina Silberer, Vittorio Ferrari, and Mirella Lapata. 2013. Models of semantic representation with visual attributes. In *ACL*, pages 572–582.

Josef Sivic and Andrew Zisserman. 2003. Video google: A text retrieval approach to object matching in videos. In *ICCV*, pages 1470–1477.

Richard Socher, Andrej Karpathy, Quoc V. Le, Christopher D. Manning, and Andrew Y. Ng. 2014. Grounded compositional semantics for finding and describing images with sentences. *Transactions of ACL*, 2:207–218.

Nitish Srivastava and Ruslan Salakhutdinov. 2014. Multimodal learning with deep Boltzmann machines. *Journal of Machine Learning Research*, 15(1):2949–2980.

Akihiro Tamura, Taro Watanabe, and Eiichiro Sumita. 2012. Bilingual lexicon extraction from comparable corpora using label propagation. In *EMNLP*, pages 24–36.

Luis von Ahn and Laura Dabbish. 2004. Labeling images with a computer game. In *CHI*, pages 319–326.

Ivan Vulić and Marie-Francine Moens. 2013a. Cross-lingual semantic similarity of words as the similarity of their semantic word responses. In *NAACL*, pages 106–116.

Ivan Vulić and Marie-Francine Moens. 2013b. A study on bootstrapping bilingual vector spaces from non-parallel data (and nothing else). In *EMNLP*, pages 1613–1624.

Ivan Vulić, Wim De Smet, and Marie-Francine Moens. 2011. Identifying word translations from comparable corpora using latent topic models. In *ACL*, pages 479–484.

Pengcheng Wu, Steven C. H. Hoi, Hao Xia, Peilin Zhao, Dayong Wang, and Chunyan Miao. 2013. Online multimodal deep similarity learning with application to image retrieval. In *ACM Multimedia*, pages 153–162.

Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. 2014. How transferable are features in deep neural networks? In *NIPS*, pages 3320–3328.

Cross-Lingual Sentiment Analysis using modified BRAE

Sarthak Jain
Department of Computer Engineering
Delhi Technological University
DL, India
successar@gmail.com

Shashank Batra
Department of Computer Engineering
Indian Institute of technology, Delhi
DL, India
shashankg@gmail.com

Abstract

Cross-Lingual Learning provides a mechanism to adapt NLP tools available for label rich languages to achieve similar tasks for label-scarce languages. An efficient cross-lingual tool significantly reduces the cost and effort required to manually annotate data. In this paper, we use the Recursive Autoencoder architecture to develop a Cross Lingual Sentiment Analysis (CLSA) tool using sentence aligned corpora between a pair of resource rich (English) and resource poor (Hindi) language. The system is based on the assumption that semantic similarity between different phrases also implies sentiment similarity in majority of sentences. The resulting system is then analyzed on a newly developed Movie Reviews Dataset in Hindi with labels given on a rating scale and compare performance of our system against existing systems. It is shown that our approach significantly outperforms state of the art systems for Sentiment Analysis, especially when labeled data is scarce.

1 Introduction

Sentiment Analysis is a NLP task that deals with extraction of opinion from a piece of text on a topic. This is used by a large number of advertising and media companies to get a sense of public opinion from their reviews. The ever increasing user generated content has always been motivation for sentiment analysis research, but majority of work has been done for English Language. However, in recent years, there has been emergence of increasing amount of text in Hindi on electronic sources but NLP Frameworks to process this data is sadly miniscule. A major cause for this is the lack of annotated datasets in Indian Languages.

One solution is to create cross lingual tools between a resource rich and resource poor language that exploit large amounts of unlabeled data and sentence aligned corpora that are widely available on web through bilingual newspapers, magazines, etc. Many different approaches have been identified to perform Cross Lingual Tasks but they depend on the presence of MT-System or Bilingual Dictionaries between the source and target language.

In this paper, we use Bilingually Constrained Recursive Auto-encoder (BRAE) given by (Zhang et al., 2014) to perform Cross Lingual sentiment analysis. Major Contributions of this paper are as follows: First, We develop a new Rating scale based Movie Review Dataset for Hindi. Second, a general framework to perform Cross Lingual Classification tasks is developed by modifying the architecture and training procedure for BRAE model. This model exploits the fact that phrases in two languages, that share same semantic meaning, can be used to learn language independent semantic vector representations. These embeddings can further be fine-tuned using labeled dataset in English to capture enough class information regarding Resource poor language. We train the resultant framework on English-Hindi Language pair and evaluate it against state of the art SA systems on existing and newly developed dataset.

2 Related Work

2.1 Sentiment Analysis in Hindi

In recent years, there have been emergence of works on Sentiment Analysis (both monolingual and cross-lingual) for Hindi. (Joshi et al., 2010) provided a comparative analysis of Unigram based In-language, MT based Cross Lingual and WordNet based Sentiment classifier, achieving highest accuracy of 78.14%. (Mittal et al., 2013) described a system based on Hindi SentiWordNet for assign-

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 159–168,
Lisbon, Portugal, 17-21 September 2015.

ing positive/negative polarity to movie reviews. In this approach, overall semantic orientation of the review document was determined by aggregating the polarity values of the words in the document assigned using the WordNet. They also included explicit rules for handling Negation and Discourse relations during preprocessing in their model to achieve better accuracies.

For Languages where labeled data is not present, approaches based on cross-lingual sentiment analysis are used. Usually, such methods need intermediary machine translation system (Wan et al., 2011; Brooke et al., 2009) or a bilingual dictionary (Ghorbel and Jacot, 2011; Lu et al., 2011) to bridge the language gap. Given the subtle and different ways in which sentiments can be expressed and the cultural diversity amongst different languages, an MT system has to be of a superior quality to perform well(Balamurali et al., 2012).

(Balamurali et al., 2012) present an alternative approach to Cross Lingual Sentiment Analysis (CLSA) using WordNet senses as features for supervised sentiment classification. A document in Resource Poor Language was tested for polarity through a classifier trained on sense marked and polarity labeled corpora in Resource rich language. The crux of the idea was to use the linked WordNets of two languages to bridge the language gap.

Recently, (Popat et al., 2013) describes a Cross Lingual Clustering based SA System. In this approach, features were generated using syntagmatic property based word clusters created from unlabeled monolingual corpora, thereby eliminating the need for Bilingual Dictionaries. These features were then used to train a linear SVM to predict positive or negative polarity on a tourism review dataset.

2.2 Autoencoders in NLP Tasks

Autoencoders are neural networks that learn a low dimensional vector representation of fixed-size inputs such as image segments or bag-of-word representations of documents. They can be used to efficiently learn feature encodings that are useful for classification. The Autoencoders were first applied in a recursive setting by Pollack (1990) in recursive auto-associative memories (RAAMs). However, RAAMs needed fixed recursive data structures to learn vector representations, whereas RAE given by (Socher et al., 2011) builds recursive data structure using a greedy algorithm. The RAE can be pre-trained with an unsupervised algorithm and then fine-tuned according to the label of the phrase, such as the syntactic category in parsing(Socher et al., 2013), the polarity in sentiment analysis, etc. The learned structures are not necessarily syntactically accurate but can capture more of the semantic information in the word vectors.

3 BRAE Framework

(Zhang et al., 2014) used the RAE along with a Bilingually Constrained Model to simultaneously learn phrase embeddings for two languages in semantic vector space. The core idea behind BRAE is that a phrase and its correct translation should share the same semantic meaning. Thus, they can supervise each other to learn their semantic phrase embeddings. Similarly, non-translation pairs should have different semantic meanings, and this information can also be used to guide learning semantic phrase embeddings. In this method, a standard recursive autoencoder (RAE) pre-trains the phrase embedding with an unsupervised algorithm by greedily minimizing the reconstruction error (Socher et al., 2011), while the bilingually-constrained model learns to finetune the phrase embedding by minimizing the semantic distance between translation equivalents and maximizing the semantic distance between non-translation pairs.

In this section, We will briefly present the structure and training algorithm for BRAE model. After that, we show how this model can be adapted to perform CLSA.

3.1 Recursive Auto-encoder Framework

In this model, each word w_k in the vocabulary V of given language corresponds to a vector $x_k \in \mathbb{R}^n$ and stacked into a single word embedding matrix $L \in \mathbb{R}^{n \times |V|}$. This matrix is learned using DNN (Collobert and Weston, 2008; Mikolov et al., 2013) and serves as input to further stages of RAE.

Using this matrix, a phrase $(w_1 w_2 \dots w_m)$ is first projected into a list of vectors $(x_1, x_2, \dots x_m)$. The RAE learns the vector representation of the phrase by combining two children vectors recursively in a bottom-up manner. For two children $c_1 = x_1, c_2 = x_2$, the auto-encoder computes the parent vector y_1:

$$y_1 = f(W^{(1)}[c_1; c_2] + b^{(1)}); y_1 \in \mathbb{R}^n \quad (1)$$

To assess how well the parent vector represents its children, the auto-encoder reconstructs the chil-

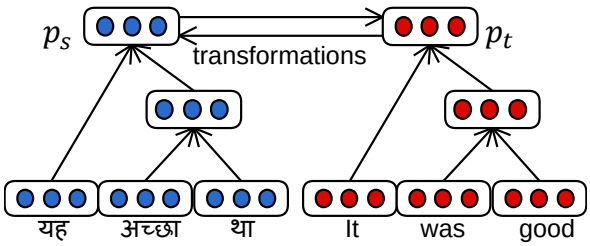


Figure 1: An illustration of BRAE structure

dren :

$$[c_1'; c_2'] = W^{(2)}p + b^{(2)} \quad (2)$$

and tries to minimize the reconstruction error (Euclidean Distance) $E_{rec}([c_1; c_2])$ between the inputs $[c_1; c_2]$ and their reconstructions $[c_1'; c_2']$.

Given y_1, Eq.1 is used again to compute y_2 by setting the children to be $[c_1; c_2] = [y_1; x_3]$. The same auto-encoder is re-used until the vector of the whole phrase is generated. For unsupervised phrase embedding, the sum of reconstruction errors at each node in binary tree y is minimized:

$$E_{rec}(x; \theta) = \arg\min_{y \in A(x)} \sum_{k \in y} E_{rec}([c_1; c_2]_k) \quad (3)$$

Where $A(x)$ denotes all the possible binary trees that can be built from inputs x. A greedy algorithm is used to generate the optimal binary tree y^*. The parameters $\theta^{rec} = (\theta^{(1)}, \theta^{(2)})$ are optimized over all the phrases in the training data. For further details, please refer (Socher et al., 2011)

3.2 Semantic Error

The BRAE model jointly learns two RAEs for source language L_S and target language L_T. Each RAE learn semantic vector representation p_s and p_t of phrases s and t respectively in translation-equivalent phrase pair (s, t) in bilingual corpora (shown in Fig.1). The transformation between the two is defined by:

$$p_t' = f(W_s^t p_s + b_s^t), p_s' = f(W_t^s p_t + b_t^s) \quad (4)$$

where $\theta_s^t = (W_s^t, b_s^t), \theta_t^s = (W_t^s, b_t^s)$ are new parameters introduced.

The semantic error between learned vector representations p_s and p_t is calculated as :

$$E_{sem}(s, t; \theta) = E_{sem}^*(t|s; \theta_t^s) + E_{sem}^*(s|t; \theta_s^t) \quad (5)$$

where $E_{sem}^*(s|t; \theta_t^s)$ is the semantic distance of p_s given p_t and vice versa. To calculate it, we first calculate Euclidean distance between original p_t and transformation p_t' as $D_{sem}(s|t, \theta_s^t) = \frac{1}{2}\|p_t - p_t'\|^2$. The max-semantic-margin distance between them is then defined as

$$E_{sem}^*(s|t, \theta_s^t) = max\{0, D_{sem}(s|t, \theta_s^t) - D_{sem}(s|t', \theta_s^t) + 1\} \quad (6)$$

where we simultaneously minimize the distance between translation pairs and maximized between non-translation pairs. Here t' in non-translation pair (s, t') is obtained by replacing the words in t with randomly chosen target language words. We calculate the $E_{sem}^*(t|s; \theta_t^s)$ in similar manner.

3.3 BRAE Objective Function

Thus, for the phrase pair (s, t), the joint error becomes:

$$\begin{aligned} E(s, t, \theta) &= E(s|t, \theta) + E(t|s, \theta) \\ E(s|t, \theta) &= \alpha E_{rec}(s; \theta_s^{rec}) + (1 - \alpha) E_{sem}^*(s|t, \theta_s^t)) \\ E(t|s, \theta) &= \alpha E_{rec}(t; \theta_t^{rec}) + (1 - \alpha) E_{sem}^*(t|s, \theta_t^s)) \end{aligned} \quad (7)$$

The hyper-parameter α weighs the reconstruction and semantic errors. The above equation indicates that the Parameter sets $\theta_t = (\theta_t^s, \theta_t^{rec})$ and $\theta_s = (\theta_s^t, \theta_s^{rec})$ on each side respectively can be optimized independently as long as the phrase representation of other side is given to compute semantic error.

The final BRAE objective over the phrase pairs training set (S, T) becomes:

$$J_{BRAE} = \frac{1}{N} \sum_{(s,t) \in (S,T)} E(s, t; \theta) + \frac{\lambda_{BRAE}}{2} \|\theta\|^2 \quad (8)$$

3.4 Unsupervised Training of BRAE

The word embedding matrices L_s and L_t are pre-trained using unlabeled monolingual data with Word2Vec toolkit (Mikolov et al., 2013). All other parameters are initialized randomly. We use SGD algorithm for parameter optimization. For full gradient calculations for each parameter set, please see (Zhang et al., 2014).

1. RAE Training Phase: Apply RAE Framework (Sec. 3.1) to pre-train the source and target phrase representations p_s and p_t respectively by optimizing θ_s^{rec} and θ_t^{rec} using unlabeled monolingual datasets.

2. Cross-Training Phase: Use target-side phrase representation p_t to update the source-side

parameters θ_s and obtain source-side phrase representation p'_s, and vice-versa for p_s. Calculate the joint error over the bilingual training corpus. On reaching a local minima or predefined no. of iterations (30 in our case), terminate this phase, otherwise set $p_s = p'_s$, $p_t = p'_t$, and repeat.

4 Adapting Model for Classifying Sentiments

At the end of previous Training procedure, we obtain high quality phrase embeddings in both source and target language and transformation function between them. We now extend that model to perform cross lingual supervised tasks, specifically CLSA.

To achieve this, we need to modify the learned semantic phrase embeddings such that they can capture information about sentiment. Since we only use monolingual labeled datasets from this point onwards, the supervised learning phases will occur independently for each RAE as we do not have any "phrase pairs" now. Thus, the new semantic vector space generated for word and phrase embeddings may no longer be in sync with their corresponding transformations.

We propose following modifications to the system to deal with this problem. Let L_S and L_T represent Resource rich and Resource poor language respectively in above model.

Modifications in architecture: We first include a softmax (σ) layer on top of each parent node in RAE for L_S to predict a K-dimensional multinomial distribution over the set of output classes defined by the task (e.g : polarity, Ratings).

$$d(p; \theta^{ce}) = \sigma(W^{ce} p) \tag{9}$$

Given this layer, we calculate cross entropy error $E_{ce}(p_k, t, W_{ce})$ generated for node p_k in binary tree, where t is target multinomial distribution or one-hot binary vector for target label. We use this layer to capture and predict actual sentiment information about the data in both L_S and L_T (described in next section). We show a node in modified architecture in Fig.2.

Penalty for Movement in Semantic Vector space: During subsequent training phases, we include the euclidean norm of the difference between the original and new phrase embeddings as penalty in reconstruction error at each node of the tree.

$$E^*_{rec}([c_1; c_2]; \theta) = E_{rec}([c_1; c_2]; \theta) + \frac{\lambda_p}{2} \|p - p^*\|^2 \tag{10}$$

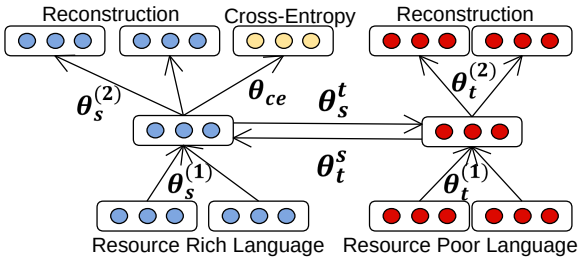


Figure 2: An illustration of BRAE segment with Cross Entropy layer

Here p is the phrase representation we get during forward propagation of current training iteration and p^* is the representation we get if we apply the parameters obtained at the end of the Cross training phase to children $[c_1; c_2]$ of that node. The reason to do this is twofold.

First, during supervised training, the error will back propagate through RAEs for both languages affecting their respective weights matrices and word embeddings. This will modify the semantic representation of phrases captured during previous phases of training procedure and adversely affect the transformations derived from them. Therefore we need to include some procedure such that the transformation information learned during Cross-training phase is not lost.

Secondly, we observe that the information about the semantic similarity of a word or phrase also implies sentiment similarity between the two. That is when dealing with bilingual data, words or phrases that appear near each other in semantic space typically represent common sentiment information and we want our model to create a decision boundary around these vectors instead of modifying them too much.

Disconnecting the RAEs: We fix the transformation weights between the two RAEs, i.e. in subsequent training steps the transformation weights(θ^t_s, θ^s_t) are not modified but rather pass the back propagated error as it is to previous layers. We observed that on optimizing the objective along with the penalty term, the transformation weights are preserved between new semantic/sentiment vector spaces, resulting in slightly degraded performance, but were still able to preserve enough information about the semantic structure of two languages.Also, it reinforced the penalty imposed on the movement of phrase embeddings in semantic vector space.On the other hand, if the weights were allowed to be updated,

the accuracies were affected severely as information learned during previous phases was lost and the weights were not been able to capture enough information about the modified phrase embeddings and generalize well on test phrases not encountered in labeled training set of Resource Scarce Language.

4.1 Supervised Training Phases

We now explain supervised training procedure using only monolingual labeled data for each language. These training phases occur at the end of BRAE training. In each training phase, we use SGD algorithm to perform parameter optimization.

4.1.1 Phase I : Resource Rich language

In this phase, we only modify the parameters of RAE_{L_S}, i.e. θ_s^{rec} and θ_{ce} by optimizing following objective over (sentence, label) pairs (x, t) in its labeled corpus.

$$J_S = \frac{1}{N} \sum_{(x,t)} E(x, t; \theta) + \frac{\lambda_S}{2} \|\theta\|^2 \qquad (11)$$

where $E(x, t; \theta)$ is the sum over the errors obtained at each node of the tree that is constructed by the greedy RAE:

$$E(x, t; \theta) = \sum_{k \in RAE_{L_S}(x)} \kappa E^*_{rec}([c_1; c_2]_k; \theta_s) + (1 - \kappa) E_{ce}(p_k, t; \theta_{ce}) \qquad (12)$$

To compute this gradient, we first greedily construct all trees and then derivatives for these trees are computed efficiently via back-propagation through structure (Goller and Kuchler, 1996). The gradient for our new reconstruction function (Eq. 10) w.r.t to p at a given node is calculated as

$$\frac{\partial E^*_{rec}}{\partial p} = \frac{\partial E_{rec}}{\partial p} + \lambda_p (p - p^*) \qquad (13)$$

The first term $\frac{\partial E_{rec}}{\partial p}$ is calculated as in standard RAE model. The partial derivative in above equation is used to compute parameter gradients in standard back-propagation algorithm.

4.1.2 Phase II : Resource Poor Language

In this phase, we modify the parameters of RAE_{L_T} and θ_{ce} by optimizing Objective J_T over (sentence, label) pairs (x, t) in labeled corpus for L_T (much smaller than that for L_S). The equation for J_T is similar to Eq.11 and Eq.12 but with θ_t and η as parameters instead of θ_s and κ respectively.

Since cross-entropy layer is only associated with L_S, we need to traverse the transformation parameters to obtain sentiment distribution for each node (green path in Fig.2). That is, we first transform p_t to source side phrase p'_s and then apply the cross entropy weights to it.

$$d(p_t, \theta_{ce}) = \sigma(W^{ce}.f(W^t_s p_t + b^t_s)) \qquad (14)$$

We use the similar back-propagation through structure approach for gradient calculation in Phase I. During back propagation, 1) we do not update the transformation weights, 2) we transfer error signals during back-propagation from Cross-entropy layer to $\theta^{(1)}_t$ as if the transformation was an additional layer in the network.

4.1.3 Predicting overall sentiment

To predict overall sentiment associated with the sentence in L_T, we use the phrase embeddings p_t of the top layer of the RAE_{L_T} and it transformation p'_s. Together, we train a softmax regression classifier on concatenation of these two vector using weight matrix $W \in \mathbb{R}^{K \times 2n}$

5 Experimental Work

We perform experiments on two kind of sentiment analysis systems : (1) that gives +ve/-ve polarity to each review and (2) assigns ratings in range 1 - 4 to each review.

5.1 External Datasets Used

For pre-training the word embeddings and RAE Training, we used HindMonoCorp 0.5(Bojar et al., 2014) with 44.49M sentences (787M Tokens) and English Gigaword Corpus.

For Cross Training, we used the bilingual sentence-aligned data from HindEnCorp[1] (Bojar et al., 2014) with 273.9k sentence pairs (3.76M English, 3.88M Hindi Tokens). This dataset contains sentence pair obtained from Bilingual New Articles, Wikipedia entries, Automated Translations, etc. Training and Validation division is 70% and 30% for all above datasets.

In Supervised Phase I, we used IMDB11 dataset available at *http://ai.stanford.edu/~amaas/data/sentiment/* and first used by (Maas et al., 2011) for +ve/-ve

[1] http://ufal.mff.cuni.cz/hindencorp

system containing 25000 +ve and 25000 -ve movie reviews.

For 4-ratings system, we use Rotten Tomatoes Review dataset (scale dataset v1.0) found at *http://www.cs.cornell.edu/People/pabo/movie-review-data*. The dataset is divided into four author-specific corpora, containing 1770, 902, 1307, and 1027 documents and each document has accompanying 4-Ratings ({0, 1, 2, 3}) label.

5.2 Rating Based Hindi Movie Review (RHMR) Dataset

We crawled the Hindi Movie Reviews Website[2] to obtain 2945 movie reviews. Each Movie Review on this site is assigned rating in range 1 to 4 by at least three reviewers. We first discard reviews that whose sum of pairwise difference of ratings is greater than two. The final rating for each review is calculated by taking the average of the ratings and rounding up to nearest integer. The fraction of Reviews obtained in ratings 1-4 are [0.20, 0.25, 0.35, 0.20] respectively. Average length of reviews is 84 words. For +ve/-ve polarity based system, we group the reviews with ratings {1, 2} as negative and {3, 4} as positive.

5.3 Experimental Settings

We used following Baselines for Sentiment Analysis in Hindi :

Majority class: Assign the most frequent class in the training set (Rating:3 / Polarity:+ve)

Bag-of-words: Softmax regression on Binary Bag-of-words

We also compare our system with state of the art Monolingual and Cross Lingual System for Sentiment Analysis in Hindi as described by (Popat et al., 2013) using the same experimental setup. The best systems in each category given by them are as below:

WordNet Based: Using Hindi-SentiWordNet[3], each word in a review was mapped to corresponding synset identifiers. These identifiers were used as features for creating sentiment classifiers based on Binary/Multiclass SVM trained on bag of words representation using libSVM library.

Cross Lingual (XL) Clustering Based: Here, joint clustering was performed on unlabeled bilingual corpora which maximizes the joint likelihood of monolingual and cross-lingual factors.. For details, please refer the work of (Popat et al., 2013). Each word in a review was then mapped to its cluster identifier and used as features in an SVM.

Our approaches

Basic RAE: We use the Semi-Supervised RAE based classification where we first trained a standard RAE using Hindi monolingual corpora, then applied supervised training procedure as described in (Socher et al., 2011). This approach doesn't use bilingual corpora, but is dependent on amount of labeled data in Hindi.

BRAE-U: We neither include penalty term, nor fix the transformations weights in our proposed system.

BRAE-P: We only include the penalty term but allow the transformation weights to be modified in proposed system.

BRAE-F: We add the penalty term and fix the transformation weights during back propagation in proposed system.

5.4 Experimental Setup

We combined the text data from all English Datasets (English Gigaword + HindEnCorp English Portion + IBMD11 + Scale Dataset) described above to train the word embeddings using Word2Vec toolkit and RAE. Similarly, we combined text data from all Hindi Datasets (HindMonoCorp + HindiEnCorp Hindi Portion + RHMR) to train word embeddings and RAE for Hindi.

We used MOSES Toolkit (Koehn et al., 2007) to obtain high quality bilingual phrase pairs from HindEnCorp to train our BRAE model. After removing the duplicates, 364.3k bilingual phrase pairs were obtained with lengths ranging from 1-6, since bigger phrases reduced the performance of the system in terms of Joint Error of BRAE model.

We randomly split our RHMR dataset into 10 segments and report the average of 10-fold cross validation accuracies for each setting for both Ratings and Polarity classifiers.

We also report 5-fold cross validation accuracy on Standard Movie Reviews Dataset (hereby referred as SMRD) given by (Joshi et al., 2010) which contains 125 +ve and 125 -ve reviews in Hindi. The dataset can be obtained at http://www.cfilt.iitb.ac.in/Resources.html.

Since this project is about reducing dependence on annotated datasets, we experiment on how accuracy varies with labeled training dataset (RHMR) size. To perform this, we train our model

[2]http://hindi.webdunia.com/bollywood-movie-review/

[3]http://www.cfilt.iitb.ac.in/

in 10% increments (150 examples) of training set size (each class sampled in proportion of original set). For each size, we sample the data 10 times with replacement and trained the model. For each sample, we calculated 10-fold cross validation accuracy as described above. Final accuracy for each size was calculated by averaging the accuracies obtained on all 10 samples. Similar kind of evaluation is done for all other Baselines explored.

In subsequent section, the word 'significant' implies that the results were statistically significant ($p < 0.05$) with paired T-test

5.5 BRAE Hyper Parameters

We empirically set the learning rate as 0.05. The word vector dimension was selected as 80 from set $[40, 60, 80, 100, 120]$ using Cross Validation. We used joint error of BRAE model to select α as 0.2 from range $[0.05, 0.5]$ in steps of 0.05. Also, λ_L was set as 0.001 for DNN trained for word embedding and λ_{BRAE} as 0.0001.

For semi-supervised phases , we used 5-fold cross validation on training set to select κ and η in range $[0.0, 1.0]$ in steps of 0.05 with optimal value obtained at $\kappa = 0.2$ and $\eta = 0.35$. Parameter λ_p was selected as 0.01 , λ_S as 0.1 and λ_T as 0.04 after selection in range $[0.0, 1.0]$ in steps of 0.01.

5.6 Results

Dataset	RHMR		SMRD
Classifier	**Ratings**	**Polarity**	**Polarity**
Majority class	35.19	51.83	52.34
Bag-of-Words	51.98	62.52	68.47
WordNet based	55.47	67.29	75.5
XL Clustering	72.34	84.46	84.71
Basic RAE	75.53	79.31	81.06
BRAE-U	76.01	82.66	84.83
BRAE-P	79.70	84.85	87.00
BRAE-F	81.22	90.50	90.21

Table 1: Accuracies obtained for various Experimental Settings. Model are trained on complete labeled training datasets

Table 1 present the results obtained for both ratings based and polarity classifier on RHMR and MRD Dataset. Our model gives significantly better performance for ratings based classification than any other baseline system currently used for SA in Hindi. The margin of accuracy obtained against next best classifier is about 8%. Also, for +ve/-ve polarity classifier, the accuracy showed an improvement of 6% over next highest baseline.

$A \downarrow / P \rightarrow$	P-1	P-2	P-3	P-4
A-1	83.19	15.28	1.53	0.00
A-2	12.23	82.20	5.57	0.00
A-3	0.00	9.03	81.26	9.71
A-4	0.00	1.87	19.69	78.44
F1-score	0.83	0.78	0.82	0.80

Table 2: Confusion Matrix for Ratings by BRAE-F, **Across**: Predicted Rating, **Downward:** Actual Rating

In Table 2, we calculate the confusion matrix for our model(BRAE-F) for the 4-Ratings case. Value in a cell (A_i, P_j) represents the percentage of examples in actual rating class i that are predicted as rating j. We also show the F1 score calculated for each individual rating class. It clearly shows that our model has low variation in F1-scores and thereby its performance among various rating classes.

In Fig. 3, we show the variation in accuracy of the classifiers with amount of sentiment labeled Training data used. We note that our approach consistently outperforms the explored baselines at all dataset sizes. Also, our model was able to attain accuracy comparable to other baselines at about 50% less labeled data showing its strength in exploiting the unlabeled resources.

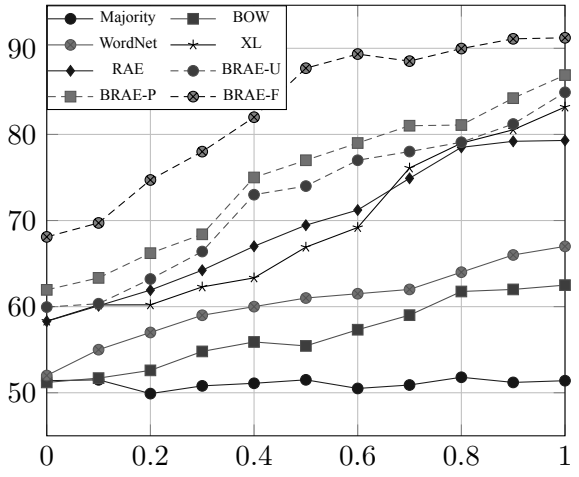


Figure 3: Variation of Accuracy (+ve/-ve Polarity) with Size of labeled Dataset(Hindi), **x-axis:** Fraction of Dataset Used, **y-axis:** %age Accuracy Obtained

We also experiment with variation of accuracies

New Word/Phrase	Similar Words/Phrases	Sentiment label
depressing निराशाजनक	gloomy उदास discouraging निराशात्मक	Rating : 1 Polarity : -ve
was painful दर्दनाक था	was difficult कठिन था was bad खराब था	Rating : 2 Polarity : -ve
should be awarded सम्मानित किया जाना चाहिए	was appreciated सराहना की गई will get accolades वाहवाही मिलना चाहिए	Rating : 4 Polarity : +ve
public won't come लोग नहीं आएगा	no one will come कोई नहीं आएगा viewers won't come दर्शक नहीं आएगा	Rating : 1 Polarity : -ve

Table 3: Semantically similar phrases obtained for new phrases and their assigned label

with amount of Unlabeled Bilingual Training Data used for Cross Lingual models explored. Again we increase size of bilingual dataset in 10% increments and calculate the accuracy as described previously. In Fig. 4, we observed that performance of the proposed approach steadily increases with amount of data added, yet even at about 50000 (20%) phrase pairs, our model produces remarkable gains in accuracy.

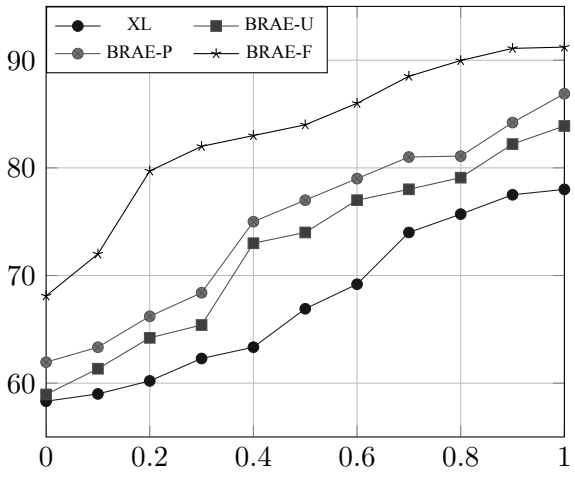


Figure 4: Variation of Accuracy (+ve/-ve polarity) with Size of Unlabeled Bilingual Corpora, **x-axis:** Fraction of Training Data Used, **y-axis:** %age Accuracy Obtained

We also observed that the model which restricts modification to transformation weights during supervised phase II does better than the one which allows the modification at all dataset sizes. This result appears to be counterintuitive to normal operation of neural network based models, but supports our hypothesis as explained in previous sections.

5.7 Performance and Error Analysis

Analysis on the test results showed that the major advantage given by our model occurs due to presence of unknown words (i.e.words not present in labeled dataset) in test data. Since we restricted the movement in semantic vector space, our model was able to infer the sentiment for a unknown word/phrase by comparing it with semantically similar words/phrases. In Table 3, we extracted the Top-2 semantically similar phrases in training set for small new phrases and sentiment labeled assigned to them by our model (the phrases are manually translated from Hindi for reader's understanding). As we can see, our model was able to extract grammatically correct phrases with similar semantic nature as given phrase and assign correct sentiment label to it.

Secondly, We found that our model was able to correctly infer word sense for polysemous words that adversely affected the quality of sentiment classifiers in our baselines. This eliminates the need for manually constructed fine grained lexical resource like WordNets and development of automated annotation resources. For example, to a phrase like *"Her acting of a schizophrenic mother made our hearts **weep"***, the baselines classifiers assigned negative polarity due to presence of words like 'weep', yet our model was correctly able to predict positive polarity and assigned it a rating of 3.

Error Analysis of test results showed that errors made by our model can be classified in two major categories :

1) A review may only give description of the object in question (in our case , the description of the film) without actually presenting any individual sentiments about it or it may express conflicting sentiments about two different aspects about the same object. This presents difficulty in assign-

ing a single polarity/rating to the review.

2) Presence of subtle contextual references affected the quality of predictions made by our classifier. For example, sentence like *"His poor acting generally destroys a movie, but this time it didn't"* got a rating of 2 due to presence of phrase with negative sense (here the phrase doesn't have ambiguous sense), yet the actual sentiment expressed is positive due to temporal dependence and generalization. Also, *"This movie made his last one looked good"* makes a reference to entities external to the review, which again forces our model to make wrong prediction of rating 3.

Analyzing these aspects and making correct predictions on such examples needs further work.

6 Conclusion and Future Work

This study focused on developing a Cross Lingual Supervised Classifier based on Bilingually Constrained Recursive Autoencoder. To achieve this, our model first learns phrase embeddings for two languages using Standard RAE, then fine tune these embeddings using Cross Training procedure. After imposing certain restrictions on these embeddings, we perform supervised training using labeled sentiment corpora in English and a much smaller one in Hindi to get the final classifier.

The experimental work showed that our model was remarkably effective for classification of Movie Reviews in Hindi on a rating scale and predicting polarity using least amount of data to achieve same accuracy as other systems explored. Moreover it reduces the need for MT System or lexical resources like Linked WordNets since the performance is not degraded too much even when we lack large quantity of labeled data.

In Future, we hope to 1) extend this system to learn phrase representations among multiple languages simultaneously, 2) apply this framework to other cross Lingual Tasks such as Paraphrase detection, Question Answering, Aspect Based Opinion Mining etc and 3) Learning different weight matrices at different nodes to capture complex relations between words and phrases.

References

Balamurali, Aditya Joshi, and Pushpak Bhattacharyya. 2012. Cross-lingual sentiment analysis for Indian languages using linked wordnets. In *Proceedings of COLING 2012: Posters*, pages 73--82. The COLING 2012 Organizing Committee.

Ondřej Bojar, Vojtěch Diatka, Pavel Rychlý, Pavel Straňák, Vít Suchomel, Aleš Tamchyna, and Daniel Zeman. 2014. HindEnCorp - Hindi-English and Hindi-only Corpus for Machine Translation. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*. European Language Resources Association (ELRA).

Julian Brooke, Milan Tofiloski, and Maite Taboada. 2009. Cross-linguistic sentiment analysis: From english to spanish. In *RANLP*, pages 50--54.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160--167. ACM.

Hatem Ghorbel and David Jacot. 2011. Further experiments in sentiment analysis of french movie reviews. In *Advances in Intelligent Web Mastering--3*, pages 19--28. Springer.

Christoph Goller and Andreas Kuchler. 1996. Learning task-dependent distributed representations by backpropagation through structure. In *Neural Networks, 1996., IEEE International Conference on*, volume 1, pages 347--352. IEEE.

Aditya Joshi, AR Balamurali, and Pushpak Bhattacharyya. 2010. A fall-back strategy for sentiment analysis in hindi: a case study. *Proceedings of the 8th ICON*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th annual meeting of the ACL on interactive poster and demonstration sessions*, pages 177--180. Association for Computational Linguistics.

Bin Lu, Chenhao Tan, Claire Cardie, and Benjamin K Tsou. 2011. Joint bilingual sentiment classification with unlabeled parallel corpora. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 320--330. Association for Computational Linguistics.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142--150. Association for Computational Linguistics.

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

Namita Mittal, Basant Agarwal, Garvit Chouhan, Nitin Bania, and Prateek Pareek. 2013. Sentiment analysis of hindi review based on negation and discourse relation. In *proceedings of International Joint Conference on Natural Language Processing*, pages 45--50.

Kashyap Popat, Balamurali A.R, Pushpak Bhattacharyya, and Gholamreza Haffari. 2013. The haves and the have-nots: Leveraging unlabelled corpora for sentiment analysis. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 412--422. Association for Computational Linguistics.

Richard Socher, Jeffrey Pennington, Eric H Huang, Andrew Y Ng, and Christopher D Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 151--161. Association for Computational Linguistics.

Richard Socher, John Bauer, Christopher D Manning, and Andrew Y Ng. 2013. Parsing with compositional vector grammars. In *In Proceedings of the ACL conference*. Citeseer.

Chang Wan, Rong Pan, and Jiefei Li. 2011. Bi-weighting domain adaptation for cross-language text classification. In *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, volume 22, page 1535.

Jiajun Zhang, Shujie Liu, Mu Li, Ming Zhou, and Chengqing Zong. 2014. Bilingually-constrained phrase embeddings for machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 111--121. Association for Computational Linguistics.

Monotone Submodularity in Opinion Summaries

Jayanth Jayanth
IIT Bombay
jayanthjaiswal10
@cse.iitb.ac.in

Jayaprakash Sundararaj
IIT Bombay
osjayaprakash
@gmail.com

Pushpak Bhattacharyya
IIT Bombay
pb
@cse.iitb.ac.in

Abstract

Opinion summarization is the task of producing the summary of a text, such that the summary also preserves the sentiment of the text. Opinion Summarization is thus a trade-off between summarization and sentiment analysis. The demand of compression may drop sentiment bearing sentences, and the demand of sentiment detection may bring in redundant sentences. We harness the power of submodularity to strike a balance between two conflicting requirements. We investigate an incipient class of submodular functions for the problem, and a partial enumeration based greedy algorithm that has performance guarantee of 63%. Our functions generate summaries such that there is good correlation between document sentiment and summary sentiment along with good ROUGE score, which outperforms the-state-of-the-art algorithms.

1 Introduction

Sentiment Analysis is often addressed as a classification task, which aims at determining the sentiment of a word, sentence, paragraph or a document as a whole into positive, negative or neutral classes (Pang et al., 2002). Summarization, on the other hand is the task of aggregating and representing information content from a single document or multiple documents in a brief and fluent manner. Due to the explosive growth of data, fine grained sentiment analysis as well as summarization on the whole chunk of data can be a very time-consuming task. Sentiment Analysis also requires filtering of text portions as either objective (factual information) or subjective (expressing some sentiment or opinion) during pre-processing and then, classifying the subjective extracts as positive or negative. Subjective extracts can also be provided to users as a summary of the sentiment-oriented content of the reviews in search engines. In this paper, we address the problem of generic extractive summarization of reviews, a task commonly known as Opinion Summarization (Liu, 2012). The goals of opinion summarization are:

1. Present a short summary that conveys the essence as well as the sentiment of the review
2. Provide a short subjective extract to NLP pipeline for faster execution (*e.g.* sentiment analysis, review clustering *etc.*).

In this paper, we use movie reviews for opinion summarization task as they often have the following parts:

1. Plot - Description of the story, which is factual in nature
2. Critique - Opinion about the movie, which is sentiment bearing

Clearly, opinion summary to be generated will have a trade-off between the two opposing parts - subjective critique and objective plot. Our goal is to strike a balance through linear combination of suitable submodular functions in our paper. Joint models of relevance and subjectivity have a great benefit in that they have a large degree of freedom as far as controlling redundancy goes. In contrast, conventional two-stage approach Pang and Lee (2004), which first generate candidate subjective sentences using min-cut and then selects top subjective sentences within budget to generate a summary, have less computational complexity than joint models. However, two-stage approaches are suboptimal for text summarization. For example, when we select subjective sentences first, the sentiment as well information content may become redundant for a particular aspect. On the

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 169–178,
Lisbon, Portugal, 17-21 September 2015.

other hand, when we extract sentences first, an important subjective sentence may fail to be selected, simply because it is long. The two stage conflict in the sense that the demand of compression may drop sentiment bearing sentences, and the demand of sentiment detection may bring in redundant sentences. We then, use partial enumeration based greedy algorithm (Khuller et al., 1999), which gives performance guarantee of $(1 - e^{-1}) \approx 0.632$ (Sviridenko, 2004). The performance guarantee reported is better than simple greedy algorithm, used by Lin and Bilmes (2010) as their proof is erroneous (Morita et al., 2013). Further, the same greedy algorithm, which was used again in Lin and Bilmes (2011) gives only performance guarantee of $\frac{1}{2}(1 - e^{1}) \approx 0.316$ (Khuller et al., 1999).

The rest of the paper is as follows - in the next section, we look at previous work and establish further motivation for our work. Following that, we build the theory and formulate suitable objectives for opinion summarization task. In the final section, we present results based on implementation and testing of the functions. Experimental results show that the functions outperform the-state-of-the-art methods.

2 Previous Work

Automatically generating opinion summaries from large review text corpora has long been studied in both information retrieval and natural language processing.

In (Pang and Lee, 2004), a mincut-based algorithm was proposed to classify each sentence as being subjective or objective. The purpose of this work was to remove objective sentences from reviews to improve document level sentiment classification. Interestingly, the cut functions are symmetrical and submodular, and the problem of finding min-cut is equivalent to minimizing a symmetric submodular function.

Lerman et al. (2009) proposed three different models - sentiment match (SM), sentiment match + aspect coverage (SMAC) and sentiment-aspect match (SAM) to perform summarization of reviews of a product. The first model is called sentiment match (SM), which extracts sentences so that the average sentiment of the summary is as close as possible to the average sentiment rating of reviews of the entity *i.e.* low MISMATCH but with high sentiment INTENSITY. The second model, called sentiment match + aspect coverage (SMAC), builds a summary that trades-off between DIVERSITY, maximally covering important aspects and MISMATCH, matching the overall sentiment of the entity along with high INTENSITY. The third model, called sentiment-aspect match (SAM), not only attempts to cover important aspects, but cover them with appropriate sentiment using KL-Divergence function. Here, INTENSITY and DIVERSITY in the first two models are linear monotone submodular functions, while KL-Divergence function *i.e.* relative entropy in last model, unlike entropy is not monotone submodular.

In (Nishikawa et al., 2010b), a more sophisticated summarization technique was proposed, which generates a traditional text summary by selecting and ordering sentences taken from multiple reviews, considering both informativeness and readability of the final summary. The readability score in this paper would have been linear monotone submodular function, if the negative polarity was not penalizing. In (Nishikawa et al., 2010a), the authors further studied this problem using an integer linear programming formulation.

On the other hand, Lin et al .(2011) treated the task of generic summarization as monotone submodular function maximization. Further, they argued that monotone non-decreasing submodular functions are an ideal class of functions to investigate for document summarization. They also show, in fact, that many well-established methods for summarization (Carbonell and Goldstein, 1998; Filatova, 2004; Riedhammer et al., 2010) correspond to submodular function optimization, a property not explicitly mentioned in these publications. Since many authors either in summarization or opinion summarization have used functions similar to submodular functions as objective, we can take this fact as testament to the value of submodular functions for opinion summarization.

3 Theoretical Background

3.1 Introduction to Submodular Functions

A submodular function is a set function ($f : 2^V \rightarrow R$) having a natural diminishing returns property. Diminishing returns property holds if the difference in the value of the function that a single element makes when added to an input set decreases as the size of the input set increases *i.e.* for every $A, B \subseteq V$ with $A \subseteq B$ and every $x \in V \backslash B$, we have that $f(A \cup \{x\}) - f(A) \geq f(B \cup \{x\}) - f(B)$.

A submodular function f is monotone if for every $A \subseteq B$, we have that $f(A) \leq f(B)$.

The extractive summarization task can be modeled as optimization problem *i.e.* finding a set $S \subseteq V$ (S is set of sentences in summary, V is set of sentences in Document) which maximizes a submodular function $f(S)$ subject to budget constraints. In the following section, we will justify the use of submodular function for opinion summarization. Another advantage of choosing monotone submodular function is that there exists a polynomial-time greedy algorithm for constrained monotone submodular objective. The greedy algorithm guarantees that the summary solution obtained is almost as good as (63%) the best possible summary solution according to the objective (Sviridenko, 2004; Wolsey, 1982).

3.2 Submodularity in Opinion Summarization

Opinion Summarization should be modeled as a monotone submodular optimization problem, since opinion summary also holds following properties:

1. Monotonicity - As more sentences are added to opinion summary, subjectivity increases along with information content as opinionated words are being added.

2. Diminishing Return - If multiple sentences of varying intensity are added to opinion summary, the effect of a lower intensity polarity bearing sentence is diluted in the presence of a higher intensity one.

To show that opinion summarization inherently follow the diminishing return property, consider the following sentences[1] with positive polarity:
A: "*Even the acting in From Hell is solid, with the dreamy Depp turning in a typically strong performance and deftly handling a British accent.*"
B: "*Worth mentioning are the supporting roles by Ians Holm and Richardsonlog.*"
(A ∪ B) : "*Even the acting in From Hell is solid, with the dreamy Depp turning in a typically strong performance and deftly handling a British accent. Worth mentioning are the supporting roles by Ians Holm and Richardsonlog.*" Compare A and its superset, A ∪ B as candidate summaries. Sentence A and B convey positive sentiment, but sentence B has less intensity compared to sentence A. After reading the text (A ∪ B), it is clear that the effect of sentence B has diminished in front of sentence A, though both are of same polarity. B can be thus, removed from the candidate summary as it does a diminishing addition in presence of sentence A to the positive sentiment over the "acting" aspect of the entity "movie". The diminishing return not only holds for same polarity but also, for opposite polarity. Consider another example[2]:
A: "*The movie is predictive with foreseeable ending.*"
B: "*Still it's very well-done that no movie in this entire year has a scene that evokes pure joy as this does.*"
(A ∪ B) : "*The movie is predictive with foreseeable ending. Still it's very well-done that no movie in this entire year has a scene that evokes pure joy as this does.*" Compare B and its superset, A ∪ B as candidate summaries. Sentence A has negative sentiment whereas sentence B conveys positive sentiment with more intensity. When we read the text (A ∪ B), it is clear that the effect of sentence A has diminished in front of sentence B in text , as usually polarity of higher intensity dominates over the polarity of lower intensity. Now, consider a general example[3],
"*Laurence plays Neo's mentor Morpheus and he does an excellent job of it. His lines flow with confidence and style that makes his acting unique and interesting. The movie has lot of special effects and action-packed scenes with part of the appeal has philosophical and religious underpinnings.*"

If the budget for summary had been only two subjective sentences, then picking up first two would have redundantly captured only single aspect (*i.e.* acting) and the redundancy of the concept (acting) also causes a diminishing return of the second sentence because of the difference in sentiment intensity. However, picking the last sentence with either one of the first two would have not just covered both the aspects (*i.e.* acting and visual effects) but since, the sentences are not overlapping in aspects, there would not have been any diminishing return of sentiment on shared aspect (acting). Thus, it can be verified that opinion polarity also holds submodular property of diminishing return, if they are on the same aspect of a distinct entity.

[1] http://www.imdb.com/reviews/295/29590.html

[2] http://www.imdb.com/reviews/159/15918.html

[3] http://www.imdb.com/title/tt0133093/reviews

4 Formulation

Let V represent the set of the sentences in a document. The task of extractive opinion summarization is to select a subset $S \in V$ to represent the entirety (ground set V) . Obviously, we should have $|S| \leq |V|$ as it is a summary and should be small. Therefore, constraints on S can naturally be modeled as knapsack constraints:

$$\sum_{i \in S} c_i \leq b \tag{1}$$

where c_i is the non-negative cost of selecting unit i (*e.g.*, the number of words in the sentence) and b is our budget. If we use a set function $F : 2^V \rightarrow R$ to measure the quality of the summary set S, the summarization problem can then be formalized as the following combinatorial optimization problem:

$$S^* \in argmax_{S \subset V} F(S) \textit{ s.t. } \sum_{i \in S} c_i \leq b \tag{2}$$

where F (S) , total utility of summary is given as a linear combination of L(S), relevance and A(S), subjective coverage of aspects.

$$F(S) = \alpha L(S) + \beta A(S) \tag{3}$$

This formulation clearly brings out the trade-off between the subjective and the objective part. The intuition behind the combination of sentiment and aspect coverage in same function A(S) is that opinion polarity holds submodular property of diminishing return only if the set of sentences talk about common aspect of the same entity as discussed in previous section. L(S) , relevance is modeled same as in (Lin and Bilmes, 2011) as it captures the summary property, while our novel function, A(S) has been modeled differently through a suitable submodular function such that it captures the subjectivity property.

$$L(S) = \sum_{i \in V} min\{c_i(S), \gamma c_i(V)\} \tag{4}$$

$$c_i(S) = \sum_{j \in S} w_{i,j} \tag{5}$$

Here, $w_{i,j} > 0$ measures the similarity between i^{th} and j^{th} sentences and $c_i(S)$ measures the similarity of summary with the document.

Since, A(S), subjective coverage of aspects has to be modeled as monotone submodular function, it has been formulated as :

1. A_1 : **Modular Function**

   A_1(S) is simple linear function, which is sum of weighted subjective scores for each sentence. No budgeting constraints are added to this formulation.

   $$A_1(S) = \sum_i \sum_{j \in (P_i \cap S)} s_j * w_i \tag{6}$$

   Here $P_i; i = 1...K$ is a partition of the ground set V (*i.e.*, $\cup_i P_i = V$), which contains sentences pertaining to different distinct aspects. w_i are the weights of the partitions, based on the corresponding aspects. s_j is the subjective score of the sentence j in summary. The subjective score s_j is calculated using sentiwordnet as sum of the positive score $\in [0, 1]$ and negative score $\in [0, 1]$ (Esuli and Sebastiani, 2006).

   $$s_j = \sum_{word \in j} (pos(word) + neg(word)) \tag{7}$$

2. A_2 : **Budget-additive Function**

   $A_2(S)$ is an extension to $A_1(S)$, where maximum subjectivity score is restricted with budget based on aspect. Here, $\lambda \in [0, 1]$ is threshold coefficient for budget additive function to avoid redundancy of high sentiment on same aspect. When aspect i is saturated by S ($min(\sum_{j \in (P_i \cap S)} s_j, \lambda) = \lambda$), any new sentence j cannot further improve coverage over i and thus, other aspects, which are not yet saturated will have a better chance of being covered. This formulation ensures that produced summary is diverse enough and conveys sentiment about different aspects by budgeting.

   $$A_2(S) = \sum_i min(\sum_{j \in (P_i \cap S)} s_j, \lambda_i) * w_i \tag{8}$$

3. A_3 : **Polarity Partitioned Budget-additive Function**

   In previous formulation we have not considered the polarity of the sentences. For example, if an aspect have many positive sentences with more intensity but few negative

sentences with less intensity, A_2 more likely to reward more positive sentences because of intensity. In this formulation budgeting applied not only on aspect but polarity scores too. This ensures that both positive and negative polarity sentences are part of summary.

$$A_3(S) = \sum_i min(\sum_{j \in (P_i \cap S \cap P_{pos})} s_j, \lambda_i) * w_i + min(\sum_{j \in (P_i \cap S \cap P_{neg})} s_j, \lambda_i) * w_i \quad (9)$$

P_{pos} and P_{neg} are the partition of the sentences in the ground set V, based on their sign of polarity score. The polarity score pol_j for partitioning sentences into P_{pos} and P_{neg} is calculated as difference of the positive and negative score.

$$pol_j = \sum_{word \in j} (pos(word) - neg(word)) \quad (10)$$

Polarity based partitions bring out contrast view on a particular aspect, which is similar to contrast view opinion summarization to give the reader a direct comparative view of different strong opinions.

4. A_4 : **Facility Location Function**

   In this formulation, we model the facility location objective function (Krause and Golovin, 2014) for opinion summarization as choosing possible sentences (facilities) out of document (set of locations) to serve aspects (customers) giving service of value s_j. If each aspect (customer) chooses the sentences (facility) with the highest value, the total value provided to all aspects (customers) is modeled by this set function.

$$A_4(S) = \sum_i max_{j \in (P_i \cap S)} s_j * w_i \quad (11)$$

So A_4 rewards only a sentence which has maximum subjectivity score in each aspect.

5. A_5 : **Polarity Partitioned Facility Location Function**

   A_5 is similar to A_4, but for each aspect, A_5 rewards two sentences with positive and negative polarity but with maximum subjectivity scores in those polarity partitions.

$$A_5(S) = \sum_i max_{j \in (P_i \cap S \cap P_{pos})} s_j * w_i + \sum_i max_{j \in (P_i \cap S \cap P_{neg})} s_j * w_i \quad (12)$$

Each of the above functions are monotone submodular as the parameters s_j and w_i are positive. Since the first function is linear, it is both submodular and supermodular, thus modular. Budget additive and facility location functions (Krause and Golovin, 2014) are special types of monotone submodular functions. Since, monotone submodularity is preserved under non-negative linear combinations, polarity based partitioned function, whose sub-parts are monotone submodular is also monotone submodular.

5 Experiment

We have created Movie ontology tree manually (figure 1). Further the ontology is enriched by adding clue words to all aspects using wordnet sense propagation algorithm (Esuli and Sebastiani, 2006) for three iterations. The algorithm does a hard clustering of the sentences by assigning the sentence aspect, which has maximum number of clue words in that sentence. Clue words for 'Plot' aspect are *story, script, storyline, chief, communicative, explain, narrate, narration, narrative, narrator, report, reporter, scheme, schemer, script, scriptural, storyteller, tell, write up..,*.

For the experiments, we have used the polarity dataset from Pang et al. (2004). The dataset contains 1000 positive and 1000 negative movie reviews with size varying between 700 to 1000 words. As summary generation is time consuming task (DUC[4] only used 25 summaries to evaluate the performance of systems), we picked 100 positive and 100 negative reviews randomly from the dataset and their abstract summaries are generated manually with 200 words limit as budget for

[4]Document Understanding Conferences, http://duc.nist.gov

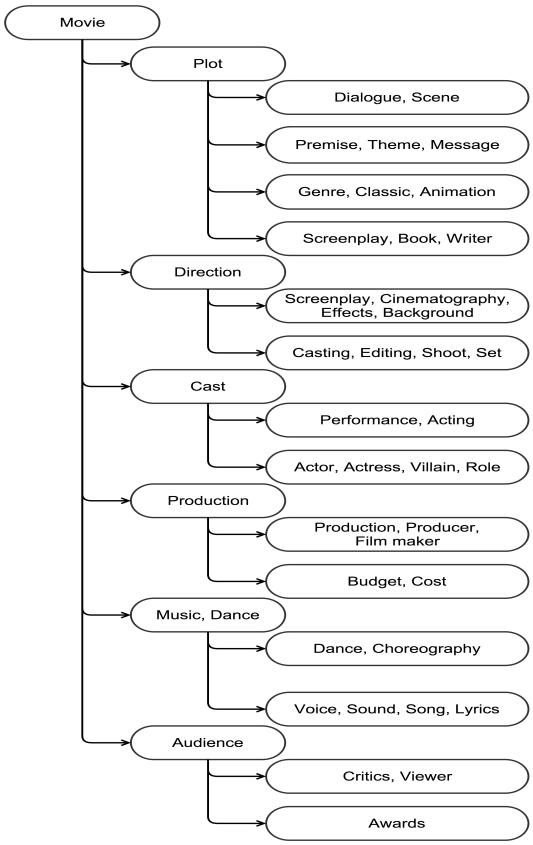


Figure 1: Movie Ontology Tree

evaluation. These 200 summaries are used as gold standard for estimating ROUGE scores of system generated summaries.

In the experiment, the partial enumeration based greedy algorithm (Khuller et al., 1999) is used for summary generation of 200 test documents within budget of 200 words. The algorithm has two parts. In the first part, the algorithm compares function values of all feasible solutions (sets) of cardinality one or two. Let $Summ_1$ be a feasible set of cardinality one or two that has the largest value of the objective function $F(S)$. In the second part, the algorithm enumerates all feasible sets of cardinality three. The algorithm, then completes each such set greedily and keeps the current solution feasible with respect to the knapsack constraint. Let $Summ_2$ be the solution obtained in the second part that has the largest value of objective function over all choices of the starting set for the greedy algorithm. Finally, the algorithm outputs $Summ_1$ if $F(Summ_1) > F(Summ_2)$ else $Summ_2$ otherwise. The algorithm does O(n^2) function calculations in first part, while O(n^5) in second part. This algorithm gives a performance guarantee of $(1-e^{-1})$ for solving monotone submodular objective with knapsack constraint (Khuller et al., 1999; Sviridenko, 2004). As far as we know, the algorithm has not been implemented for such problems because of complexity constraints (Lin and Bilmes, 2011).

Algorithm 1 Overall Algorithm - Summary Extraction

$B \Leftarrow 200$
for *Sentence s* $\in$ *Document* V **do**
  Assign sentence s to one of aspects in movie ontology.
end for
$Summ_1 \Leftarrow$ argmax { F(S), such that S $\subseteq$ V, |S| < 3, and cost(S) $\leq$ B }
$Summ_2 \Leftarrow \emptyset$
for all S $\subseteq$ V, |S|=3, and cost(S) $\leq$ B **do**
  $U \Leftarrow V \backslash S$
  while U $\neq \emptyset$ **do**
    maxReturn $\Leftarrow$ *0.0*
    newSentence $\Leftarrow \emptyset$
    for *Sentence s* $\in$ U **do**
      $S^* \Leftarrow S \cup \{s\}$
      $F(S^*) \Leftarrow \alpha L(S^*) + (1-\alpha)A(S^*)$
      $return \Leftarrow \frac{F(S^*)-F(S)}{len(s)}$
      if $return \geq maxReturn$ **then**
        $maxReturn \Leftarrow return$
        $newSentence \Leftarrow s$
      end if
    end for
    if cost($S \cup \{newSentence\}$) $\leq$ B **then**
      $S \Leftarrow S \cup \{newSentence\}$
    end if
    U $\Leftarrow$ U $\backslash \{newSentence\}$
  end while
  if F(S) $\geq F(Summ_2)$ **then**
    $Summ_2 \Leftarrow$ S
  end if
end for
if F($Summ_1$) $\geq$ F($Summ_2$) **then**
  Summary $\Leftarrow Summ_1$
else
  Summary $\Leftarrow Summ_2$
end if

In the algorithm, the sentences are clustered in different partitions, corresponding to different aspects in the ontology tree using the clue words. In the experiment, hard clustering of the sentences in aspect-based partitions is considered but soft-clustering of the sentences will also work with this approach, which has been left out to avoid further parameter tuning for soft clustering assigments. The weights of the partitions as well as the threshold parameters for the A(S) are currently kept proportional to the inverse of the depth of that aspect in the ontology-tree as sentiment expressed on the concepts at higher level in the ontology tree should have more weightage.

$\forall$ Aspects i,

$$w_i = \lambda_i = \frac{1}{Level(i)}$$

The linear combination parameter β is set as $1-\alpha$ to bring out the trade-off between relevance and subjective coverage of aspects and α is varied from 0 to 1 with step size 0.05 to find optimal α. γ in L(S) is set to 0.5. The parameter learning, esp. α and its impact have been already studied in (Lin and Bilmes, 2011) and thus, is not addressed in the paper. We have used the same approach of grid search to find the optimal value of α.

6 Results

We use ROUGE (Lin, 2004) for evaluating the content of summaries. We have used the 200 test documents that are manually summarized as gold standard data for ROUGE evaluation. For figuring out the sentiment correlation between manual and system generated summaries, we trained Naive Bayes sentiment classifier (Pang et al., 2002) on training data using bag of words approach with features as unigrams and bigrams and then, using minimum Pearson's chi-square score of 3 for feature extraction (Pecina and Schlesinger, 2006) before calculating the sentiment. The measure of sentiment preservation is calculated as Pearson correlation between the sentiment score of the document and the corresponding summary sentiment, both calculated by the Naive Bayes sentiment classifier while the measure of coverage of information content is given by ROUGE-1 and ROUGE-2 f-scores. Mathematically,

$$Correlation(X,Y) = \frac{Covariance(X,Y)}{std.dev(X) * std.dev(Y)} \quad (13)$$

Here, random variable X is the sentiment score of the document sample and random variable Y is the sentiment score of the corresponding summary sample. For 200 documents, it will be $[(X_1,Y_1),(X_2,Y_2),...,(X_{200},Y_{200})]$ sample points for the above correlation function.

Following five baselines are used for comparison:

1. **Baseline-1/TOP** : Sentences selected consecutively from the start of the review within the budget.

2. **Baseline-2/TOP-SUBJ** : Sentences ranked based on their subjectivity and then, selected with in the budget.

3. **Baseline-3/LER-SM** : (Lerman et al., 2009) Sentences which have sentiment close to document sentiment are chosen as Summary. We have used same NaiveBayes classifier (Pang et al., 2002) trained on imdb corpus to predict the sentiment of a sentence and document.

$$min_{S \subset V} \sum_{j \in S} (|senti(V) - senti(j)|) \quad (14)$$

4. **Baseline-4/TEXTRANK** : TextRank summarizer is based on Graph based unsupervised algorithm. Graph is constructed by creating a vertex for each sentence in the document and edges between vertices based on the number of words two sentences (of vertices) have in common and then, ranking them by applying PageRank to the resulting graph. Summary is generated with sentences having more vertex score (Mihalcea and Tarau, 2004).

5. **Baseline-5/MINCUT** : Mincut algorithm (Pang and Lee, 2004) classifies the sentences as subjective and objective sentences, by finding minimum s-t cuts in graph of sentences using maximum flow algorithm. In the graph, each sentence is a vertex and the edge between the vertex to the source or sink is taken as probability of the sentence being subjective or objective (individual scores). To ensure the graph connectivity, edges are drawn between every pair of sentence vertices, with edge weights taken proportional to the degree of proximity (association scores). After maximum flow algorithm, the cut in which source vertex lies is classified as subjective and vice-versa. We pick top subjective sentences within the budget as summary.

Among the five baselines, TOP and TOP-SUBJ are simplistic. Though both TEXTRANK and MINCUT were not originally proposed for opinion summarization but a number of papers in opinion summarization have built over these two methods and also, used them as baselines and thus, comparing with the "well-known" baselines will give the readers from the sentiment analysis field an intuitive idea of the performance of our system. MINCUT, however was reproposed specifically for subjective summarization by Pang and Lee (2004) and we use that formulation for comparison.

Table 1 compares the five functions with the above baselines based on optimal values of tradeoff α. From the table, it can be inferred that all the

System	ROUGE1	ROUGE2	S. Corr.
TOP	0.43001	0.16591	0.86144
TOP-SUBJ	0.41807	0.14362	0.82953
LER-SM	0.42608	0.14533	0.96545
TEXTRANK	0.41987	0.14644	0.88967
MINCUT	0.39368	0.11047	0.84017
Submod-A_1	0.43223	0.15702	0.95306
Submod-A_2	0.43594	0.15977	0.97538
Submod-A_3	0.43247	0.15436	0.93155
Submod-A_4	**0.43602**	**0.15760**	**0.98566**
Submod-A_5	0.42976	0.15551	0.95415

Table 1: ROUGE F-score and sentiment correlation for optimal values of α with baselines 1-5

System	ROUGE1	ROUGE2	S. Corr.
LIN-A_1	0.43112	0.15795	0.89850
LIN-A_2	0.42704	0.15382	0.90212
LIN-A_3	0.42612	0.15297	0.93155
LIN-A_4	0.42688	0.15245	0.93905
LIN-A_5	0.43359	0.15922	0.91019
Submod-A_1	0.43223	0.15702	0.95306
Submod-A_2	0.43594	0.15977	0.97538
Submod-A_3	0.43247	0.15436	0.93155
Submod-A_4	**0.43602**	**0.15760**	**0.98566**
Submod-A_5	0.42976	0.15551	0.95415

Table 2: ROUGE F-score and sentiment correlation for optimal values of α with baseline 6

proposed functions not only outperform the baselines in terms of ROUGE scores for optimal parameters but also, give better correlation with the document sentiment. This can be quantitively verified by test of significance, unpaired one-tailed t-test without assuming equal variance between the baselines and the systems. The p-values are 0.0203 and 0.0066 respectively for ROUGE-1 F Score and Sentiment Correlation, justifying that the performance improvement by our system over the baselines is statistically significant at $p < 0.05$. The main reason being that the functions with optimal values of trade-off parameter α strike out a balance between relevance and subjectivity. Clearly, the facility location based monotone submodular functions are the best choice as objective for opinion summarization task as they select sentences with maximum subjectivity (facilities giving best service).

Our system is able to access the information of aspect and polarity of each sentence, while some baselines do not. So, the improvement over the baselines may be attributed to those additional information rather than the optimality of the partial enumeration greedy algorithm over submodular functions. So, we therefore, introduced the following baseline to question this misdoubt on the experiment:

6. **Baseline-6/LIN** :

In this baseline, the greedy algorithm (Lin and Bilmes, 2010) is used for summary generation, using the same functions and information in the formulation. This algorithm fills the empty summary set greedily by adding a single sentence in each

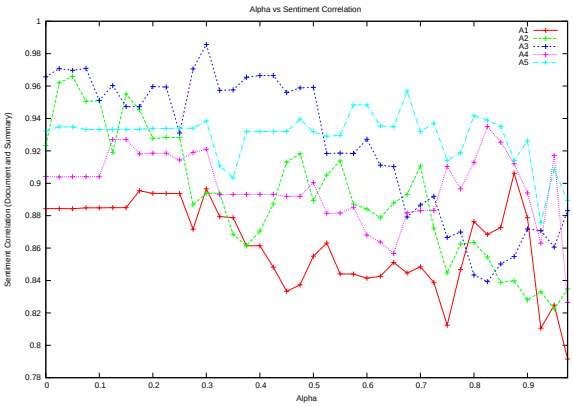


Figure 2: Sentiment Correlation *vs* α

iteration, which gives maximum return over cost ratio ($\frac{F(S^*)-F(S)}{len(s)}$), ensuring that current solution is feasible with respect to the knapsack constraint (cost($S \cup \{newSentence\}$) $\leq$ B). This algorithm has a complexity of O(n^2) but gives only performance guarantee of $\frac{1}{2}(1 - e^1) \approx 0.316$ (Khuller et al., 1999).

Table 2 compares the same five functions in our system with (Lin and Bilmes, 2010) system based on optimal values of tradeoff α. From the table, it can be inferred that our system also outperforms this baseline both in terms of ROUGE scores and sentiment correlation, which can be quantitively verified by test of significance, unpaired one-tailed t-test without assuming equal variance between the baselines and the systems. The p-values are 0.02517 and 0.003965 respectively for ROUGE-1 F Score and Sentiment Correlation, justifying that the performance improvement by our system over LIN system is statistically significant at $p < 0.05$.

The figures 2 and 3 plot the value of senti-

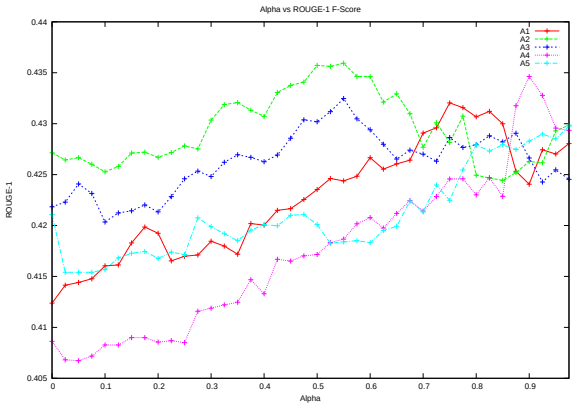


Figure 3: ROUGE-1 F-score *vs* α

Sys	ROUGE1	ROUGE2	Senti. Corr.
A_1	0.43223	0.15702	0.84827
A_2	0.43594	**0.15977**	0.88601
A_3	0.43247	0.15436	0.87038
A_4	**0.43602**	0.15760	0.87818
A_5	0.42976	0.15551	**0.90147**

Table 3: Maximum ROUGE F-score and their corresponding sentiment correlation

ment correlation and ROUGE-1 F score for the formulated submodular functions with respect to the trade-off parameter, α respectively. Looking at the graph 2, we can observe that more weightage to relevance over subjective coverage of aspects decreases the sentiment correlation, which was expected because the summary generated misses out on subjective sentiment due to trade-off. Similarly, by looking at the graph 3, we also observe that more weightage to relevance over subjective coverage of aspects increases the ROUGE score as expected. The erratic behaviour in figure 3 can be explained by arguing that subjective words are also important for summary and thus, giving less weightage to them over relevance, ROUGE score will increase but not properly.

The table 3 presents the value of sentiment correlation corresponding to maximum ROUGE score (for $\alpha \approx 1$). Clearly, A_4 and A_2 have maximum ROUGE scores as they neglect polarities and instead, reward on aspect based partitions, thus increasing coverage. The table 4 presents the value of ROUGE score corresponding to maximum sentiment correlation (for $\alpha \approx 0$). Clearly, A_4 also has maximum sentiment correlation as it rewards maximum subjectivity, irrespective of polarities and

Sys	Senti. Corr.	ROUGE1	ROUGE2
A_1	0.95306	**0.42572**	0.14939
A_2	0.97538	0.41764	0.14836
A_3	0.93155	0.42415	0.14782
A_4	**0.98566**	0.42492	**0.14942**
A_5	0.95415	**0.42572**	0.14266

Table 4: Maximum sentiment correlation and corresponding ROUGE F-Score

the corresponding ROUGE-2 F-score is also highest among all functions. Tables 1 and 2 contain the ROUGE F-score and sentiment correlation for optimal values of α, found after grid search while tables 3 and 4 contain the peak values in the figures 2 and 3. For example, table 3 contains the peak value of ROUGE-1 F score from figure 3 and the corresponding value of Sentiment Correlation from figure 2, at the same α.

7 Conclusion

In this paper, we show that conflict between subjectivity and relevance naturally arises in opinion summarization. To address this problem, we introduce new monotone submodular functions that are well suited to document summarization (Lin and Bilmes, 2010; Lin and Bilmes, 2011; Morita et al., 2013) by modeling two important properties of opinion summary - relevance and subjective coverage of aspects. We then, design different possible combinations of objective functions to model the task. To solve the algorithm effectively, we use the partial enumeration based algorithm, which is though computationally expensive (O(n^5) function calls), gives a performance guarantee of 63% for an NP-hard problem like summarization (McDonald, 2007). We have justified the submodular property of opinion summary through examples and significant performance of the system over the baselines. Further, this optimal trade-off between relevance and subjectivity can be used to design an evaluation framework for opinion summarization task as both part of the objective functions are proportional to the ROUGE and Sentiment Correlation respectively, which are widely used evaluation measures (Kim et al., 2011). As opinion summarization task lies in the intersection of opinion mining and summarization problems, both IR and NLP communities will benefit from our work.

References

Jaime Carbonell and Jade Goldstein. 1998. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *SIGIR*, pages 335–336.

Andrea Esuli and Fabrizio Sebastiani. 2006. Sentiwordnet: A publicly available lexical resource for opinion mining. In *Proceedings of LREC*, volume 6, pages 417–422.

Elena Filatova. 2004. Event-based extractive summarization. In *Proceedings of ACL Workshop on Summarization*, pages 104–111.

Samir Khuller, Anna Moss, and Joseph Seffi Naor. 1999. The budgeted maximum coverage problem. *Information Processing Letters*, 70(1):39–45.

Hyun Duk Kim, Kavita Ganesan, Parikshit Sondhi, and ChengXiang Zhai. 2011. Comprehensive review of opinion summarization.

Andreas Krause and Daniel Golovin. 2014. Submodular function maximization. In *Tractability: Practical Approaches to Hard Problems (to appear)*. Cambridge University Press, February.

Kevin Lerman, Sasha Blair-Goldensohn, and Ryan McDonald. 2009. Sentiment summarization: evaluating and learning user preferences. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 514–522.

Hui Lin and Jeff Bilmes. 2010. Multi-document summarization via budgeted maximization of submodular functions. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 912–920.

Hui Lin and Jeff Bilmes. 2011. A class of submodular functions for document summarization. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 510–520.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, pages 74–81.

Bing Liu. 2012. Sentiment analysis and opinion mining. *Synthesis Lectures on Human Language Technologies*, 5(1):1–167.

Ryan McDonald. 2007. A study of global inference algorithms in multi-document summarization. In *Advances in Information Retrieval*, pages 557–564. Springer.

Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into texts. In *Proceedings of EMNLP*, volume 4, page 275.

Hajime Morita, Hiroya Takamura, Ryohei Sasano, and Manabu Okumura. 2013. Subtree extractive summarization via submodular maximization. In *Proceedings of 51st Annual Meeting of the Association for Computational Linguistics (to appear)*.

Hitoshi Nishikawa, Takaaki Hasegawa, Yoshihiro Matsuo, and Genichiro Kikui. 2010a. Opinion summarization with integer linear programming formulation for sentence extraction and ordering. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 910–918.

Hitoshi Nishikawa, Takaaki Hasegawa, Yoshihiro Matsuo, and Genichiro Kikui. 2010b. Optimizing informativeness and readability for sentiment summarization. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 325–330.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd annual meeting on Association for Computational Linguistics*, page 271.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 79–86. Association for Computational Linguistics.

Pavel Pecina and Pavel Schlesinger. 2006. Combining association measures for collocation extraction. In *Proceedings of the COLING/ACL on Main conference poster sessions*, pages 651–658. Association for Computational Linguistics.

Korbinian Riedhammer, Benoit Favre, and Dilek Hakkani-Tür. 2010. Long story short–global unsupervised models for keyphrase based meeting summarization. *Speech Communication*, 52(10):801–815.

Maxim Sviridenko. 2004. A note on maximizing a submodular set function subject to a knapsack constraint. *Operations Research Letters*, 32(1):41–43.

Laurence A Wolsey. 1982. An analysis of the greedy algorithm for the submodular set covering problem. *Combinatorica*, 2(4):385–393.

Joint Prediction for Entity/Event-Level Sentiment Analysis using Probabilistic Soft Logic Models

Lingjia Deng
Intelligent Systems Program
University of Pittsburgh
lid29@pitt.edu

Janyce Wiebe
Intelligent Systems Program
Department of Computer Science
University of Pittsburgh
wiebe@cs.pitt.edu

Abstract

In this work, we build an entity/event-level sentiment analysis system, which is able to recognize and infer both explicit and implicit sentiments toward entities and events in the text. We design Probabilistic Soft Logic models that integrate explicit sentiments, inference rules, and +/-effect event information (events that positively or negatively affect entities). The experiments show that the method is able to greatly improve over baseline accuracies in recognizing entity/event-level sentiments.

1 Introduction

There are increasing numbers of opinions expressed in various genres, including reviews, newswire, editorials, and forums. While much early work was at the document or sentence level, to fully understand and utilize opinions, researchers are increasingly carrying out more fine-grained sentiment analysis to extract components of **opinion frames**: the source (whose sentiment is it), the polarity, and the target (what is the sentiment toward). Much fine-grained analysis is span or aspect based (Yang and Cardie, 2014; Pontiki et al., 2014). In contrast, this work contributes to **entity/event**-level sentiment analysis. A system that could recognize sentiments toward entities and events would be valuable in an application such as Automatic Question Answering, to support answering questions such as "Who is negative/positive toward X?" (Stoyanov et al., 2005), where X could be any entity or event.

Let us consider an example from the MPQA opinion annotated corpus (Wiebe et al., 2005a; Wilson, 2007; Deng and Wiebe, 2015).

Ex(1) When the Imam (may God be satisfied with him $_1$) issued the fatwa against $_2$ Salman Rushdie for insulting $_3$ the Prophet (peace be upon him $_4$), the countries that are so-called $_5$ supporters of human rights protested against $_6$ the fatwa.

There are several sentiment expressions annotated in MPQA. In the first clause, the writer is positive toward Imam and Prophet as expressed by *may God be satisfied with him* (1) and *peace be upon him* (4), respectively. Imam is negative toward Salman Rushdie and the insulting event, as revealed by the expression *issued the fatwa against* (2). And Salman Rushdie is negative toward *Prophet*, as revealed by the expression *insulting* (3). In the second clause, the writer is negative toward the countries, as expressed by *so-called* (5). And the countries are negative toward fatwa, as revealed by the expression *protested against* (6). Using the source and the target, we summarize the positive opinions above in a set P, and the negative opinions above in another set N. Thus, P contains {(writer, Imam), (writer, Prophet)}, and N contains {(Imam, Rushdie), (Imam, insulting), (Rushdie, Prophet), (writer, countries), (countries, fatwa)}.[1]

An (ideal) explicit sentiment analysis system is expected to extract the above sentiments expressed by (1)-(6). However, there are many more sentiments communicated by the writer but not expressed via explicit expressions. First, Imam is positive toward the Prophet, because Rushdie insults the Prophet and Imam is angry that he does

[1] Sources in MPQA are nested, having the form ⟨writer⟩ or ⟨writer, $S_1, \ldots, S_n$⟩. This work only deals with the rightmost source, writer or S_n. Also, actions like *issuing a fatwa* are treated the same as private states. Please see (Wiebe et al., 2005a).

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 179–189,
Lisbon, Portugal, 17-21 September 2015.

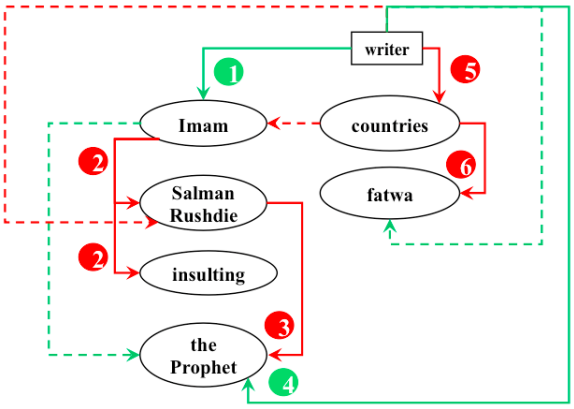


Figure 1: Explicit and implicit sentiments in Ex(1).

so. Second, the writer is negative toward Rushdie, because the writer is positive toward the Prophet but Rushdie insults him! Also, the writer is probably positive toward the fatwa since it is against Rushdie. Third, the countries are probably negative toward Imam, because the countries are negative toward fatwa and it is Imam who issued the fatwa. Thus, the set P should also contain {(Imam, Prophet), (writer, fatwa)}, and the set N should also contain {(writer, Rushdie), (countries, Imam)}. These opinions are not directly expressed, but are **inferred** by a human reader.[2] The explicit and implicit sentiments are summarized in Figure 1, where each green line represents a positive sentiment and each red line represents a negative sentiment. The solid lines are explicit sentiments and the dashed lines are implicit sentiments.

In this work, we detect sentiments such as those in P and N, where the sources are entities (or the writer) and the targets are entities and events.

Previous work in sentiment analysis mainly focuses on detecting explicit opinions. Recently there is emerging focus on sentiment inference, which recognizes implicit sentiments by inferring them from explicit sentiments via inference rules. Current works in sentiment inference differ on how the sentiment inference rules are defined and how they are expressed. For example, Zhang and Liu (2011) define linguistic templates to recognize phrases that express implicit sentiments, while previously we (Deng et al., 2014) represent a few simple rules as (in)equality constraints in Integer Linear Programming. In contrast to previous work, we propose a more general set of inference rules and encode them in a probabilistic soft logic (PSL) framework (Bach et al., 2015). We chose PSL because it is designed to have efficient inference and, as similar methods in Statistical Relational Learning do, it allows probabilistic models to be specified in first-order logic, an expressive and natural way to represent if-then rules, and it supports joint prediction. Joint prediction is critical for our task because it involves multiple, mutually constraining ambiguities (the source, polarity, and target).

Thus, this work aims at detecting both implicit and explicit sentiments expressed by an entity toward another entity/event (i.e., an **eTarget**) within the sentence. The contributions of this work are: (1) defining a method for entity/event-level sentiment analysis to provide a deeper understanding of the text; (2) exploiting first-order logic rules to infer such sentiments, where the source is not limited to the writer, and the target may be any entity, event, or even another sentiment; and (3) developing a PSL model to jointly resolve explicit and implicit sentiment ambiguities by integrating inference rules.

2 Related Work

Fined-grained sentiment analysis. Most fine-grained sentiment analysis is span or aspect based. Previous work differs from the entity/event-level sentiment analysis task we address in terms of targets and sources. In terms of targets, in a span-based sentiment analysis system, the target is a span instead of the exact head of the phrase referring to the target. The target in a span-based system is evaluated by measuring the overlapping proportion of an extracted span against the gold standard phrase (Yang and Cardie, 2013), while the eTarget in an entity/event-level system is evaluated against the exact word (i.e., head of NP/VP) in the gold standard. It is a stricter evaluation. While the targets in aspect-based sentiment analysis are often entity targets, they are mainly product aspects, which are a predefined set.[3] In contrast, the target in the entity/event-level task may be any noun or verb. In terms of sources, previous work in sentiment analysis trained on review data assumes that the source is the writer of the review (Hu and Liu, 2004; Titov and McDonald, 2008).

[2] Note that the inferences are conversational implicatures; they are defeasible and may not go through in context (Deng et al., 2014; Wiebe and Deng, 2014).

[3] As stated in SemEval-2014: "we annotate only aspect terms naming particular aspects".

Our work is rare in that it allows sources other than the writer *and* finds sentiments toward eTargets which may be any entity or event.

Sentiment Inference. There is some recent work investigating features that directly indicate implicit sentiments (Zhang and Liu, 2011; Feng et al., 2013). That work assumes the source is only the writer. Further, as it uses features to directly extract implicit sentiments, it does not perform general sentiment inference.

Previously, we (Deng et al., 2013; Deng and Wiebe, 2014; Deng et al., 2014) develop rules and models to infer sentiments related to *+/-effect events*, events that positively or negatively affect entities. That work assumes that the source is only the writer, and the targets are limited to entities that participate in +/-effect events. Further, our previous models all require certain manual (oracle) annotations to be input. In this work we use an expanded set of more general rules. We allow sources other than the writer, and targets that may be any entity or event. In fact, under our new rules, the targets of sentiments may be other sentiments; we model such novel "sentiment toward sentiment" structures in Section 4.3. Finally, our method requiring no manual annotations as input when the inference is conducted.

Previously, we also propose a set of sentiment inference rules and develop a rule-based system to infer sentiments (Wiebe and Deng, 2014). However, the rule-based system requires *all* information regarding explicit sentiments and +/-effect events to be provided as oracle information by manual annotations.

Probabilistic Soft Logic. Probabilistic Soft Logic (PSL) is a variation of Markov Logic Networks, which is a framework for probabilistic logic that employs weighted formulas in first-order logic to compactly encode complex undirected probabilistic graphical models (i.e., Markov networks) (Bach et al., 2015; Beltagy et al., 2014). PSL is a new statistical relational learning method that has been applied to many NLP and other machine learning tasks in recent years (Beltagy et al., 2014; London et al., 2013; Pujara et al., 2013; Bach et al., 2013; Huang et al., 2013; Memory et al., 2012). Previously, PSL has not been applied to entity/event-level sentiment analysis.

3 Task Definition

In this section, we introduce the definition of the entity/event-level sentiment analysis task, followed by a description of the gold standard corpus.

For each sentence s, we define a set E consisting of entities, events, and the writer of s, and sets P and N consisting of positive and negative sentiments, respectively. Each element in P is a tuple, representing a **positive pair** of two entities, (e_1, e_2) where e_1, $e_2 \in E$, and e_1 is positive toward e_2. A positive pair (e_1,e_2) aggregates all the positive sentiments from e_1 to e_2 in the sentence. N is the corresponding set for **negative pairs**.

The goal of this work is to automatically recognize a set of positive pairs (P_{auto}) and a set of negative pairs (N_{auto}). We compare the system output ($P_{\text{auto}} \cup N_{\text{auto}}$) against the gold standard ($P_{\text{gold}} \cup N_{\text{gold}}$) for each sentence.

3.1 Gold Standard Corpus: MPQA 3.0

MPQA 3.0 is a recently developed corpus with entity/event-level sentiment annotations (Deng and Wiebe, 2015).[4] It is built on the basis of MPQA 2.0 (Wiebe et al., 2005b; Wilson, 2007), which includes editorials, reviews, news reports, and scripts of interviews from different news agencies, and covers a wide range of topics.

In both MPQA 2.0 and 3.0, the top-level annotations include **direct subjective**s (**DS**). Each DS has a **nested-source** annotation. Each DS has one or more attitude links, meaning that all of the attitudes share the same nested source. The attitudes differ from one another in their attitude types, polarities, and/or targets. Moreover, both corpora contain **expressive subjective element** (**ESE**) annotations, which pinpoint specific expressions used to express subjectivity. We ignore neutral ESEs and only consider ESEs whose polarity is positive or negative.

MPQA 2.0 and 3.0 differ in their target annotations. In 2.0, each target is a span. A target annotation of an opinion captures the most important target this opinion is expressed toward. Since the exact boundaries of the spans are hard to define even for human annotators (Wiebe et al., 2005a; Yang and Cardie, 2013), the target span in MPQA 2.0 could be a single word, an NP or VP, or a text span covering more than one constituent. In contrast, in MPQA 3.0, each target is anchored to the head of an NP or VP, which is a single word. It is called an

[4] Available at http://mpqa.cs.pitt.edu/corpora/

eTarget since it is an entity or an event. In MPQA 2.0, only attitudes have target-span annotations. In MPQA 3.0, both attitudes and ESEs have eTarget annotations. Importantly, the eTargets include the targets of both explicit and implicit sentiments.

Recall Ex(1) in Section 1. $P_{\text{gold}} = \{$(writer, Imam), (writer, Prophet), (Imam, Prophet), (writer, fatwa)$\}$, and $N_{\text{gold}} = \{$(Imam, Rushdie), (Imam, insulting), (Rushdie, Prophet), (writer, countries), (countries, fatwa), (writer, Rushdie), (countries, Imam)$\}$.

4 PSL for Sentiment Analysis

We need to resolve three components for an opinion frame: the source, the polarity, and the eTarget. Each of these ambiguities has several candidates. For example in Ex(1), the eTarget of the opinion expression *insulting* is an ambiguity. The candidates include *Prophet*, *countries*, and so on.

In this work, we use Probabilistic Soft Logic (PSL). A PSL model is defined using a set of atoms to be grounded, and a set of weighted if-then rules expressed in first-order logic. For example, we define the atom ETARGET(y,t) to represent an opinion y having eTarget t. If y and t are constants, then ETARGET(y,t) is a ground atom (e.g., ETARGET(insulting, Prophet)). Each ground atom is assigned a score by a local system. PSL takes as input all the local scores as well as the constraints defined by the rules among atoms, so that it is able to jointly resolve all the ambiguities. In the final output, for example, the score ETARGET(insulting, Prophet) > 0 means that PSL considers Prophet to be an eTarget of *insulting*, while ETARGET(insulting, countries) $= 0$ means that PSL does not consider *countries* to be an eTarget of *insulting*.

In this section, we first introduce PSL in Section 4.1. We then present three PSL models in turn. PSL1 (Section 4.2) aggregates span-based opinions into P_{auto} and N_{auto}. PSL2 (Section 4.3) adds sentiment inference rules to PSL1. For PSL3 (Section 4.4), rules involving +/-effect events are added to PSL2, resulting in the richest overall model.

4.1 Probabilistic Soft Logic

PSL (Bach et al., 2015) uses logical representations to compactly define large graphical models with continuous variables, and includes methods for performing efficient probabilistic inference for the resulting models (Beltagy et al., 2014). As mentioned above, a PSL model is defined using a set of atoms to be grounded, and a set of weighted if-then rules in first-order logic. For example,

friend(x,y) $\wedge$ votesFor(y,z) $\Rightarrow$ votesFor(x,z)

means that a person may vote for the same person as his/her friend. Each predicate in the rule is an atom (e.g., friend(x,y)). A ground atom is produced by replacing variables with constants (e.g., friend(Tom, Mary)). Each rule is associated with a weight, indicating the importance of this rule in the whole rule set.

A key distinguishing feature of PSL is that each ground atom a has a soft, continuous truth value in the interval [0, 1], denoted as $I(a)$, rather than a binary truth value as in Markov Logic Networks and most other probabilistic logic frameworks (Beltagy et al., 2014). To compute soft truth values for logical formulas, Lukasiewicz relaxations are used:

$l_1 \wedge l_2 = \max\{0, I(l_1) + I(l_2) - 1\}$
$l_1 \vee l_2 = \min\{I(l_1) + I(l_2), 1\}$
$\neg l_1 = 1 - I(l_1)$

A rule $r \equiv r_{body} \rightarrow r_{head}$, is satisfied (i.e. $I(r) = 1$) iff $I(r_{body}) \leq I(r_{head})$. Otherwise, a distance to satisfaction $d(r)$ is calculated, which defines how far a rule r is from being satisfied: $d(r) = \max\{0, I(r_{body}) - I(r_{head})\}$. Using $d(r)$, PSL defines a probability distribution over all possible interpretations I of all ground atoms:

$$p(I) = \frac{1}{Z} \exp\{-1 * \sum_{r \in R} \lambda_r (d(r))^p\}$$

where Z is the normalization constant, λ_r is the weight of rule r, R is the set of all rules, and p defines loss functions. PSL seeks the interpretation with the minimum distance $d(r)$ and which satisfies all rules to the extent possible.

4.2 PSL for Sentiment Aggregation (PSL1)

The first PSL model, PSL1, aggregates span-based opinions into P_{auto} and N_{auto}. We call this *sentiment aggregation* because, instead of building an entity/event-level sentiment system from scratch, we choose to fully utilize previous work on span-based sentiment analysis. PSL1 aggregates span-based opinions into entity/event-level opinions.

Consistent with the task definition in Section 3, we define two atoms in PSL:
(1) POSPAIR(s,t): a positive pair from s toward t
(2) NEGPAIR(s,t): a negative pair from s toward t

Both s and t are chosen from the set E. The values of ground atoms (1) and (2) are not observed and are inferred by PSL.

Then, we define atoms to model an entity/event-level opinion:

(3) POS(y): y is a positive sentiment
(4) NEG(y): y is a negative sentiment
(5) SOURCE(y,s): the source of y is s
(6) ETARGET(y,t): the eTarget of y is t

Two rules are defined to aggregate various opinions extracted by span-based systems into positive pairs and negative pairs, shown in Part 1 of Table 1 as Rules 1.1 and 1.2. Thus, under our representation, the PSL model not only finds a set of eTargets of an opinion (ETARGET(y,t)), but also represents the aggregated sentiments among entities and events (POSPAIR(s,t) and NEGPAIR(s,t)) in the sentence.

Next, we turn to assigning local scores to ground atoms (3)-(6).

POS(y) and NEG(y): We build upon three span-based sentiment analysis systems. The first, S1 (Yang and Cardie, 2013), and the second, S2 (Yang and Cardie, 2014), are both trained on MPQA 2.0, which does not contain any eTarget annotations. S1 extracts triples of ⟨source span, opinion span, target span⟩, but does not extract opinion polarities. S2 extracts opinion spans and opinion polarities, but it does not extract sources or targets. The third system, S3 (Socher et al., 2013), is trained on movie review data. It extracts opinion spans and polarities. The source is always assumed to be the writer.

We take the union set of opinions extracted by S1, S2 and S3. For each opinion y, a ground atom is created, depending on the polarity (POS(y) if y is positive and NEG(y) is y is negative). The polarity is determined as follows. If S2 assigns a polarity to y, then that polarity is used. If S3 but not S2 assigns a polarity to y, then S3's polarity is used. In both cases, the score assigned to the ground atom is 1.0. If neither S2 nor S3 assigns a polarity to y, we use the MPQA subjectivity lexicon to determine its polarity. The score assigned to the ground atom is the proportion of the words in the opinion span that are included in the subjectivity lexicon.

SOURCE(y,s): S1 extracts the source of each opinion, S2 does not extract the source, and S3 assumes the source is always the writer. Thus, for an opinion y, if the source s is assigned by S1, a ground atom SOURCE(y,s) is created with score 1.0. Otherwise, if S3 extracts opinion y, a ground atom SOURCE(y,writer) is created with score 1.0 (since S3 assumes the source is always the writer). Otherwise, we run the Stanford named entity recognizer (Manning et al., 2014; Finkel et al., 2005) to extract named entities in the sentence. The nearest named entity to the opinion span on the dependency parse graph will be treated as the source. The score is the reciprocal of the length of the path between the opinion span and the source span in the dependency parse.

ETARGET(y,t): Though each eTarget is an entity or event, it is difficult to determine which nouns and verbs should be considered. Taking into consideration the trade-off between precision and recall, we experimented with three methods to select eTarget candidates. For each opinion y, a ground atom ETARGET(y,t) is created for each eTarget candidate t.

ET1 considers all the nouns and verbs in the sentence, to provide a full recall of eTargets.

ET2 considers all the nouns and verbs in the target spans and opinion spans that are automatically extracted by systems S1, S2 and S3. We hypothesized that ET2 would be useful because most of the eTargets in MPQA 3.0 appear within the opinion or the target spans of MPQA 2.0.

ET3 considers the heads of the target and opinion spans that are automatically extracted by systems S1, S2 and S3.[5] ET3 also considers the heads of siblings of target spans and opinion spans. Among the three methods, ET3 has the lowest recall but the highest precision.

In addition, for the eTarget candidate set extracted by ET2, or ET3, we run the Stanford co-reference system (Manning et al., 2014; Recasens et al., 2013; Lee et al., 2013) to expand the set in two ways. First, for each eTarget candidate t, the co-reference system extracts the entities that co-refer with t. We add the referring entities into the candidate set. Second, the co-reference system extracts words which the Stanford system judges to be entities, regardless of whether they have any referent or not. We add this set of entities to the candidate set as well.

We train an SVM classifier (Cortes and Vapnik, 1995) to assign a score to the ground atom ETARGET(y,t). Syntactic features describing the

[5]The head of a phrase is extracted by the Collins head finder in the Stanford parser (Manning et al., 2014).

Part 1.	Aggregation Rules.	
1.1	SOURCE(y,s) $\wedge$ ETARGET(y,t) $\wedge$ POS(y)	$\Rightarrow$ POSPAIR(s,t)
1.2	SOURCE(y,s) $\wedge$ ETARGET(y,t) $\wedge$ NEG(y)	$\Rightarrow$ NEGPAIR(s,t)
Part 2.	Inference Rules.	
2.1	POSPAIR(s_1,y_2) $\wedge$ SOURCE(y_2,s_2)	$\Rightarrow$ POSPAIR(s_1,s_2)
2.2	POSPAIR(s_1,y_2) $\wedge$ ETARGET(y_2,t_2) $\wedge$ POS(y_2)	$\Rightarrow$ POSPAIR(s_1,t_2)
2.3	POSPAIR(s_1,y_2) $\wedge$ ETARGET(y_2,t_2) $\wedge$ NEG(y_2)	$\Rightarrow$ NEGPAIR(s_1,t_2)
2.4	NEGPAIR(s_1,y_2) $\wedge$ SOURCE(y_2,s_2)	$\Rightarrow$ NEGPAIR(s_1,s_2)
2.5	NEGPAIR(s_1,y_2) $\wedge$ ETARGET(y_2,t_2) $\wedge$ POS(y_2)	$\Rightarrow$ NEGPAIR(s_1,t_2)
2.6	NEGPAIR(s_1,y_2) $\wedge$ ETARGET(y_2,t_2) $\wedge$ NEG(y_2)	$\Rightarrow$ POSPAIR(s_1,t_2)
Part 3.	Inference Rules w.r.t +/-Effect Event Information.	
3.1	POSPAIR(s,x) $\wedge$ AGENT(x,a)	$\Rightarrow$ POSPAIR(s,a)
3.2	POSPAIR(s,x) $\wedge$ THEME(x,h) $\wedge$ +EFFECT(x)	$\Rightarrow$ POSPAIR(s,h)
3.3	POSPAIR(s,x) $\wedge$ THEME(x,h) $\wedge$ -EFFECT(x)	$\Rightarrow$ NEGPAIR(s,h)
3.4	NEGPAIR(s,x) $\wedge$ AGENT(x,a)	$\Rightarrow$ NEGPAIR(s,a)
3.5	NEGPAIR(s,x) $\wedge$ THEME(x,h) $\wedge$ +EFFECT(x)	$\Rightarrow$ NEGPAIR(s,h)
3.6	NEGPAIR(s,x) $\wedge$ THEME(x,h) $\wedge$ -EFFECT(x)	$\Rightarrow$ POSPAIR(s,h)
3.7	POSPAIR(s,a) $\wedge$ AGENT(x,a)	$\Rightarrow$ POSPAIR(s,x)
3.8	POSPAIR(s,h) $\wedge$ THEME(x,h) $\wedge$ +EFFECT(x)	$\Rightarrow$ POSPAIR(s,x)
3.9	POSPAIR(s,h) $\wedge$ THEME(x,h) $\wedge$ -EFFECT(x)	$\Rightarrow$ NEGPAIR(s,x)
3.10	NEGPAIR(s,a) $\wedge$ AGENT(x,a)	$\Rightarrow$ NEGPAIR(s,x)
3.11	NEGPAIR(s,h) $\wedge$ THEME(x,h) $\wedge$ +EFFECT(x)	$\Rightarrow$ NEGPAIR(s,x)
3.12	NEGPAIR(s,h) $\wedge$ THEME(x,h) $\wedge$ -EFFECT(x)	$\Rightarrow$ POSPAIR(s,x)

Table 1: Rules in First-Order Logic.

relations between an eTarget and the extracted opinion span and target span are considered, including: (1) whether the eTarget is in the opinion/target span; (2) the unigrams and bigrams on the path from the eTarget to the opinion/target span in the constituency parse tree; and (3) the unigrams and bigrams on the path from the eTarget to the opinion/target word in the dependency parse graph. We normalize the SVM scores into the range of a ground atom score, [0,1].

4.3 PSL for Sentiment Inference (PSL2)

The two rules defined in Section 4.2 aggregate various opinions into positive pairs and negative pairs, but inferences have not yet been introduced. PSL2 is defined using the atoms and rules in PSL1. But it also includes some rules defined in (Wiebe and Deng, 2014), represented here in first-order logic in Part 2 of Table 1. Let us go through an example inference for Ex(1), in particular, the inference that Imam is positive toward the Prophet. Rule 2.6 supports this inference. Recall the two explicit sentiments: Imam is negative toward the insulting sentiment (revealed by *issued the fatwa against*), and Rushdie is negative toward the Prophet (revealed by *insulting*). Thus, we can instantiate Rule 2.6, where s_1 is Imam, y_2 is the negative sentiment (*insulting*), and t_2 is the Prophet. The inference is: since Imam is negative that there is any negative opinion expressed toward the Prophet, we infer that Imam is positive toward the Prophet.

NEGPAIR(Imam, insulting)
$\wedge$ ETARGET(insulting, Prophet)
$\wedge$ NEG(insulting)
$\Rightarrow$ POSPAIR(Imam, Prophet).

The inference rules in Part 2 of Table 1 are novel in that eTargets may be sentiments (e.g., NEGPAIR(Imam,insulting) means that Imam is negative toward the negative sentiment revealed by *insulting*). The inference rules link sentiments to sentiments and, transitively, link entities to entities (e.g., from Imam to Rushdie to the Prophet).

To support such rules, more groundings of ETARGET(y,t) are created in PSL2 than in PSL1. For two opinions y_1 and y_2, if the target span of y_1 overlaps with the opinion span of y_2, we create ETARGET(y_1,y_2) as a ground atom representing that y_2 is an eTarget of y_1.

4.4 PSL Augmented with +/-Effect Events (PSL3)

Finally, for PSL3, +/-effect event atoms and rules are added to PSL2 for the inference of additional sentiments.

According to (Deng et al., 2013), a +effect event has positive effect on the theme (examples are *help*, *increase*, and *save*), and a -effect event has negative effect on the theme (examples are *obstruct*, *decrease*, and *kill*).[6] We define the following atoms to represent such events:

(7) +EFFECT(x): x is a +effect event
(8) -EFFECT(x): x is a -effect event
(9) AGENT(x,a): the agent of x is a
(10) THEME(x,h): the theme of x is h

Next we assign scores to these ground atoms.

+EFFECT(x) and -EFFECT(x): We use the +/-effect sense-level lexicon (Choi and Wiebe, 2014)[7] to extract the +/-effect events in each sentence. The score of +EFFECT(x) is the fraction of that word's senses that are +effect senses according to the lexicon, and the score of -EFFECT(x) is the fraction of that word's senses that are -effect senses according to the lexicon. If a word does not appear in the lexicon, we do not treat it as a +/-effect event, and thus assign 0 to both +EFFECT(x) and -EFFECT(x).

AGENT(x,a) and THEME(x,h): We consider all nouns in the same or in sibling constituents of a +/-effect event as potential agents or themes. An SVM classifier is run to assign scores to AGENT(x,a), and another SVM classifier is run to assign scores to THEME(x,h). Both SVM classifiers are trained on a separate corpus, the +/-effect corpus (Deng et al., 2013) used in (Deng et al., 2014), which is annotated with +/-effect event, agent, and theme spans. The features we use to train the agent and theme classifier include unigram, bigram and syntax information.

Generalizations of the inference rules used in (Deng et al., 2014) are expressed in first-order logic, shown in Part 3 of Table 1. Let us go through an example inference for Ex(1), in particular, the inference that the countries are negative toward Imam. Recall that we infer this because the countries are negative toward the fatwa and it is Imam who issued the fatwa. The rules supporting this inference are Rules 3.11 and 3.4 in Table 1, where s is the countries, h is the fatwa, x is the issue event, and a is Imam.

The application of Rule 3.11 can be explained as follows. The countries are negative toward the fatwa, and the issue event is a +effect event with theme fatwa (the issue event is +effect for the fatwa because it creates the fatwa; creation is one type of +effect event identified in (Deng et al., 2013)); thus, the countries are negative toward the issue event.

NEGPAIR(countries, fatwa)
$\wedge$ THEME(issue, fatwa)
$\wedge$ +EFFECT(issue)
$\Rightarrow$ NEGPAIR(countries, issue) .

The application of Rule 3.4 can be explained as follows. The countries are negative toward the issue event, and it is Imam who conducted the event; thus, the countries are negative toward Imam.

NEGPAIR(countries, issue)
$\wedge$ AGENT(issue, Imam)
$\Rightarrow$ NEGPAIR(countries, Imam) .

Finally, to support the new inferences, more groundings of ETARGET(y,t) are defined in PSL3. For a +/-effect event x whose agent is a, if one of x and a is an eTarget candidate of y, the other will be added to the eTarget candidate set for y (sentiments toward both +effect and -effect events and their agents have the same polarity according to the rules (Deng et al., 2014)). For +effect event x whose theme is h, if one of x and h is an eTarget candidate of y, the other is added to the eTarget candidate set for y (sentiments toward +effect events and their themes have the same polarity).

5 Experiments

We carry out experiments on the MPQA 3.0 corpus. Currently, there are 70 documents, 1,634 sentences, and 1,921 DS and ESEs in total. The total number of POSPAIR(s,t) and NEGPAIR(s,t) are 867 and 1,975, respectively. Though the PSL inference does not need supervision and the SVM classifier for agents and themes in Section 4.4 is trained on a separate corpus, we still have to train the eTarget SVM classifier to assign local scores as described in Section 4.2. Thus, the experiments are carried out using 5-fold cross validation. For each fold test set, the eTarget classifier is trained on the other folds. The trained classifier is then run on the test set, and PSL inference is carried out on the test set.

[6] In (Deng et al., 2013), such events are called *goodFor/badFor* events; they are later renamed as *+/-effect* events.

[7] Available at: http://mpqa.cs.pitt.edu/lexicons/effect_lexicon/

In total, we have three methods for eTarget candidate selection (ET1, ET2, ET3) and three models for sentiment analysis (PSL1, PSL2, PSL3).

Baselines. Since each noun and verb may be an eTarget, the first baseline (All NP/VP) regards all the nouns and verbs as eTargets. The first baseline estimates the difficulty of this task.

The second baseline (SVM) uses the SVM local classification results from Section 4.2. The score of ETARGET(y,t) is assigned by the SVM classifier. Then it is normalized as input into PSL. Before normalization, if the score assigned by the SVM classifier is above 0, the SVM baseline considers it as a correct eTarget.

5.1 Evaluations

First, we examine the performance of the PSL models on correctly recognizing eTargets of a particular opinion. This evaluation is carried out on a subset of the corpus: we only examine the opinions which are automatically extracted by the span-based systems (S1, S2 and S3). If an opinion expression in the gold standard is not extracted by any span-based system, it is not input into PSL, so PSL cannot possibly find its eTargets.

The second and third evaluations assess performance of the PSL models on correctly extracting positive and negative pairs. Note that our sentiment analysis system has the capability, through inference, to recognize positive and negative pairs even if corresponding opinion expressions are not extracted. Thus, the second and third evaluations are carried out on the entire corpus. The second evaluation uses ET3, and compares PSL1, PSL2 and PSL3. The third evaluation uses PSL3 and compares performance using ET1, ET2 and ET3. The results for the other combinations follow the same trends.

ETargets of An Opinion. According to the gold standard in Section 3.1, each opinion has a set of eTargets. But not all eTargets are equally important. Thus, our first evaluation assesses the performance of extracting the most important eTarget. As introduced in Section 3.1, a span-based target annotation of an opinion in MPQA 2.0 captures the most important target this opinion is expressed toward. Thus, the head of the target span can be considered to be the most important eTarget of an opinion. We model this as a ranking problem to compare models. For an opinion y automatically extracted by a span-based system, both the SVM baseline and PSL assign scores to ETARGET(y,t). We rank the eTargets according to the scores. Because the ALL NP/VP baseline does not assign scores to the nouns and verbs, we do not compare with that baseline in this ranking experiment. We use the Precision@N evaluation metric. If the top N eTargets of an opinion contain the head of target span, we consider it as a correct hit. The results are in Table 2.

	Prec@1	Prec@3	Prec@5
SVM	0.0370	0.0556	0.0820
PSL1	0.5105	0.6905	0.7831
PSL2	0.5317	0.7486	0.7883
PSL3	0.5503	0.7434	0.8148

Table 2: Precision@N of Most Important ETarget.

Table 2 shows that SVM is poor at ranking the most important eTarget. The PSL models are much better, even PSL1, which does not include any inference rules. This shows that SVM, which only uses local features, cannot distinguish the most important eTarget from the others. But the PSL models consider all the opinions, and can recognize a true negative even if it ranks high in the local results. The ability of PSL to rule out true negative candidates will be repeatedly shown in the later evaluations.

We not only evaluate the ability to recognize the most important eTarget of a particular opinion, we also evaluate the ability to extract all the eTargets of that opinion. The F-measure of SVM is 0.2043, while the F-measures of PSL1, PSL2 and PSL3 are 0.3135, 0.3239, and 0.3275, respectively. Correctly recognizing all the eTargets is difficult, but all the PSL models are better than the baseline.

Positive Pairs and Negative Pairs. Now we evaluate the performance in a stricter way. We compare automatically extracted sets of sentiment pairs: $P_{\text{auto}} = \{\text{POSPAIR}(s,t) > 0\}$ and $N_{\text{auto}} = \{\text{NEGPAIR}(s,t) > 0\}$, against the gold standard sets P_{gold} and N_{gold}. Table 3 shows the accuracies using ET3. Note that higher accuracies can be achieved, as shown later. Here we use ET3 just to show the trend of results.

As shown in Table 3, the low accuracy of baseline All NP/VP shows that entity/event-level sentiment analysis is a difficult task. Even the SVM baseline does not have good accuracy. Note that the SVM baseline in Table 3 uses ET3. The baseline classifies the heads of target spans and opin-

	PosPair	NegPair
All NP/VP	0.1280	0.1654
SVM	0.0765	0.0670
PSL1	0.3356	0.3754
PSL2	0.3705	0.3705
PSL3	0.4315	0.3892

Table 3: Accuracy comparing PSL models (ET3 used for all)

ion spans, which are extracted by state-of-the-art span-based sentiment analysis systems. This shows the results from span-based sentiment analysis systems do not provide enough accurate information for the more fine-grained entity/event-level sentiment analysis task. In contrast, PSL1 achieves much higher accuracy than the baselines. PSL2 and PSL3, which add sentiment toward sentiment and +/-effect event inferences, give further improvements. A reason is that SVM uses a hard constraint to cut off many eTarget candidates, while the PSL models take the scores as soft constraints.

A more critical reason is due to the definition of accuracy: (TruePositive+TrueNegative)/All. A significant benefit of using PSL is correctly recognizing true negative eTarget candidates and eliminating them from the set. Interestingly, even though both PSL2 and PSL3 introduce more eTarget candidates, both are able to recognize more true negatives and improve the accuracy.

Note that F-measure does not count true negatives. Precision is $\frac{TP}{TP+FP}$, and recall is $\frac{TP}{TP+FN}$; neither considers true negatives (TN). As shown in Table 4, the increment of PSL model over baselines on F-measure is not as large as the increase in accuracy. Comparing PSL2 and PSL3 to PSL1, the inference rules largely increase recall but lower precision. However, the accuracy in Table 3 keeps growing. Thus, the biggest advantage of PSL models is to correctly rule out true negative eTargets. For the baselines, though the SVM baseline has higher precision, it eliminates so many eTarget candidates that the F-measure is not high.

ETarget Selection. To assess the methods for eTarget selection, we run PSL3 (the fullest PSL model) using each method in turn. The F-measures and accuracies are listed in Table 5. The F-measure of ET1 is slightly lower than the F-measures of ET2 and ET3, while the accuracy of

	Precision	Recall	F-measure
	PosPair		
All NP/VP	0.1481	0.4857	0.2270
SVM	0.3791	0.0870	0.1415
PSL1	0.2234	0.2687	0.2440
PSL2	0.1666	0.2738	0.2072
PSL3	0.1659	0.3523	0.2256
	NegPair		
All NP/VP	0.1824	0.6408	0.2840
SVM	0.3568	0.0761	0.1254
PSL1	0.2857	0.3872	0.3288
PSL2	0.2772	0.3883	0.3235
PSL3	0.2586	0.4529	0.3292

Table 4: F-measure comparing PSL models (ET3 used for all)

ET1 is much better than the accuracies of ET2 and ET3. Again, this is because PSL recognizes true negatives in the eTarget candidates. Since ET1 considers more eTarget candidates, ET1 gives PSL a greater opportunity to remove true negatives, leading to an overall increase in accuracy.

	PosPair		NegPair	
	F	Acc.	F	Acc.
ET1	0.2192	0.4963	0.3157	0.4461
ET2	0.2374	0.4433	0.3261	0.3969
ET3	0.2256	0.4315	0.3295	0.3892

Table 5: Comparison of eTarget selection methods (PSL3 used for all)

6 Conclusion

This work builds upon state-of-the-art span-based sentiment analysis systems to perform entity/event-level sentiment analysis covering both explicit and implicit sentiments expressed among entities and events in text. Probabilistic Soft Logic models incorporating explicit sentiments, inference rules and +/-effect event information are able to jointly disambiguate the ambiguities in the opinion frames and improve over baseline accuracies in recognizing entity/event-level sentiments.

Acknowledgements. This work was supported in part by DARPA-BAA-12-47 DEFT grant #12475008. We thank the anonymous reviewers for their helpful comments.

References

Stephen H Bach, Bert Huang, and Lise Getoor. 2013. Learning latent groups with hinge-loss markov random fields. In *Inferning: ICML Workshop on Interactions between Inference and Learning*.

Stephen H. Bach, Matthias Broecheler, Bert Huang, and Lise Getoor. 2015. Hinge-loss markov random fields and probabilistic soft logic. arXiv:1505.04406 [cs.LG].

Islam Beltagy, Katrin Erk, and Raymond Mooney. 2014. Probabilistic soft logic for semantic textual similarity. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1210–1219, Baltimore, Maryland, June. Association for Computational Linguistics.

Yoonjung Choi and Janyce Wiebe. 2014. +/-effectwordnet: Sense-level lexicon acquisition for opinion inference. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1181–1191, Doha, Qatar, October. Association for Computational Linguistics.

Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine learning*, 20(3):273–297.

Lingjia Deng and Janyce Wiebe. 2014. Sentiment propagation via implicature constraints. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 377–385, Gothenburg, Sweden, April. Association for Computational Linguistics.

Lingjia Deng and Janyce Wiebe. 2015. Mpqa 3.0: An entity/event-level sentiment corpus. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1323–1328, Denver, Colorado, May–June. Association for Computational Linguistics.

Lingjia Deng, Yoonjung Choi, and Janyce Wiebe. 2013. Benefactive/malefactive event and writer attitude annotation. In *ACL 2013 (short paper)*. Association for Computational Linguistics.

Lingjia Deng, Janyce Wiebe, and Yoonjung Choi. 2014. Joint inference and disambiguation of implicit sentiments via implicature constraints. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 79–88, Dublin, Ireland, August. Dublin City University and Association for Computational Linguistics.

Song Feng, Jun Sak Kang, Polina Kuznetsova, and Yejin Choi. 2013. Connotation lexicon: A dash of sentiment beneath the surface meaning. In *Proceedings of the 51th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, Sofia, Bulgaria, Angust. Association for Computational Linguistics.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 363–370. Association for Computational Linguistics.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177. ACM.

Bert Huang, Angelika Kimmig, Lise Getoor, and Jennifer Golbeck. 2013. A flexible framework for probabilistic models of social trust. In *Social Computing, Behavioral-Cultural Modeling and Prediction*, pages 265–273. Springer.

Heeyoung Lee, Angel Chang, Yves Peirsman, Nathanael Chambers, Mihai Surdeanu, and Dan Jurafsky. 2013. Deterministic coreference resolution based on entity-centric, precision-ranked rules. *Computational Linguistics*, 39(4):885–916.

Ben London, Sameh Khamis, Stephen H. Bach, Bert Huang, Lise Getoor, and Larry Davis. 2013. Collective activity detection using hinge-loss Markov random fields. In *CVPR Workshop on Structured Prediction: Tractability, Learning and Inference*.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60.

Alex Memory, Angelika Kimmig, Stephen Bach, Louiqa Raschid, and Lise Getoor. 2012. Graph summarization in annotated data using probabilistic soft logic. In *Proceedings of the 8th International Workshop on Uncertainty Reasoning for the Semantic Web (URSW 2012)*, volume 900, pages 75–86.

Maria Pontiki, Haris Papageorgiou, Dimitrios Galanis, Ion Androutsopoulos, John Pavlopoulos, and Suresh Manandhar. 2014. Semeval-2014 task 4: Aspect based sentiment analysis. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 27–35.

Jay Pujara, Hui Miao, Lise Getoor, and William Cohen. 2013. Knowledge graph identification. In *The Semantic Web–ISWC 2013*, pages 542–557. Springer.

Marta Recasens, Marie-Catherine de Marneffe, and Christopher Potts. 2013. The life and death of discourse entities: Identifying singleton mentions. In *HLT-NAACL*, pages 627–633.

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng,

and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1631–1642. Citeseer.

Veselin Stoyanov, Claire Cardie, and Janyce Wiebe. 2005. Multi-Perspective Question Answering using the OpQA corpus. In *Proceedings of the Human Language Technologies Conference/Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP-2005)*, pages 923–930, Vancouver, Canada.

Ivan Titov and Ryan T McDonald. 2008. A joint model of text and aspect ratings for sentiment summarization. In *ACL*, volume 8, pages 308–316. Citeseer.

Janyce Wiebe and Lingjia Deng. 2014. An account of opinion implicatures. *arXiv*, 1404.6491[cs.CL].

Janyce Wiebe, Theresa Wilson, and Claire Cardie. 2005a. Annotating expressions of opinions and emotions in language. *Language resources and evaluation*, 39(2-3):165–210.

Janyce Wiebe, Theresa Wilson, and Claire Cardie. 2005b. Annotating expressions of opinions and emotions in language ann. *Language Resources and Evaluation*, 39(2/3):164–210.

Theresa Wilson. 2007. *Fine-grained Subjectivity and Sentiment Analysis: Recognizing the Intensity, Polarity, and Attitudes of private states*. Ph.D. thesis, Intelligent Systems Program, University of Pittsburgh.

Bishan Yang and Claire Cardie. 2013. Joint inference for fine-grained opinion extraction. In *ACL (1)*, pages 1640–1649.

Bishan Yang and Claire Cardie. 2014. Context-aware learning for sentence-level sentiment analysis with posterior regularization. In *Proceedings of ACL*.

Lei Zhang and Bing Liu. 2011. Identifying noun product features that imply opinions. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 575–580, Portland, Oregon, USA, June. Association for Computational Linguistics.

Learning to Recognize Affective Polarity in Similes

Ashequl Qadir and Ellen Riloff
School of Computing
University of Utah
Salt Lake City, UT 84112, USA
{asheq,riloff}@cs.utah.edu

Marilyn A. Walker
Natural Language & Dialogue Systems Lab
University of California Santa Cruz
Santa Cruz, CA 95064, USA
mawalker@ucsc.edu

Abstract

A *simile* is a comparison between two essentially unlike things, such as *"Jane swims like a dolphin"*. Similes often express a positive or negative sentiment toward something, but recognizing the polarity of a simile can depend heavily on world knowledge. For example, *"memory like an elephant"* is positive, but *"memory like a sieve"* is negative. Our research explores methods to recognize the polarity of similes on Twitter. We train classifiers using lexical, semantic, and sentiment features, and experiment with both manually and automatically generated training data. Our approach yields good performance at identifying positive and negative similes, and substantially outperforms existing sentiment resources.

1 Introduction

A *simile* is a form of figurative language that compares two essentially unlike things (Paul, 1970), such as *"Jane swims like a dolphin"*. Similes often express a positive or negative view toward an entity, object, or experience (Li et al., 2012; Fishelov, 2007). Sometimes, the sentiment of a simile is expressed explicitly, such as *"Jane swims **beautifully** like a dolphin!"*. But in many cases the sentiment is implicit, evoked entirely from the comparison itself. *"Jane swims like a dolphin"* is easily understood to be a compliment toward Jane's swimming ability because dolphins are known to be excellent swimmers.

A simile consists of four key components: the **topic** or **tenor** (subject of the comparison), the **vehicle** (object of the comparison), the **event** (act or state), and a **comparator** (usually "as", "like", or "than") (Niculae and Danescu-Niculescu-Mizil, 2014). A **property** (shared attribute) can be optionally included as well (e.g., *"He is as **red** as a tomato"*). Our research aims to identify the affective polarity of a simile as *positive*, *negative*, or *neutral*, based on its component phrases.

Table 1 shows examples of similes and their polarity. A simile can have *neutral* polarity if it offers an objective observation. Example (a) is a neutral simile because, although bananas have a distinctive smell, it is not generally considered to be a particularly good or bad scent. Example (b) illustrates that using the subjective verb "stink" instead of "smell" indicates a negative polarity toward the scent of bananas. Example (c) shows that including a subjective adjective such as "rotten" suggests a negative sentiment. Example (d) has negative polarity because the vehicle term, "garbage", carries a strong negative connotation.

	Simile	Polarity
(a)	smells like bananas	*neutral*
(b)	stinks like bananas	*negative*
(c)	smells like rotten bananas	*negative*
(d)	smells like garbage	*negative*
(e)	memory like an elephant	*positive*
(f)	memory like a sieve	*negative*
(g)	looks like a celebrity	*positive*
(h)	acts like a celebrity	*negative*

Table 1: Simile Examples with Affective Polarity.

However, the affective polarity of a simile often emerges from multiple component terms. For instance, all of the words in Examples (e) and (f) have neutral polarity. But Example (e) is positive because elephants are widely known to have excellent memories, while Example (f) is negative because a sieve has holes, which is metaphorical with memory lapses. Examples (g) and (h) illustrate that a prior connotation can even be overridden depending upon the property being compared. In general, the word "celebrity" tends to have a positive connotation and looking like a celebrity is generally a compliment. But acting like a celebrity is a negative simile because it alludes to negative attributes such as narcissism or entitlement.

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 190–200,
Lisbon, Portugal, 17-21 September 2015.

Our research explores the challenge of identifying the affective polarity of similes. First, we introduce a new data set of similes extracted from Twitter. We describe a manual annotation study to label them with affective polarity. We also present several approaches for identifying some instances of positive and negative similes using existing sentiment resources, to automatically create labeled data to train a classifier. Second, we describe a machine learning classifier to recognize the affective polarity of similes by considering lexical, semantic, and sentiment properties of their components. Third, we present experimental results for the simile polarity classifier, using both manually annotated training data and automatically labeled training data. Our evaluation shows that the classifier trained with manually labeled data achieves good performance at identifying positive and negative similes. Training with automatically labeled data produces classifiers that are not quite as good as those trained with manually labeled data, but they still substantially outperform existing sentiment resources and offer a way to easily train simile classifiers for different domains.

2 Related Work

Although similes are a popular form of comparison, there has been relatively little prior research on understanding affective polarity in similes. Veale and Hao (2007) created a large simile case-base using the pattern "as ADJ as a/an NOUN". They collected similes by querying the web after instantiating part of the pattern with adjectives, and then had a human annotate 30,991 of the extracted similes for validity. Their focus was on extracting salient properties associated with simile **vehicles**, and the affective perception on vehicles that the salient properties bring about.

Veale (2012) took a step further and automatically recognized the affect toward vehicles when properties reinforce each other (e.g., hot and humid). They built a support graph of properties and determined how they connect to unambiguous positive and negative words. Li et al. (2012) used similar patterns to retrieve similes and determine basic sentiment toward simile vehicles across different languages using the compared properties. One major difference with their work and ours is that they determine sentiment or affective perception toward entities or concepts extracted from simile **vehicles**. In contrast, our work is focused on determining affective polarity of a simile as a whole, where the affective polarity typically relates to an act or state of the **tenor**. In many cases, a simile vehicle does not have positive or negative polarity by itself. For example, "sauna" is not a positive or negative concept, but "room feels like a sauna" is a negative simile because it suggests that the room is humid and unpleasant.

Niculae and Danescu-Niculescu-Mizil (2014) created a simile data set from Amazon product reviews, and determined when comparisons are figurative. They did not identify affective polarity, but showed that sentiment and figurative comparisons are correlated. Fishelov (2007) conducted a study of 16 similes where the connection between tenor and vehicle is obvious or not obvious, and when a conventional or unconventional explicit property is present or absent. Fishelov analyzed responses from participants to understand the positive and negative impression a simile conveys toward its tenor. Hanks (2005) presented an analysis of semantic categories of simile vehicles (animal, roles in society, artifact, etc.) that people most commonly use in similes.

Previous research has also explored sentiment expressed through metaphor. Rumbell et al. (2008) presented an analysis of animals that are metaphorically used to describe a person. Rentoumi et al. (2009) determined use of figurative language by disambiguating word senses, and then determined sentiment polarity at the sense level using ngram graph similarity. Wallington et al. (2011) identified affect in metaphor and similes when a comparison is made with an animal (e.g., dog, fox) or mythical creature (e.g., dragon, angel) by analyzing WordNet sense glosses of the compared terms. More recently, the SemEval-2015 Shared Task 11 (Ghosh et al., 2015) has addressed the sentiment analysis of figurative language such as irony, metaphor and sarcasm in Twitter.

Our work is also related to sentiment analysis in general. The most common approach applies supervised classification with features such as ngrams, parts-of-speech, punctuation, lexicon features, etc. (e.g., (Kouloumpis et al., 2011; Davidov et al., 2010; Mohammad et al., 2013)). To overcome the challenge of acquiring manually labeled data, some work automatically collects noisy training data using emoticons and hashtags (e.g., (Go et al., 2009; Purver and Battersby, 2012)). In addition to determining overall sen-

timent, research has also focused on understanding people's sentiment during specific events such as stock market fluctuations, presidential elections, Oscars, tsunamis, or toward entities such as movies, companies, or aspects of a product (Bollen et al., 2011; Thelwall et al., 2011; Jiang et al., 2011; Hu and Liu, 2004; Jo and Oh, 2011).

To our knowledge, we are the first to explore recognition of affective polarity in similes as a whole, where the polarity relates to an act or state of the tenor. Unlike previous work, we do not rely on the presence of explicit properties. We also present a data set annotated with affective polarity in similes, and experiment with both manually annotated and automatically acquired training data.

3 Simile Data Set Creation

One of the major challenges of supervised classification is acquiring sufficient labeled data for training, since manual annotation is time consuming. However, similes sometimes contain words with explicit polarity (e.g., "bed feels like *heaven*" or "he *cries* like a baby"). Many of these cases can be identified with existing sentiment resources and then used to provide a classifier with training instances. But because sentiment resources have limitations (e.g., sentiment classifiers are not perfect, sentiment lexicons do not possess knowledge of context), these instances will have some noise. Therefore, we experiment with both manually labeled data sets that are smaller in size but high quality, and automatically labeled data sets that are comparatively larger but noisier.

Twitter is a popular microblogging platform and is widely used for sentiment analysis. Thus it is an excellent source for collecting similes that people use in everyday conversation. For this research, we extracted similes from 140 million tweets we harvested using the Twitter streaming API from March 2013 to April 2014. We started by selecting tweets containing three common **comparator** keywords: *"like"*, *"as"*, and *"than"*. We removed tweets with exact duplicate content, and tweets containing a retweet token. An additional challenge of tweets is that many are "near duplicates" (e.g., shared tweets with an added symbol or comment). So we performed an additional de-duplication step using Jaccard similarity of trigrams to measure overlap in the text content between pairs of tweets. When Jaccard similarity between two tweets was > 0.5, we kept only the longer tweet, and repeated the process.

We used the UIUC Chunker (Punyakanok and Roth, 2001) to identify phrase sequences with the syntax of similes (e.g., NP_1 + VP + PP-like + NP_2, or NP_1 + VP + ADJP + PP-like + NP_2, where NP_1 is the tenor, NP_2 is the vehicle, VP is the event, and ADJP is any explicitly mentioned property). We generalized over the extracted similes by removing the comparator, and the optional explicit property component. Our simile representation is thus a triple of the tenor, event and vehicle. We also lemmatized all words using Stanford CoreNLP (Manning et al., 2014). For a tenor phrase, we kept only the head noun, which is usually sufficient to understand the affective polarity target. We kept the entire noun phrase for the vehicle, since vehicles like "ice box" and "gift box" may represent two different concepts with different polarities in similes. We replaced personal pronouns (e.g., he, she) with a general PERSON token and other pronouns (e.g., it, this, that) with a general IT token. Table 2 presents examples of positive and negative similes in the annotated data set.

Positive	**Negative**
PERSON, smile, sun	PERSON, look, zombie
PERSON, feel, kid	PERSON, treat, stranger
PERSON, be, older brother	PERSON, feel, poo
IT, sound, heaven	PERSON, look, clown
PERSON, look, superman	word, cut, knife
IT, be, old time	PERSON, act, child
IT, feel, home	PERSON, look, voldemort
IT, fit, glove	PERSON, look, wet dog
IT, would be, dream	PERSON, treat, baby
IT, smell, spring	PERSON, look, drug addict

Table 2: Sample Similes from Annotated Data.

Sometimes, vehicle phrases contain adjective modifiers indicating explicit sentiment (e.g., "she looks like a *beautiful* model"). Since a simile is trivial to classify with such a modifier, we removed the instances that already had positive or negative adjective modifiers. To identify these cases, we used the AFINN sentiment lexicon (Nielsen, 2011). Similes that contain profanity (e.g., "You look like *crap*") are nearly always negative, and trivial to classify, so we filtered out these cases using a freely available profanity list[1]. We also removed any simile where the vehicle is a pronoun (e.g., "it looks like that"), and discard similes appearing fewer than 5 times. Our final data set contains 7,594 similes.

[1] http://www.bannedwordlist.com/lists/swearWords.txt

3.1 Manually Annotated Simile Data Set

To obtain manual annotation, we randomly selected 1500 similes occurring at least 10 times, from the 7,594 similes. Our expectation was that more frequent similes will be easier for the annotators to understand. We used Amazon's Mechanical Turk to obtain gold standard annotations for affective polarity. We asked the annotators to determine if a simile expresses affective polarity toward the subject (i.e., the **tenor** component), and to assign one of four labels: *positive*, *negative*, *neutral*, or *invalid*. The first two labels are for similes that clearly express positive polarity (e.g., *"Jane swims like a dolphin"*) or negative polarity (e.g., *"Fred's hair looks like a bird's nest"*). The *neutral* label is for similes that do not have positive or negative polarity (e.g., *"the cloud looks like a turtle"* isn't a positive or negative comment about the cloud) or similes that are ambiguous without the benefit of context (e.g., *"he is like my dog"* could be good or bad depending on the context).

The data also contained many misidentified similes, typically due to parsing errors. For example, sometimes there is an entire clause in place of the vehicle (e.g., "I feel like im gonna puke"). Other times, the informal text of Twitter makes the tweet hard to parse (e.g., "he is like whatttt") or a verb occurs after "like" (e.g., "he is like hyperventilating"). The *invalid* label covers these types of erroneously extracted similes.

The annotation task was first conducted on a small sample of 50 similes, to select workers that had high annotation agreement with each other and gold standard labels we prepared. The best three workers then all annotated the official set of 1500 similes. The average Cohen's Kappa (κ) (Carletta, 1996) between each pair of annotators was 0.69. We then assigned the final label through majority vote. However, none of the annotators agreed on the same label for 78 of the 1500 similes, and 303 instances were labeled as *invalid* similes by the annotators. So we removed these 381 instances from the annotated data set. Finally, we randomly divided the remaining similes into an evaluation (Eval) set of 741 similes, and a development (Dev) set of 378 similes. Table 3 shows the label distribution of these sets.

Label	# of Similes (Dev Data)	# of Similes (Eval Data)
Positive	164	312
Negative	181	343
Neutral	33	86
Total	378	741

Table 3: Manually Annotated Data.

3.2 Automatically Labeled Similes

For any new domain (e.g., Amazon product reviews), manual annotations for supervised training may not be readily available, and being able to automatically obtain training instances can be valuable. We therefore create and experiment with six types of automatically labeled training data.

Using AFINN Sentiment Lexicon Words: Our first training data set is created using the AFINN sentiment lexicon (Nielsen, 2011) containing 2,477 manually labeled words with integer values ranging from -5 (negativity) to 5 (positivity). For each simile, we sum the sentiment scores for all lexicon words in the simile components, assigning positive/negative polarity depending on whether the sum is positive/negative. This method yields 460 positive and 423 negative similes.

Using MPQA Sentiment Lexicon Words: Our second training data set is created using the 2,718 positive words and 4,910 negative words from the MPQA lexicon (Wilson et al., 2005). We applied the CMU part-of-speech tagger for tweets (Owoputi et al., 2013) to match the MPQA parts-of-speech for each word. We assign positive/negative polarity to similes with more positive/negative lexicon words. This method yields 629 positive and 522 negative similes.

Using Sentiment Classifiers: We create our third training data set using a state-of-the-art sentiment classifier designed for tweets. For this, we reimplemented the NRC Canada sentiment classifier (Zhu et al., 2014) using the same set of features described by the authors. We use a Java implementation[2] of SVM from LIBLINEAR (Fan et al., 2008), with the original parameter values used by the NRC Canada system. We trained the sentiment classifier with all of the tweet training data from SemEval 2013 subtask B (Nakov et al., 2013). We label a simile as positive or negative if the sentiment classifier labels it as positive or negative, respectively. This method yields 1185 positive and 402 negative similes.

Using Sentiment in Surrounding Words: The previous approaches for labeling training instances will primarily identify similes that contain one or more strongly affective words. This

[2]http://liblinear.bwaldvogel.de/

can potentially bias the training data and limit the classifier's ability to learn to recognize affective similes that do not contain words with a positive or negative connotation. Therefore, we explore an additional approach where instead of judging the sentiment of the words in the simile, we analyze the words in the tweet surrounding the simile. We hypothesize that there are often redundant sentiment indicators in the tweet. For example, "I *hate* it when my room is as cold as Antarctica". For each simile, we identify all tweets that contain the simile and collect all of the words surrounding the simile in these tweets as a collective "context" for the tweet. We then count the number of distinct positive and negative sentiment words and compute the probability of positive or negative polarity given all the sentiment words surrounding a simile, and retain positive or negative similes with probability higher than a threshold (here, 0.7 to ensure high quality). As our sentiment lexicon, we combined the MPQA and the AFINN lexicon.

One issue is that when people feel amused (e.g., "he looks like a complete zombie, haha") or sarcastic (e.g., "my room feels like an igloo. great! LOL."), seemingly positive words in the context can be misleading because the sentiment is actually negative. As a simple measure to mitigate this issue, we manually removed a small set of laughter indicators from the lexicons (e.g., lol, haha).

This method yielded 492 positive and 181 negative similes.

Combination of Training Instances: As our last two training sets, we combined sets of instances labeled using the different methods above. As the fifth set, we combined training instances collected using the MPQA and AFINN lexicons and the NRC Canada sentiment classifier, which yielded a total of 2274 positive similes and 1347 negative similes. As our sixth set, we added the instances recognized from the surrounding words of a simile, producing the largest data set of 2766 positive and 1528 negative similes.

We also select neutral instances that are not identified as positive or negative by the above approaches and that also do not contain a sentiment lexicon (AFINN + MPQA) word in their collective context. For each approach, we then randomly select our final training instances for positive, negative and neutral classes maintaining the distribution of the development data. The final training data sizes are reported in Table 5.

4 Classifying Simile Polarity

Our goal is to create a classifier that can determine whether a simile expresses positive or negative affective polarity toward its subject. We present a classifier designed to label similes as Positive, Negative, or Neutral polarity. In this section, we describe the feature set and the classification framework of the supervised classifiers.

4.1 Feature Set

We extract three types of features from a simile, representing the lexical, semantic, and sentiment properties of the simile components.

4.1.1 Lexical Features

Unigrams: A binary feature indicates the presence of a unigram in a simile. This feature is not component specific, so the unigram can be from any simile component (tenor, event or vehicle).

Simile Components: We define a binary feature for each tenor, event and vehicle phrase in the data set. This feature is component specific, (e.g., "dog" as a tenor is a different feature from "dog" as a vehicle).

Paired Components: We use a binary feature for each pair of simile components. Our intuition is that a pair of components may indicate affective polarity when used together. For example, "event:feel, vehicle:ice box" is negative for many different tenors (e.g., house, room, hotel). Similarly, "tenor:person, vehicle:snail" is negative for many different events (e.g., move, run, drive).

Explicit Properties Associated with Vehicle: Sometimes a simile explicitly mentions a property that is common to the tenor and the vehicle (e.g., "my pillow is *soft* like a cloud"). Although the properties are not part of our triples because they are optional components, we can still use them as valuable features, whenever present in the original corpus. For each simile vehicle, we therefore extract all explicit properties mentioned with that vehicle in our corpus, and create a binary feature for each (e.g., "Jane swims like a dolphin" and "Jim runs like a cheetah" can both share the feature *fast*, if *fast* appears with both "dolphin" and "cheetah" in the corpus as an explicit property).

Vehicle Pre-modifiers: We use a binary feature for each noun or adjective pre-modifier that appears with the vehicle (the vehicle head noun itself is excluded). Our intuition is that the same pre-modifiers appearing with different vehicles indi-

cate the same affective polarity (e.g., "smells like *wet* dog" and "smells like *wet* clothes").

4.1.2 Semantic Features

Hypernym Class: We obtain up to two levels of hypernym classes for each simile component head, using WordNet (Miller, 1995). For words with multiple senses, we only use the first synset of a word from WordNet, for simplicity. Once the hypernym classes are obtained for a word, we no longer keep the level information, and use a binary feature to represent each hypernym class. Our intuition is that groups of similar words can be used in different similes with the same affective polarity (e.g., room, bedroom).

Perception Verb: We create a binary feature to indicate if the event component is a perception verb. Perception verbs are fairly common in similes (e.g., "*looks* like a model", "*smells* like garbage"). We use a set of the 5 most common perception verbs in similes (look, feel, sound, smell, taste).

4.1.3 Sentiment Features

We add sentiment features that can be recognized in the simile using existing sentiment resources. For this purpose, we combined the MPQA (Wilson et al., 2005), and the AFINN lexicon (Nielsen, 2011) to use as our sentiment lexicon.

Component Sentiment: We use 3 binary features (one for each component) to indicate the presence of a positive sentiment word, and 3 binary features to indicate the presence of a negative sentiment word in each simile component.

Explicit Property Sentiment: We use 2 numeric features that count the number of positive and negative properties that appear with the vehicle in our corpus. We look for the property words in the combined AFINN and MPQA sentiment lexicons.

Sentiment Classifier Label: We use 2 binary features (one for positive and one for negative) to represent the label that the NRC-Canada Sentiment Classifier assigns to a simile.

Simile Connotation Polarity: We use 2 binary features (one for positive and one for negative) to indicate the overall connotation of a simile. We count whether the number of positive (or negative) connotation words is greater in a simile using a Connotation Lexicon (Feng et al., 2013), which contains 30,881 words with positive connotation and 33,724 words with negative connotation.

4.2 Classification Framework

As our supervised classification algorithm, we use a linear SVM classifier from LIBLINEAR (Fan et al., 2008), with its default parameter settings. Our goal is to assign one of three labels to a simile: *Positive*, *Negative*, or *Neutral*. We train two binary classifiers, one for positive and one for negative polarity. For positive polarity, we use similes labeled *positive* as positive training instances, and similes labeled *negative* or *neutral* as the negative training instances. For the negative polarity classifier, we use similes labeled *negative* as the positive training instances, and similes labeled *positive* or *neutral* as the negative instances.

To classify a simile, we apply both classifiers. If the simile is labeled as *positive* or *negative*, then it is assigned that label. If the simile is labeled as both *positive* and *negative*, or not labeled as either, then it is assigned a *neutral* label. We did not create a classifier to solely identify neutral similes because neutral similes are much less common than positive/negative similes, making up only 8.7% of the extracted similes in our development set (Table 3). Consequently, obtaining a large set of neutral similes via manual annotation would have required substantially more manual annotation effort. Secondly, we did not have a good way to reliably identify neutral similes automatically.

5 Evaluation

5.1 Classification Performance with Manually Annotated Data

Table 4 presents the results for supervised classification with our manually annotated data set using 10-fold cross-validation. As baselines, we used existing sentiment resources as described in Section 3.2, but now applied to evaluation data. We also used the connotation lexicon (Feng et al., 2013) the same way as the MPQA sentiment lexicon (Wilson et al., 2005) to compare as an additional baseline. The top section of Table 4 shows how effective these four existing sentiment resources are at assigning polarity to similes. Although precision was sometimes very high, recall was low across the board.

The lower section of Table 4 shows results for our classifiers. We first trained a classifier using only the sentiment features in order to shed light on the effectiveness of traditional sentiment indicators. Row (a) in Table 4 shows that this classifier produces reasonable precision (65-72%) but recall

	Positive			Negative			Neutral		
	P	R	F	P	R	F	P	R	F
Sentiment Resource Baselines									
AFINN Lexicon	**88**	17	28	**95**	18	31	13	**95**	23
MPQA Lexicon	83	21	34	90	15	26	13	**95**	24
Connotation Lexicon	61	38	47	63	40	49	17	63	26
NRC Canada Sentiment Classifier	72	34	47	94	16	27	13	83	23
Affective Polarity Simile Classifiers									
(a) Sentiment Features	65	54	59	72	48	58	19	37	25
(b) Unigrams	73	52	61	74	70	72	21	47	29
(c) Unigrams + Other Lexical	73	56	63	75	76	75	**26**	45	**33**
(d) Unigrams + Other Lexical + Semantic	68	59	63	76	72	74	24	40	30
(e) Unigrams + Other Lexical + Semantic + Sentiment	75	**60**	**67**	77	**79**	**78**	25	40	31

Table 4: Results with Manually Annotated Training Data (P = Precision, R = Recall, F = F1-score).

Classifier	# of Training Instances			Positive			Negative			Neutral		
	Pos	**Neg**	**Neu**	P	R	F	P	R	F	P	R	F
(a) SVM with labeled data using AFINN	384	423	78	**78**	32	45	85	31	45	**14**	80	**24**
(b) SVM with labeled data using MPQA	475	522	94	65	44	53	81	27	41	12	59	20
(c) SVM with labeled data using NRC Canada	365	402	74	72	34	47	**94**	16	27	13	**83**	23
(d) SVM with labeled data from (a), (b), + (c)	1085	1193	216	69	50	58	88	30	45	13	62	22
(e) SVM with labeled data using sentiment in surrounding words	164	181	34	60	57	59	62	**57**	**60**	13	20	16
(f) SVM with labeled data from (a), (b), (c), + (e)	1221	1342	242	64	**61**	**62**	75	48	59	11	30	16

Table 5: Results with Automatically Labeled Training Data (P = Precision, R = Recall, F = F1-score).

levels only around 50% for both positive and negative polarity. The Neutral class has extremely low precision, which indicates that many unrecognized positive and negative similes are being classified as Neutral.

Row (b) shows the results for a baseline classifier trained only with unigram features. Unigrams perform substantially better than the sentiment features for negative polarity, but only slightly better for positive polarity. Row (c) shows that the additional lexical features described in Section 4.1.1 further improve performance.

Row (d) shows that adding the semantic features did not improve performance. One reason could be that some WordNet hypernym classes are very specific and may not generalize well. Also, similes can have different polarities with vehicle words from the same general semantic class (e.g., "he runs like a *cheetah*" vs "he runs like a *turtle*").

Finally, Row (e) shows that adding the sentiment features along with all the other features yields a precision gain for positive polarity and a recall gain for negative polarity. Overall, the full feature set improves the F score from 61% to 67% for positive polarity, and from 72% to 78% for negative polarity, over the unigram baseline.

5.2 Classification Performance with Automatically Acquired Training Data

Table 5 shows the performance of the classifiers (using our full feature set) when they are trained with automatically acquired training instances. The upper section of Table 5 shows results using training instances labeled by three different sentiment resources. Row (d) shows that combining the training instances labeled by all three resources produces the best results.

Row (e) of Table 5 shows the performance of the classifiers when they are trained with instances selected by analyzing sentiment in the surrounding words of the similes. We observe a substantial recall gain, which validates our hypothesis that similes obtained by recognizing sentiment in their surrounding words provide the classifier with a more diverse set of training examples. Finally, Row (f) shows that using both types of training instances further improves performance for positive polarity, and increases precision for negative polarity but with some loss of recall.

Comparing these results with Table 4, we see that there is still a gap between the performance of classifiers trained with manually annotated data versus automatically acquired data. However, the classifiers trained with automatically acquired data produce substantially higher F scores than all of

the baseline systems in Table 4. Using automatically acquired training data is a practical approach for creating simile classifiers for specific domains, such as Amazon product reviews (e.g., "headphone sounds like garbage", or "each song is like a snow-flake") which were studied in previous work on figurative comparisons in similes (Niculae and Danescu-Niculescu-Mizil, 2014).

5.3 Impact of Training Data Size

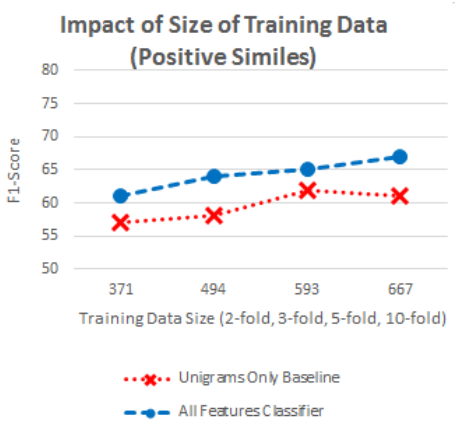


Figure 1: Learning Curve for Positive Similes.

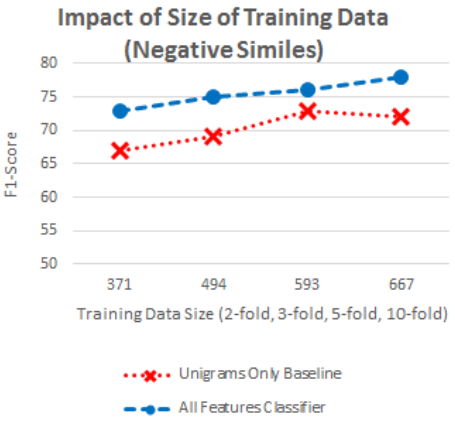


Figure 2: Learning Curve for Negative Similes.

We also generated learning curves to determine how much the size of the training set matters. Figures 1 and 2 show the performance of classifiers trained using varying amounts of manually annotated data. We show results for the classifiers trained only with unigram features and classifiers trained with our full feature set, for positive similes in Figure 1 and negative similes in Figure 2. The results were produced from 2-fold, 3-fold, 5-fold and 10-fold cross-validation experiments, with the size of the corresponding training sets shown on the X-axis. These figures show that the classifiers with unigram features hit a plateau at about 600 training instances. However the classifiers with the full feature set continually benefited from more training data. Table 6 presents a sample of similes where the vehicle appears only once in our data set. The unigram-based classifier could not classify these instances, but the classifier with the full feature set could.

Positive	Negative
PERSON, feel, superhero	PERSON, feel, old woman
PERSON, be, friend	PERSON, be, hurricane
beast, look, beauty	IT, feel, eternity
PERSON, feel, hero	PERSON, feel, peasant
PERSON, feel, champion	PERSON, eat, savage
PERSON, seem, sweetheart	PERSON, be, witch
IT, be, sleepover	PERSON, feel, prisoner
IT, be, reunion	IT, be, north pole
PERSON, feel, president	IT, feel, winter
ronaldo, be, messi	PERSON, be, wolf

Table 6: Similes with unique vehicles that were correctly classified using the full feature set.

6 Analysis and Discussion

We also conducted a qualitative analysis of our new corpus of similes and the behavior of the classifiers. We hypothesized that there are at least two reasons why similes might be difficult to classify. First, the interpretation of a simile can be highly context-dependent and subjective, depending on the speaker or the perceiver. To illustrate, Table 7 presents examples of similes that can have different polarity depending on the speaker or perceiver's personal experience or location, and other subjective aspects of the context. For example, *it looks like snow* may be a good thing to someone who lives in Utah where people look forward to skiing, but a bad thing to someone living in Boston during the winter of 2015. Similarly *she smells like a baby* is typically positive to new mothers, but was viewed as negative by the Mechanical Turk annotators.

Polarity	Simile	Context
positive	PERSON, smell, baby	young mother
negative	PERSON, smell, baby	MTurkers
negative	IT, look, snow	lives in Boston
positive	IT, look, snow	lives in Utah
negative	IT, look, rain	lives in England
positive	IT, look, rain	lives in California

Table 7: Similes with Context-dependent Polarity.

Second, we hypothesized that the polarity of a simile might interact with the distinction made in previous work between figurative and literal

uses of similes (Bredin, 1998; Addison, 1993), for example Niculae and Danescu-Niculescu-Mizil (2014) showed that sentiment and figurative comparisons are strongly correlated. Thus our expectation was that most literal comparisons would be neutral while most figurative comparisons would carry polarity. To explore this issue, we conducted an informal analysis of the 378 similes in our development data set to examine the literal vs. figurative distinction. For this analysis, we looked at the simile component triples as well as the context of ten tweets in which the simile appeared.

Our analysis suggests that the picture is more complex than we initially hypothesized. We found that, 1) the distinction between positive and negative similes in our data is orthogonal to the figurative vs. literal distinction, 2) some similes are used both figuratively and literally, and cannot be differentiated without context, 3) even in cases when all sample uses were literal, it is easy to invent contexts where the simile might be used figuratively, and vice versa, and 4) for a particular instance (simile + context), it is usually possible to tell whether a figurative or literal use is intended by examining the simile context, but some cases remain ambiguous. Table 8 shows examples of some similes that we identified as being figurative, literal, or both depending on context.

Use	Polarity	Simile
fig	positive	house, smell, heaven
fig	positive	PERSON, look, queen
fig	negative	PERSON, look, tomato
lit	negative	hair, smell, smoke
lit	neutral	PERSON, look, each other
both	neutral	house, smell, pizza
both	negative	IT, smell, skunk
both	negative	PERSON, look, frankenstein

Table 8: Similes with figurative (fig) or literal (lit) interpretation, or ambiguous depending on the context.

These observations reinforce the difficulty with making the figurative/literal distinction noted by Niculae and Danescu-Niculescu-Mizil (2014), whose annotation task required Turkers to label comparisons on a scale of 1 to 4 ranging from very literal to very figurative. Even with Master Turkers, a qualification task, filtering annotators by gold standard items, and collapsing scalar 1,2 values to literal and 3,4 values to figurative, the inter-annotator agreement with Fleiss' κ was 0.54. They note that out of 2400 automatically extracted comparison candidates, only 12% end up being selected confidently as figurative comparisons.

Selected cases that the classifiers fail on are further illustrated in Table 9. Examples S1 to S9 could be related to the difficulties noted above with subjectivity of interpretation. Many people for example like the smell of coffee and pizza, but perhaps not when a person smells that way. Similarly, *a baby* is often positive as a vehicle, but smelling and sounding like a baby may not be positive depending on the circumstances, while the positive or negative interpretation of *sounding like a pirate* and *looking like a pirate* might also be context dependent.

ID	Simile	Gold	Man	Auto
S1	PERSON, smell, coffee	neg	pos	pos
S2	PERSON, smell, pizza	pos	neg	neut
S3	IT, smell, pizza	neut	pos	neut
S4	PERSON, sleep, baby	pos	pos	neut
S5	PERSON, smell, baby	neg	neg	neut
S6	PERSON, feel, baby	pos	neg	pos
S7	PERSON, sound, baby	neg	pos	neut
S8	PERSON, sound, pirate	pos	neg	neg
S9	PERSON, look, pirate	neg	neg	neut

Table 9: Error Analysis of Classifier Output (Man = Classifier trained with manually annotated instances, Auto = Classifier trained with automatically annotated instances).

7 Conclusions

Similes are one example of a tractable case of sentiment-bearing expressions that are not recognized well by current sentiment analysis tools or lexicons. Making progress on sentiment analysis may require tackling many different types of linguistic phenomena such as this one. To this end, we have presented a simile data set labeled with affective polarity and have presented a supervised classification framework for recognizing affective polarity in similes. We have also presented our experiments with both manually labeled and automatically acquired training instances. We have shown that with manually labeled data, our feature set can substantially improve performance over a unigram only baseline. We have also shown that good performance can be achieved with automatically acquired training instances, when manually labeled data may not be available.

Acknowledgments

This material is based upon work supported in part by the National Science Foundation under grants IIS-1450527 and IIS-1302668.

References

Catherine Addison. 1993. From literal to figurative: An introduction to the study of simile. *College English*, pages 402–419.

Johan Bollen, Huina Mao, and Alberto Pepe. 2011. Modeling public mood and emotion: Twitter sentiment and socio-economic phenomena.

Hugh Bredin. 1998. Comparisons and similes. *Lingua*, 105(1):67–78.

Jean Carletta. 1996. Assessing agreement on classification tasks: The kappa statistic. *Comput. Linguist.*, 22(2):249–254, June.

Dmitry Davidov, Oren Tsur, and Ari Rappoport. 2010. Enhanced sentiment learning using twitter hashtags and smileys. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 241–249. Association for Computational Linguistics.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. Liblinear: A library for large linear classification. *J. Mach. Learn. Res.*, 9:1871–1874, June.

Song Feng, Jun Seok Kang, Polina Kuznetsova, and Yejin Choi. 2013. Connotation lexicon: A dash of sentiment beneath the surface meaning. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1774–1784, Sofia, Bulgaria, August. Association for Computational Linguistics.

David Fishelov. 2007. Shall i compare thee? simile understanding and semantic categories. *Journal of literary semantics*, 36(1):71–87.

Aniruddha Ghosh, Guofu Li, Tony Veale, Paolo Rosso, Ekaterina Shutova, John Barnden, and Antonio Reyes. 2015. Semeval-2015 task 11: Sentiment analysis of figurative language in twitter. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 470–478, Denver, Colorado, June. Association for Computational Linguistics.

Alec Go, Richa Bhayani, and Lei Huang. 2009. Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford*, pages 1–12.

Patrick Hanks. 2005. Similes and sets: The english preposition like. *Languages and Linguistics: Festschrift for Fr. Cermak. Charles University, Prague*.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177. ACM.

Long Jiang, Mo Yu, Ming Zhou, Xiaohua Liu, and Tiejun Zhao. 2011. Target-dependent twitter sentiment classification. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 151–160. Association for Computational Linguistics.

Yohan Jo and Alice H Oh. 2011. Aspect and sentiment unification model for online review analysis. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 815–824. ACM.

Efthymios Kouloumpis, Theresa Wilson, and Johanna Moore. 2011. Twitter sentiment analysis: The good the bad and the omg! *ICWSM*, 11:538–541.

Bin Li, Haibo Kuang, Yingjie Zhang, Jiajun Chen, and Xuri Tang. 2012. Using similes to extract basic sentiments across languages. In *Web Information Systems and Mining*, pages 536–542. Springer.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60.

George A. Miller. 1995. Wordnet: A lexical database for english. *Commun. ACM*, 38(11):39–41, November.

Saif M. Mohammad, Svetlana Kiritchenko, and Xiaodan Zhu. 2013. Nrc-canada: Building the state-of-the-art in sentiment analysis of tweets. In *Proceedings of the seventh international workshop on Semantic Evaluation Exercises (SemEval-2013)*, Atlanta, Georgia, USA, June.

Preslav Nakov, Sara Rosenthal, Zornitsa Kozareva, Veselin Stoyanov, Alan Ritter, and Theresa Wilson. 2013. Semeval-2013 task 2: Sentiment analysis in twitter. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 312–320, Atlanta, Georgia, USA, June. Association for Computational Linguistics.

Vlad Niculae and Cristian Danescu-Niculescu-Mizil. 2014. Brighter than gold: Figurative language in user generated comparisons. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2008–2018. Association for Computational Linguistics.

Finn Årup Nielsen. 2011. "A new ANEW: Evaluation of a word list for sentiment analysis in microblogs",. In *ESWC2011 Workshop on Making Sense of Microposts: Big things come in small packages*.

Olutobi Owoputi, Chris Dyer, Kevin Gimpel, Nathan Schneider, and Noah A. Smith. 2013. Improved part-of-speech tagging for online conversational text with word clusters. In *In Proceedings of NAACL*.

Anthony M Paul. 1970. Figurative language. *Philosophy & Rhetoric*, pages 225–248.

V. Punyakanok and D. Roth. 2001. The use of classifiers in sequential inference. In *NIPS*, pages 995–1001. MIT Press.

Matthew Purver and Stuart Battersby. 2012. Experimenting with distant supervision for emotion classification. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 482–491. Association for Computational Linguistics.

Vassiliki Rentoumi, George Giannakopoulos, Vangelis Karkaletsis, and A. George Vouros. 2009. Sentiment analysis of figurative language using a word sense disambiguation approach. In *Proceedings of the International Conference RANLP-2009*, pages 370–375. Association for Computational Linguistics.

Tim Rumbell, John Barnden, Mark Lee, and Alan Wallington. 2008. Affect in metaphor: Developments with wordnet. In *Proceedings of the AISB Convention on Communication, Interaction and Social Intelligence*, volume 1, page 21.

Mike Thelwall, Kevan Buckley, and Georgios Paltoglou. 2011. Sentiment in twitter events. *Journal of the American Society for Information Science and Technology*, 62(2):406–418.

Tony Veale and Yanfen Hao. 2007. Learning to understand figurative language: from similes to metaphors to irony. In *Proceedings of CogSci*.

Tony Veale. 2012. A context-sensitive, multi-faceted model of lexico-conceptual affect. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pages 75–79. Association for Computational Linguistics.

Alan Wallington, Rodrigo Agerri, John Barnden, Mark Lee, and Tim Rumbell. 2011. Affect transfer by metaphor for an intelligent conversational agent. In *Affective Computing and Sentiment Analysis*, pages 53–66. Springer.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pages 347–354, Stroudsburg, PA, USA. Association for Computational Linguistics.

Xiaodan Zhu, Svetlana Kiritchenko, and Saif Mohammad. 2014. Nrc-canada-2014: Recent improvements in the sentiment analysis of tweets. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 443–447, Dublin, Ireland, August. Association for Computational Linguistics and Dublin City University.

Cross-document Event Coreference Resolution based on Cross-media Features

Tongtao Zhang[1], Hongzhi Li[2], Heng Ji[1], Shih-Fu Chang[2]
[1]Computer Science Department, Rensselaer Polytechnic Institute
{zhangt13, jih}@rpi.edu
[2]Department of Computer Science, Columbia University
{hongzhi.li, shih.fu.chang}@columbia.edu

Abstract

In this paper we focus on a new problem of event coreference resolution across television news videos. Based on the observation that the contents from multiple data modalities are complementary, we develop a novel approach to jointly encode effective features from both closed captions and video key frames. Experiment results demonstrate that visual features provided 7.2% absolute F-score gain on state-of-the-art text based event extraction and coreference resolution.

1 Introduction

TV news is the medium that broadcasts events, stories and other information via television. The broadcast is conducted in programs with the name of "**Newscast**". Typically, newscasts require one or several anchors who are introducing stories and coordinating transition among topics, reporters or journalists who are presenting events in the fields and scenes that are captured by cameramen. Similar to newspapers, the same stories are often reported by multiple newscast agents. Moreover, in order to increase the impact on audience, the same stories and events are reported for mutliple times. TV audience passively receives redundant information, and often has difficulty in obtaining clear and useful digest of ongoing events. These properties lead to needs for automatic methods to cluster information and remove redundancy. We propose a new research problem of event coreference resolution across multiple news videos.

To tackle this problem, a good starting point is processing the **C**losed **C**aptions (**CC**) which is accompanying videos in newcasts. The **CC** is either generated by automatic speech recognition (ASR) systems or transcribed by a human stenotype operator who inputs phonetics which are instantly and automatically translated into texts, where events can be extracted. There exist some previous event coreference resolution work such as (Chen and Ji, 2009b; Chen et al., 2009; Lee et al., 2012; Bejan and Harabagiu, 2010). However, they only focused on formally written newswire articles and utilized textual features. Such approaches do not perform well on CC due to (1). the propagated errors from upper stream components (e.g., automatic speech/stenotype recognition and event extraction); (2). the incompleteness of information. Different from written news, newscasts are often limited in time due to fixed TV program schedules, thus, anchors and journalists are trained and expected to organize reports

Figure 1: Similar visual contents improve detection of a coreferential event pair which has a low text-based confidence score.
Closed Captions: "*It 's not clear when it was **killed**.*"; "*Jordan just executed two ISIS prisoners, direct retaliation for the capture of the **killing** Jordanian pilot.*"

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 201–206,
Lisbon, Portugal, 17-21 September 2015.

which are comprehensively informative with complementary visual and CC descriptions within a short time. These two sides have minimal overlapped information while they are inter-dependent. For example, anchors and reporters introduce the background story which are not presented in the videos, and thus the events extracted from CC often lack information about participants.

For example, as shown in Figure 1, these two **Conflict.Attack** event mentions are coreferential. However, in the first event mention, a mistake in Closed Caption ("*he was killed*" → "*it was killed*") makes event extraction and text based coreference systems unable to detect and link "*it*" to the entity of "*Jordanian pilot*". Fortunately, videos often illustrate brief descriptions by vivid visual contents. Moreover, diverse anchors, reporters and TV channels tend to use similar or identical video contents to describe the same story, even though they usually use different words and phrases. Therefore, the challenges in coreference resolution methods based on text information can be addressed by incorporating visual similarity. In this example, the visual similarity between the corresponding video frames is high because both of them show the scene of the Jordanian pilot.

Similar work such as (Kong et al., 2014), (Ramanathan et al., 2014), (Motwani and Mooney, 2012) and (Ramanathan et al., 2013) have explored methods of linking visual materials with texts. However, these methods mainly focus on connecting image concepts with entities in text mentions; and some of them do not clearly distinguish *entity* and *event* in the documents since the definition of visual concepts often require both of them. Moreover, the aforementioned work mainly focuses on improving visual contents recognition by introducing text features while our work will take the opposite route, which takes advantage of visual information to improve event coreference resolution.

In this paper, we propose to jointly incorporate features from both speech (textual) and video (visual) channels for the first time. We also build a newscast crawling system that can automatically accumulate video records and transcribe closed captions. With the crawler, we created a benchmark dataset which is fully annotated with cross-document coreferential events [1].

[1]Dataset can be found at http://www.ee.columbia.edu/dvmm/newDownloads.htm

2 Approach

2.1 Event Extraction

Given unstructured transcribed CC, we extract entities and events and present them in structured forms. We follow the terminologies used in **ACE** (Automatic Content Extraction) (NIST, 2005):

- Entity: an object or set of objects in the world, such as person, organization and facility.
- Entity mention: words or phrases in the texts that mention an entity.
- Event: a specific occurrence involving participants.
- Event trigger: the word that most clearly expresses an event's occurrence.
- Event argument: an entity, or a temporal expression or a value that has a certain role (e.g., Time-Within, Place) in an event.
- Event mention: a sentence (or a text span extent) that mentions an event, including a distinct trigger and arguments involved.

2.2 Text based Event Coreference Resolution

Coreferential events are defined as the same specific occurrence mentioned in different sentences, documents and transcript texts. Coreferential events should happen in the same place and within the same time period, and the entities involved and their roles should be identical. From the perspective of extracted events, each specific attribute and argument from those events should match. However, mentions for the same event may appear in forms of diverse words and phrases; and they do not always cover all arguments or attributes.

To tackle these challenges, we adopt a Maximum Entropy (MaxEnt) model as in (Chen and Ji, 2009b). We consider every pair of event mentions which share the same event type as a candidate and exploit features proposed in (Chen and Ji, 2009b; Chen et al., 2009). Note that the goal in (Chen and Ji, 2009b; Chen et al., 2009) was to resolve event coreference within the same document, whereas our scenario yields to a cross-document/video transcript setting, so we remove some improper and invalid features. We also investigated the approaches by (Lee et al., 2012) and (Bejan and Harabagiu, 2010), but the confidence estimation results from these alternative methods are not reliable. Moreover, the input of event coreference are automatic results from event extraction instead of gold standard, so the noise and errors significantly impact the corefer-

ence performance, especially for unsupervised approaches (Bejan and Harabagiu, 2010). Nevertheless, we still incorporate features from the aforementioned methods. Table 1 shows the features that constitute the input of the MaxEnt model.

2.3 Visual Similarity

Visual content provides useful cues complementary with those used in text-based approach in event coreference resolution. For example, two coreferential events typically show similar or even duplicate scenes, objects, and activities in the visual channel. Coherence of such visual content has been used in grouping multiple video shots into the same video story (Hsu et al., 2003), but it has not been used for event coreference resolution. Recent work in computer vision has demonstrated tremendous progress in large-scale visual content recognition. In this work, we adopt the state-of-the-art techniques (Krizhevsky et al., 2012) and (Simonyan and Zisserman, 2014) that train robust convolutional neural networks (CNN) over millions of web images to detect 20,000 semantic categories defined in ImageNet (Deng et al., 2009) from each image. The 2nd to the last layer features from such deep network can be considered as high-level visual representation that can be used to discriminate various semantic classes (scenes, objects, activity). It has been found effective in computing visual similarity between images, by directly computing the L2 distance of such features or through further metric learning. To compute the similarity between videos associated with two candidate event mentions, we sample multiple frames from each video and aggregate the similarity scores of the few most similar image pairs between the videos. Let $\{f_1^i, f_2^i, ..., , f_l^i\}$ be the key frames sampled from video V_i and $\{f_1^j, f_2^j, ..., , f_l^j\}$ be key frames sampled from video V_j. All the frames are resized to a fixed resolution of 256 x 256 and fed into our pretrained CNN model. We get the high-level visual representation $F_m = FC7(f_m)$ for each frame f_m from the output of the 2nd to the last fully connected layer (FC7) of CNN model. F_m is a 4096 dimension vector. The visual distance of frames f_m and f_n is defined by L_2 distance, which is

$$D_{mn} = ||FC7(f_m^i) - FC7(f_n^j)||_2. \quad (1)$$

The distance of video pair (V_i, V_j) is computed as

$$\bar{D}_{ij} = \frac{1}{k} * \sum_{(f_m, f_n)} D_{mn} \quad (2)$$

, where (f_m, f_n) is the top k of most similar frame pairs. In our experiment, we use $k = 3$. Such aggregation method among the top matches is intended to capture similarity between videos that share only partially overlapped content.

Each news video story typically starts with an introduction by an anchor person followed by news footages showing the visual scenes or activities of the event. Therefore, when computing visual similarity, it's important to exclude the anchor shot and focus on the story-related clips. Anchor frame detection (Hsu et al., 2003) is a well studied problem. In order to detect anchor frames automatically, a face detector is applied to all I-frames of a video. We can obtain the location and size of each detected face. After checking the temporal consistency of the detected faces within each shot, we get a set of candidate anchor faces. The detected face regions are further extended to regions of interest that may include hair and upper body. All the candidate faces detected from the same video are clustered based on their HSV color histogram. It is reasonable to assume that the most frequent face cluster is the one corresponding to the anchor faces. Once the anchor frames are detected, they are excluded and only the non-anchor frames are used to compute the visual similarity between videos associated with event mentions.

2.4 Joint Re-ranking

Using the visual distance calculated from Section 2.3, we can rerank the confidence values from Section 2.2 using the text-based MaxEnt model. We use the following empirical equation to adjust the confidence:

$$W'_{ij} = W_{ij} * e^{-\frac{\bar{D}_{ij}}{\alpha}+1}, \quad (3)$$

where W_{ij} denotes the original coreference confidence between event mentions i and j, D_{ij} denotes the visual distance between the corresponding video frames where the event mentions were spoken and α is a parameter which is used to adjust the impact of visual distance. In the current implementation, we empirically set it as the average of pair-wised visual distances between videos of all event coreference candidates. With this α

Category	Features	Remarks (EM_i: the first event mention, EM_j: the second event mention)
Baseline	type_subtype	pair of event type and subtype in EM_i
	trigger_pair	trigger pair of EM_i and EM_j
	pos_pair	part-of-speech pair of triggers of EM_i and EM_j
	nominal	1 if the trigger of EM_i is nominal
	nom_number	"plural" or "singular" if the trigger of EM_i is nominal
	pronominal	1 if the trigger of EM_i is pronominal
	exact_match	1 if the trigger spelling in EM_i matches that in EM_j
	stem_match	1 if the trigger stem in EM_i matches that in EM_j
	trigger_sim	the semantic similarity scores between triggers of EM_i and EM_j using WordNet(Miller, 1995)
Arguments	argument_match	1 if arguments holding the same roles in both EM_i and EM_j matches
Attributes	mod,pol,gen,ten	four event attributes in EM_i: modality, polarity, genericity and tense
	mod_conflict, pol_conflict, gen_conflict, ten_conflict	1 if the attributes of EM_i and EM_j conflict

Table 1: Features for Event Coreference Resolution

we generally enhance the confidence of event pairs with small visual distances and penalize those with large ones. An alternative way for setting the alpha parameter is through cross validation over separate data partitions.

3 Experiments

3.1 Data and Setting

We establish a system that actively monitors over 100 U.S. major broadcast TV channels such as ABC, CNN and FOX, and crawls newscasts from these channels for more than two years (Li et al., 2013a). With this crawler, we retrieve 100 videos and their correspondent transcribed CC with the topic of "ISIS"[2]. This system also temporally aligns the CC text with the transcribed text from automatic speech recognition following the methods in (Huang et al., 2003). This provides accurate time alignment between the CC text and the video frames. As CC consists of capitalized letters, we apply the true-casing tool from Standford CoreNLP (Manning et al., 2014) on CC. Then we apply a state-of-the-art event extraction system (Li et al., 2013b; Li et al., 2014) to extract event mentions from CC. We asked two human annotators to investigate all event pairs and annotate coreferential pairs as the ground truth. Kappa coefficient for measuring inter-annotator agreement is 74.11%. In order to evaluate our system performance, we rank the confidence scores of all event mention pairs and present the results in Precision vs. Detection Depth curve. Finally we find the video frames corresponding to the event mentions, remove the anchor frames and calculate the visual similarity between the videos. Our final dataset consists of 85 videos, 207 events and 848 event pairs, where 47 pairs are considered coreferential.

We adopt the MaxEnt-based coreference resolution system from (Chen and Ji, 2009b; Chen et al., 2009) as our baseline, and use ACE 2005 English Corpus as the training set for the model. A 5-fold cross-validation is conducted on the training set and the average f-score is 56%. It is lower than results from (Chen and Ji, 2009a) since we remove some features which are not available for the cross-document scenario.

3.2 Results

The peak F-score for the baseline system is 44.23% while our cross-media method boosts it to 51.43%. Figure 2 shows the improvement after incorporating the visual information. We adopt Wilcoxon signed-rank test to determine the significance between the pairs of precision scores at the same depth. The z-ratio is 3.22, which shows the improvement is significant.

For example, the event pair "*So why hasn't U.S. air strikes targeted Kobani within the city*

[2]abbreviation for *Islamic State of Iraq and Syria*

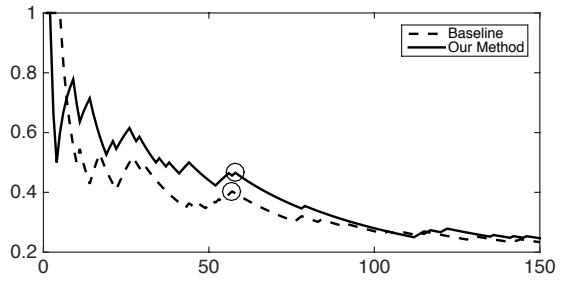


Figure 2: Performance comparison between baseline and our cross-media method on top 150 pairs. Circles indicate the peak F-scores.

limits" and "*Our strikes continue alongside our partners.*" was mistakenly considered coreferential by text features. In fact, the former "strikes" mentions the airstrike and the latter refers to the war or battle, therefore, they are not coreferential. The corresponding video shots demonstrate two different scenes: the former one shows bombing while the latter shows that the president is giving a speech about the strike. Thus the visual distance successfully corrected this error.

3.3 Error Analysis

However, from Figure 2 we can also notice that there are still some errors caused by the visual features. One major error type resides in the negative pairs with both "relatively" high textual coreference confidence scores and "relatively" high visual similarity. From the text side, the event pair contains similar events, for example: "*The Penn(tagon) says coalition air strikes in and around the Syrian city of Kobani have kill hundreds of ISIS fighters but more are streaming in even as the air campaign intensifies.*" and "*Throughout the day, explosions from coalition air strikes sent plums of smoke towering into the sky.*". They talk about two airstrikes during different time periods and are not coreferential, but the baseline system produces a high rank. Our current approach limits the image frames to those overlapped with the speech of an event mention, and in this error, both videos show "battle" scene, yielding a small visual distance. The aforementioned assumption that anchors and journalists tend to use similar videos when describing the same events , which may introduce risk of error caused by similar text event mentions with similar video shots. For such errors, one potential solution is to expand the video frame windows to capture more events and concepts from videos. Expanding the detection range to include visual events in the temporal neighborhood can also differentiate the events.

3.4 Discussion

A systematic way of choosing α in Equation 3 will be useful. One idea is to adapt the α value for different types of events, e.g., we expect some event types are more visually oriented than others and thus use a smaller α value.

We also notice the impact of the errors from the upstream event extraction system. According to (Li et al., 2014) the F-score of event trigger labeling is 65.3%, and event argument labeling is 45%. Missing arguments in events is a main problem, thus the performance on automatically extracted event mentions is significantly worse. About 20 more coreferential pairs could be detected if events and arguments are perfectly extracted.

4 Conclusions and Future Work

In this paper, we improved event coreference resolution on newscast speech by incorporating visual similarity. We also build a crawler that provides a benchmark dataset of videos with aligned closed captions. This system can also help create more datasets to conduct research on video description generation. In the future, we will focus on improving event extraction from texts by introducing more fine-grained cross-media information such as object, concept and event detection results from videos. Moreover, joint detection of events from both sides is our ultimate goal, however, we need to explore the mapping among events from both text and visual sides, and automatic detection of a wide range of objects and events from news video itself is still challenging.

Acknowledgement

This work was supported by the U.S. DARPA DEFT Program No. FA8750-13-2-0041, ARL NS-CTA No. W911NF-09-2-0053, NSF CAREER Award IIS-1523198, AFRL DREAM project, gift awards from IBM, Google, Disney and Bosch. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

References

Cosmin Adrian Bejan and Sanda Harabagiu. 2010. Unsupervised event coreference resolution with rich linguistic features. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1412–1422.

Zheng Chen and Heng Ji. 2009a. Event coreference resolution: Algorithm, feature impact and evaluation. In *Proceedings of Events in Emerging Text Types (eETTs) Workshop, in conjunction with RANLP, Bulgaria*.

Zheng Chen and Heng Ji. 2009b. Graph-based event coreference resolution. In *Proceedings of the 2009 Workshop on Graph-based Methods for Natural Language Processing*, TextGraphs-4, pages 54–57.

Zheng Chen, Heng Ji, and Robert Haralick. 2009. A pairwise event coreference model, feature impact and evaluation for event coreference resolution. In *Proceedings of the Workshop on Events in Emerging Text Types*, eETTs '09, pages 17–22.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 2009.*, pages 248–255.

Winston Hsu, Shih-Fu Chang, Chih-Wei Huang, Lyndon Kennedy, Ching-Yung Lin, and Giridharan Iyengar. 2003. Discovery and fusion of salient multimodal features toward news story segmentation. In *Electronic Imaging 2004*, pages 244–258.

Chih-Wei Huang, Winston Hsu, and Shin-Fu Chang. 2003. Automatic closed caption alignment based on speech recognition transcripts. *Rapport technique, Columbia*.

Chen Kong, Dahua Lin, Mohit Bansal, Raquel Urtasun, and Sanja Fidler. 2014. What are you talking about? text-to-image coreference. In *Proceedings of 2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3558–3565.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

Heeyoung Lee, Marta Recasens, Angel Chang, Mihai Surdeanu, and Dan Jurafsky. 2012. Joint entity and event coreference resolution across documents. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 489–500.

Hongzhi Li, Brendan Jou, Jospeh G Ellis, Daniel Morozoff, and Shih-Fu Chang. 2013a. News rover: Exploring topical structures and serendipity in heterogeneous multimedia news. In *Proceedings of the 21st ACM international conference on Multimedia*, pages 449–450.

Qi Li, Heng Ji, and Liang Huang. 2013b. Joint event extraction via structured prediction with global features. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 73–82.

Qi Li, Heng Ji, Yu HONG, and Sujian Li. 2014. Constructing information networks using one single model. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1846–1851.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60.

George A. Miller. 1995. Wordnet: A lexical database for english. *Commun. ACM*, 38(11):39–41, November.

Tanvi S Motwani and Raymond J Mooney. 2012. Improving video activity recognition using object recognition and text mining. In *Proceedings of the 20th European Conference on Artificial Intelligence*, pages 600–605.

NIST. 2005. The ace 2005 evaluation plan. `http://www.itl.nist.gov/iad/mig/tests/ace/ace05/doc/ace05-evaplan.v3.pdf`.

Vignesh Ramanathan, Percy Liang, and Li Fei-Fei. 2013. Video event understanding using natural language descriptions. In *Proceedings of 2013 IEEE International Conference on Computer Vision*, pages 905–912.

Vignesh Ramanathan, Armand Joulin, Percy Liang, and Li Fei-Fei. 2014. Linking people in videos with their names using coreference resolution. In *Computer Vision–ECCV 2014*, pages 95–110.

Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

A Survey of Current Datasets for Vision and Language Research

Francis Ferraro[1*], **Nasrin Mostafazadeh**[2*], **Ting-Hao (Kenneth) Huang**[3], **Lucy Vanderwende**[4], **Jacob Devlin**[4], **Michel Galley**[4], **Margaret Mitchell**[4]

Microsoft Research

1 Johns Hopkins University, 2 University of Rochester, 3 Carnegie Mellon University,
4 Corresponding authors: {lucyv,jdevlin,mgalley,memitc}@microsoft.com

Abstract

Integrating vision and language has long been a dream in work on artificial intelligence (AI). In the past two years, we have witnessed an explosion of work that brings together vision and language from images to videos and beyond. The available corpora have played a crucial role in advancing this area of research. In this paper, we propose a set of quality metrics for evaluating and analyzing the vision & language datasets and categorize them accordingly. Our analyses show that the most recent datasets have been using more complex language and more abstract concepts, however, there are different strengths and weaknesses in each.

1 Introduction

Bringing together language and vision in one intelligent system has long been an ambition in AI research, beginning with SHRDLU as one of the first vision-language integration systems (Winograd, 1972) and continuing with more recent attempts on conversational robots grounded in the visual world (Kollar et al., 2013; Cantrell et al., 2010; Matuszek et al., 2012; Kruijff et al., 2007; Roy et al., 2003). In the past few years, an influx of new, large vision & language corpora, alongside dramatic advances in vision research, has sparked renewed interest in connecting vision and language. Vision & language corpora now provide alignments between visual content that can be recognized with Computer Vision (CV) algorithms and language that can be understood and generated using Natural Language Processing techniques.

Fueled in part by the newly emerging data, research that blends techniques in vision and in language has increased at an incredible rate. In just the past year, recent work has proposed methods for image and video captioning (Fang et al., 2014; Donahue et al., 2014; Venugopalan et al., 2015), summarization (Kim et al., 2015), reference (Kazemzadeh et al., 2014), and question answering (Antol et al., 2015; Gao et al., 2015), to name just a few. The newly crafted large-scale vision & language datasets have played a crucial role in defining this research, serving as a foundation for training/testing and helping to set benchmarks for measuring system performance.

Crowdsourcing and large image collections such as those provided by Flickr[1] have made it possible for researchers to propose methods for vision and language tasks alongside an accompanying dataset. However, as more and more datasets have emerged in this space, it has become unclear how different methods generalize beyond the datasets they are evaluated on, and what data may be useful for moving the field beyond a single task, towards solving larger AI problems.

In this paper, we take a step back to document this moment in time, making a record of the major available corpora that are driving the field. We provide a quantitative analysis of each of these corpora in order to understand the characteristics of each, and how they compare to one another. The quality of a dataset must be measured and compared to related datasets, as low quality data may distort an entire subfield. We propose a set of criteria for analyzing, evaluating and comparing the quality of vision & language datasets against each other. Knowing the details of a dataset compared to similar datasets allows researchers to define more precisely what task(s) they are trying to solve, and select the dataset(s) best suited to their goals, while being aware of the implications and biases the datasets could impose on a task.

We categorize the available datasets into three major classes and evaluate them against these cri-

*F.F. and N.M. contributed equally to this work.

[1] http://www.flickr.com

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 207–213,
Lisbon, Portugal, 17-21 September 2015.

teria. The datasets we present here were chosen because they are all available to the community and cover the data that has been created to support the recent focus on image captioning work. More importantly, we provide an evolving website[2] containing pointers and references to many more vision-to-language datasets, which we believe will be valuable in unifying the quickly expanding research tasks in language and vision.

2 Quality Criteria for Language & Vision Datasets

The quality of a dataset is highly dependent on the sampling and scraping techniques used early in the data collection process. However, the content of datasets can play a major role in narrowing the focus of the field. Datasets are affected by both *reporting bias* (Gordon and Durme, 2013), where the frequency with which people write about actions, events, or states does not directly reflect real-world frequencies of those phenomena; they are also affected by *photographer's bias* (Torralba and Efros, 2011), where photographs are somewhat predictable within a given domain. This suggests that new datasets may be useful towards the larger AI goal if provided alongside a set of quantitative metrics that show how they compare against similar corpora, as well as more general "background" corpora. Such metrics can be used as indicators of dataset bias and language richness. At a higher level, we argue that clearly defined metrics are necessary to provide quantitative measurements of how a new dataset compares to previous work. This helps clarify and benchmark how research is progressing towards a broader AI goal as more and more data comes into play.

In this section, we propose a set of such metrics that characterize vision & language datasets. We focus on methods to measure *language quality* that can be used across several corpora. We also briefly examine metrics for *vision quality*. We evaluate several recent datasets based on all proposed metrics in Section 4, with results reported in Tables 1, 2, and Figure 1.

2.1 Language Quality

We define the following criteria for evaluating the captions or instructions of the datasets:

• **Vocabulary Size** (*#vocab*), the number of unique vocabulary words.

• **Syntactic Complexity** (*Frazier*, *Yngve*) measures the amount of embedding/branching in a sentence's syntax. We report mean Yngve (Yngve, 1960) and Frazier measurements (Frazier, 1985); each provides a different counting on the number of nodes in the phrase markers of syntactic trees.

• **Part of Speech Distribution** measures the distribution of nouns, verbs, adjectives, and other parts of speech.

• **Abstract:Concrete Ratio** (*#Conc*, *#Abs*, *%Abs*) indicates the range of visual and non-visual concepts the dataset covers. Abstract terms are ideas or concepts, such as 'love' or 'think' and concrete terms are all the objects or events that are mainly available to the senses. For this purpose, we use a list of most common abstract terms in English (Vanderwende et al., 2015), and define concrete terms as all other words except for a small set of function words.

• **Average Sentence Length** (*Sent Len.*) shows how rich and descriptive the sentences are.

• **Perplexity** provides a measure of data skew by measuring how expected sentences are from one corpus according to a model trained on another corpus. We analyze perplexity (*Ppl*) for each dataset against a 5-gram language model learned on a generic 30B words English dataset. We further analyze pair-wise perplexity of datasets against each other in Section 4.

2.2 Vision Quality

Our focus in this survey is mainly on language, however, the characteristics of images or videos and their corresponding annotations is as important in vision & language research. The quality of vision in a dataset can be characterized in part by the variety of visual subjects and scenes provided, as well as the richness of the annotations (e.g., segmentation using bounding boxes (*BB*) or visual dependencies between boxes). Moreover, a vision corpus can use abstract or real images (*Abs/Real*).

3 The Available Datasets

We group a representative set of available datasets based on their content. For a complete list of datasets and their descriptions, please refer to the supplementary website.[2]

3.1 Captioned Images

Several recent vision & language datasets provide one or multiple captions per image. The captions

[2] http://visionandlanguage.net

of these datasets are either the original photo title and descriptions provided by online users (Ordonez et al., 2011; Thomee et al., 2015), or the captions generated by crowd workers for existing images. The former datasets tend to be larger in size and contain more contextual descriptions.

3.1.1 User-generated Captions

• **SBU Captioned Photo Dataset** (Ordonez et al., 2011) contains 1 million images with original user generated captions, collected in the wild by systematic querying of Flickr. This dataset is collected by querying Flickr for specific terms such as objects and actions and then filtered images with descriptions longer than certain mean length.

• **Déjà Images Dataset** (Chen et al., 2015) consists of 180K unique user-generated captions associated with 4M Flickr images, where one caption is aligned with multiple images. This dataset was collected by querying Flickr for 693 high frequency nouns, then further filtered to have at least one verb and be judged as "good" captions by workers on Amazon's Mechanical Turk (Turkers).

3.1.2 Crowd-sourced Captions

• **UIUC Pascal Dataset** (Farhadi et al., 2010) is probably one of the first datasets aligning images with captions. Pascal dataset contains 1,000 images with 5 sentences per image.

• **Flickr 30K Images** (Young et al., 2014) extends previous Flickr datasets (Rashtchian et al., 2010), and includes 158,915 crowd-sourced captions that describe 31,783 images of people involved in everyday activities and events.

• **Microsoft COCO Dataset (MS COCO)** (Lin et al., 2014) includes complex everyday scenes with common objects in naturally occurring contexts. Objects in the scene are labeled using per-instance segmentations. In total, this dataset contains photos of 91 basic object types with 2.5 million labeled instances in 328k images, each paired with 5 captions. This dataset gave rise to the CVPR 2015 image captioning challenge and is continuing to be a benchmark for comparing various aspects of vision and language research.

• **Abstract Scenes Dataset (Clipart)** (Zitnick et al., 2013) was created with the goal of representing real-world scenes with clipart to study scene semantics isolated from object recognition and segmentation issues in image processing. This removes the burden of low-level vision tasks. This dataset contains 10,020 images of children playing outdoors associated with total 60,396 descriptions.

3.1.3 Captions of Densely Labeled Images

Existing caption datasets provide images paired with captions, but such brief image descriptions capture only a subset of the content in each image. Measuring the magnitude of the reporting bias inherent in such descriptions helps us to understand the discrepancy between what we can learn for the specific task of image captioning versus what we can learn more generally from the photographs people take. One dataset useful to this end provides image annotation for content selection:

• **Microsoft Research Dense Visual Annotation Corpus** (Yatskar et al., 2014) provides a set of 500 images from the Flickr 8K dataset (Rashtchian et al., 2010) that are densely labeled with 100,000 textual labels, with bounding boxes and facets annotated for each object. This approximates "gold standard" visual recognition.

To get a rough estimate of the reporting bias in image captioning, we determined the percentage of top-level objects[3] that are mentioned in the captions for this dataset out of all the objects that are annotated. Of the average 8.04 available top-level objects in the image, each of the captions only reports an average of 2.7 of these objects.[4] A more detailed analysis of reporting bias is beyond the scope of this paper, but we found that many of the biases (e.g., people selection) found with abstract scenes (Zitnick et al., 2013) are also present with photos.

3.2 Video Description and Instruction

Video datasets aligned with descriptions (Chen et al., 2010; Rohrbach et al., 2012; Regneri et al., 2013; Naim et al., 2015; Malmaud et al., 2015) generally represent limited domains and small lexicons, which is due to the fact that video processing and understanding is a very compute-intensive task. Available datasets include:

• **Short Videos Described with Sentences** (Yu and Siskind, 2013) includes 61 video clips (each 35 seconds in length, filmed in three different

[3]This visual annotation consists of a two-level hierarchy, where multiple Turkers enumerated and located objects and stuff in each image, and these objects were then further labeled with finer-grained object information (*Has* attributes).

[4]We did not use an external synonym or paraphrasing resource to perform the matching between labels and captions, as the dataset itself provides paraphrases for each object: each object is labeled by multiple Turkers, who labeled *Isa* relations (e.g., "eagle" is a "bird").

		Size(k)		Language								Vision	
	Dataset	**Img**	**Txt**	**Frazier**	**Yngve**	**Vocab Size (k)**	**Sent Len.**	**#Conc**	**#Abs**	**%Abs**	**Ppl**	**(A)bs/ (R)eal**	**BB**
Balanced	**Brown**	-	52	18.5	77.21	47.7	20.82	40411	7264	15.24%	194	-	-
User-Gen	**SBU**	1000	1000	9.70	26.03	254.6	13.29	243940	9495	3.74%	346	R	-
	Deja	4000	180	4.13	4.71	38.3	4.10	34581	3714	9.70%	184	R	-
Crowd-sourced	**Pascal**	1	5	8.03	25.78	3.4	10.78	2741	591	17.74%	123	R	-
	Flickr30K	32	159	9.50	27.00	20.3	12.98	17214	3033	14.98%	118	R	-
	COCO	328	2500	9.11	24.92	24.9	11.30	21607	3218	12.96%	121	R	Y
	Clipart	10	60	6.50	12.24	2.7	7.18	2202	482	17.96%	126	A	Y
Video	**VDC**	2	85	6.71	15.18	13.6	7.97	11795	1741	12.86%	148	R	-
Beyond	**VQA**	10	330	6.50	14.00	6.2	7.58	5019	1194	19.22%	113	A/R	-
	CQA	123	118	9.69	11.18	10.2	8.65	8501	1636	16.14%	199	R	Y
	VML	11	360	6.83	12.72	11.2	7.56	9220	1914	17.19%	110	R	Y

Table 1: Summary of statistics and quality metrics of a sample set of major datasets. For Brown, we report Frazier and Yngve scores on automatically acquired parses, but we also compute them for the 24K sentences with gold parses: in this setting, the mean Frazier score is 15.26 while the mean Yngve score is 58.48.

outdoor environments), showing multiple simultaneous events between a subset of four objects: a person, a backpack, a chair, and a trash-can. Each video was manually annotated (with very restricted grammar and lexicon) with several sentences describing what occurs in the video.

• **Microsoft Research Video Description Corpus (MS VDC)** (Chen and Dolan, 2011) contains parallel descriptions (85,550 English ones) of 2,089 short video snippets (10-25 seconds long). The descriptions are one sentence summaries about the actions or events in the video as described by Amazon Turkers. In this dataset, both paraphrase and bilingual alternatives are captured, hence, the dataset can be useful translation, paraphrasing, and video description purposes.

3.3 Beyond Visual Description

Recent work has demonstrated that n-gram language modeling paired with scene-level understanding of an image trained on large enough datasets can result in reasonable automatically generated captions (Fang et al., 2014; Donahue et al., 2014). Some works have proposed to step beyond description generation, towards deeper AI tasks such as question answering (Ren et al., 2015; Malinowski and Fritz, 2014). We present two of these attempts below:

• **Visual Madlibs Dataset (VML)** (Yu et al., 2015) is a subset of 10,783 images from the MS COCO dataset which aims to go beyond describing which objects are in the image. For a given image, three Amazon Turkers were prompted to complete one of 12 fill-in-the-blank template questions, such as 'when I look at this picture, I feel –', selected automatically based on the image content. This dataset contains a total of 360,001 MadLib question and answers.

• **Visual Question Answering (VQA) Dataset** (Antol et al., 2015) is created for the task of open-ended VQA, where a system can be presented with an image and a free-form natural-language question (e.g., 'how many people are in the photo?'), and should be able to answer the question. This dataset contains both real images and abstract scenes, paired with questions and answers. Real images include 123,285 images from MS COCO dataset, and 10,000 clip-art abstract scenes, made up from 20 'paperdoll' human models with adjustable limbs and over 100 objects and 31 animals. Amazon Turkers were prompted to create 'interesting' questions, resulting in 215,150 questions and 430,920 answers.

• **Toronto COCO-QA Dataset (CQA)** (Ren et al., 2015) is also a visual question answering dataset, where the questions are automatically generated from image captions of MS COCO dataset. This dataset has a total of 123,287 images with 117,684 questions with one-word answer about objects, numbers, colors, or locations.

4 Analysis

We analyze the datasets introduced in Section 3 according to the metrics defined in Section 2, using the Stanford CoreNLP suite to acquire parses and part-of-speech tags (Manning et al., 2014). We also include the Brown corpus (Francis and Kucera, 1979; Marcus et al., 1999) as a reference point. We find evidence that the VQA dataset captures more abstract concepts than other datasets, with almost 20% of the words found in our abstract concept resource. The Deja corpus has the least number of abstract concepts, followed by COCO and VDC. This reflects differences in col-

	Brown	Clipart	Coco	Flickr30K	CQA	VDC	VQA	Pascal	SBU
Brown	237.1	99.6	560.8	405.0	354.039	187.3	126.5	47.8	621.5
Clipart	233.6	11.2	117.4	109.4	210.8	82.5	114.7	28.7	130.6
Coco	274.6	59.2	36.2	75.3	137.0	87.1	236.9	39.3	111.0
Flickr30K	247.8	78.5	54.3	37.8	181.5	72.1	192.2	39.9	125.0
CQA	489.4	186.1	137.0	244.5	33.8	259.0	72.1	74.9	200.1
VDC	200.5	52.4	61.5	51.1	289.9	30.0	180.1	28.7	154.5
VQA	425.9	368.8	366.8	665.8	317.7	455.0	19.6	119.3	281.0
Pascal	265.2	64.5	43.2	63.4	174.2	83.0	228.2	36.0	105.3
SBU	473.9	107.1	346.4	344.0	328.5	230.7	194.3	78.2	119.8
#vocab	*14.0k*	*1.1k*	*13k*	*9.4k*	*5.3k*	*4.9k*	*1.4k*	*1.0k*	*65.1k*

Table 2: Perplexities across corpora, where rows represent test sets (20k sentences) and columns training sets (remaining sentences). To make perplexities comparable, we used the same vocabulary frequency cutoff of 3. All models are 5-grams.

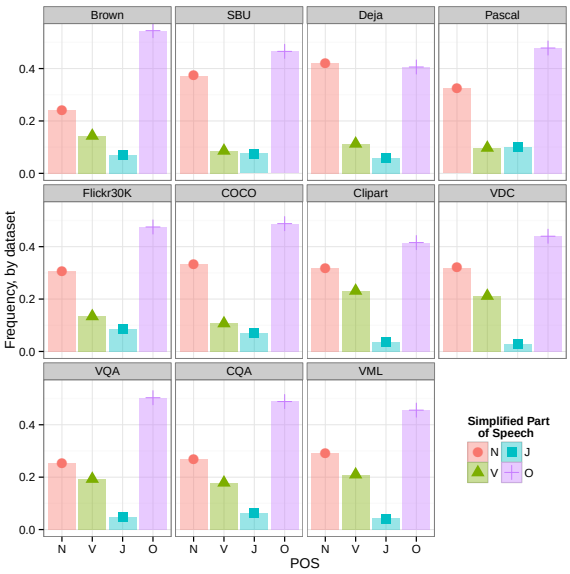


Figure 1: Simplified part-of-speech distributions for the eight datasets. We include the POS tags from the balanced Brown corpus (Marcus et al., 1999) to contextualize any very shallow syntactic biases. We mapped all nouns to "N," all verbs to "V," all adjectives to "J" and all other POS tags to "O."

lecting the various corpora: For example, the Deja corpus was collected to find *specifically* visual phrases that can be used to describe multiple images. This corpus also has the most syntactically simple phrases, as measured by both Frazier and Yngve; this is likely caused by the phrases needing to be general enough to capture multiple images.

The most syntactically complex sentences are found in the Flickr30K, COCO and CQA datasets. However, the CQA dataset suffers from a high perplexity against a background corpus relative to the other datasets, at odds with relatively short sentence lengths. This suggests that the automatic caption-to-question conversion may be creating unexpectedly complex sentences that are less reflective of general language usage. In contrast, the COCO and Flickr30K dataset's relatively high syntactic complexity is in line with their relatively high sentence length.

Table 2 illustrates further similarities between datasets, and a more fine-grained use of perplexity to measure the usefulness of a given training set for predicting words of a given test set. Some datasets such as COCO, Flickr30K, and Clipart are generally more useful as out-domain data compared to the QA datasets. Test sets for VQA and CQA are quite idiosyncratic and yield poor perplexity unless trained on in-domain data. As shown in Figure 1, the COCO dataset is balanced across POS tags most similarly to the balanced Brown corpus (Marcus et al., 1999). The Clipart dataset provides the highest proportion of verbs, which often correspond to actions/poses in vision research, while the Flickr30K corpus provides the most nouns, which often correspond to object/stuff categories in vision research.

We emphasize here that the distinction between a qualitatively good or bad dataset is task dependent. Therefore, all these metrics and the obtained results provide the researchers with an objective set of criteria so that they can make the decision whether a dataset is suitable to a particular task.

5 Conclusion

We detail the recent growth of vision & language corpora and compare and contrast several recently released large datasets. We argue that newly introduced corpora may measure how they compare to similar datasets by measuring *perplexity*, *syntactic complexity*, *abstract:concrete* word ratios, among other metrics. By leveraging such metrics and comparing across corpora, research can be sensitive to how datasets are biased in different directions, and define new corpora accordingly.

Acknowledgements We thank the three anonymous reviewers for their feedback, and Benjamin Van Durme for discussions on reporting bias.

References

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh. 2015. VQA: visual question answering. *arXiv preprint arXiv:1505.00468*.

Rehj Cantrell, Matthias Scheutz, Paul W. Schermerhorn, and Xuan Wu. 2010. Robust spoken instruction understanding for hri. In Pamela J. Hinds, Hiroshi Ishiguro, Takayuki Kanda, and Peter H. Kahn Jr., editors, *HRI*, pages 275–282. ACM.

David L. Chen and William B. Dolan. 2011. Collecting highly parallel data for paraphrase evaluation. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 190–200, Stroudsburg, PA, USA. Association for Computational Linguistics.

David L. Chen, Joohyun Kim, and Raymond J. Mooney. 2010. Training a multilingual sportscaster: Using perceptual context to learn language. *J. Artif. Int. Res.*, 37(1):397–436, January.

Jianfu Chen, Polina Kuznetsova, David Warren, and Yejin Choi. 2015. Déjà image-captions: A corpus of expressive descriptions in repetition. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 504–514, Denver, Colorado, May–June. Association for Computational Linguistics.

Jeff Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. 2014. Long-term recurrent convolutional networks for visual recognition and description. *CoRR*, abs/1411.4389.

Hao Fang, Saurabh Gupta, Forrest N. Iandola, Rupesh Srivastava, Li Deng, Piotr Dollár, Jianfeng Gao, Xiaodong He, Margaret Mitchell, John C. Platt, C. Lawrence Zitnick, and Geoffrey Zweig. 2014. From captions to visual concepts and back. *CoRR*, abs/1411.4952.

Ali Farhadi, Mohsen Hejrati, Mohammad Amin Sadeghi, Peter Young, Cyrus Rashtchian, Julia Hockenmaier, and David Forsyth. 2010. Every picture tells a story: Generating sentences from images. In *Proceedings of the 11th European Conference on Computer Vision: Part IV*, ECCV'10, pages 15–29, Berlin, Heidelberg. Springer-Verlag.

W Nelson Francis and Henry Kucera. 1979. Brown Corpus manual: Manual of information to accompany a standard corpus of present-day edited American English for use with digital computers. *Brown University, Providence, Rhode Island, USA*.

L. Frazier. 1985. Syntactic complexity. In D. R. Dowty, L. Karttunen, and A. M. Zwicky, editors, *Natural Language Parsing: Psychological, Computational, and Theoretical Perspectives*, pages 129–189. Cambridge University Press, Cambridge.

Haoyuan Gao, Junhua Mao, Jie Zhou, Zhiheng Huang, Lei Wang, and Wei Xu. 2015. Are you talking to a machine? dataset and methods for multilingual image question answering. *CoRR*, abs/1505.05612.

Jonathan Gordon and Benjamin Van Durme. 2013. Reporting bias and knowledge extraction. In *Automated Knowledge Base Construction (AKBC) 2013: The 3rd Workshop on Knowledge Extraction, at CIKM 2013*, AKBC'13.

Sahar Kazemzadeh, Vicente Ordonez, Mark Matten, and Tamara Berg. 2014. ReferItGame: Referring to Objects in Photographs of Natural Scenes. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 787–798, Doha, Qatar, October. Association for Computational Linguistics.

Gunhee Kim, Seungwhan Moon, and Leonid Sigal. 2015. Joint Photo Stream and Blog Post Summarization and Exploration. In *28th IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2015)*.

Thomas Kollar, Jayant Krishnamurthy, and Grant Strimel. 2013. Toward interactive grounded language acquisition. In *Robotics: Science and Systems*.

Geert-Jan M. Kruijff, Hendrik Zender, Patric Jensfelt, and Henrik I. Christensen. 2007. Situated dialogue and spatial organization: What, where... and why? *International Journal of Advanced Robotic Systems, Special Issue on Human and Robot Interactive Communication*, 4(2), March.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollar, and C. Lawrence Zitnick. 2014. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312.

Mateusz Malinowski and Mario Fritz. 2014. A multi-world approach to question answering about real-world scenes based on uncertain input. In *Advances in Neural Information Processing Systems 27*, pages 1682–1690.

Jonathan Malmaud, Jonathan Huang, Vivek Rathod, Nicholas Johnston, Andrew Rabinovich, and Kevin Murphy. 2015. Whats cookin? interpreting cooking videos using text, speech and vision. In *North American Chapter of the Association for Computational Linguistics Human Language Technologies (NAACL HLT 2015), May 31 - June 5, 2015, Denver, Colorado USA*.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60.

Mitchell Marcus, Beatrice Santorini, Mary Ann Marcinkiewicz, and Ann Taylor. 1999. Brown corpus, treebank-3.

Cynthia Matuszek, Nicholas FitzGerald, Luke Zettlemoyer, Liefeng Bo, and Dieter Fox. 2012. A Joint Model of Language and Perception for Grounded Attribute Learning. In *Proc. of the 2012 International Conference on Machine Learning*, Edinburgh, Scotland, June.

Iftekhar Naim, Young C. Song, Qiguang Liu, Liang Huang, Henry Kautz, Jiebo Luo, and Daniel Gildea. 2015. Discriminative unsupervised alignment of natural language instructions with corresponding video segments. In *North American Chapter of the Association for Computational Linguistics Human Language Technologies (NAACL HLT 2015), May 31 - June 5, 2015, Denver, Colorado USA*.

Vicente Ordonez, Girish Kulkarni, and Tamara L. Berg. 2011. Im2text: Describing images using 1 million captioned photographs. In *Neural Information Processing Systems (NIPS)*.

Cyrus Rashtchian, Peter Young, Micah Hodosh, and Julia Hockenmaier. 2010. Collecting image annotations using amazon's mechanical turk. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, CSLDAMT '10, pages 139–147, Stroudsburg, PA, USA. Association for Computational Linguistics.

Michaela Regneri, Marcus Rohrbach, Dominikus Wetzel, Stefan Thater, Bernt Schiele, and Manfred Pinkal. 2013. Grounding action descriptions in videos. *Transactions of the Association for Computational Linguistics (TACL)*, 1:25–36.

Mengye Ren, Ryan Kiros, and Richard Zemel. 2015. Question answering about images using visual semantic embeddings. In *Deep Learning Workshop, ICML 2015*.

Marcus Rohrbach, Sikandar Amin, Mykhaylo Andriluka, and Bernt Schiele. 2012. A database for fine grained activity detection of cooking activities. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, IEEE, June.

Deb Roy, Kai-Yuh Hsiao, and Nikolaos Mavridis. 2003. Conversational robots: Building blocks for grounding word meaning. In *Proceedings of the HLT-NAACL 2003 Workshop on Learning Word Meaning from Non-linguistic Data - Volume 6*, HLT-NAACL-LWM '04, pages 70–77, Stroudsburg, PA, USA. Association for Computational Linguistics.

Bart Thomee, David A. Shamma, Gerald Friedland, Benjamin Elizalde, Karl Ni, Douglas Poland, Damian Borth, and Li-Jia Li. 2015. The new data and new challenges in multimedia research. *arXiv preprint arXiv:1503.01817*.

A. Torralba and A. A. Efros. 2011. Unbiased look at dataset bias. In *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '11, pages 1521–1528, Washington, DC, USA. IEEE Computer Society.

Lucy Vanderwende, Arul Menezes, and Chris Quirk. 2015. An amr parser for english, french, german, spanish and japanese and a new amr-annotated corpus. Proceedings of NAACL 2015, June.

Subhashini Venugopalan, Huijuan Xu, Jeff Donahue, Marcus Rohrbach, Raymond Mooney, and Kate Saenko. 2015. Translating videos to natural language using deep recurrent neural networks. In *Proceedings the 2015 Conference of the North American Chapter of the Association for Computational Linguistics – Human Language Technologies (NAACL HLT 2015)*, pages 1494–1504, Denver, Colorado, June.

Terry Winograd. 1972. *Understanding Natural Language*. Academic Press, New York.

Mark Yatskar, Michel Galley, Lucy Vanderwende, and Luke Zettlemoyer. 2014. See no evil, say no evil: Description generation from densely labeled images. In *Proceedings of the Third Joint Conference on Lexical and Computational Semantics (*SEM 2014)*, pages 110–120, Dublin, Ireland, August. Association for Computational Linguistics and Dublin City University.

Victor H. Yngve. 1960. A model and an hypothesis for language structure. *Proceedings of the American Philosophical Society*, 104:444–466.

Peter Young, Alice Lai, Micah Hodosh, and Julia Hockenmaier. 2014. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *Transactions of the Association for Computational Linguistics*, 2:67–78.

Haonan Yu and Jeffrey Mark Siskind. 2013. Grounded language learning from video described with sentences. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 53–63, Sofia, Bulgaria. Association for Computational Linguistics. Best Paper Award.

Licheng Yu, Eunbyung Park, Alexander C. Berg, and Tamara L. Berg. 2015. Visual Madlibs: Fill in the blank Image Generation and Question Answering. *arXiv preprint arXiv:1506.00278*.

C. Lawrence Zitnick, Devi Parikh, and Lucy Vanderwende. 2013. Learning the visual interpretation of sentences. In *IEEE International Conference on Computer Vision, ICCV 2013, Sydney, Australia, December 1-8, 2013*, pages 1681–1688.

Combining Geometric, Textual and Visual Features for Predicting Prepositions in Image Descriptions

Arnau Ramisa*[1] **Josiah Wang***[2] **Ying Lu**[3] **Emmanuel Dellandrea**[3]
Francesc Moreno-Noguer[1] **Robert Gaizauskas**[2]

[1] Institut de Robòtica i Informàtica Industrial (UPC-CSIC), Barcelona, Spain
[2] Department of Computer Science, University of Sheffield, UK
[3] LIRIS, École Centrale de Lyon, France

{aramisa, fmoreno}@iri.upc.edu {j.k.wang, r.gaizauskas}@sheffield.ac.uk
{ying.lu, emmanuel.dellandrea}@ec-lyon.fr

Abstract

We investigate the role that geometric, textual and visual features play in the task of predicting a preposition that links two visual entities depicted in an image. The task is an important part of the subsequent process of generating image descriptions. We explore the prediction of prepositions for a pair of entities, both in the case when the labels of such entities are known and unknown. In all situations we found clear evidence that all three features contribute to the prediction task.

1 Introduction

In recent years, there has been an increased interest in the task of automatic generation of natural language image descriptions at sentence level, compared to earlier work that annotates images with a laundry list of terms (Duygulu et al., 2002). The task is important in that such detailed annotations are more informative and discriminative compared to isolated textual labels, and are essential for improved text and image retrieval.

The most standard approach to generating such descriptions involves first detecting instances of pre-defined concepts in the image, and then reasoning about these concepts to generate image descriptions e.g. (Kulkarni et al., 2011; Yang et al., 2011). Our work is also based on this paradigm. However, we assume that object instances have already been pre-detected by visual recognisers, and concentrate on a specific subtask of description generation. More specifically, given two visual entity instances where one could potentially act as a modifier to the other, we address the problem of identifying the appropriate preposition to connect these two entities (Figure 1). The inferred prepositional relations will subsequently act as an important intermediate representation towards the eventual goal of generating image descriptions.

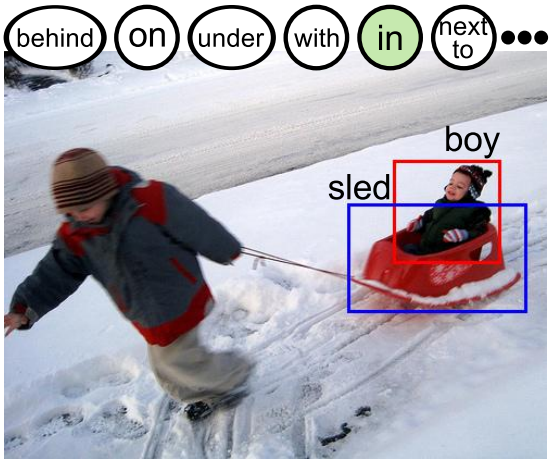


Figure 1: Given a subject *boy* and an object *sled* and their location in the image, what would the best preposition be to connect the two entities?

The main contribution of this paper is therefore to learn to predict the most suitable preposition given its context, and to learn this jointly from images and their descriptions. In particular, we concentrate on learning from (i) geometric relations between two visual entities from image annotations; (ii) textual features from textual descriptions; (iii) visual features from images. Previous work exists (Yang et al., 2011) that uses text corpora to 'guess' the prepositions given the context without considering the appropriate spatial relations between the entities in the image, signifying a gap between visual content and its corresponding description. For example, although *person on horse* might commonly occur in text corpora, a particular image might actually depict a person standing beside a horse. On the other hand, work that does consider the image content for generating prepositions (Kulkarni et al., 2011; Elliott and Keller, 2013) map geometric relations to a limited set of prepositions using manually defined rules,

*A. Ramisa and J. Wang contributed equally to this work.

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 214–220,
Lisbon, Portugal, 17-21 September 2015.

not as humans would naturally use them with a richer vocabulary. We would like to have the best of both worlds, by considering image content as well as textual information to select the preposition best used to express the relation between two entities. Our hypothesis is that the combination of geometric, textual and visual features can help with the task of predicting the most appropriate preposition, since incorporating geometric and visual information should help generate a relation that is consistent with the image content, whilst incorporating textual information should help generate a description that is consistent with natural language.

2 Related Work

The Natural Language Processing Community has significant interest in different aspects of prepositions. The Prepositions Project (Litkowski and Hargraves, 2005) analysed and produced a lexicon of English prepositions and their senses, and subsequently used them in the Word Sense Disambiguation of Prepositions task in SemEval-2007 (Litkowski and Hargraves, 2007). In SemEval-2012, Kordjamshidi et al. (2012) introduce the more fine-grained task of spatial role labelling to detect and classify spatial relations expressed by triples *(trajector, landmark, spatial_indicator)*. In the latest edition of SemEval-2015, the SpaceEval task (Pustejovsky et al., 2015) introduce further tasks of identifying spatial and motion signals, as well as spatial configurations/orientation and motion relation.

In work that links prepositions more strongly to image content, Gupta and Davis (2008) model prepositions implicitly to disambiguate image regions, rather than for predicting prepositions. Their work also require manual annotation of prepositional relations. In image description generation work, Kulkarni et al. (2011) manually map spatial relations to pre-defined prepositions, whilst Yang et al. (2011) predict prepositions from large-scale text corpora solely based on the complement term, with the prepositions constrained to describing scenes (*on the street*). Elliott and Keller (2013) define a list of eight spatial relations and their corresponding prepositional term for sentence generation. Although they also present alternative models that use text corpora for descriptions that are more human-like, they are limited to verbs and do not cover prepositions. Le et al. (2014) examine prepositions modifying human actions (verbs), and conclude that these relate to positional information to a certain extent. Other related work include training classifiers for prepositions with spatial relation features to improve image segmentation and detection (Fidler et al., 2013); this work is however limited to four prepositions.

3 Task Definition

We formally define the task of predicting prepositions as follows: Let P be the set of possible prepositions. Let L be the set of possible *landmark* entities acting as the complement of a preposition, and let T be the set of possible *trajector* entities modified by the prepositional phrase comprising a preposition and its landmark[1]. For example, for the phrase *person on bicycle*, *on* would be the preposition, *bicycle* the landmark, and *person* the trajector. For this paper, we constrain *trajector* and *landmark* to be entities that are visually identifiable in an image since we are interested in discovering the role of visual features and geometric configurations between two entities in the preposition prediction task.

Let $D = \{d_1, d_2, ..., d_N\}$ be the set of N observations, where each d_i for $i = 1, 2..., N$ is represented by $d_i = (x_i, y_i, r_i)$, where x_i and y_i are the feature representations for the trajector and the landmark entities respectively, and r_i the relative geometric feature between the two visual entities.

Given d_i, the objective of the preposition prediction task is to produce a ranked list of prepositions $(p_1, p_2, ...p_{|P|})$ according to how likely they are to express the appropriate spatial relation between the given trajector and landmark entities that are either known (Section 6.1) or only represented by visual features (Section 6.2).

4 Dataset

We base the preposition prediction task on two large-scale image datasets with human authored descriptions, namely MSCOCO (Lin et al., 2014) and Flickr30k (Young et al., 2014; Plummer et al., 2015). To extract instances of triples *(trajector, preposition, landmark)* from image descriptions, we used the Neural Network, transition-based dependency parser of Chen and Manning (2014) as implemented in Stanford CoreNLP (Manning et al., 2014). Dependencies signifying prepositional

[1]The terminologies *trajector* and *landmark* are adopted from spatial role labelling (Kordjamshidi et al., 2011)

Bounding Box feature (number of dimensions)
• Vector (x, y) from centroid of trajector to centroid of landmark, normalised by the size of the bounding box enclosing both objects (2)
• Area of trajector bounding box relative to landmark (1)
• Aspect ratio of each bounding box (2)
• Area of each bounding box w.r.t. enclosing box (2)
• Intersection over union of the bounding boxes (1)
• Euclidean distance between the trajector and landmark bounding boxes, normalised by the image size (1)
• Area of each bounding box w.r.t. the whole image (2)

Table 1: Geometric features derived from bounding boxes.

relations are retained where both the governor and its dependent overlap with the entity mentions in the descriptions, and where both mentions have corresponding bounding boxes. The MSCOCO validation set is further annotated to remove errors arising from dependency parsing (notably PP attachment errors), and is used as our clean test set. Our final dataset comprises 8,029 training and 3,431 test instances for MSCOCO, and 46,847 training and 20,010 test instances for Flickr30k. Details on how the triples were extracted from captions and matched to instances in images are available in the supplementary material.

We consider two variants of trajector and landmark terms in our experiments: (i) using the provided high level categories as terms (80 for MSCOCO and 8 for Flickr30k); (ii) using the terms occurring in the sentence directly, which constitute a bigger and more realistic challenge. For Flickr30k, the descriptive phrases may cause data sparseness (*the furry, black and white dog*). Thus, we extracted the lemmatised head word of each phrase, using a 'semantic head' variant of the head finding rules of Collins (2003) in Stanford CoreNLP. Entities from the same coreference chain are denoted with a common head noun chosen by majority vote among the group, with ties broken by the most frequent head noun in the corpus, and further ties broken at random.

5 Features

Geometric Features: Geometric features between a trajector and a landmark entity are derived from bounding box annotations. We defined an 11-dimensional vector of bounding box features, covering geometric relations such as distance, orientation, relative bounding box sizes and overlaps between bounding boxes (Table 1). We chose to use continuous features as we felt these may be more powerful and expressive compared to discrete, binned features. Despite some of these features being correlated, we left it to the classifier to determine the most useful features for discrimination without having to withhold any unnecessarily.

Textual features: We consider two textual features to encode the trajector and landmark terms w_i^t and w_i^l. The first feature is a one-hot indicator vector x_i^I and y_i^I for the trajector and landmark respectively, where $x_{i,t}^I = 1$ if index t corresponds to the trajector term w_i^t and 0 elsewhere (and similarly for landmark). As data sparseness may be an issue, we also explore an alternative textual feature which encodes the terms as word2vec embeddings (Mikolov et al., 2013). This encodes each term as a vector such that semantically related terms are close in the vector space. This allows information to be transferred across semantically related terms during training (e.g. information from *person on boat* can help predict the preposition that mediates *man* and *boat*).

Image Features: While it is ideal to have vision systems produce a firm decision about the visual entity instance detected in an image, in reality it may be beneficial to defer the decision by allowing several possible interpretations of the instance being detected. In such cases, we will not have a single concept label for the entity, but instead a high-level visual representation. For this scenario, we extracted visual representations from the final layer of a Convolutional Neural Network trained on ImageNet (Krizhevsky et al., 2012), and used them as representations for entity instances in place of textual features.

6 Preposition Prediction

Here we highlight interesting findings from experiments performed for the task of predicting prepositions for two different scenarios (Sections 6.1 and 6.2). Detailed results can be found in the supplementary material.

Evaluation metrics. As there may be more than one 'correct' preposition for a given context (*person on horse* and *person atop horse*), we propose the mean rank of the correct preposition as the main evaluation metric, as it accommodates

		IND	W2V	GF	IND+GF	W2V+GF	Baseline
Mean rank	MSCOCO (max rank 17)	1.45	1.43	1.72	1.44	**1.42**	2.14
	MSCOCO (balanced)	3.20	3.10	4.60	3.00	**2.90**	5.40
	Flickr30k (max rank 52)	1.91	1.87	2.35	1.88	**1.85**	2.54
	Flickr30k (balanced)	11.10	9.04	15.55	10.23	**8.90**	15.13
Accuracy	MSCOCO	79.7%	80.3%	68.4%	79.8%	**80.4%**	40.2%
	MSCOCO (balanced)	52.5%	**54.2%**	31.5%	52.7%	53.9%	11.9%
	Flickr30k	75.4%	75.2%	58.5%	**75.8%**	75.4%	53.7%
	Flickr30k (balanced)	24.6%	25.9%	9.0%	25.2%	**26.9%**	4.0%

Table 2: Top: Mean rank of the correct preposition (lower is better). Bottom: Accuracy with different feature configurations. All results are with the original trajector/landmark terms from descriptions. IND stands for Indicator Vectors, W2V for Word2Vec, and GF for Geometric Features. As baseline we rank the prepositions by their relative frequencies in the training dataset.

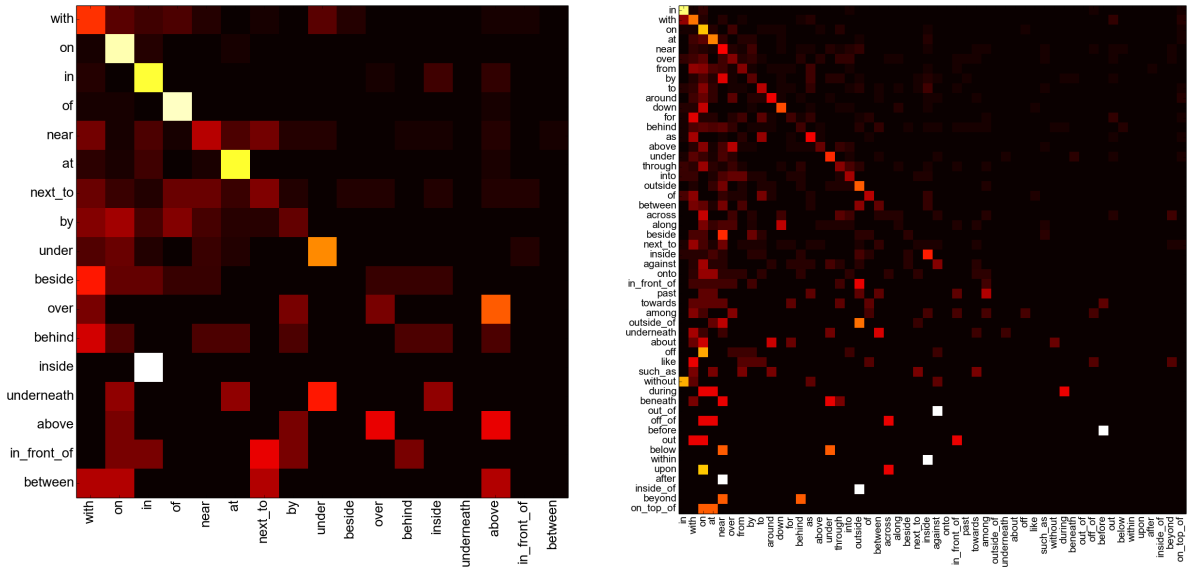


Figure 2: Normalised confusion matrices on the balanced test subsets for the two datasets (left: MSCOCO, right: Flickr30k), using geometric features and word2vec with the original terms.

multiple possible prepositions that may be equally valid. For completeness we also report classification accuracy results.

Baseline. As baseline, we rank the prepositions by their relative frequencies in the training dataset. We found this to be a sufficiently strong baseline, as ubiquitous prepositions such as *with* and *in* tend to occur frequently in the dataset.

6.1 Ranking with known entity labels

In this section, we focus on predicting the best preposition given the geometric and textual features of the trajector and landmark entities. This simulates the scenario of a vision detector providing a firm decision on the concept label for the detected entities. We use a multi-class logistic regression classifier (Fan et al., 2008), and concatenate multiple features into a single vector. We compare high-level categories and terms from descriptions as trajector/landmark labels. Prepositions are ranked in descending order of the classifier output scores.

We found a few prepositions (e.g. *with*) dominating the datasets. Thus, we also evaluated our models on a balanced subset where each preposition is limited to a maximum of 50 random test samples. The training samples are weighted according to their class frequency in order to train non-biased classifiers to predict this balanced test set. The results on both the original and balanced

Dataset	Prep (known labels)		Preposition		Trajector		Landmark	
	acc	rank	acc	rank	acc	rank	acc	rank
MSCOCO	79.8%	1.46 (17)	62.9%	1.92 (17)	65.6%	4.64 (74)	44.5%	7.30 (77)
Flickr30k	67.1%	2.16 (52)	61.7%	2.28 (52)	77.3%	1.43 (8)	66.4%	1.64 (8)

Table 3: Accuracy (**acc**) and mean rank (**rank**, with max rank in parenthesis) for each variable of the CRF model, trained using the high-level concept labels. Columns under **Prep (known labels)** refer to the results of predicting prepositions with the trajector and landmark labels fixed to the correct values.

test sets are compared.

As shown in Table 2, the system performed significantly better than the baseline in most cases. In general, geometric features perform better than the baseline, and when combined with text features further improve the results. In a per-preposition analysis, the geometric features show up to 14% improvement in the mean rank for Flickr30k.

In feature ablation tests on MSCOCO (balanced), we found the y component of the trajector to landmark vector to be important to most prepositions, especially for *under*, *above* and *on*. Other important geometric features include the final two features in Table 1 (Euclidean distance and area).

The benefit of the word2vec text feature is clear when moving from high-level categories to original terms from descriptions, where it consistently improves the mean rank (up to 25%). In contrast, the indicator vectors resulted in a less significant improvement, if not worse performance, when using the sparse original terms.

We also evaluated the relative importance of the trajector and the landmark, by withholding either from the textual feature vector. We found that the landmark plays a larger role in preposition prediction as omitting the trajector produces 10%-30% better results than omitting the landmark.

Figure 2 shows the confusion matrices of the best-performing systems. Note that many mistakes arise from prepositions that are often equally valid (e.g. predicting *near* instead of *next_to*).

6.2 Ranking with unknown entity labels

Here, we investigate the task of jointly predicting prepositions with the entity labels given geometric and visual features (*without* the trajector and landmark labels). This simulates the scenario of a vision detector output. For this structured prediction task, we use a 3-node chain CRF model[2], with the centre node representing the preposition and the two end nodes representing the trajector and landmark. We use *image features* for the entity nodes, and *geometric features* for the preposition node (Section 5). Due to computational constraints only high-level category labels are used, but as seen in Section 6.1, this may actually be *hurting* the performance.

Table 3 shows the results of the structured model used to predict the most likely *(trajector, preposition, landmark)* combination. To facilitate comparison with Section 6.1, column **Prep (known labels)** shows the results with the trajector and landmark labels as known conditions and fixed to the correct values, thus only needing to predict the preposition. The model achieved excellent performance considering the added difficulty of the task.

7 Conclusions and Future Work

We explored the role of geometric, textual and visual features in learning to predict a preposition given two bounding box instances in an image, and found clear evidence that all three features play a part in the task. Our system performs well even with uncertainties surrounding the entity labels. Future work could include non-prepositional terms like verbs, having prepositions modify verbs, adding word2vec embeddings to the structured prediction model, and providing stronger features – whether textual, visual or geometric.

Acknowledgments

This work was funded by the ERA-Net CHIST-ERA D2K VisualSense project (Spanish MINECO PCIN-2013-047, UK EPSRC EP/K019082/1 and French ANR Grant ANR-12-CHRI-0002-04) and the Spanish MINECO RobInstruct project TIN2014-58178-R. Ying Lu was also supported by the China Scholarship Council.

[2]We used the toolbox by Mark Schmidt: `http://www.cs.ubc.ca/~schmidtm/Software/UGM.html`

References

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 740–750, Doha, Qatar, October. Association for Computational Linguistics.

Michael Collins. 2003. Head-driven statistical models for natural language parsing. *Computational Linguistics*, 29(4):589–637, December.

Pinar Duygulu, Kobus Barnard, Nando de Freitas, and David A. Forsyth. 2002. Object recognition as machine translation: Learning a lexicon for a fixed image vocabulary. In *Proceedings of the European Conference on Computer Vision*, pages 97–112.

Desmond Elliott and Frank Keller. 2013. Image description using visual dependency representations. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1292–1302, Seattle, Washington, USA, October. Association for Computational Linguistics.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874.

Sanja Fidler, Abhishek Sharma, and Raquel Urtasun. 2013. A sentence is worth a thousand pixels. In *Proceedings of the IEEE Conference on Computer Vision & Pattern Recognition*.

Abhinav Gupta and Larry S. Davis. 2008. Beyond nouns: Exploiting prepositions and comparative adjectives for learning visual classifiers. In *Proceedings of the European Conference on Computer Vision*, pages 16–29.

Parisa Kordjamshidi, Martijn van Otterlo, and Marie-Francine Moens. 2011. Spatial role labeling: Towards extraction of spatial relations from natural language. *ACM Transactions on Speech and Language Processing*, 8(3):article 4, 36 p.

Parisa Kordjamshidi, Steven Bethard, and Marie-Francine Moens. 2012. Semeval-2012 task 3: Spatial role labeling. In **SEM 2012: The First Joint Conference on Lexical and Computational Semantics – Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, pages 365–373, Montréal, Canada, 7-8 June. Association for Computational Linguistics.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J.C. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105.

Girish Kulkarni, Visruth Premraj, Sagnik Dhar, Siming Li, Yejin Choi, Alexander C. Berg, and Tamara L. Berg. 2011. Baby talk: Understanding and generating image descriptions. In *Proceedings of the IEEE Conference on Computer Vision & Pattern Recognition*.

Dieu-Thu Le, Jasper Uijlings, and Raffaella Bernardi. 2014. TUHOI: Trento Universal Human Object Interaction dataset. In *Proceedings of the Third Workshop on Vision and Language*, pages 17–24, Dublin, Ireland, August. Dublin City University and the Association for Computational Linguistics.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. 2014. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312.

Kenneth C. Litkowski and Orin Hargraves. 2005. The preposition project. In *ACL-SIGSEM Workshop on The Linguistic Dimensions of Prepositions and Their Use in Computational Linguistic Formalisms and Applications*, pages 171–179.

Kenneth C. Litkowski and Orin Hargraves. 2007. Semeval-2007 task 06: Word-sense disambiguation of prepositions. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 24–29, Prague, Czech Republic, June. Association for Computational Linguistics.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, Baltimore, Maryland, June. Association for Computational Linguistics.

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

Bryan Plummer, Liwei Wang, Chris Cervantes, Juan Caicedo, Julia Hockenmaier, and Svetlana Lazebnik. 2015. Flickr30k entities: Collecting region-to-phrase correspondences for richer image-to-sentence models. *CoRR*, abs/1505.04870.

James Pustejovsky, Parisa Kordjamshidi, Marie-Francine Moens, Aaron Levine, Seth Dworman, and Zachary Yocum. 2015. Semeval-2015 task 8: Spaceeval. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 884–894, Denver, Colorado, June. Association for Computational Linguistics.

Yezhou Yang, Ching Teo, Hal Daumé III, and Yiannis Aloimonos. 2011. Corpus-guided sentence generation of natural images. In *Proceedings of the Conference on Empirical Methods in Natural Language*

Processing (EMNLP), pages 444–454. Association for Computational Linguistics.

Peter Young, Alice Lai, Micah Hodosh, and Julia Hockenmaier. 2014. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *Transactions of the Association for Computational Linguistics*, 2:67–78, February.

On a Strictly Convex IBM Model 1

Andrei Simion
Columbia University
New York, NY, 10027
aas2148@columbia.edu

Michael Collins*
Columbia University
Computer Science
New York, NY, 10027
mc3354@columbia.edu

Clifford Stein
Columbia University
IEOR Department
New York, NY, 10027
cs2035@columbia.edu

Abstract

IBM Model 1 is a classical alignment model. Of the first generation word-based SMT models, it was the only such model with a concave objective function. For concave optimization problems like IBM Model 1, we have guarantees on the convergence of optimization algorithms such as Expectation Maximization (EM). However, as was pointed out recently, the objective of IBM Model 1 is not strictly concave and there is quite a bit of alignment quality variance within the optimal solution set. In this work we detail a strictly concave version of IBM Model 1 whose EM algorithm is a simple modification of the original EM algorithm of Model 1 and does not require the tuning of a learning rate or the insertion of an l_2 penalty. Moreover, by addressing Model 1's shortcomings, we achieve AER and F-Measure improvements over the classical Model 1 by over 30%.

1 Introduction

The IBM translation models were introduced in (Brown et al., 1993) and were the first-generation Statistical Machine Translation (SMT) systems. In the current pipeline, these word-based models are the seeds for more sophisticated models which need alignment tableaus to start their optimization procedure. Among the original IBM Models, only IBM Model 1 can be formulated as a concave optimization problem. Recently, there has been some research on IBM Model 2 which addresses either the model's non-concavity (Simion et al., 2015) or over parametrization (Dyer et al., 2013). We make the following contributions in this paper:

- We utilize and expand the mechanism introduced in (Simion et al., 2015) to construct *strictly* concave versions of IBM Model 1[1]. As was shown in (Toutanova and Galley, 2011), IBM Model 1 is not a strictly concave optimization problem. What this means in practice is that although we can initialize the model with random parameters and get to the same objective cost via the EM algorithm, there is quite a bit of alignment quality variance within the model's optimal solution set and ambiguity persists on which optimal solution truly is the best. Typically, the easiest way to make a concave model strictly concave is to append an l_2 regularizer. However, this method does not allow for seamless EM training: we have to either use a learning-rate dependent gradient based algorithm directly or use a gradient method within the M step of EM training. In this paper we show how to get via a simple technique an infinite supply of models that still allows a straightforward application of the EM algorithm.

- As a concrete application of the above, we detail a very simple strictly concave version of IBM Model 1 and study the performance of different members within this class. Our strictly concave models combine some of the elements of word association and positional dependance as in IBM Model 2 to yield a significant model improvement. Furthermore,

*Currently on leave at Google Inc. New York.

[1]Please refer as needed to the Appendix for examples and definitions of convexity/concavity and strict convexity/concavity.

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 221–226,
Lisbon, Portugal, 17-21 September 2015.

we now have guarantees that the solution we find is unique.

- We detail an EM algorithm for a subclass of strictly concave IBM Model 1 variants. The EM algorithm is a small change to the original EM algorithm introduced in (Brown et al., 1993).

Notation. Throughout this paper, for any positive integer N, we use $[N]$ to denote $\{1 \dots N\}$ and $[N]_0$ to denote $\{0 \dots N\}$. We denote by $\mathbb{R}^n_+$ the set of nonnegative n dimensional vectors. We denote by $[0,1]^n$ the $n-$dimensional unit cube.

2 IBM Model 1

We begin by reviewing IBM Model 1 and introducing the necessary notation. To this end, throughout this section and the remainder of the paper we assume that our set of training examples is $(e^{(k)}, f^{(k)})$ for $k = 1 \dots n$, where $e^{(k)}$ is the k'th English sentence and $f^{(k)}$ is the k'th French sentence. Following standard convention, we assume the task is to translate from *French* (the "source" language) into *English* (the "target" language). We use E to denote the English vocabulary (set of possible English words), and F to denote the French vocabulary. The k'th English sentence is a sequence of words $e^{(k)}_1 \dots e^{(k)}_{l_k}$ where l_k is the length of the k'th English sentence, and each $e^{(k)}_i \in E$; similarly the k'th French sentence is a sequence $f^{(k)}_1 \dots f^{(k)}_{m_k}$ where each $f^{(k)}_j \in F$. We define $e^{(k)}_0$ for $k = 1 \dots n$ to be a special NULL word (note that E contains the NULL word).

For each English word $e \in E$, we will assume that $D(e)$ is a *dictionary* specifying the set of possible French words that can be translations of e. The set $D(e)$ is a subset of F. In practice, $D(e)$ can be derived in various ways; in our experiments we simply define $D(e)$ to include all French words f such that e and f are seen in a translation pair.

Given these definitions, the IBM Model 1 optimization problem is given in Fig. 1 and, for example, (Koehn, 2008). The parameters in this problem are $t(f|e)$. The $t(f|e)$ parameters are *translation* parameters specifying the probability of English word e being translated as French word f. The objective function is then the log-likelihood of the training data (see Eq. 3):

$$\frac{1}{n}\sum_{k=1}^{n}\sum_{j=1}^{m_k}\log p(f^{(k)}_j|e^{(k)}) \ ,$$

where $\log p(f^{(k)}_j|e^{(k)})$ is

$$\log\sum_{i=0}^{l_k}\frac{t(f^{(k)}_j|e^{(k)}_i)}{1+l_k} = C + \log\sum_{i=0}^{l_k} t(f^{(k)}_j|e^{(k)}_i) \ ,$$

and C is a constant that can be ignored.

Input: Define E, F, L, M, $(e^{(k)}, f^{(k)}, l_k, m_k)$ for $k = 1 \dots n$, $D(e)$ for $e \in E$ as in Section 2.

Parameters:

• A parameter $t(f|e)$ for each $e \in E, f \in D(e)$.

Constraints:

$$\forall e \in E, f \in D(e), \quad t(f|e) \geq 0 \tag{1}$$

$$\forall e \in E, \quad \sum_{f \in D(e)} t(f|e) = 1 \tag{2}$$

Objective: Maximize

$$\frac{1}{n}\sum_{k=1}^{n}\sum_{j=1}^{m_k}\log\sum_{i=0}^{l_k} t(f^{(k)}_j|e^{(k)}_i) \tag{3}$$

with respect to the $t(f|e)$ parameters.

Figure 1: The IBM Model 1 Optimization Problem.

While IBM Model 1 is concave optimization problem, it is not strictly concave (Toutanova and Galley, 2011). Therefore, optimization methods for IBM Model 1 (specifically, the EM algorithm) are typically only guaranteed to reach a global maximum of the objective function (see the Appendix for a simple example contrasting convex and strictly convex functions). In particular, although the objective cost is the same for any optimal solution, the translation quality of the solutions is not fixed and will still depend on the initialization of the model (Toutanova and Galley, 2011).

3 A Strictly Concave IBM Model 1

We now detail a very simple method to make IBM Model 1 strictly concave with a unique optimal solution without the need for appending an l_2 loss.

Theorem 1. *Consider IBM Model 1 and modify its objective to be*

$$\frac{1}{n}\sum_{k=1}^{n}\sum_{j=1}^{m_k}\log\sum_{i=0}^{l_k} h_{i,j,k}(t(f^{(k)}_j|e^{(k)}_i)) \tag{4}$$

where $h_{i,j,k} : \mathbb{R}_+ \rightarrow \mathbb{R}_+$ is strictly concave. With the new objective and the same constraints as IBM Model 1, this new optimization problem is strictly concave.

Proof. To prove concavity, we now show that the new likelihood function

$$L(\mathbf{t}) = \frac{1}{n}\sum_{k=1}^{n}\sum_{j=1}^{m_k}\log\sum_{i=0}^{l_k} h_{i,j,k}(t(f_j^{(k)}|e_i^{(k)}))\ ,$$

is strictly concave (concavity follows in the same way trivially). Suppose by way of contradiction that there is $(\mathrm{t}) \neq (\mathrm{t}')$ and $\theta \in (0,1)$ such that equality hold for Jensen's inequality. Since $h_{i,j,k}$ is strictly concave and $(\mathrm{t}) \neq (\mathrm{t}')$ we must have that there must be some (k,j,i) such that $t(f_j^{(k)}|e_i^{(k)}) \neq t'(f_j^{(k)}|e_i^{(k)})$ so that Jensen's inequality is strict for $h_{i,j,k}$ and we have

$$\begin{aligned}&h_{i,j,k}(\theta t(f_j^{(k)}|e_i^{(k)}) + (1-\theta)t'(f_j^{(k)}|e_i^{(k)}))\\ &\quad > \theta h_{i,j,k}(t(f_j^{(k)}|e_i^{(k)})) + (1-\theta)h_{i,j,k}(t'(f_j^{(k)}|e_i^{(k)}))\end{aligned}$$

Using Jensen's inequality, the monotonicity of the log, and the above strict inequality we have

$$\begin{aligned}&L(\theta\mathbf{t} + (1-\theta)\mathbf{t}')\\ &= \sum_{k=1}^{n}\sum_{j=1}^{m_k}\log\sum_{i=0}^{l_k} h_{i,j,k}(\theta t(f_j^{(k)}|e_i^{(k)}) + (1-\theta)t'(f_j^{(k)}|e_i^{(k)}))\\ &> \sum_{k=1}^{n}\sum_{j=1}^{m_k}\log\sum_{i=0}^{l_k} \theta h_{i,j,k}(t(f_j^{(k)}|e_i^{(k)})) + (1-\theta)h_{i,j,k}(t'(f_j^{(k)}|e_i^{(k)}))\\ &\geq \theta\sum_{k=1}^{n}\sum_{j=1}^{m_k}\log\sum_{i=0}^{l_k} h_{i,j,k}(t(f_j^{(k)}|e_i^{(k)}))\\ &+ (1-\theta)\sum_{k=1}^{n}\sum_{j=1}^{m_k}\log\sum_{i=0}^{l_k} h_{i,j,k}(t'(f_j^{(k)}|e_i^{(k)}))\\ &= \theta L(\mathbf{t}) + (1-\theta)L(\mathbf{t}')\end{aligned}$$

□

The IBM Model 1 strictly concave optimization problem is presented in Fig. 2. In (7) it is crucial that each $h_{i,j,k}$ be strictly concave within $\sum_{i=0}^{l_k} h_{i,j,k}(t(f_j^{(k)}|e_i^{(k)}))$. For example, we have that $\sqrt{x_1} + x_2$ is concave but not strictly concave and the proof of Theorem 1 would break down. To see this, we can consider $(x_1, x_2) \neq (x_1, x_3)$ and note that equality holds in Jensen's inequality. We should be clear: the main reason why Theorem 1 works is that we have $h_{i,j,k}$ are strictly concave (on $\mathbb{R}_+$) and all the lexical probabilities that are arguments to L are present within the log-likelihood.

Input: Define E, F, L, M, $(e^{(k)}, f^{(k)}, l_k, m_k)$ for $k = 1 \ldots n$, $D(e)$ for $e \in E$ as in Section 2. A set of strictly concave functions $h_{i,j,k} : \mathbb{R}_+ \rightarrow \mathbb{R}_+$.

Parameters:

• A parameter $t(f|e)$ for each $e \in E$, $f \in D(e)$.

Constraints:

$$\forall e \in E, f \in D(e), \quad t(f|e) \geq 0 \tag{5}$$

$$\forall e \in E, \quad \sum_{f \in D(e)} t(f|e) = 1 \tag{6}$$

Objective: Maximize

$$\frac{1}{n}\sum_{k=1}^{n}\sum_{j=1}^{m_k}\log\sum_{i=0}^{l_k} h_{i,j,k}(t(f_j^{(k)}|e_i^{(k)})) \tag{7}$$

with respect to the $t(f|e)$ parameters.

Figure 2: The IBM Model 1 strictly concave optimization problem.

4 Parameter Estimation via EM

For the IBM Model 1 strictly concave optimization problem, we can derive a clean EM Algorithm if we base our relaxation of

$$h_{i,j,k}(t(f_j^{(k)}|e_i^{(k)})) = \alpha(e_i^{(k)}, f_j^{(k)})(t(f_j^{(k)}|e_i^{(k)}))^{\beta(e_i^{(k)}, f_j^{(k)})}$$

with $\beta(e_i^{(k)}, f_j^{(k)}) < 1$. To justify this, we first need the following:

Lemma 1. *Consider $h : \mathbb{R}_+ \rightarrow \mathbb{R}_+$ given by $h(x) = x^\beta$ where $\beta \in (0,1)$. Then h is strictly concave.*

Proof. The proof of this lemma is elementary and follows since the second derivative given by $h''(x) = \beta(\beta-1)x^{\beta-2}$ is strictly negative. □

For our concrete experiments, we picked a model based on Lemma 1 and used $h(x) = \alpha x^\beta$ with $\alpha, \beta \in (0,1)$ so that

$$h_{i,j,k}(t(f_j^{(k)}|e_i^{(k)})) = \alpha(f_j^{(k)}, e_i^{(k)})(t(f_j^{(k)}|e_i^{(k)}))^{\beta(f_j^{(k)}, e_i^{(k)})}\ .$$

Using this setup, parameter estimation for the new model can be accomplished via a slight modification of the EM algorithm for IBM Model 1. In particular, we have that the posterior probabilities of this model factor just as those of the standard Model 1 and we have an M step that requires optimizing

$$\sum_{a^{(k)}} q(a^{(k)}|e^{(k)}, f^{(k)}) \log p(f^{(k)}, a^{(k)}|e^{(k)})$$

Input: Define E, F, L, M, $(e^{(k)}, f^{(k)}, l_k, m_k)$ for $k = 1 \ldots n$, $D(e)$ for $e \in E$ as in Section 2. An integer T specifying the number of passes over the data. A set of weighting parameter $\alpha(e, f)$, $\beta(e, f) \in (0, 1)$ for each $e \in E$, $f \in D(e)$. A tuning parameter $\lambda > 0$.
Parameters:
• A parameter $t(f|e)$ for each $e \in E$, $f \in D(e)$.
Initialization:
• $\forall e \in E,\ f \in D(e)$, set $t(f|e) = 1/|D(e)|$.
EM Algorithm:
for all $t = 1 \ldots T$ **do**
  $\forall e \in E,\ f \in D(e),\ count(f, e) = 0$
  $\forall e \in E,\ count(e) = 0$
  EM Algorithm: Expectation
  for all $k = 1 \ldots n$ **do**
    for all $j = 1 \ldots m_k$ **do**
      $\delta_1[i] = 0\ \forall i \in [l_k]_0$
      $\Delta_1 = 0$
      for all $i = 0 \ldots l_k$ **do**
        $\delta_1[i] = \alpha(f_j^{(k)}, e_i^{(k)})(t(f_j^{(k)}|e_i^{(k)}))^{\beta(f_j^{(k)}, e_i^{(k)})}$
        $\Delta_1 \mathrel{+}= \delta_1[i]$
      for all $i = 0 \ldots l_k$ **do**
        $\delta_1[i] = \frac{\delta_1[i]}{\Delta_1}$
        $count(f_j^{(k)}, e_i^{(k)}) \mathrel{+}= \beta(f_j^{(k)}, e_i^{(k)})\delta_1[i]$
        $count(e_i^{(k)}) \mathrel{+}= \beta(f_j^{(k)}, e_i^{(k)})\delta_1[i]$
  EM Algorithm: Maximization
  for all $e \in E$ **do**
    for all $f \in D(e)$ **do**
      $t(f|e) = \frac{count(e,f)}{count(e)}$
Output: t parameters

Figure 3: Pseudocode for T iterations of the EM Algorithm for the IBM Model 1 strictly concave optimization problem.

where

$$q(a^{(k)}|e^{(k)}, f^{(k)}) \propto \prod_{j=1}^{m_k} h_{a_j^{(k)}, j, k}(t(f_j^{(k)}|e_{a_j^{(k)}}^{(k)}))$$

are constants gotten in the E step. This optimization step is very similar to the regular Model 1 M step since the β drops down using $\log t^\beta = \beta \log t$; the exact same count-based method can be applied. The details of this algorithm are in Fig. 3.

5 Choosing α and β

The performance of our new model will rely heavily on the choice of $\alpha(e_i^{(k)}, f_j^{(k)})$, $\beta(e_i^{(k)}, f_j^{(k)}) \in (0, 1)$ we use. In particular, we could make β depend on the association between the words, or the words' positions, or both. One classical measure of word association is the dice coefficient (Och and Ney, 2003) given by

$$dice(e, f) = \frac{2c(e, f)}{c(e) + c(f)}\ .$$

In the above, the count terms c are the number of training sentences that have either a particular word or a pair of of words (e, f). As with the other choices we explore, the dice coefficient is a fraction between 0 and 1, with 0 and 1 implying less and more association, respectively. Additionally, we make use of positional constants like those of the IBM Model 2 distortions given by

$$d(i|j, l, m) = \begin{cases} \frac{1}{(l+1)Z(j,l,m)} & : i = 0 \\ \frac{le^{-\lambda|\frac{i}{l} - \frac{j}{m}|}}{(l+1)Z(j,l,m)} & : i \neq 0 \end{cases}$$

In the above, $Z(j, l, m)$ is the partition function discussed in (Dyer et al., 2013). The previous measures all lead to potential candidates for $\beta(e, f)$, we have $t(f|e) \in (0, 1)$, and we want to enlarge competing values when decoding (we use αt^β instead of t when getting the Viterbi alignment). The above then implies that we will have the word association measures inversely proportional to β, and so we set $\beta(e, f) = 1 - dice(e, f)$ or $\beta(e, f) = 1 - d(i|j, l, m)$. In our experiments we picked $\alpha(f_j^{(k)}, e_i^{(k)}) = d(i|j, l_k, m_k)$ or 1; we hold λ to a constant of either 16 or 0 and do not estimate this variable ($\lambda = 16$ can be chosen by cross validation on a small trial data set).

6 Experiments

6.1 Data Sets

For our alignment experiments, we used a subset of the Canadian Hansards bilingual corpus with 247,878 English-French sentence pairs as training data, 37 sentences of development data, and 447 sentences of test data (Michalcea and Pederson, 2003). As a second validation corpus, we considered a training set of 48,706 Romanian-English sentence-pairs, a development set of 17 sentence pairs, and a test set of 248 sentence pairs (Michalcea and Pederson, 2003).

6.2 Methodology

Below we report results in both AER (lower is better) and F-Measure (higher is better) (Och and Ney, 2003) for the English → French translation direction. To declare a better model we have to settle on an alignment measure. Although the relationship between AER/F-Measure and translation quality varies (Dyer et al., 2013), there are some positive experiments (Fraser and Marcu, 2004) showing that F-Measure may be more useful, so perhaps a comparison based on F-Measure is ideal.

Table 1 contains our results for the Hansards data. For the smaller Romanian data, we obtained similar behavior, but we leave out these results due

(α, β)	$(1,1)$	$(d,1)$	$(1,1-dice)$	$(1,1-d)$	$(d,1-d)$
Iteration			AER		
0	0.8716	0.6750	0.6240	0.6597	**0.5570**
1	0.4426	0.2917	0.4533	**0.2738**	0.3695
2	0.3383	0.2323	0.4028	**0.2318**	0.3085
3	0.3241	**0.2190**	0.3845	0.2252	0.2881
4	0.3191	**0.2141**	0.3751	0.2228	0.2833
5	0.3175	**0.2118**	0.3590	0.2229	0.2812
6	0.3160	**0.2093**	0.3566	0.2231	0.2793
7	0.3203	**0.2090**	0.3555	0.2236	0.2783
8	0.3198	**0.2075**	0.3546	0.2276	0.2777
9	0.3198	**0.2066**	0.3535	0.2323	0.2769
10	0.3177	**0.2065**	0.3531	0.2352	0.2769
Iteration			F-Measure		
0	0.0427	0.1451	**0.2916**	0.1897	0.2561
1	0.4213	0.5129	0.4401	**0.5453**	0.4427
2	0.5263	0.5726	0.4851	**0.5940**	0.5014
3	0.5413	0.5852	0.5022	**0.6047**	0.5199
4	0.5480	0.5909	0.5111	**0.6085**	0.5255
5	0.5500	0.5939	0.5264	**0.6101**	0.5273
6	0.5505	0.5959	0.5282	**0.6101**	0.5286
7	0.5449	0.5965	0.5298	**0.6096**	0.5296
8	0.5456	0.5977	0.5307	**0.6068**	0.5300
9	0.5451	0.5985	0.5318	**0.6040**	0.5309
10	0.5468	0.5984	0.5322	**0.6024**	0.5311

Table 1: Results on the English-French data for various (α, β) settings as discussed in Section 5. For the d parameters, we use $\lambda = 16$ throughout. The standard IBM Model 1 is column 1 and corresponds to a setting of (1,1). The not necessarily strictly concave model with (d,1) setting gives the best AER, while the strictly concave model given by the $(1, 1-d)$ setting has the highest F-Measure.

to space limitations. Our experiments show that using

$$h_{i,j,k}(t(f_j^{(k)}|e_i^{(k)})) = (t(f_j^{(k)}|e_i^{(k)}))^{1-d(i|j,l_k,m_k)}$$

yields the best F-Measure performance and is not far off in AER from the "fake"[2] IBM Model 2 (gotten by setting $(\alpha, \beta) = (d, 1)$) whose results are in column 2 (the reason why we use this model at all is since it should be better than IBM 1: we want to know how far off we are from this obvious improvement). Moreover, we note that dice does not lead to quality β exponents and that, unfortunately, combining methods as in column 5 $((\alpha, \beta) = (d, 1-d))$ does not necessarily lead to additive gains in AER and F-Measure performance.

[2]Generally speaking, when using

$$h_{i,j,k}(t(f_j^{(k)}|e_i^{(k)})) = d(i|j,l_k,m_k)t(f_j^{(k)}|e_i^{(k)})$$

with d constant we cannot use Theorem 3 since h is linear. Most likely, the strict concavity of the model will hold because of the asymmetry introduced by the d term; however, there will be a necessary dependency on the data set.

7 Comparison with Previous Work

In this section we take a moment to also compare our work with the classical IBM 1 work of (Moore, 2004). Summarizing (Moore, 2004), we note that this work improves substancially upon the classical IBM Model 1 by introducing a set of heuristics, among which are to (1) modify the lexical parameter dictionaries (2) introduce an initialization heuristic (3) modify the standard IBM 1 EM algorithm by introducing smoothing (4) tune additional parameters. However, we stress that the main concern of this work is not just heuristic-based empirical improvement, but also structured learning. In particular, although using an regularizer l_2 and the methods of (Moore, 2004) would yield a strictly concave version of IBM 1 as well (with improvements), it is not at all obvious how to choose the learning rate or set the penalty on the lexical parameters. The goal of our work was to offer a new, alternate form of regularization. Moreover, since we are changing the original log-likelihood, our method can be thought of as way of *bringing the l_2 regularizer inside the log likelihood*. Like (Moore, 2004), we also achieve appreciable gains but have just one tuning parameter (when $\beta = 1 - d$ we just have the centering λ parameter) and do not break the probabilistic interpretation any more than appending a regularizer would (our method modifies the log-likelihood but the simplex constrains remain).

8 Conclusion

In this paper we showed how IBM Model 1 can be made into a strictly convex optimization problem via functional composition. We looked at a specific member within the studied optimization family that allows for an easy EM algorithm. Finally, we conducted experiments showing how the model performs on some standard data sets and empirically showed 30% important over the standard IBM Model 1 algorithm. For further research, we note that picking the optimal $h_{i,j,k}$ is an open question, so provably finding and justifying the choice is one topic of interest.

Acknowledgments

Andrei Simion was supported by a Google research award. Cliff Stein is supported in part by NSF grants CCF-1349602 and CCF-1421161.

References

Steven Boyd and Lieven Vandenberghe. 2004. Convex Optimization. Cambridge University Press.

Peter F. Brown, Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert. L. Mercer. 1993. The Mathematics of Statistical Machine Translation: Parameter Estimation. *Computational Linguistics*, 19:263-311.

A. P. Dempster, N. M. Laird, and D. B. Rubin. 1977. Maximum Likelihood From Incomplete Data via the EM Algorithm. *Journal of the royal statistical society, series B*, 39(1):1-38.

Chris Dyer , Victor Chahuneau, Noah A. Smith. 2013. A Simple, Fast, and Effective Reparameterization of IBM Model 2. *In Proceedings of NAACL.*

Alexander Fraser and Daniel Marcu. 2007. Measuring Word Alignment Quality for Statistical Machine Translation. *Journal Computational Linguistics*, 33(3): 293-303.

Phillip Koehn. 2008. Statistical Machine Translation. Cambridge University Press.

Rada Michalcea and Ted Pederson. 2003. An Evaluation Exercise in Word Alignment. *HLT-NAACL 2003: Workshop in building and using Parallel Texts: Data Driven Machine Translation and Beyond.*

Robert C. Moore. 2004. Improving IBM Word-Alignment Model 1. *In Proceedings of the ACL.*

Stephan Vogel, Hermann Ney and Christoph Tillman. 1996. HMM-Based Word Alignment in Statistical Translation. *In Proceedings of COLING*.

Franz Och and Hermann Ney. 2003. A Systematic Comparison of Various Statistical Alignment Models. *Computational-Linguistics*, 29(1): 19-52.

Andrei Simion, Michael Collins and Cliff Stein. 2013. A Convex Alternative to IBM Model 2. *In Proceedings of EMNLP*.

Andrei Simion, Michael Collins and Cliff Stein. 2015. A Family of Latent Variable Convex Relaxations for IBM Model 2. *In Proceedings of the AAAI*.

Kristina Toutanova and Michel Galley. 2011. Why Initialization Matters for IBM Model 1: Multiple Optima and Non-Strict Convexity. *In Proceedings of the ACL.*

Ashish Vaswani, Liang Huang and David Chiang. 2012. Smaller Alignment Models for Better Translations: Unsupervised Word Alignment with the $L0$-norm. *In Proceedings of the ACL.*

Factorization of Latent Variables in Distributional Semantic Models

Arvid Österlund and **David Ödling**
KTH Royal Institute of Technology, Sweden
arvidos|dodling@kth.se

Magnus Sahlgren
Gavagai, Sweden
mange@gavagai.se

Abstract

This paper discusses the use of factorization techniques in distributional semantic models. We focus on a method for redistributing the weight of latent variables, which has previously been shown to improve the performance of distributional semantic models. However, this result has not been replicated and remains poorly understood. We refine the method, and provide additional theoretical justification, as well as empirical results that demonstrate the viability of the proposed approach.

1 Introduction

Distributional Semantic Models (DSMs) have become standard paraphernalia in the natural language processing toolbox, and even though there is a wide variety of models available, the basic parameters of DSMs (context type and size, frequency weighting, and dimension reduction) are now well understood. This is demonstrated by the recent convergence of state-of-the-art results (Baroni et al., 2014; Levy and Goldberg, 2014).

However, there are a few notable exceptions. One is the performance improvements demonstrated in two different papers using a method for redistributing the weight of principal components (PCs) in factorized DSMs (Caron, 2001; Bullinaria and Levy, 2012). In the latter of these papers, the factorization of latent variables in DSMs is used to reach a perfect score of 100% correct answers on the TOEFL synonym test. This result is somewhat surprising, since the factorization method is *the inverse of what is normally used*.

Neither the result nor the method has been replicated, and therefore remains poorly understood. The goal of this paper is to replicate and explain the result. In the following sections, we first provide a brief review of DSMs and factorization, and review the method for redistributing the weight of latent variables. We then replicate the 100% score on the TOEFL test and provide additional state-of-the-art scores for the BLESS test. We also provide a more principled reformulation of the factorization method that is better suited for practical applications.

2 Distributional Semantics

Consider a set of words $W = \{w_1, \ldots, w_n\}$ and a set of context words $C = \{c_1, \ldots, c_m\}$. The DSM representation is created by registering an occurrence of a word w_i with a set of context words $c_j, \ldots, c_k$ with a corresponding increment of the projection of w_i on the $c_j, \ldots, c_k$ bases. In other words, each cell f_{ij} in the matrix representation F represents a co-occurrence count between a word w_i and a context c_j. In the following, we use $W = C$, making the co-occurrence matrix symmetric $F_{n \times n}$. We also adhere to standard practice of weighting the co-occurrence counts with Positive Pointwise Mutual Information (PPMI) (Niwa and Nitta, 1994), which is a variation of the standard PMI weighting,[1] which simply discards non-positive PMI values.

3 Singular Value Decomposition

The high dimensionality of the co-occurrence matrix makes it necessary in most practical applications to apply some form of dimensionality reduction to F, with the goal of finding a basis $\{\hat{x_j}, ..., \hat{x_k}\}$ that restates the original basis $\{x_k, ...\}$ in a lower-dimensional space $\hat{F}$, where $\hat{F}$ denotes the rank-k approximation of F:

$$\min_{\hat{F} \in R^n \times R^k} |F - \hat{F}| \quad (1)$$

[1] $\text{PMI}(f_{ij}) = \log \frac{f_{ij}(\sum_{ij} f_{ij})^2}{\sum_i f_{ij} \sum_j f_{ij} \sum_{ij} f_{ij}}$.

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 227–231,
Lisbon, Portugal, 17-21 September 2015.

Assuming Gaussian-like distributions,[2] a canonical way of achieving this is to maximize the variance of the data in the new basis. This enables ordering of the new basis according to how much of the variance in the original data each component describes.

A standard co-occurrence matrix is positive and symmetric and thus has, by the spectral theorem, a spectral decomposition of an ordered set of positive eigenvalues and an orthogonal set of eigenvalues:

$$F = U\Sigma V^T \tag{2}$$

where U holds the eigenvectors of F, Σ holds the eigenvalues, and $V \in U(w)$ is a unitary matrix mapping the original basis of F into its eigenbasis. Hence, by simply choosing the first k eigenvalues and their respective eigenvectors we have the central result:

$$\min_k |F - \hat{F}| \rightarrow \hat{F} \approx U_k \Sigma_k V_k^T \tag{3}$$

where $\hat{F}$ is the best rank-k approximation in the Frobenius-norm. This is commonly referred to as *truncated* Singular Value Decomposition (SVD).

Finally, using cosine similarity,[3] V is redundant due to invariance under unitary transformations, which means we can represent the principal components of $\hat{F}$ in its most compact form $\hat{F} \equiv U\Sigma$ without any further comment.

This projection onto the eigenbasis does not only provide an efficient compression of the sparse co-occurrence data, but has also been shown to improve the performance and noise tolerance of DSMs (Schütze, 1992; Landauer and Dumais, 1997; Bullinaria and Levy, 2012).

4 The Caron p-transform

Caron (2001) introduce a method for renormalization of the latent variables through an exponent factor $p \in R$:

$$U\Sigma \rightarrow U\Sigma^p \tag{4}$$

which is shown to improve the results of factorized models using both information retrieval and question answering test collections. We refer to this renormalization as the *Caron p-transform*. Bullinaria and Levy (2012) further corroborate Caron's result, and show that the optimum exponent parameter p for DSMs is with strong statistical significance $p < 1$. Moreover, due to the redistribution of weight to the lower variance PCs, Bullinaria and Levy (2012) show that similar performance improvements can be achieved by simply *removing* the first PCs. We refer to this as *PC-removal*. A highlight of their results is a perfect score of 100% on the TOEFL synonym test.

Apart form the perfect score on the TOEFL test, it is noteworthy that the PC-removal scheme is the inverse of how SVD is normally used in DSMs; instead of retaining only the first PCs – which is the standard way of using the SVD in DSMs – the PC-removal scheme *deletes* them, and instead retains all the rest.

5 Experiments

We replicate the experiment setup of Bullinaria and Levy (2012) by removing punctuation and decapitalizing the ukWaC corpus (Baroni et al., 2009). The DSM includes the 50,000 most frequent words along with the remaining 23 TOEFL words and is populated using a ± 2-sized context window. Co-occurrence counts are weighted with PPMI, and SVD is applied to the resulting matrix, reducing the dimensionality to 5,000. The results of removing the first PCs versus applying the Caron p-transform are shown in Figure 1, which replicates the results from Bullinaria and Levy (2012).

In order to better understand what influence the transform has on the representations, we also provide results on the BLESS test (Baroni and Lenci, 2011), which lists a number of related terms to 200 target terms. The related terms represent 8 different kinds of semantic relations (co-hyponymy, hypernymy, meronymy, attribute, event, and three random classes corresponding to randomly selected nouns, adjectives and verbs), and it is thus possible to use the BLESS test to determine what type of semantic relation a model favors. Since our primary interest here is in *paradigmatic* relations, we focus on the hypernymy and co-hyponymy relations, and require that the model scores one of the related terms from these classes higher than the related terms from the other classes. The corpus was split into different sizes to test the statistical significance of the weight redistribution effect. Furthermore, it shows that the optimal weight distribution depends on the size of the data.

[2] It is well known that the Gaussian assumption does not hold in reality, and consequently there are also other approaches to dimensionality reduction based on multinomial distributions, which we will not consider in this paper.

[3] $\cos(w_i, w_j) = \frac{w_i \cdot w_j}{|w_i||w_j|}$

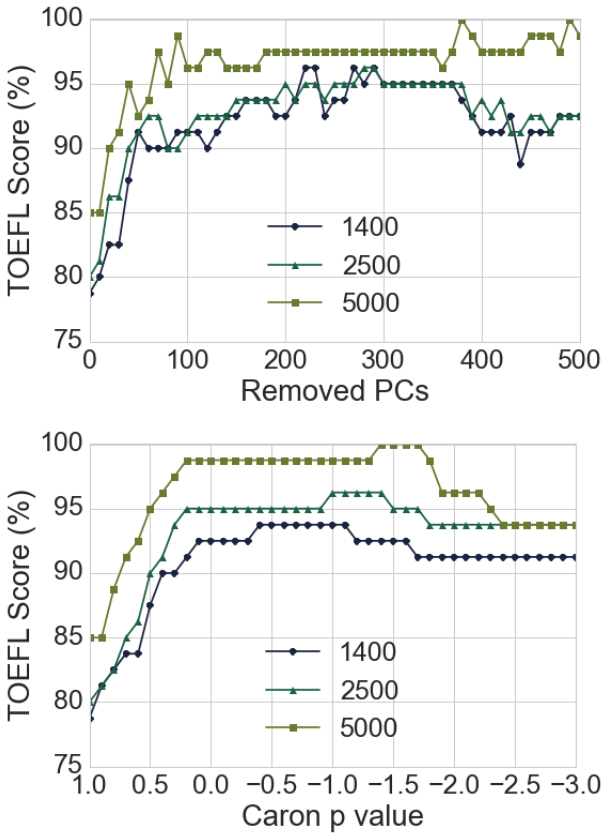


Figure 1: TOEFL score for the PC-removal scheme and the Caron p-transform for the span of PCs.

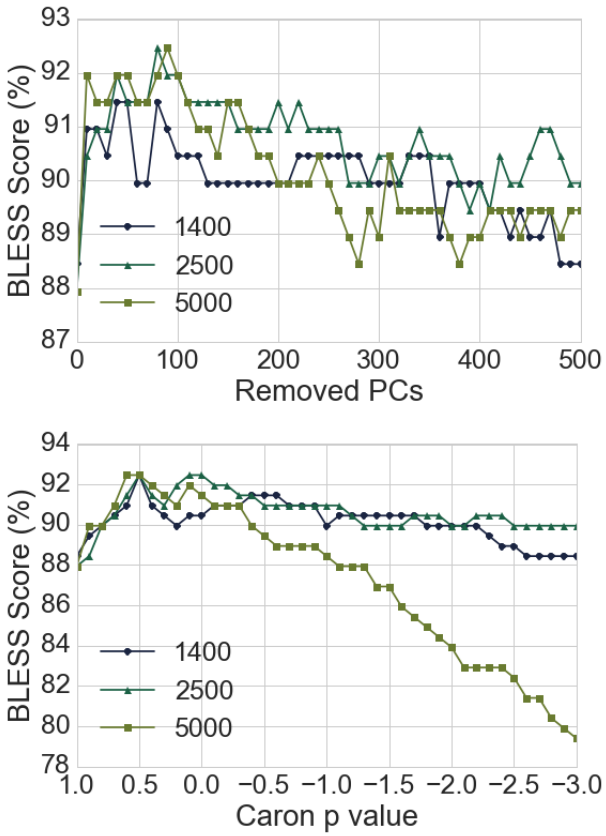


Figure 2: BLESS score for the PC-removal scheme and the Caron p-transform for the span of PCs.

Figure 2 shows the BLESS results for both the PC removal scheme and the Caron p-transform for different sizes of the corpus. The best score is 92.96% for the PC removal, and 92.46% for the Caron p-transform, both using the full data set. Similarly to the TOEFL results, we see better results for a larger number of removed PCs. Interestingly, there is clearly a larger improvement in performance of the Caron p-transform than for the PC removal scheme.

Figure 3 shows how the redistribution affects the different relations in the BLESS test. The violin plots are based on the maximum values of each relation, and the width of the violin represents the normalized probability density of cosine measures. The cosine distributions, Θ_i, are based on the best matches for each category i, and normalized by the total mean and variance amongst all categories $\hat{\Theta}_i = \frac{\Theta_i - \mu}{\sigma}$. Thus, the figure illustrates how well each category is separated from each other, the larger separation the better.

The results in Figure 3 indicate that the top 120 PCs contain a higher level of co-hyponymy relations than the lower; removing the top 120 PCs gives a violin shape that resembles the inverse of the plot for the top 120 PCs. Although neither part of the PC span is significantly better in separating the categories, it is clear that removing the first 120 PCs increases the variance within the categories

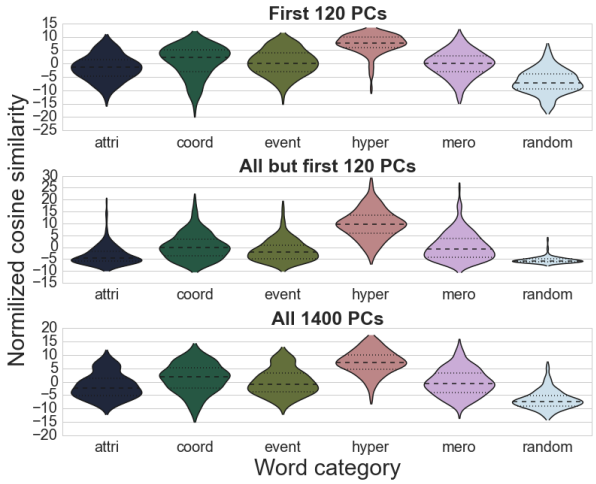


Figure 3: BLESS targets versus categories from 1,400 PCs representation of the entire corpus.

and especially amongst the coord-category. This is an interesting result, since it seems to contradict the hypothesis that removing the first PCs improves the semantic quality of the representations – there is obviously valuable semantic information in the first PCs.

Table 1 summarizes our top results on the TOEFL, BLESS, and also the SimLex-999 similarity test (Hill et al., 2014), and compares them to a baseline score from the Skipgram model (Mikolov et al., 2013a), trained on the same data using a window size of 2, negative samples, and 400-dimensional vectors.

	TOEFL	BLESS	SimLex-999
PC removal	100	92.96	46.52
Caron p	100	92.46	46.66
Skipgram	83.75	83.00	39.91

Table 1: Top results for the PC removal and Caron p on each test compared to the Skipgram model.

Unfortunately, the optimal redistribution of weight on the PCs for the respective top scores differ between the experiments. For the PC removal the optimal number of removed PCs is 379 for TOEFL, 15 for BLESS and 128 for SimLex-999, while the optimal number for the Caron p-transform is -1.4 for TOEFL, 0.5 for BLESS and -0.40 for SimLex-999. Hence, there is likely no easy way to find a general expression of the optimal redistribution of weight on the PCs for a given application.

6 The Pareto Principle

It is common practice to reduce the dimensionality of an n-dimensional space to as many PCs as it takes to cover 80% of the total eigenvalue mass. This convention is known as the *Pareto principle* (or 80/20-rule), and generally gives a good trade-off between compression and precision. The results presented in the previous section suggest a type of inversion of this principle in the case of DSMs.

Given a computational and practical limit of the number of PCs m with weights $\Sigma = \{\sigma_1, ..., \sigma_m\}$, the optimal redistribution of weight on these components is such that the first $l - m$ components $\sigma_1, ..., \sigma_{m-l}$ is transformed such that they constitute 20% of the new total mass. Where $l - m$ is the number of components representing the last 20 % of the original mass. In other words, the function $f : \Sigma \rightarrow \hat{\Sigma}$ performing this redistribution is such that:

$$\frac{\sum_{i=1}^{m-l} \hat{\sigma}_i}{\sum_{i=1}^{m} \hat{\sigma}_i} \approx 20\% \quad (5)$$

In this formulation, we can consider the Caron p-transform and the PC-removal scheme as special cases, where the Caron p-transform is given by:

$$f(\sigma_i) = \sigma_i^p \quad \forall i,\ p \in R \quad (6)$$

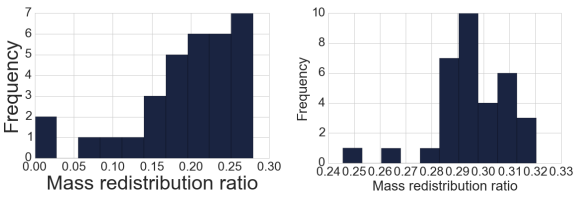
and the PC-removal scheme by:

$$f(\sigma_i) = (1 - \delta(F))\sigma_i \quad \forall i,\ F = \{1....l\} \quad (7)$$

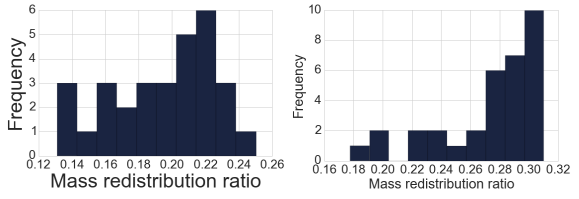
where $\delta(F)$ denotes the generalized Kronecker delta function.

To test this claim, we form this quotient for the distributions of weights at the optimal parameters for the Caron p-transform and the PC-removal scheme for both the BLESS and TOEFL tests for each of the 40 sub-corpora.

Even though the results are not as optimal for the BLESS test as for the TOEFL, the results in Figure 4 point in favor of this measure. The optimal mass distributions for the Caron p-transform and the PC removal are all around 20 %.



(a) TOEFL - PC removal (b) BLESS - PC removal



(c) TOEFL - Caron p (d) BLESS - Caron p

Figure 4: The mass redistribution ratio for the best results on the 1,400 PC models.

This result does not only apply for 1,400 PCs, but has also been verified on a smaller set of matrices with sizes of 2,500 PCs, 4,000 PCs and 5,000 PCs. The results for 1,400 PCs and 5,000 PCs are shown in Table 2. As can be seen in this table, the rule of thumb yields reasonable good guesses for both Caron p and PC removal, over the different tests and for various number of PCs.

5,000 PC representation

	PC removal	Caron p
Parameter	493	-0.80
TOEFL	100	98.75
BLESS	89.45	88.95
SimLex	44.82	45.74

1,400 PC representation

	PC removal	Caron p
Parameter	204	-0.80
TOEFL	93.75	93.75
BLESS	89.95	90.95
SimLex	45.45	46.47

Table 2: Results for the PC removal and Caron p using the 80/20 rule

7 Conclusions and future work

This paper has discussed the method of redistributing the weight of the first PCs in factorized DSMs. We have replicated previously published results, and provided additional empirical justification for the method. The method significantly outperforms the baseline Skipgram model on all tests used in the experiments. Our results also suggest a slight refinement of the method, for which we have provided both theoretical and empirical justification. The resulting rule of thumb method leads to stable results that may be useful in practice.

Although the experiments in this paper has provided further evidence for the usefulness of redistributing the weight in factorized models, it also raises additional interesting research questions. For example, does the method also improve models that have been trained on smaller data sets? Does it also hold for non-Gaussian factorization like Non-negative Matrix Factorization? How does the method affect the (local) structural properties of the representations; do factorized models display the same type of structural regularities as has been observed in word embeddings (Mikolov et al., 2013b), and would it be possible to use methods such as relative neighborhood graphs (Gyllensten and Sahlgren, 2015) to explore the local effects of the transformation?

References

Marco Baroni and Alessandro Lenci. 2011. How we blessed distributional semantic evaluation. In *Proceedings of GEMS*, pages 1–10.

Marco Baroni, Silvia Bernardini, Adriano Ferraresi, and Eros Zanchetta. 2009. The wacky wide web: a collection of very large linguistically processed web-crawled corpora. *Language resources and evaluation*, 43(3):209–226.

Marco Baroni, Georgiana Dinu, and Germán Kruszewski. 2014. Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of ACL*, pages 238–247.

John Bullinaria and Joseph Levy. 2012. Extracting semantic representations from word co-occurrence statistics: stop-lists, stemming, and SVD. *Behavior Research Methods*, 44(3):890–907.

John Caron. 2001. Experiments with LSA scoring: Optimal rank and basis. In Michael Berry, editor, *Computational Information Retrieval*, pages 157–169.

Amaru Cuba Gyllensten and Magnus Sahlgren. 2015. Navigating the semantic horizon using relative neighborhood graph. In *Proceedings of EMNLP*.

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

Thomas Landauer and Susan Dumais. 1997. A solution to Plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological review*, 104(2):211.

Omer Levy and Yoav Goldberg. 2014. Neural word embedding as implicit matrix factorization. In *Proceedings of NIPS*, pages 2177–2185.

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

Tomas Mikolov, Wen tau Yih, and Geoffrey Zweig. 2013b. Linguistic regularities in continuous space word representations. In *Proceedings NAACL-HLT*, pages 746–751.

Yoshiki Niwa and Yoshihiko Nitta. 1994. Co-occurrence vectors from corpora vs. distance vectors from dictionaries. In *Proceedings of COLING*, pages 304–309.

Hinrich Schütze. 1992. Dimensions of meaning. In *Proceedings of Supercomputing*, pages 787–796.

Non-lexical neural architecture for fine-grained POS Tagging

Matthieu Labeau, Kevin Löser, Alexandre Allauzen
Université Paris-Sud and LIMSI-CNRS,
Rue John von Neumann
91403 Orsay cedex
France
firstname.lastname@limsi.fr

Abstract

In this paper we explore a POS tagging application of neural architectures that can infer word representations from the raw character stream. It relies on two modelling stages that are jointly learnt: a convolutional network that infers a word representation directly from the character stream, followed by a prediction stage. Models are evaluated on a POS and morphological tagging task for German. Experimental results show that the convolutional network can infer meaningful word representations, while for the prediction stage, a well designed and structured strategy allows the model to outperform state-of-the-art results, without any feature engineering.

1 Introduction

Most modern statistical models for natural language processing (NLP) applications are strongly or fully lexicalized, for instance part-of-speech (POS) and named entity taggers, as well as language models, and parsers. In these models, the observed word form is considered as the elementary unit, while its morphological properties remain neglected. As a result, the vocabulary observed on training data heavily restricts the generalization power of lexicalized models.

Designing subword-level systems is appealing for several reasons. First, words sharing morphological properties often share grammatical function and meaning, and leveraging that information can yield improved word representations. Second, a subword-level analysis can address the out-of-vocabulary issue *i.e* the fact that word-level models fail to meaningfully process unseen word forms. This allows a better processing of morphologically rich languages in which there is a combinatorial explosion of word forms, most of which are not observed during training. Finally, using subword units could allow processing of noisy text such as user-generated content on the Web, where abbreviations, slang usage and spelling mistakes cause the number of word types to explode.

This work investigates models that do not rely on a fixed vocabulary to make a linguistic prediction. Our main focus in this paper is POS tagging, yet the proposed approach could be applied to a wide variety of language processing tasks. Our main contribution is to show that neural networks can successfully learn unlexicalized models that infer a useful word representation from the character stream. This approach achieves state of-the-art performance on a German POS tagging task. This task is difficult because German is a morphologically rich language[1], as reflected by the large number of morphological tags (255) in our study, yielding a grand total of more than 600 POS+MORPH tags. An aggravating factor is that these morphological categories are overtly marked by a handful of highly ambiguous inflection marks (suffixes). We therefore believe that this case study is well suited to assess both the representation and prediction power of our models.

The architecture we explore in section 2 differs from previous work that only consider the character level. Following (Santos and Zadrozny, 2014), it consists in two stages that are jointly learnt. The lower stage is a convolutional network that infers a word embedding from a character string of arbitrary size, while the higher network infers the POS tags based on this word embedding sequence. For the latter, we investigate different architectures of increasing complexities: from a feedforward and context-free inference to a bi-recurrent network that predicts the global sequence. Experimental results (section 4) show that the proposed approach can achieve state of the art performance

[1]Besides inflected forms, German is characterized by a possibly infinite and evolving set of compound nouns.

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 232–237,
Lisbon, Portugal, 17-21 September 2015.

and that the choice of architecture for the prediction part of the model has a significant impact.

2 Network Architectures

The different architectures we propose act in two stages to infer, for a sentence $s = \{w_1, \ldots, w_{|s|}\}$, a sequence of tags $\{t_1, \ldots, t_{|s|}\}$. Each tag belongs to the tagset $\mathcal{T}$. The first stage is designed to represent each word *locally*, and focuses on capturing the meaningful morphological information. In the second stage, we investigate different ways to predict the tag sequence that differ in how the *global* information is used.

2.1 From character to word level

To obtain word embeddings, the usual approach introduced by (Bengio et al., 2003) relies on a fixed vocabulary $\mathcal{W}$ and each word $w \in \mathcal{W}$ is mapped to a vector of n_f real valued features by a look-up matrix $W \in \mathbb{R}^{|\mathcal{W}|*n_f}$. To avoid the use of a fixed vocabulary, we propose to derive a word representation from a sequence of character embedding: if $\mathcal{C}$ denotes the finite set of characters, each character is mapped on a vector of n_c features gathered in the look-up matrix C.

To infer a word embedding , we use a *convolution layer* (Waibel et al., 1990; Collobert et al., 2011), build as in (Santos and Zadrozny, 2014). As illustrated in figure 1, a word w is a character sequence $\{c_1, .., c_{|w|}\}$ represented by their embeddings $\{C_{c_1}, .., C_{c_{|w|}}\}$, where C_{c_i} denotes the row in C associated to the character c_i. A convolution filter $W^{conv} \in \mathbb{R}^{n_f} \times \mathbb{R}^{d_c * n_c}$ is applied over a sliding window of d_c characters, producing local features :

$$x_n = W^{conv}(C_{c_{n-d_c+1}} : .. : C_{c_n})^T + b^{conv},$$

where x_n is a vector of size n_f obtained for each position n in the word[2]. The i-th element of the embedding of w is the maximum over the i-th elements of the feature vectors :

$$[f]_i = \tanh(\max_{1 \leq n \leq |s|} [x_n]_i)$$

Using a maximum after a sliding convolution window ensures that the embedding combines local features from the whole word, and selects the more useful ones. The parameters of the layer are the matrices C and W^{conv} and the bias b^{conv}.

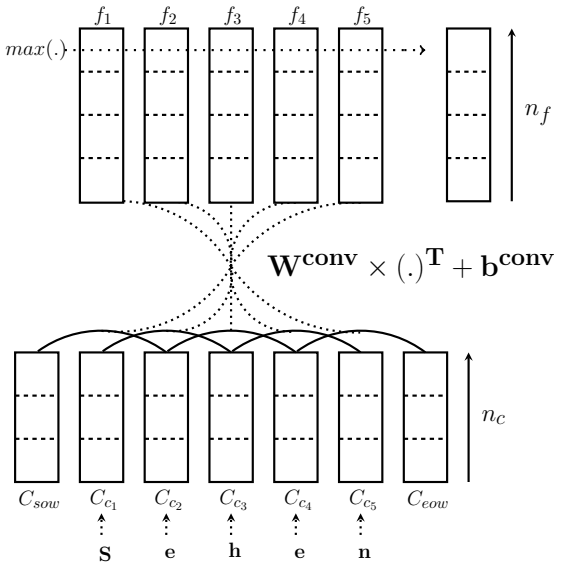


Figure 1: Architecture of the layer for character-level encoding of words.

2.2 From words to prediction

To predict the tag sequence associated to a sentence s, we first use **a feedforward architecture**, with a single hidden layer. To compute the probability of tagging the n-th word in the sentence with tag t_i, we use a window of d_w word embeddings[3] centered around the word w_n:

$$x_n = f_{n-\frac{d_w-1}{2}} : ... : f_{n+\frac{d_w-1}{2}},$$

followed by a hidden and output layers:

$$s_n = W^o \tanh(W^h x_n + b^h) + b^o. \qquad (1)$$

The parameters of the hidden an output layers are respectively W^h, b^h and W^o, b^o.

We also experiment with a **a bidirectional recurrent layer**, as described in (Graves et al., 2013). The forward and backward passes allow each prediction to be conditioned on the complete past and future contexts, instead of merely a neighboring window. As illustrated in figure 2, the forward hidden state, at position n, will be computed using the previous forward hidden state and the word embedding in position n:

$$\overrightarrow{h^n} = \tanh(\overrightarrow{W^{fh}} f_n + \overrightarrow{W^{hh}} \overrightarrow{h^{n-1}} + b^h)$$

[2]Two padding character tokens are used to deal with border effects. The first is added at the beginning and the second at the end of the word, as many times as it is necessary to obtain the same number of windows than the length of the word. Their embeddings are added to C.

[3]Similarly, we use special word tokens for padding.

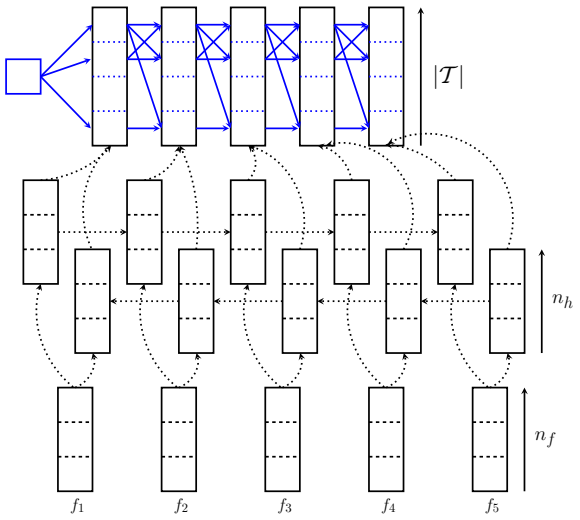


Figure 2: Bidirectional recurrent architecture for tag prediction. The upper part is used in the case of structured inference.

$\overrightarrow{W^{fh}}$ and $\overrightarrow{W^{hh}}$ are the transition matrices of the forward part of the layer, and b_h is the bias. The backward hidden states are computed similarly, and the hidden states of each direction are concatenated to pass through an output layer:

$$s_n = W^o(\overrightarrow{h^n} : \overleftarrow{h^n}) + b^o. \quad (2)$$

2.3 Inference and Training

To infer the tag sequence from the sequence of output layers defined by equations 1 or 2, we explore two strategies. The first simply applies a softmax function to the output layer of the network described in the previous section. In this case, each tag prediction is made independently of the surrounding predictions.

For sequence labeling, a more appropriate solution relies on the approach of (Collobert, 2011), also used in (Santos and Zadrozny, 2014). Let consider each possible tag sequence $\{t_1, \ldots, t_{|s|}\}$ as a possible path over a sequence of hidden states. We can add a transition matrix W^{trans} and then compute the score of a sequence as follows:

$$s(\{t\}_1^{|s|}, \{w\}_1^{|s|}) = \sum_{1 \leq n \leq |s|} \left(W^{trans}_{t_{n-1}, t_n} + [s_n]_{t_n} \right)$$

The Viterbi algorithm (Viterbi, 1967) offers an exact solution to infer the path that gives the maximum score. It is worth noticing that both these strategies can be applied to the feedforward and bidirectional recurrent networks. For both strategies, the whole network can estimate conditional log-likelihood of a tag sequence given a sentence s and the set of parameters θ. This criterion can then be optimized using a stochastic gradient ascent with the back-propagation algorithm.

3 Related Work

The choice to consider words from the character level has recently been more and more explored. While its raw application to language modeling did not achieve clear improvement over the word-based models (Mikolov et al., 2012), this approach shown impressive results for text generation (Sutskever et al., 2011; Graves, 2013). However, for this line of work, the main issue is to learn long range dependencies at the character level since the word level is not considered by the model.

More recently, the character level was considered as more interpretable and convenient way to explore and understand recurrent networks (Karpathy et al., 2015). In (Zhang and LeCun, 2015), the authors build a text understanding model that does not require any knowledge and uses hierarchical feature extraction. Here the character level allows the model to ignore the definition *a priori* of a vocabulary and let the model build its own representation of a sentence or a document, directly from the character level. To some extent, our work can be considered as an extension of their work, tailored for POS tagging.

(Santos and Zadrozny, 2014) applies a very similar model to the POS tagging of Portuguese and English. (Luong et al., 2013) also descends lower than the word level, using a dictionary of morphemes and recursive neural networks to model the structure of the words. Similarly, this allows a better representation of rare and complex words, evaluated on a word similarity task.

4 Experiments and Results

Experiments are carried out on the Part-of-Speech and Morphological tagging tasks using the German corpus TIGER Treebank (Brants et al., 2002). To the best of our knowledge, the best results on this task were published in (Mueller et al., 2013), who applied a high-order CRF that includes an intensive feature engineering to five different languages. German was highlighted as having 'the most ambiguous morphology'. The corpus, de-

Architecture	Encoding	Output	POS		POS+Morph	
			Dev	Test	Dev	Test
Feedforward	Lex.	Simple	4.22 ± 0.05	5.89 ± 0.07	13.97 ± 0.14	17.46 ± 0.14
		Struct.	3.90 ± 0.05	5.33 ± 0.09	12.22 ± 0.13	15.34 ± 0.13
	Non-lex.	Simple	3.31 ± 0.07	4.22 ± 0.07	13.50 ± 0.16	16.23 ± 0.13
		Struct.	2.92 ± 0.02	3.82 ± 0.04	11.65 ± 0.11	14.43 ± 0.19
	Both	Simple	2.59 ± 0.05	3.34 ± 0.09	11.89 ± 0.14	14.63 ± 0.22
		Struct.	$2.22 \pm 0.03^*$	$2.86 \pm 0.03^*$	9.11 ± 0.14	11.29 ± 0.06
biRNN	Lex	Simple	6.03 ± 0.06	8.05 ± 0.05	17.83 ± 0.11	21.33 ± 0.26
		Struct.	3.89 ± 0.06	5.26 ± 0.05	11.88 ± 0.05	17.78 ± 0.12
	Non-Lex	Simple	4.46 ± 0.08	5.84 ± 0.19	16.61 ± 0.18	19.39 ± 0.12
		Struct.	2.74 ± 0.07	3.59 ± 0.07	10.09 ± 0.09	12.88 ± 0.28
	Both	Simple	3.63 ± 0.06	4.63 ± 0.04	14.83 ± 0.11	17.54 ± 0.13
		Struct.	$2.21 \pm 0.04^*$	$2.86 \pm 0.05^*$	$\mathbf{8.63 \pm 0.21}^*$	$\mathbf{10.97 \pm 0.19}^*$
	CRF		**2.06**	**2.56**	9.40	11.42

Table 1: Comparison of the feedforward and bidirectional recurrent architectures for predictions, with different settings. The non-lexical encoding is convolutional. *CRF* refers to state-of-the-art system of (Mueller et al., 2013). *Simple* and *Struct.* respectively denote the position-by-position and structured prediction. $*$ indicates our best configuration.

scribed in details in (Fraser et al., 2013), contains a training set of 40472 sentences, a development and a test set of both 5000 sentences. We consider the two tagging tasks, with first a coarse tagset (54 tags), and then a morpho-syntactical rich tagset (619 items observed on the the training set).

4.1 Experimental settings

All the models are implemented[4] with the Theano library (Bergstra et al., 2010). For optimization, we use Adagrad (Duchi et al., 2011), with a learning rate of 0.1. The other hyperparameters are: the window sizes, d_c and d_w, respectively set to 5 and 9, the dimension of character embeddings, word embeddings and of the hidden layer, n_c, n_f and n_h, that are respectively of 100, 200 and 200[5]. The models were trained on 7 epochs. Parameter initialization and corpus ordering are random, and the results presented are the average and standard deviation of the POS Tagging error rate over 5 runs.

[4]Implementation is available at `https://github.com/MatthieuLabeau/NonlexNN`

[5]For both the learning rate and the embedding sizes, results does not differ in a significant way in a large range of hyperparameters, and their impact resides more in convergence speed and computation time

4.2 Results

The first experiment aims to evaluate the efficiency of a convolutional encoding with the basic feedforward architecture for prediction. We compare a completely non-lexicalized model which relies only on a character-level encoding with a lexicalized model where we use conventional word embeddings stored with a fixed vocabulary[6]. Results are reported in Table 1 along with with the state-of-the-art results published in (Mueller et al., 2013). Results show that a character-level encoding yields better results than the conventional word-level encoding. Moreover, the structured inference allows the model to achieve accuracy reasonably close to the performance of a high-order CRF that uses handcrafted features. Finally, the model that uses the concatenation of both the character and word-level embeddings outperforms the state-of-the-art system on the more difficult task, without any feature engineering.

To give an idea of how a simple model would perform on such task, the reader can refer to (Schmid and Laws, 2008) and (Mueller et al., 2013). For instance in the former, by choosing the most probable tag position-by-position, the error rate on the development set of the TIGER dataset

[6]Every word that appears in the training set.

is 32.7 for the simple POS Tagging task.

We further analyze the results by looking at the error rates respectively on known and unknown words[7]. From table 2, we observe that the number of unknown words wrongly labeled is divided by 3 for POS and almost divided by 2 for POS+Morph tagging, showing the ability of character-level encoding to generalize to new words. Moreover, a strictly non-lexical encoding makes slightly more mistakes on words already seen, whereas the model that concatenates both embeddings will make less mistakes for both unknown and known words.

This shows that information from the context and from the morphology are complementary, which is conjectured in (Mueller et al., 2013) by using a morphological analyzer in complement of higher-order CRF.

		Lex.	Non-lex.	Both
POS	Unknown	2970	1054	1010
	Known	1974	2981	1620
POS+Morph	Unknown	5827	3472	3384
	Known	8652	10205	7232

Table 2: Error counts for known/unknown words in the test set, with a structured feedforward prediction model for the tagging task.

In the second set of experiments, we evaluate the convolutional encoding with a bidirectional recurrent network for prediction. Results are presented in the second half of Table 1. Surprisingly, this architecture performs poorly with simple inference, but clearly improves when predicting a structured output using the Viterbi algorithm, both for training and testing. Moreover, a non-lexical model trained to infer a tag sequence with the Viterbi algorithm achieves results that are close to the state-of-the-art, thus validating our approach. We consider that this improvement comes from the synergy between using a global training objective with a global hidden representation, complexifying the model but allowing a more efficient solution. Finally, the model that uses the combination of both the character and word-level embeddings yields the best results. It is interesting to notice that the predictive architecture has no influence on the results of the simple task when the prediction is structured, but improves them on the difficult task. This also shows that the contribution of word embeddings to our model corresponds to a difference of 1.5 to 2 points in performance.

[7]Unknown words refer to words present in the development or test sets, but not in the training set.

5 Conclusion

In this paper, we explored new models that can infer meaningful word representations from the raw character stream, allowing the model to exploit the morphological properties of words without using any handcrafted features or external tools. These models can therefore efficiently process words that were unseen in the training data. The evaluation was carried out on a POS and morphological tagging task for German. We described different architectures that act in two stages: the first stage is a convolutional network that infers a word representation directly from the character stream, while the second stage performs the prediction. For the prediction stage, we investigated different solutions showing that a bidirectional recurrent network can outperform state-of-the-art results when using a structured inference algorithm.

Our results showed that character-level encoding can address the unknown words problem for morphologically complex languages. In the future, we plan to extend these models to other tasks such as syntactic parsing and machine translation. Moreover, we will also investigate other architectures to infer word embeddings from the character level. For instance, preliminary experiments show that bidirectional recurrent network can achieve very competitive and promising results.

Acknowledgments

We would like to thank the anonymous reviewers for their helpful comments and suggestions. This work has been partly funded by the European Unions Horizon 2020 research and innovation programme under grant agreement No. 645452 (QT21).

References

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137 1155.

James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. 2010. Theano: a CPU and

GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*, June. Oral Presentation.

Sabine Brants, Stefanie Dipper, Silvia Hansen, Wolfgang Lezius, and George Smith. 2002. The TIGER treebank. In *Proceedings of the workshop on treebanks and linguistic theories*, pages 24–41.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.*, 12:2493–2537, November.

Ronan Collobert. 2011. Deep learning for efficient discriminative parsing. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2011, Fort Lauderdale, USA, April 11-13, 2011*, pages 224–232.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.*, 12:2121–2159, July.

Alexander Fraser, Helmut Schmid, Richárd Farkas, Renjing Wang, and Hinrich Schütze. 2013. Knowledge sources for constituent parsing of German, a morphologically rich and less-configurational language. *Comput. Linguist.*, 39(1):57–85, March.

Alex Graves, Navdeep Jaitly, and Abdel-rahman Mohamed. 2013. Hybrid speech recognition with deep bidirectional LSTM. In *2013 IEEE Workshop on Automatic Speech Recognition and Understanding, Olomouc, Czech Republic, December 8-12, 2013*, pages 273–278.

Alex Graves. 2013. Generating sequences with recurrent neural networks. *CoRR*, abs/1308.0850.

Andrej Karpathy, Justin Johnson, and Fei-Fei Li. 2015. Visualizing and understanding recurrent networks. *CoRR*, abs/1506.02078.

Thang Luong, Richard Socher, and Christopher D. Manning. 2013. Better word representations with recursive neural networks for morphology. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning, CoNLL 2013, Sofia, Bulgaria, August 8-9, 2013*, pages 104–113.

Tomas Mikolov, Ilya Sutskever, Anoop Deoras, Hai-Son Le, Stefan Kombrink, and Jan Cernocky. 2012. Subword language modeling with neural networks. Unpublished.

Thomas Mueller, Helmut Schmid, and Hinrich Schütze. 2013. Efficient higher-order CRFs for morphological tagging. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 322–332, Seattle, Washington, USA, October. Association for Computational Linguistics.

Cicero D. Santos and Bianca Zadrozny. 2014. Learning character-level representations for part-of-speech tagging. In Tony Jebara and Eric P. Xing, editors, *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1818–1826. JMLR Workshop and Conference Proceedings.

Helmut Schmid and Florian Laws. 2008. Estimation of conditional probabilities with decision trees and an application to fine-grained pos tagging. In *Proceedings of the 22Nd International Conference on Computational Linguistics - Volume 1*, COLING '08, pages 777–784, Stroudsburg, PA, USA. Association for Computational Linguistics.

Ilya Sutskever, James Martens, and Geoffrey Hinton. 2011. Generating text with recurrent neural networks. In Lise Getoor and Tobias Scheffer, editors, *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, ICML '11, pages 1017–1024, New York, NY, USA, June. ACM.

Andrew Viterbi. 1967. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Trans. Inf. Theor.*, 13(2):260–269, April.

Alexander Waibel, Toshiyuki Hanazawa, Geofrey Hinton, Kiyohiro Shikano, and Kevin J. Lang, 1990. *Readings in Speech Recognition*, chapter Phoneme Recognition Using Time-delay Neural Networks, pages 393–404. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Xiang Zhang and Yann LeCun. 2015. Text understanding from scratch. *CoRR*, abs/1502.01710.

Online Representation Learning in Recurrent Neural Language Models

Marek Rei
The ALTA Institute
Computer Laboratory
University of Cambridge
United Kingdom
marek.rei@cl.cam.ac.uk

Abstract

We investigate an extension of continuous online learning in recurrent neural network language models. The model keeps a separate vector representation of the current unit of text being processed and adaptively adjusts it after each prediction. The initial experiments give promising results, indicating that the method is able to increase language modelling accuracy, while also decreasing the parameters needed to store the model along with the computation required at each step.

1 Introduction

In recent years, neural network models have shown impressive performance on many natural language processing tasks, such as speech recognition (Chorowski et al., 2014; Graves et al., 2013), machine translation (Kalchbrenner and Blunsom, 2013; Cho et al., 2014), text classification (Le and Mikolov, 2014; Kalchbrenner et al., 2014) and image description generation (Kiros et al., 2014). One of the main advantages of these methods is the ability to learn smooth vector representations for words, thereby reducing the sparsity problem inherent in any natural language dataset.

Language modelling is another task where neural networks have delivered excellent results (Bengio et al., 2003; Mikolov et al., 2011). Chelba et al. (2014) have recently benchmarked several well-known language models by training on very large datasets. They found that a recurrent neural network language model (RNNLM) combined with a 9-gram MaxEnt model was able to give the best results and lowest perplexity.

In this work we investigate a possible extension of RNNLM, by allowing it to continue learning and adapting during testing. The model keeps a vector representation of the current sentence that is being processed, and continuously modifies it based on an error signal. We refer to this as a version of online learning, as gradient descent is used to optimise the vector even during testing.

The technique is inspired by work on representation learning (Collobert and Weston, 2008; Mnih and Hinton, 2008; Mikolov et al., 2013), especially Le and Mikolov (2014) who use a related model to learn representations for text classification. We extend the idea to recurrent models and apply it to the task of language modelling. Our results indicate that by exchanging some existing model parameters for a component using online learning, the system is able to achieve lower perplexity while also reducing the necessary computation.

2 RNNLM

We base our implementation of the RNNLM on Mikolov et al. (2011), shown in Figure 1. The input layer to the network consists of a 1-hot vector representing the previous word in the sequence, and the hidden vector from the previous time step. These are multiplied by corresponding weight matrices and the resulting vectors are passed through an activation function to calculate the hidden vec-

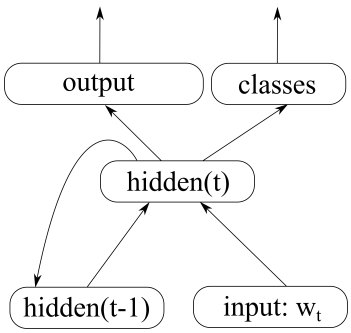


Figure 1: Recurrent neural network language model (RNNLM)

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 238–243,
Lisbon, Portugal, 17-21 September 2015.

tor at the current time step.[1]

Class-based output architecture is used to avoid calculating the softmax over all words in the vocabulary. The probability distributions over words and classes are calculated by multiplying the hidden vector with the corresponding weight matrix and applying the softmax function:

$$hidden_t = \sigma(E \cdot input_t + W_h \cdot hidden_{t-1})$$

$$classes = softmax(W_c \cdot hidden_t)$$

$$output = softmax(W_o^{(c)} \cdot hidden_t)$$

where σ is the logistic function and $W_o^{(c)}$ is the weight matrix between the hidden layer and the output words in class c.

Finally, we multiply the probability of the next word belonging to class c with the output probability of the next word given the class to get the overall probability of the next word given the previous words:

$$P(w_{t+1}|w_1^t) \approx classes_c \cdot output_{w_{t+1}}$$

Negative log-probability is used as the loss function, which optimises the network to assign a high probability to the correct words. The network is trained using gradient descent and backpropagation through time. In the basic model, this means unrolling the recurrent network for a fixed number of time steps, essentially turning it into a deep feedforward network which outputs probability distributions on different layers. Instead of using a fixed number of steps, our implementation unrolls each sentence from the last word to the first word, making it more suitable for processing individual sentences as opposed to longer texts.

In addition, we introduce a special vector to use as the hidden vector at the start of each sentence. The values in this vector are treated as parameters and optimised during training. This allows the network to learn a suitable starting point when no other information is available, giving slight performance improvements in our experiments.

3 RNNLM with online learning

We extend the RNNLM by introducing an additional document/context vector, shown as *doc* in Figure 2. This vector will represent the current

[1]Explicit multiplication for the word vectors can be avoided by using data structures that retrieve the correct vector in constant time.

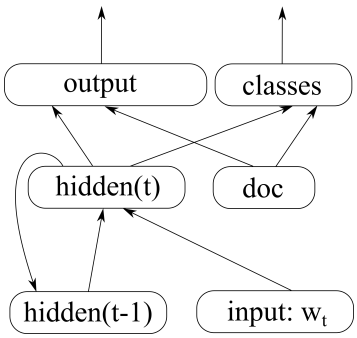


Figure 2: RNNLM with an additional document vector for active learning

document being processed, whether that is a sentence, paragraph or a larger text. When calculating output probabilities over classes and words, we also condition them on this new document vector:

$$classes = softmax(W_c \cdot hidden_t + W_{dc} \cdot doc)$$

$$output = softmax(W_o^{(c)} \cdot hidden_t + W_{do}^{(c)} \cdot doc)$$

where W_{dc} is the weight matrix between the document vector and class layer, and $W_{do}^{(c)}$ is the weight matrix between the document vector and output words in class c.

We construct the document vector by treating the values as parameters and optimising them during both training and testing using backpropagation. At each time step, the system first performs a forward pass through the network and outputs probability distributions over classes and words. We then use the next word in the sequence to calculate the error derivatives in the output and backpropagate them back into the document vector. The update is not able to to affect the output at the current time step, but it will modify the document vector which will be used in the next time step. The same word that is used for modifying the document vector for the next time step is also available in the input layer of the next time step, therefore the system receives no additional knowledge as input.

We are interested in modelling individual sentences, therefore at the beginning of each sentence the document vector is reset to a specific starting state, which is optimised during training and shared between all sentences. During testing, the values in the document vector are continuously modified depending on the error derivatives being backpropagated from the output layer, while all other parameters in the model stay constant.

When dealing with larger texts and domain-specific corpora, similar ideas of iterative learning can be applied to any language model. After processing a certain amount of data during testing, a new model could be trained using the previously seen testing examples as additional training data. Since this process adds more training data which is likely to be similar to upcoming testing examples, the system is likely to achieve a better performance.

However, when dealing with independent sentences, online learning becomes more difficult to apply. Each sentence contains very little additional data, and even if the language model is adjusted after every individual word, it only obtains evidence of previous words in the sentence, whereas these words are relatively unlikely to occur again in the same sentence. Therefore, instead of adjusting individual word representations, our approach learns a distributed document vector to represent the specific unit of text that is currently being processed. This vector is then used as additional evidence when calculating output probabilities.

Le and Mikolov (2014) use a similar method for learning vector representations of documents and paragraphs. They construct a feedforward language model and include a paragraph vector as an additional vector in the input layer. The model parameters are trained on the training set, and when given unseen test data, the system optimises the paragraph vector according to the error signal. They use these vectors as input to a logistic regression classifier and achieve state-of-the-art performance on sentiment classification of movie reviews. However, they did not consider the effect of this model modification directly on the task of language modelling.

While the system of Le and Mikolov (2014) uses a basic feedforward language model, we extend the idea to recurrent neural network language models, as they are currently used in state-of-the-art language modelling systems (Chelba et al., 2014). Attaching the document vector to the input layer is not preferable for RNNLM, as the error is only backpropagated into the input layer after several time steps. When this time step is reached and the network is unrolled to perform backpropagation through time, several words have already passed without receiving any additional information. Since our implementation performs the unrolling only at the end of each sentence, the updates would not have any effect. Therefore, we attach the document vector directly to the output layer, in parallel with the recurrent hidden component. Parameters in the document vector can then be updated at each time step, while the unrolling and backpropagation through time still happens at the end of the sentence.

4 Experiments

We constructed a dataset from English Wikipedia to evaluate language modelling performance over individual sentences. The text was tokenised, sentence split and lowercased. The sentences were shuffled, in order to minimise any transfer effects between consecutive sentences, and then split into training, development and test sets. The final sentences were sampled randomly, in order to obtain reasonable training times for the experiments. The dataset sizes are shown in Table 2.

	Train	Dev	Test
Words	9,990,782	237,037	4,208,847
Sentences	419,278	10,000	176,564

Table 2: Dataset sizes

Model performance is measured using perplexity, therefore lower values indicate a model which is able to better predict the data. Special tokens are used to mark the beginning and end of a sentence. The sentence end token is also included in the evaluation, whereas the sentence start token is only used as context in the input layer. Any words that occur less than 30 times in the training data were replaced by a special token for unknown words, leaving a vocabulary of 16,514 unique words. General learning rate was set to 0.1 and decreased during training, whereas the learning rate of the document vector was fixed at 0.1 for both training and testing.

As the baseline, we use the regular RNNLM with 100-dimensional hidden layers and word vectors ($M = 100$). In the experiments we increase the capacity of the model and measure how that affects the perplexity on the datasets. First, we increase the value of M, allowing more information to be stored into word representations, while also increasing the number of hidden-hidden and hidden-output connections. As can be seen in Table 1, this improves the overall performance of the

	Train PPL	Dev PPL	Test PPL	+Parameters	+Operations
Baseline M=100	92.65	103.56	102.51	–	–
M=120	88.60	98.78	97.79	666,960	7,400
M=100, D=20	**87.28**	**95.36**	**94.39**	**332,300**	**6,000**
M=135	85.17	96.33	95.71	1,167,705	13,475
M=100, D=35	**80.11**	**91.05**	**90.29**	**581,525**	**10,500**

Table 1: Perplexity and additional parameters/operations for different language model configurations

model – setting M to 120 and 135 leads to progressively lower perplexity.

Next, instead of increasing M, we add a D-dimensional document vector to the model and use this for online learning. When the same number of elements is added to M or D, our results show consistently better performance when using the document vector. Increasing M by 35 gives perplexity 95.71, whereas using a 35-dimensional document vector gives perplexity 90.29. We also performed the same experiment using only half of the training data, and the difference was even larger – 105.50 and 98.23 correspondingly.

One reason why online learning during model deployment is not commonly used is because it is computationally expensive. Continuously retraining the model and adjusting parameters can be very time-consuming compared to a simple feed-forward process through the network. However, extra computation is also needed when using a hidden vector of size M, as opposed to using a smaller value. When increasing the value of M to $M + X$, the RNNLM will contain

$$X \cdot C + 2 \cdot X \cdot V + 2 \cdot X \cdot M + X^2$$

additional parameters and needs to perform

$$2 \cdot X \cdot M + X^2 + X \cdot C + X \cdot E[O]$$

additional operations at each time step.[2] C is the number of classes, V is vocabulary size, and $E[O]$ is the expected number of words that need to be processed in the output layer during one step.

The corresponding number of additional parameters in a RNNLM model using a D-dimensional document vector for online learning is

$$D + D \cdot V + D \cdot C$$

and additional operations

$$2 \cdot D \cdot E[O] + 2 \cdot D \cdot C$$

which includes the error backpropagation at each time step. For our experiments $V = 16,514$, $C = 100$ and $E[O] \approx 50$. Table 1 contains the additional values for the experiments, showing that replacing some hidden vector parameters with the actively learned document vector leads to fewer total parameters and fewer operations, along with lower perplexity.

Figure 3 presents the relationship between perplexity and the number of additional parameters, when increasing either M or D. The results are averaged over 10 runs with different random initialisations. As can be seen, using a small document vector lowers the perplexity with fewer parameters, compared to simply increasing the main components of the network. The graph of perplexity with respect to additional operations in the model also has a very similar shape.

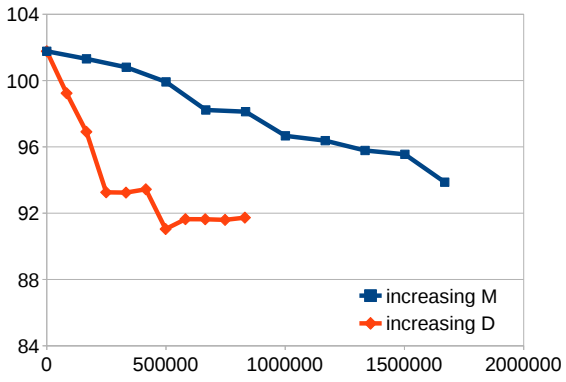


Figure 3: Perplexity as a function of additional parameters when increasing either M or D. The x-axis shows the number of additional parameters in the model, with respect to the baseline of $M = 100$, $D = 0$. The y-axis shows the perplexity on the test set.

[2]We only count the matrix multiplication operations, as they take the majority of the time in a neural network language model.

Both Hufnagel and Marston also joined the long-standing technical death metal band Gorguts. 1. The band eventually went on to become the post-hardcore band Adair. 2. The band members originally came from different death metal bands, bonding over a common interest in d-beat. 3. The proceeds went towards a home studio, which enabled him to concentrate on his solo output and songs that were to become his debut mini-album "Feeding The Wolves".
The Chiefs reclaimed the title on September 29, 2014 in a Monday Night Football game against the New England Patriots, hitting 142.2 decibels. 1. He played in twenty-four regular season games for the Colts, all off the bench. 2. In May 2009 the Warriors announced they had re-signed him until the end of the 2011 season. 3. The team played inconsistently throughout the campaign from the outset, losing the opening two matches before winning four consecutive games during September 1927.
He was educated at Llandovery College and Jesus College, Oxford, where he obtained an M.A. degree. 1. He studied at the Orthodox High School, then at the Faculty of Mathematics. 2. Kaigama studied for the priesthood at St. Augustine's Seminary in Jos with further study in theology in Rome. 3. Under his stewardship, Zahira College became one of the leading schools in the country.

Table 3: Examples of using the document vectors to find similar sentences in the development data.

In order to further explore the relationship between D and M, we trained a number of smaller models with different values, under the constraint $D + M = 100$. To reduce computation time, only half of the training data was used in these experiments. The lowest perplexity was achieved in the region of $D = 23$ and $M = 77$, and making the document vectors much smaller or larger led to a decrease in performance. This indicates that including the document vector does help increase model accuracy, but as it contains no information about the training data, this vector should be small compared to the main model.

Intuitively, this approach works by having the document vector capture the unique aspects of each sentence. While the general RNNLM is a smooth static representation of the entire training data, the document vector is optimised to represent how each sentence differs from the main language model. Therefore we performed a qualitative evaluation and found that the learned sentence vectors were also very good predictors of semantic similarity. The RNN language model was trained on the training set, and then used to process the development set. The last state of the document vector of each sentence was used to calculate cosine similarity. Table 3 contains randomly sampled sentences from the development set, together with corresponding development sentences that have the highest similarity (excluding the original sentence). Even though there is almost no word overlap, the retrieved sentences are semantically very similar.

5 Conclusion

We have described a possible extension of RNNLM which uses continuous online learning. The model includes a separate vector to represent the unit of text, such as a sentence, being currently processed. The vector starts in a default state and is continuously updated using backpropagation, leading to a more informative representation. The modified language model achieves lower perplexity with a more optimal use of parameters.

The idea of continuous training and adaptation is natural and also established in biological learning processes, yet it is not widely used due to computational complexity. Our experiments indicate that by including this active learning component in the neural network model, the system is able to achieve higher accuracy, while also decreasing the parameters needed to store the model and decreasing the computation required.

References

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A Neural Probabilistic Language Model. 3:1137–1155.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2014. One Billion Word Benchmark for Measuring Progress in Statistical Language Modeling. In *INTERSPEECH 2014*.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. June.

Jan Chorowski, Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. End-to-end Continuous Speech Recognition using Attention-based Recurrent NN : First Results.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. *Proceedings of the 25th international conference on Machine learning*.

Alex Graves, Navdeep Jaitly, and Abdel-rahman Mohamed. 2013. Hybrid speech recognition with deep bidirectional LSTM. In *ASRU 2013*.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent Continuous Translation Models. In *EMNLP*.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *The 52nd Annual Meeting of the Association for Computational Linguistics. Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*.

Ryan Kiros, Ruslan Salakhutdinov, and Richard Zemel. 2014. Multimodal Neural Language Models. In *ICML 2014*.

Quoc Le and Tomas Mikolov. 2014. Distributed Representations of Sentences and Documents. In *ICML*, volume 32.

Tomaš Mikolov, Stefan Kombrink, Anoop Deoras, Lukáš Burget, and Jan Černocký. 2011. RNNLM - Recurrent neural network language modeling toolkit. In *ASRU 2011 Demo Session*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. *ICLR Workshop*, pages 1–12.

Andriy Mnih and Geoffrey Hinton. 2008. A Scalable Hierarchical Distributed Language Model. *Advances in Neural Information Processing Systems*, pages 1–8.

A Model of Zero-Shot Learning of Spoken Language Understanding

Majid Yazdani
Computer Science Department
University of Geneva
majid.yazdani@unige.ch

James Henderson
Xerox Research Center Europe
james.henderson@xrce.xerox.com

Abstract

When building spoken dialogue systems for a new domain, a major bottleneck is developing a spoken language understanding (SLU) module that handles the new domain's terminology and semantic concepts. We propose a statistical SLU model that generalises to both previously unseen input words and previously unseen output classes by leveraging unlabelled data. After mapping the utterance into a vector space, the model exploits the structure of the output labels by mapping each label to a hyperplane that separates utterances with and without that label. Both these mappings are initialised with unsupervised word embeddings, so they can be computed even for words or concepts which were not in the SLU training data.

1 Introduction

Spoken Language Understanding (SLU) in dialogue systems is the task of taking the utterance output by a speech recognizer and assigning it a semantic label that represents the dialogue actions of that utterance accompanied with their associated attributes and values. For example, the utterance "I would like Chinese food" is labelled with *inform(food=Chinese)*, in which *inform* is the dialogue action that provides the value of the attribute *food* that is *Chinese*.

Dialogue systems often use hand-crafted grammars for SLU, such as Phoenix (Ward, 1994), which are expensive to develop, and expensive to extend or adapt to new attributes and values. Statistical SLU models are usually trained on the data obtained from a specific domain and location, using a structured output classifier that can be discriminative (Pradhan et al., 2004; Kate and Mooney, 2006; Henderson et al., 2012) or generative (Schwartz et al., 1996; He and Young, 2005). Gathering and annotating SLU data is costly and time consuming and therefore SLU datasets are small compare to the number of possible labels.

Because training sets for a new domain are small, or non-existent, learning is often an instance of Zero-shot or One-shot learning problems (Palatucci et al., 2009; L. Fei-Fei; Fergus, 2006), in which zero or few examples of some output classes are available during the training. For example, in the restaurant reservation domain, not all possible combinations of foods and dialogue actions may be included in the training set. The general idea to solve this type of problems is to map the input and class labels to a semantic space of usually lower dimension in which similar classes are represented by closer points in the space (Palatucci et al., 2009; Weston et al., 2011; Weston et al., 2010). Usually unsupervised knowledge sources are used to form semantic codes of the labels that helps us to generalize to unseen labels.

On the other hand, there are also different ways to express the same meaning, and similarly, most of them can not be included in the training set. For instance, the system may have seen "Please give me the telephone number" in training, but the user might ask "Please give me the phone" at test time. This problem, feature sparsity, is a common issue in many NLP tasks. Decomposition of input feature parameters using vector-matrix multiplication (Bengio et al., 2003; Collobert et al., 2011; Collobert and Weston, 2008) has addressed this sparsity issue successfully in previous work. In this way, by sharing the word representations and composition matrices, we can overcome feature sparsity by producing similar representations for similar utterances.

In order to represent words and concepts we use word embeddings, which are a form of vector space model. Word embeddings have proven to be effective models of semantic representation

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 244–249,
Lisbon, Portugal, 17-21 September 2015.

of words in various NLP tasks (Baroni et al., 2014; Yazdani and Popescu-Belis, 2013; Collobert et al., 2011; Collobert and Weston, 2008; Huang et al., 2012; Mikolov et al., 2013b). In addition to parameter sharing, these representations enable us to leverage large scale unlabelled data. Because word embeddings trained on unlabeled data reflect the similarity between words, they help the model generalize from the words in the original training corpora to the words in the new extended domain, and help generalize from small amounts of data in the extended domain.

The contribution of this paper is to build a representation learning classifier for the SLU task that can generalize to unseen words and labels. For every utterance we learn how to compose the word vectors to form the semantics of that utterance for this task of language understanding. Furthermore, we learn how to compose the semantics of each label from the semantics of the words used to name that label. This enables us to generalize to unseen labels.

In this work we use the word2vec software of Mikolov et al. (2013a)[1] to induce unsupervised word embeddings that are used to initialize word embedding parameters. For this, we use an English Wikipedia dump as our unlabelled training corpus, which is a diverse broad-coverage corpus. It has been shown (Baroni et al., 2014; Mikolov et al., 2013b) that these embeddings capture lexical similarities even when they are trained on a diverse corpus like Wikipedia. We test our models on a restaurant booking domain. We investigate domain adaptation by adding new attribute types (e.g. *goodformeal*) and new attribute values (e.g. *Hayes Valley* as a restaurant location). Our experiments indicate that our model has better performance compared to a hand-crafted system as well as a SVM baseline.

2 SLU Datasets

The dialogue utterances used to build the SLU dataset were collected during a trial of online dialogue policy adaptation for a restaurant reservation system based in San Francisco. The trial began with (*area*, *pricerange* and *food*), and adapted the Interaction Manager online to handle the additional attribute types *near*, *allowedforkids*, and *goodformeal* (Gašic et al., 2014). User utterances from these trials were transcribed and annotated with dialogue acts by an expert, and afterwards edited by another expert[2]. Each user utterance was annotated with a set of labels, where each label consists of an act type (e.g. inform, request), an attribute type (e.g. foodtype, pricerange), and an attribute value (e.g. Chinese, Cheap).

The dataset is separated into four subsets, `SFCore`, `SF1Ext`, `SF2Ext` and `SF3Ext`, each with an increasing set of attribute types, as specified in Table 1. This table also gives the total number of utterances in each data set. For our first experiment, we split each dataset into about 15% for the testing set and 85% for the training set. For our second experiment we use each extended subset for testing and its preceding subsets for training.

Ontology	Attribute types (# of values)	# of utterances
SFCore	*food*(59), *area*(155), *pricerange*(3)	1103
SF1Ext	SFCore + *near*(39)	1810
SF2Ext	SF1Ext + *allowedforkids*(2)	1571
SF3Ext	SF2Ext +*goodformeal*(4)	1518

Table 1: Domains for San Francisco (SF) restaurants expanding in complexity

3 A Dialogue Act Representation Learning Classifier

The SLU model is run on each hypothesis output by the ASR component, and tries to predict the correct set of dialogue act labels for each hypothesis. This problem is in general an instance of multi-label classification, because a single utterance can have multiple dialogue act labels. Also, these labels are structured, since each label consist of an act type, an attribute type, and an attribute value. Each label component also has canonical text associated with it, which is the text used to name the label component (e.g. "Chinese" as a value).

The number of possible dialogue acts grows rapidly as the domain is extended with new attribute types and values, making this task one of multi-label classification with a very large number of labels. One natural approach to this task is to train one binary classifier for each possible label, to decide whether or not to include it in the output. In our case, this requires training a large number of classifiers, and it is impossible to generalize to

[1] `https://code.google.com/p/word2vec/`

[2] This data is publically available from `https://sites.google.com/site/parlanceprojectofficial/home/datarepository`

dialogue acts that include attributes or values that were not in the training set since there won't be any parameter sharing among label classifiers.

In our alternative approach, we build the representation of the utterance and the representation of the label from their constituent words, then we check if these representations match or not. In the following we explain in details this representation learning model.

3.1 Utterance Representation Learning

In this section we explain how to build the utterance representation from its constituent words. In addition to words, we use bigrams, since they have been shown previously to be effective features for this task (Henderson et al., 2012). Following the success in transfer learning from parsing to understanding tasks (Henderson et al., 2013; Socher et al., 2013), we use dependency parse bigrams in our features as well. We learn to build a local representation at each word position in the utterance by using the word representation, adjacent word representations, and the head word representation. Let $\phi(w)$ be a d dimensional vector representing the word w, and $\phi(U_i)$ be a h dimensional vector which is the local representation at word position i. We compute the local representation as follows:

$$\phi(U_i) = \sigma(\phi(w_i)W_{word} + \phi(w_h)W_{parse_{R_k}} + \phi(w_j)W_{previous} + \phi(w_k)W_{next}) \quad (1)$$

in which w_h is the head word with the dependency relation R_k to w_i, and w_j and w_k are the previous and next words. W_{word} is a $d \times h$ matrix that transforms the word embedding to hidden representation inputs. $W_{parse_{R_k}}$ is a $d \times h$ matrix for the relation R_k that similarly transforms the head word embedding (so W_{parse} is a tensor), and $W_{previous}$ and W_{next} similarly transform the previous and next words' embeddings. Figure 1 depicts this representation building at each word.

3.2 Label Representation Learning

One standard way to address the problem of multi-label classification is building binary classifiers for each possible label. Large margin classifiers have been shown to be an effective tool for this task (Pradhan et al., 2004; Kate and Mooney, 2006). We use the same idea of binary classifiers to learn one hyperplane per label, which separates the utterances with this label from all other utterances, with a large margin. In the standard way of

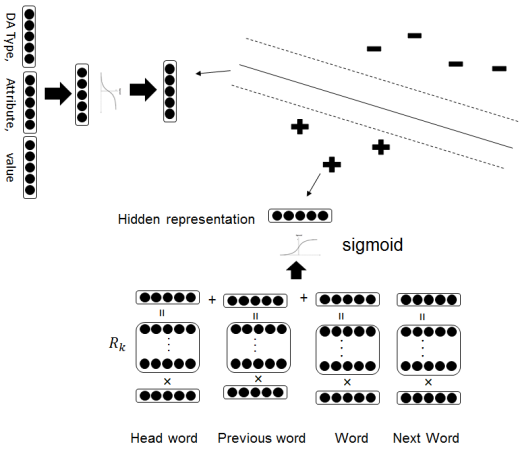


Figure 1: The multi-label classifier

building the classifier, each label's hyperplane is independent of other labels. To extend this model to a zero-shot learning classifier, we use parameter sharing among label hyperplanes so that similar labels have similar hyperplanes.

We exploit the structure of labels by assuming that each hyperplane representation is a composition of representations of the label's constituent components, namely dialogue action, attribute and attribute value. We learn the composition function and the constituent representations while training the classifiers, using the labelled SLU data. The constituent representations are initialised as the word embeddings for the label constituent's name string, such as "inform", "food" and "Chinese", where these embeddings are trained on the unlabelled data. Figure 1 depicts the classifier model.

We define the hyperplane of the label $a_j(att_k = val_l)$ with its normal vector W_{a_j,att_k,val_l} as:

$$W_{a_j,att_k,val_l} = \sigma([\phi(a_j), \phi(att_k), \phi(val_l)]W_{ih})W_{ho}$$

where $\phi(\cdot)$ is the same mapping to d dimensional word vectors that is used above in the utterance representation, W_{ih} is a $3d \times h$ matrix and W_{ho} is a $h \times h$ matrix. The score of each local representation vector $\phi(U_i)$ is its distance from this label hyperplane, which is computed as the dot product of the local vector $\phi(U_i)$ with the normal vector W_{a_j,att_k,val_l}.

We sum these local scores for each position i to build the whole utterance score: $\sum_i \phi(U_i)W^T_{a_j,att_k,val_l}$. Alternatively we can think of this computation as summing the local vectors to get a whole-utterance representation $\phi(U) = \sum_i \phi(U_i)$ and then doing the dot product. The

pooling method (sum) used in the model is (intentionally) over-simplistic. We did not want to distract from the main contribution of the paper, and our dataset did not justify any more complex solution since utterances are short. It can be replaced by more powerful approaches if it is needed.

To train a large margin classifier, we train all the parameters such that the score of an utterance is bigger than a margin for its labels and less than the negative margin for all other labels. Thus, the loss function is as follows:

$$\min_{\theta} \frac{\lambda}{2}\theta^2 + \sum_{U} \max(0, 1 - y \sum_{i} \phi(U_i) W^T_{a_j, att_k, val_l}) \quad (2)$$

where θ is all the parameters of the model, namely $\phi(w_i)$ (word embeddings), W_{word}, W_{Parse}, $W_{previous}$, W_{next}, W_{ih}, and W_{ho}. y is either 1 or -1 depending whether the input U has that label or not.

To optimize this large margin classifier we perform stochastic gradient descent by using the adagrad algorithm on this primal loss function, similarly to Pegasos SVM (Shalev-Shwartz et al., 2007), but here we backpropagate the errors to the representations to train the word embeddings and composition functions. In each iteration of the stochastic training algorithm, we randomly select an utterance and its labels as positive examples and choose randomly another utterance with a different label as a negative example. When choosing the negative sample randomly, we sample utterances with the same dialogue act but different attribute or value with 4 times higher probability than utterances with a different dialogue act. This biased negative sampling speeds up the training process since it provides more difficult training examples to the learner.

The model is able to address the adaptivity issues because the utterance and the dialogue act representations are in the same space using the same shared parameters $\phi(w)$, which are initialised with unsupervised word embeddings. It has been shown that such word embeddings capture word similarities and hence the classifier is no longer ignorant about any new attribute type or attribute value. Also, there is parameter sharing between dialogue acts because these word/label embeddings are shared, and the matrices for the composition of these representations are the same across all dialogue acts. This can help overcome sparsity in the SLU training set by transferring learning between similar situations and similar dialogue act triples. For example, if the training set does not contain any examples of the act "request(postcode)", but many examples of "request(phone)", sharing the parameters can help with the recognition of "request(postcode)" in utterances similar to "request(phone)". Moreover, the SLU model is to some extent robust against paraphrasing in the input utterance because it maps the utterance to a semantic space, and uses parse bigrams. More sophisticated vector-space semantic representations of the utterance are an area for future work, but should be largely orthogonal to the contribution of this paper.

To find the set of compatible dialogue acts for a given utterance, we should check all possible dialogue acts. This can severely slow down SLU. To avoid testing all possible dialogue combinations, we build three different classifiers: The first one recognises the act types in the utterance, the second one recognises the attribute types for each of the chosen act types, and the third classifier recognises the full dialogue acts as we described above, but only for the chosen pairs of act types and attribute types.

4 SLU Experiments

In the first experiment, we measure SLU performance trained on all available data, by building a dataset that is the union of all the above datasets. This measures the performance of SLU when there is a small amount of data for an extended domain. This dataset, similarly to `SF3Ext`, has 6 main attribute types. Table 2 shows the performance of this model. We report as baselines the performance of the Phoenix system (hand crafted for this domain) and a binary linear SVM trained on the same data. The hidden layers have size h=d=50. For this experiment, we split each dataset into about 15% for the testing set and 85% for the training set.

System	Outputs	Precision	Recall	F-core
Phoenix	516	84.10	41.65	55.71
SVM	690	65.03	52.45	58.06
Our	932	90.24	81.15	85.45

Table 2: Performance on union of data (SF-Core+SF1Ext+SF2Ext+SF3Ext)

Our SLU model can adapt well to the extended domain with more attribute types. We observe

model, train set	Test set SF1Ext P—R—F	SF2Ext P—R—F	SF3Ext P—R—F
Our SFcore	73.36—66.11—69.54	74.61—59.73—66.34	72.54—53.86—61.81
SVM SFcore	50.66— 38.7— 43.87	49.64—34.70— 40.84	48.99—30.91—37.90
Our SF1Ext		83.18—66.08—73.65	78.32—59.98—67.93
SVM SF1Ext		58.72—41.71—48.77	53.25—34.88—42.15
Our SF2Ext			84.12—67.78—75.07
SVM SF2Ext			59.27—42.80—49.70

Table 3: SLU performance: trained on a smaller domain and tested on more inclusive domains.

particularly that the recall is almost twice as high as the hand-crafted baseline. This shows that our SLU can recognise most of the dialogue acts in an utterance, where the rule-based Phoenix system and a classifier without composed output cannot. Overall there are 1042 dialogue acts in the test set. SLU recall is very important in the overall dialogue system performance, as the effect of a missed dialogue act is hard to handle for the Interaction Manager. Both hand-crafted and our system show relatively high precision.

In the next experiment, we measure how well the new SLU model performs in an extended domain without any training examples from that extended domain. We train a SLU model on each subset, and test it on each of the more inclusive subsets. Table 3 shows the results.

Not surprisingly, the performance is better if SLU is trained on a similar domain to the test domain, and adding more attribute types and values decreases the performance more. But our SLU can generalise very well to the extended domain, achieving much better generalisation that the SVM model.

4.1 Conclusion

In this paper, we describe a new SLU model that is designed for improved domain adaptation. The multi-label classification problem of dialogue act recognition is addressed with a classifier that learns to build an utterance representation and a dialogue act representation, and decides whether or not they are compatible. The dialogue act representation is a vector composition of its constituent labels' embeddings, and is trained as the hyperplane of a large margin binary classifier for that dialogue act. The utterance representation is trained as a composition of word embeddings. Since the utterance and the dialogue act representations are both built using unsupervised word embeddings and share these embedding parameters, the model can address the issues of domain adaptation. Word embeddings capture word similarities, and hence the classifier is able to generalise from known attribute types or values to similar novel attribute types or values. We tested this SLU model on datasets where the number of attribute types and values is increased, and show much better results than the baselines, especially in recall. The model succeeds in both adapting to an extended domain using relatively few training examples and in recognising novel attribute types and values.

Acknowledgments

The research leading to this work was funded by the EC FP7 programme FP7/2011-14 under grant agreement no. 287615 (PARLANCE), and Hasler foundation project no. 15019, Deep Neural Network Dependency Parser for Context-aware Representation Learning. The authors also would like to thank Dr.Helen Hastie for her help in annotating the dataset.

References

Marco Baroni, Georgiana Dinu, and Germán Kruszewski. 2014. Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 238–247.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3:1137–1155.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 160–167.

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

M Gašic, D Kim, P Tsiakoulis, C Breslin, M Henderson, M Szummer, B Thomson, and S Young. 2014. Incremental on-line adaptation of pomdp-based dialogue managers to extended domains.

Yulan He and Steve Young. 2005. Semantic processing using the hidden vector state model. *Computer Speech and Language*, 19:85–106.

Matthew Henderson, Milica Gašić, Blaise Thomson, Pirros Tsiakoulis, Kai Yu, and Steve Young. 2012. Discriminative Spoken Language Understanding Using Word Confusion Networks. In *Spoken Language Technology Workshop, 2012.*

James Henderson, Paola Merlo, Ivan Titov, and Gabriele Musillo. 2013. Multilingual joint parsing of syntactic and semantic dependencies with a latent variable model. *Comput. Linguist.*, 39(4):949–998.

Eric H. Huang, Richard Socher, Christopher D. Manning, and Andrew Y. Ng. 2012. Improving Word Representations via Global Context and Multiple Word Prototypes. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Rohit J. Kate and Raymond J. Mooney. 2006. Using string-kernels for learning semantic parsers. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, ACL-44, pages 913–920.

R.; Perona L. Fei-Fei; Fergus. 2006. One-shot learning of object categories. *IEEE Transactions on Pattern Analysis Machine Intelligence*, 28:594–611, April.

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

Tomas Mikolov, Wen tau Yih, and Geoffrey Zweig. 2013b. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT-2013)*.

Mark Palatucci, Dean Pomerleau, Geoffrey E. Hinton, and Tom M. Mitchell. 2009. Zero-shot learning with semantic output codes. In *Advances in Neural Information Processing Systems 22*, pages 1410–1418.

Sameer Pradhan, Wayne Ward, Kadri Hacioglu, and James H. Martin. 2004. Shallow semantic parsing using support vector machines.

Richard Schwartz, Scott Miller, David Stallard, and John Makhoul. 1996. Language understanding using hidden understanding models. In *Spoken Language, 1996. ICSLP 96. Proceedings., Fourth International Conference on*, volume 2, pages 997–1000. IEEE.

Shai Shalev-Shwartz, Yoram Singer, and Nathan Srebro. 2007. Pegasos: Primal estimated sub-gradient solver for svm. In *Proceedings of the 24th International Conference on Machine Learning*, pages 807–814.

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

Wayne Ward. 1994. Extracting information in spontaneous speech. In *ICSLP*.

Jason Weston, Samy Bengio, and Nicolas Usunier. 2010. Large scale image annotation: Learning to rank with joint word-image embeddings. *Mach. Learn.*, 81.

Jason Weston, Samy Bengio, and Nicolas Usunier. 2011. Wsabie: Scaling up to large vocabulary image annotation. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Three*, pages 2764–2770.

Majid Yazdani and Andrei Popescu-Belis. 2013. Computing text semantic relatedness using the contents and links of a hypertext encyclopedia. *Artif. Intell.*, 194:176–202.

Modeling Tweet Arrival Times using Log-Gaussian Cox Processes

Michal Lukasik,[1] **P.K. Srijith,**[1] **Trevor Cohn,**[2] and **Kalina Bontcheva**[1]
[1]Department of Computer Science,
The University of Sheffield
[2]Department of Computing and Information Systems,
The University of Melbourne
{m.lukasik, pk.srijith, k.bontcheva}@shef.ac.uk
t.cohn@unimelb.edu.au

Abstract

Research on modeling time series text corpora has typically focused on predicting what text will come next, but less well studied is predicting when the next text event will occur. In this paper we address the latter case, framed as modeling continuous inter-arrival times under a log-Gaussian Cox process, a form of inhomogeneous Poisson process which captures the varying rate at which the tweets arrive over time. In an application to rumour modeling of tweets surrounding the 2014 Ferguson riots, we show how inter-arrival times between tweets can be accurately predicted, and that incorporating textual features further improves predictions.

1 Introduction

Twitter is a popular micro-blogging service which provides real-time information on events happening across the world. Evolution of events over time can be monitored there with applications to disaster management, journalism etc. For example, Twitter has been used to detect the occurrence of earthquakes in Japan through user posts (Sakaki et al., 2010). Modeling the temporal dynamics of tweets provides useful information about the evolution of events. Inter-arrival time prediction is a type of such modeling and has application in many settings featuring continuous time streaming text corpora, including journalism for event monitoring, real-time disaster monitoring and advertising on social media. For example, journalists track several rumours related to an event. Predicted arrival times of tweets can be applied for ranking rumours according to their activity and narrow the interest to investigate a rumour with a short inter-arrival time over that of a longer one.

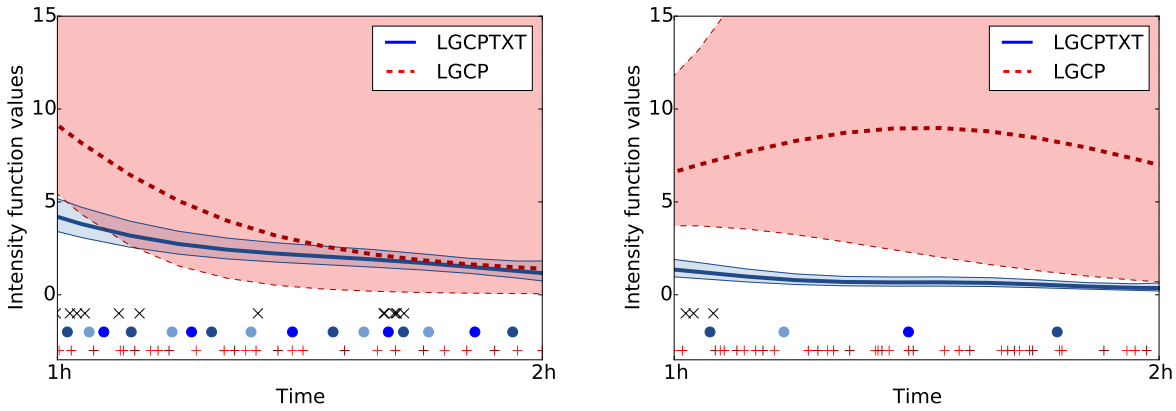
Modeling the inter-arrival time of tweets is a challenging task due to complex temporal patterns exhibited. Tweets associated with an event stream arrive at different rates at different points in time. For example, Figure 1a shows the arrival times (denoted by black crosses) of tweets associated with an example rumour around Ferguson riots in 2014. Notice the existence of regions of both high and low density of arrival times over a one hour interval. We propose to address inter-arrival time prediction problem with log-Gaussian Cox process (LGCP), an inhomogeneous Poisson process (IPP) which models tweets to be generated by an underlying intensity function which varies across time. Moreover, it assumes a non-parametric form for the intensity function allowing the model complexity to depend on the data set. We also provide an approach to consider textual content of tweets to model inter-arrival times. We evaluate the models using Twitter rumours from the 2014 Ferguson unrest, and demonstrate that they provide good predictions for inter-arrival times, beating the baselines e.g. homogeneous Poisson Process, Gaussian Process regression and univariate Hawkes Process. Even though the central application is rumours, one could apply the proposed approaches to model the arrival times of tweets corresponding to other types of memes, e.g. discussions about politics.

This paper makes the following contributions: 1. Introduces log-Gaussian Cox process to predict tweet arrival times. 2. Demonstrates how incorporating text improves results of inter-arrival time prediction.

2 Related Work

Previous approaches to modeling inter-arrival times of tweets (Perera et al., 2010; Sakaki et al., 2010; Esteban et al., 2012; Doerr et al., 2013) were not complex enough to consider their time varying characteristics. Perera et al. (2010) modeled

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 250–255,
Lisbon, Portugal, 17-21 September 2015.



(a) rumour #39 (b) rumour #60

Figure 1: Intensity functions and corresponding predicted arrival times for different methods across example Ferguson rumours. Arrival times predicted by LGCP are denoted by red pluses, LGCPTXT by blue dots, and ground truth by black crosses. Light regions denote uncertainty of predictions.

inter-arrival times as independent and exponentially distributed with a constant rate parameter. A similar model is used by Sakaki et al. (2010) to monitor the tweets related to earthquakes. The renewal process model used by Esteban et al. (2012) assumes the inter-arrival times to be independent and identically distributed. Gonzalez et al. (2014) attempts to model arrival times of tweets using a Gaussian process but assumes the tweet arrivals to be independent every hour. These approaches do not take into account the varying characteristics of arrival times of tweets.

Point processes such as Poisson and Hawkess process have been used for spatio-temporal modeling of meme spread in social networks (Yang and Zha, 2013; Simma and Jordan, 2010). Hawkes processes (Yang and Zha, 2013) were also found to be useful for modeling the underlying network structure. These models capture relevant network information in the underlying intensity function. We use a log-Gaussian cox process which provides a Bayesian method to capture relevant information through the prior. It has been found to be useful e.g. for conflict mapping (Zammit-Mangion et al., 2012) and for frequency prediction in Twitter (Lukasik et al., 2015).

3 Data & Problem

In this section we describe the data and we formalize the problem of modeling tweet arrival times.

Data We consider the Ferguson rumour data set (Zubiaga et al., 2015), consisting of tweets on rumours around 2014 Ferguson unrest. It consists of conversational threads that have been manually labeled by annotators to correspond to rumours[1]. Since some rumours have few posts, we consider only those with at least 15 posts in the first hour as they express interesting behaviour (Lukasik et al., 2015). This results in 114 rumours consisting of a total of 4098 tweets.

Problem Definition Let us consider a time interval $[0, 2]$ measured in hours, a set of rumours $R = \{E_i\}_{i=1}^{n}$, where rumour E_i consists of a set of m_i posts $E_i = \{p_j^i\}_{j=1}^{m_i}$. Posts are tuples $p_j^i = (\mathbf{x}_j^i, t_j^i)$, where $\mathbf{x}_j^i$ is text (in our case a vector of Brown clusters counts, see section 5) and t_j^i is time of occurrence of post p_j^i, measured in time since the first post on rumour E_i.

We introduce the problem of predicting the exact time of posts in the future unobserved time interval, which is studied as *inter-arrival time prediction*. In our setting, we observe posts over a target rumour i for one hour and over reference rumours (other than i) for two hours. Thus, the training data set is $R^O = \{E_i^O\}_{i=1}^{n}$, where $E_i^O = \{p_j^i\}_1^{m_i^O}$ (m_i^O represents number of posts observed for i^{th} rumour). We query the model for a complete set of times $\{t_j^i\}_{m_i^O+1}^{m_i}$ of posts about rumour i in the future one hour time interval.

[1]For a fully automated approach, a system for early detection of rumours (Zhao et al., 2015) could be run first and our models then applied to the resulting rumours.

4 Model

The problem of modeling the inter-arrival times of tweets can be solved using Poisson processes (Perera et al., 2010; Sakaki et al., 2010). A homogeneous Poisson process (HPP) assumes the intensity to be constant (with respect to time and the rumour statistics). It is not adequate to model the inter-arrival times of tweets because it assumes constant rate of point arrival across time. Inhomogeneous Poisson process (IPP) (Lee et al., 1991) can model tweets occurring at a variable rate by considering the intensity to be a function of time, *i.e.* $\lambda(t)$. For example, in Figure 1a we show intensity functions learnt for two different IPP models. Notice how the generated arrival times vary according to the intensity function values.

Log-Gaussian Cox process We consider a log-Gaussian Cox process (LGCP) (Møller and Syversveen, 1998), a special case of IPP, where the intensity function is assumed to be stochastic. The intensity function $\lambda(t)$ is modeled using a latent function $f(t)$ sampled from a Gaussian process (Rasmussen and Williams, 2005). To ensure positivity of the intensity function, we consider $\lambda(t) = \exp(f(t))$. This provides a non-parametric Bayesian approach to model the intensity function, where the complexity of the model is learnt from the training data. Moreover, we can define the functional form of the intensity function through appropriate GP priors.

Modeling inter-arrival time Inhomogeneous Poisson process (unlike HPP) uses a time varying intensity function and hence, the distribution of inter-arrival times is not independent and identically distributed (Ross, 2010). In IPP, the number of tweets y occurring in an interval $[s, e]$ is Poisson distributed with rate $\int_s^e \lambda(t)dt$.

$$p(y|\lambda(t),[s,e]) = \text{Poisson}(y|\int_s^e \lambda(t)dt) = \frac{(\int_s^e \lambda(t)dt)^y \exp(-\int_s^e \lambda(t)dt)}{y!} \quad (1)$$

Assume that n^{th} tweet occurred at time $E_n = s$ and we are interested in the inter-arrival time T_n of the next tweet. The arrival time of next tweet E_{n+1} can be obtained as $E_{n+1} = E_n + T_n$. The cumulative distribution for T_n, which provides the probability that a tweet occurs by time $s + u$ can be obtained as[2]

$$\begin{aligned} p(T_n \le u) &= 1 - p(T_n > u|\lambda(t), E_n = s) \\ &= 1 - p(0 \text{ events in } [s, s+u]|\lambda(t)) \\ &= 1 - \exp(-\int_s^{s+u} \lambda(t)dt) \\ &= 1 - \exp(-\int_0^u \lambda(s+t)dt) \end{aligned} \quad (2)$$

The derivation is obtained by considering a Poisson probability for 0 counts with rate parameter given by $\int_s^{s+u} \lambda(t)dt$ and applying integration by substitution to obtain (2). The probability density function of the random variable T_n is obtained by taking the derivative of (2) with respect to u:

$$p(T_n = u) = \lambda(s+u)\exp(-\int_0^u \lambda(s+t)dt). \quad (3)$$

The computational difficulties arising from integration are dealt by assuming the intensity function to be constant in an interval and approximating the inter-arrival time density as (Møller and Syversveen, 1998; Vanhatalo et al., 2013)

$$p(T_n = u) = \lambda(s+u)\exp(-u\lambda(s+\frac{u}{2})). \quad (4)$$

We associate a distinct intensity function $\lambda_i(t) = \exp(f_i(t))$ with each rumour E_i as they have varying temporal profiles. The latent function f_i is modelled to come from a zero mean Gaussian process (GP) (Rasmussen and Williams, 2005) prior with covariance defined by a squared exponential (SE) kernel over time, $k_{time}(t,t') = a\exp(-(t-t')^2/l)$. We consider the likelihood of posts E_i^{O} over the entire training period to be product of Poisson distribution (1) over equal length sub-intervals with the rate in a sub-interval $[s, e]$ approximated as $(e-s)\exp(f_i(\frac{1}{2}(s+e)))$. The likelihood of posts in the rumour data is obtained by taking the product of the likelihoods over individual rumours.

The distribution of the posterior $p(f_i|E_i^{\text{O}})$ is intractable and a Laplace approximation (Rasmussen and Williams, 2005) is used to obtain the posterior. The predictive distribution $f_i(t_*^i)$ at time t_*^i is obtained using the approximated posterior. The intensity function value at the point t_*^i is then obtained as

$$\lambda_i(t_*^i|E_i^{\text{O}}) = \int \exp\left(f_i(t_*^i)\right) p\left(f_i(t_*^i)|E_i^{\text{O}}\right) df_i(t_*^i).$$

[2] We suppress the conditioning variables for brevity.

Algorithm 1 Importance sampling for predicting the next arrival time

1: **Input:** Intensity function $\lambda(t)$, previous arrival time s, proposal distribution $q(t) = \exp(t; 2)$, number of samples N
2: **for** $i = 1$ **to** N **do**
3: Sample $u_i \sim q(t)$.
4: Obtain weights $w_i = \frac{p(u_i)}{q(u_i)}$, where $p(t)$ is given by (4).
5: **end for**
6: Predict expected inter-arrival time as $\bar{u} = \sum_{i=1}^{N} u_i \frac{w_i}{\sum_{j=1}^{N} w_j}$
7: Predict the next arrival time as $\bar{t} = s + \bar{u}$.
8: **Return:** $\bar{t}$

Importance sampling We are interested in predicting the next arrival time of a tweet given the time at which the previous tweet was posted. This is achieved by sampling the inter-arrival time of occurrence of the next tweet using equation (4). We use the importance sampling scheme (Gelman et al., 2003) where an exponential distribution is used as the proposal density. We set the rate parameter of this exponential distribution to 2 which generates points with a mean value around 0.5. Assuming the previous tweet occurred at time s, we obtain the arrival time of next tweet as outlined in Algorithm 1. We run this algorithm sequentially, i.e. the time $\bar{t}$ returned from Algorithm 1 becomes starting time s in the next iteration. We stop at the end of the interval of interest, for which a user wants to find times of post occurrences.

Incorporating text We consider adding the kernel over text from posts to the previously introduced kernel over time. We join text from the observed posts together, so a different component is added to kernel values across different rumours. The full kernel then takes form $k_{\text{TXT}}((t,i),(t',i')) = k_{time}(t,t') + k_{text}\big(\sum_{p_j^i \in E_i^O} \mathbf{x}_j^i, \sum_{p_j^{i'} \in E_{i'}^O} \mathbf{x}_j^{i'}\big)$. We compare text via linear kernel with additive underlying base similarity, expressed by $k_{text}(\mathbf{x}, \mathbf{x}') = b + c\mathbf{x}^T\mathbf{x}'$.

Optimization All model parameters (a, l, b, c) are obtained by maximizing the marginal likelihood $p(E_i^O) = \int p(E_i^O|f_i)p(f_i)df_i$ over all rumour data sets.

5 Experiments

Data preprocessing In our experiments, we consider the first two hours of each rumour lifespan. The posts from the first hour of a target rumour is considered as observed (training data) and we predict the arrival times of tweets in the second hour. We consider observations over equal sized time intervals of length six minutes in the rumour lifespan for learning the intensity function. The text in the tweets is represented by using Brown cluster ids associated with the words. This is obtained using 1000 clusters acquired on a large scale Twitter corpus (Owoputi et al., 2013).

Evaluation metrics Let the arrival times predicted by a model be $(\hat{t}_1, \ldots, \hat{t}_M)$ and let the actual arrival times be $(t_1, \ldots, t_N)$. We introduce two metrics based on root mean squared error (RMSE) for evaluating predicted inter-arrival times. First is aligned root mean squared error (ARMSE), where we align the initial $K = \min(M, N)$ arrival times and calculate the RMSE between such two subsequences. The second is called penalized root mean squared error (PRMSE). In this metric we penalize approaches which predict a different number of inter-arrival times than the actual number. The PRMSE metric is defined as the square root of the following expression.

$$\frac{1}{K}\sum_{i=1}^{K}(\hat{t}_i - t_i)^2 + \mathbb{I}[M > N] \sum_{i=N+1}^{M} (T - \hat{t}_i)^2 + \mathbb{I}[M < N] \sum_{i=M+1}^{N} (T - t_i)^2 \quad (5)$$

The second and third term in (5) respectively penalize for the excessive or insufficient number of points predicted by the model.

Baselines We consider a homogeneous Poisson process (HPP) (Perera et al., 2010) as a baseline which results in exponentially distributed inter-arrival times with rate λ. The rate parameter is set to the maximum likelihood estimate, the reciprocal of the mean of the inter-arrival times in the training data. The second baseline is a GP with a linear kernel (GPLIN), where the inter-arrival time is modeled as a function of time of occurrence of last tweet. This model tends to predict small inter-arrival times yielding a huge number of points. We limit the number of predicted points

method	ARMSE	PRMSE
GPLIN	20.60±22.01⋆	1279.78±903.90⋆
HPP	21.85±22.82⋆	431.4±96.5⋆
HP	15.94±18.20	363.70±59.01⋆
LGCP	13.31±14.28	261.26±92.97⋆
LGCP Pooled	19.18±20.36⋆	183.25±102.20⋆
LGCPTXT	15.52±18.79	154.05±115.70

Table 1: ARMSE and PRMSE between the true event times and the predicted event times expressed in minutes (lower is better) over the 114 Ferguson rumours, showing mean ± std. dev. Key ⋆ denotes significantly worse than LGCPTXT method according to one-sided Wilcoxon signed rank test ($p < 0.05$). In case of ARMSE, LGCP is not significantly better than LGCP TXT according to Wilcoxon test.

to 1000 (above the maximum count yielded by any rumour from our dataset), thus reducing the error from this method.

We also compare against Hawkes Process (HP) (Yang and Zha, 2013), a self exciting point process where an occurrence of a tweet increases the probability of tweets arriving soon afterwards. We consider a univariate Hawkes process where the intensity function is modeled as $\lambda_i(t) = \mu + \sum_{t^i_j < t} k_{time}(t^i_j, t)$. The kernel parameters and μ are learnt by maximizing the likelihood. We apply the importance sampling algorithm discussed in Algorithm 1 for generating arrival times for Hawkess process. We consider this baseline only in the single-task setting, where reference rumours are not considered.

LGCP settings In the case of LGCP, the model parameters of the intensity function associated with a rumour are learnt from the observed inter-arrival times from that rumour alone. LGCP Pooled and LGCPTXT consider a different setting where this is learnt additionally using the inter-arrival times of all other rumours observed over the entire two hour life-span.

Results Table 1 reports the results of predicting arrival times of tweets in the second hour of the rumour lifecycle. In terms of ARMSE, LGCP is the best method, performing better than LGCP-TXT (though not statistically significantly) and outperforming other approaches. However, this metric does not penalize for the wrong number of predicted arrival times. Figure 1b depicts an example rumour, where LGCP greatly overestimates the number of points in the interval of interest. Here, the three points from the ground truth (denoted by black crosses) and the initial three points predicted by the LGCP model (denoted by red pluses), happen to lie very close, yielding a low ARMSE error. However, LGCP predicts a large number of arrivals in this interval making it a bad model compared to LGCPTXT which predicts only four points (denoted by blue dots). ARMSE fails to capture this and hence we use PRMSE. Note that Hawkes Process is performing worse than the LGCP approach.

According to PRMSE, LGCPTXT is the most successful method, significantly outperforming all other according to Wilcoxon signed rank test. Figure 1a depicts the behavior of LGCP and LGCP-TXT on rumour 39 with a larger number of points from the ground truth. Here, LGCPTXT predicts relatively less number of arrivals than LGCP. The performance of Hawkes Process is again worse than the LGCP approach. The self excitory nature of Hawkes process may not be appropriate for this dataset and setting, where in the second hour the number of points tends to decrease as time passes.

We also note, that GPLIN performs very poorly according to PRMSE. This is because the inter-arrival times predicted by GPLIN for several rumours become smaller as time grows resulting in a large number of arrival times.

6 Conclusions

This paper introduced the log-Gaussian Cox processes for the problem of predicting the inter-arrival times of tweets. We showed how text from posts helps to achieve significant improvements. Evaluation on a set of rumours from Ferguson riots showed efficacy of our methods comparing to baselines. The proposed approaches are generalizable to problems other than rumours, e.g. disaster management and advertisement campaigns.

Acknowledgments

Work partially supported by the European Union under grant agreement No. 611233 PHEME.

References

Christian Doerr, Norbert Blenn, and Piet Van Mieghem. 2013. Lognormal infection times of online information spread. *PLOS ONE*, 8.

J. Esteban, A. Ortega, S. McPherson, and M. Sathiamoorthy. 2012. Analysis of Twitter Traffic based on Renewal Densities. *ArXiv e-prints*.

Andrew Gelman, John B. Carlin, Hal S. Stern, and Donald B. Rubin. 2003. *Bayesian Data Analysis*. Chapman and Hall/CRC.

Roberto Gonzalez, Alfonso Muñoz, José Alberto Hernández, and Ruben Cuevas. 2014. On the tweet arrival process at twitter: analysis and applications. *Trans. Emerging Telecommunications Technologies*, 25(2):273–282.

S. H. Lee, M. M. Crawford, and J. R. Wilson. 1991. Modeling and simulation of a nonhomogeneous poisson process having cyclic behavior. *Communications in Statistics Simulation, 20(2):777–809*.

Michal Lukasik, Trevor Cohn, and Kalina Bontcheva. 2015. Point process modelling of rumour dynamics in social media. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015*, pages 518–523.

Jesper Møller and Anne Randi Syversveen. 1998. Log Gaussian Cox processes. *Scandinavian Journal of Statistics*, pages 451–482.

Olutobi Owoputi, Chris Dyer, Kevin Gimpel, Nathan Schneider, and Noah A. Smith. 2013. Improved part-of-speech tagging for online conversational text with word clusters. In *In Proceedings of NAACL*.

Rohan DW Perera, Sruthy Anand, KP Subbalakshmi, and R Chandramouli. 2010. Twitter analytics: Architecture, tools and analysis. In *Military Communications Conference, 2010-MILCOM 2010*, pages 2186–2191.

Carl Edward Rasmussen and Christopher K. I. Williams. 2005. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press.

Sheldon M. Ross. 2010. *Introduction to Probability Models, Tenth Edition*. Academic Press, Inc., Orlando, FL, USA.

Takeshi Sakaki, Makoto Okazaki, and Yutaka Matsuo. 2010. Earthquake shakes twitter users: Real-time event detection by social sensors. In *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, pages 851–860.

Aleksandr Simma and Michael I. Jordan. 2010. Modeling events with cascades of poisson processes. In *UAI*, pages 546–555.

Jarno Vanhatalo, Jaakko Riihimäki, Jouni Hartikainen, Pasi Jylänki, Ville Tolvanen, and Aki Vehtari. 2013. Gpstuff: Bayesian modeling with gaussian processes. *J. Mach. Learn. Res.*, 14(1):1175–1179.

Shuang-Hong Yang and Hongyuan Zha. 2013. Mixture of mutually exciting processes for viral diffusion. In *ICML (2)*, volume 28 of *JMLR Proceedings*, pages 1–9.

Andrew Zammit-Mangion, Michael Dewar, Visakan Kadirkamanathan, and Guido Sanguinetti. 2012. Point process modelling of the afghan war diary. In *Proceedings of the National Academy of Sciences*, Vol. 109, No. 31, pages 12414–12419.

Zhe Zhao, Paul Resnick, and Qiaozhu Mei. 2015. Early detection of rumors in social media from enquiry posts. In *International World Wide Web Conference Committee (IW3C2)*.

Arkaitz Zubiaga, Maria Liakata, Rob Procter, Kalina Bontcheva, and Peter Tolmie. 2015. Towards detecting rumours in social media. In *AAAI Workshop on AI for Cities*.

Pre-Computable Multi-Layer Neural Network Language Models

Jacob Devlin
Microsoft Research
Redmond, WA, USA
jdevlin@microsoft.com

Chris Quirk
Microsoft Research
Redmond, WA, USA
chrisq@microsoft.com

Arul Menezes
Microsoft Research
Redmond, WA, USA
arulm@microsoft.com

Abstract

In the last several years, neural network models have significantly improved accuracy in a number of NLP tasks. However, one serious drawback that has impeded their adoption in production systems is the slow runtime speed of neural network models compared to alternate models, such as maximum entropy classifiers. In Devlin et al. (2014), the authors presented a simple technique for speeding up feed-forward embedding-based neural network models, where the dot product between each word embedding and part of the first hidden layer are *pre-computed* offline. However, this technique cannot be used for hidden layers beyond the first. In this paper, we explore a neural network architecture where the embedding layer feeds into multiple hidden layers that are placed "next to" one another so that each can be pre-computed independently. On a large scale language modeling task, this architecture achieves a 10x speedup at runtime and a significant reduction in perplexity when compared to a standard multi-layer network.

1 Introduction

Neural network models have become extremely popular in the last several years for a wide variety of NLP tasks, including language modeling (Schwenk, 2007), sentiment analysis (Socher et al., 2013), translation modeling (Devlin et al., 2014), and many others (Collobert et al., 2011). However, a serious drawback of neural network models is their slow speeds in training and test time (runtime) relative to alternative models such as maximum entropy (Berger et al., 1996) or back-off models (Kneser and Ney, 1995).

One popular application of neural network models in NLP is using neural network language models (NNLMs) as an additional feature in an existing machine translation (MT) or automatic speech recognition (ASR) engines. NNLMs are particularly costly in this scenario, since decoding a single sentence typically requires tens of thousands or more n-gram lookups. Although we will focus on this particular scenario in this paper, it is important to note that the techniques presented generalize to any feed-forward embedding-based neural network model.

One popular technique for improving the runtime speed of NNLMs involves training the network to be "approximately normalized," so that the softmax normalizer does not have to be computed after training. Two algorithms have been proposed to achieve this: (1) noise-contrastive estimation (NCE) (Mnih and Teh, 2012; Vaswani et al., 2013) and (2) explicit self-normalization (Devlin et al., 2014), which is used in this paper.

However, even with self-normalized networks, computing the output of an intermediate hidden layer still requires a costly matrix-vector multiplication. To mitigate this, Devlin et al. (2014) made the observation that for 1-layer NNLMs, the dot product between each embedding+position pair and the first hidden layer can be *pre-computed* after training is complete, which allows the matrix-vector multiplication to be replaced by a handful of vector additions. Using these two techniques in combination improves the runtime speed of NNLMs by several orders of magnitude with no degradation to accuracy.

To understand pre-computation, first assume that we are training a NNLM that uses 250-dimensional word embeddings, a four word context window, and a 500-dimensional hidden layer. The weight matrix for the first hidden layer is thus 1000×500. For each word in the vocabulary and each of the four positions in the context vector, we

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 256–260,
Lisbon, Portugal, 17-21 September 2015.

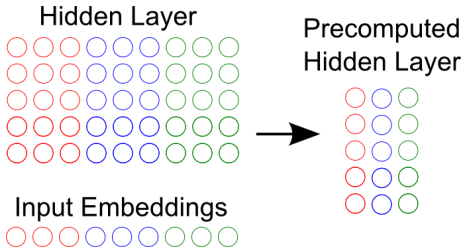


Figure 1: The "pre-computation trick." The dot product between each word embedding and each section of the hidden layer can be computed offline.

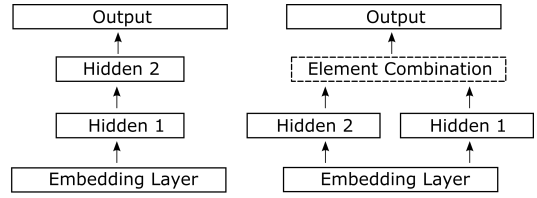


Figure 2: Network architectures.

can pre-compute the dot product between the 250-dimensional word embedding and the 250×500 section of the hidden layer. This results in four 500-dimensional vectors for each word that can be stored in a lookup table. At test time, we can simply sum four vectors to obtain the output of the first hidden layer. This is shown visually in Figure 1. Note that this is *not* an approximation, and the resulting output vector is identical to the original matrix-vector product. However, the major limitation of the "pre-computation trick" is that it only works with 1-hidden layer architectures, even though more accurate models can nearly always be obtained by training multi-layer networks.

In this paper, we explore a network architecture where multiple hidden layers are placed "next to" one another instead of "on top of" one another, as is usually done. The output of these *lateral* layers are combined using an inexpensive element-wise function and fed into the output layer. Crucially, then, we can apply the pre-computation trick to *each* hidden layer independently, allowing for very powerful models that are orders of magnitude faster at runtime than a standard multi-layer network.

Mathematically, this can be thought of as a generalization of maxout networks (Goodfellow et al., 2013), where different element-wise combination functions are explored rather than just the max function.

2 Lateral Network

In a standard feed-forward embedding-based neural network, the input tokens are mapped into a continuous vector using an embedding table[1], and this embedding vector is fed into the first hidden layer. The output of each hidden layer is then fed into the next hidden layer. We refer to this as the *stacked* architecture. For a two layer network, we can represent the output of the final hidden layer as:

$$H = \phi(W_2\phi(W_1E(x)))$$

where x is the input vector, $E(x)$ is the output of the embedding layer, W_i is the weight matrix for layer i, and ϕ is the transfer function such as $tanh$. Generally, H is then multiplied by an output matrix and a softmax is performed to obtain the output probabilities.

In the lateral network architecture, the embedding layer is fed into two or more "side-by-side" hidden layers, and the outputs of these hidden layers are combined using an element-wise function such as maximum or multiplication. This is represented as:

$$H = C(\phi(W_1E(x)), \phi(W_2E(x)))$$

Where C is a combination function that takes two or more k-dimensional vectors as inputs and produces as k-dimensional vector as output. If $C(h_1, h_2) = max(h_1, h_2)$ then this is equivalent to a maxout network (Goodfellow et al., 2013). To generalize this, we explore three different combination functions: [2]

$$C_{max}(h_1, h_2) = max(h_1, h_2)$$
$$C_{mul}(h_1, h_2) = h_1 * (h_2 + 1)$$
$$C_{add}(h_1, h_2) = h_1 + h_2$$

The three-or-more hidden layer versions are constructed as expected.[3]

A visualization is given in Figure 2. Crucially, for the lateral architecture, each hidden layer can be pre-computed independently, allowing for very fast n-gram probability lookups at runtime.

[1]The embeddings may or may not be trained jointly with the rest of the model.

[2]Note that C is an element-wise function, so these represent the operation on a single dimension of the input vectors.

[3]The + 1 in C_{mul} is used for all hidden layers after the first. This is used to prevent the value from being very close to 0.

3 Language Modeling Results

In this section we report results on a large scale language modeling task.

3.1 Data

Our LM training corpus consists of 120M words from the New York Times portion of the English GigaWord data set. This was chosen instead of the commonly used 1M word Penn Tree Bank corpus in order to better represent real world LM training scenarios. We use all data from 2008 and 2009 as training, the first 100k words from June 2010 as validation, and the first 100k words from December 2010 as test. The data is segmented and tokenized using the Stanford Word Segmenter with default settings.

3.2 Neural Network Training

Training was performed with an in-house toolkit using stochastic gradient descent. The vocabulary is limited to 16k words so that the output layer can be trained using a basic softmax with self-normalization. All experiments use 250-dimensional word embeddings and a $tanh$ activation function. The weights were initialized in the range [-0.05, 0.05], the batch size was 256, and the initial learning rate was 0.25.

3.3 5-gram LM Perplexity

5-gram results are shown in Table 1. The 1-layer NNLM achieves a 13.2 perplexity improvement over the Kneser-Ney smoothed baseline (Kneser and Ney, 1995). Consistent with Schwenk et al. (2014), using additional hidden layers to the stacked (standard) network results in 2.0-3.0 perplexity improvements on top of the 1-layer model.

The lateral architecture significantly outperforms any of the stacked networks, achieving a 6.5 perplexity reduction over the 1-layer model. The multiplicative combination function performs better than the additive and max functions by a small margin, which suggests that it better allows for modeling complex relationships between input words.

Perhaps most surprisingly, the additive function performs as well as the max function, despite the fact that it provides no additional modeling power compared to a 1-layer network. However, it does allow the model to *generalize* better than a 1-layer network by explicitly tying together two or three hidden nodes from each node in the output layer.

Condition	PPL
5-gram KNLM	91.1
1-Layer (k=500)	77.9
1-Layer (k=1000)	77.7
2-Stacked (k=500)	76.3
2-Stacked (k=1000)	76.2
3-Stacked (k=1000)	74.8
2-Lateral Max (k=500)	73.8
2-Lateral Mul ...	72.7
2-Lateral Add	73.7
3-Lateral Max	73.1
3-Lateral Mul	71.1
3-Lateral Add	72.3

Table 1: Perplexity of 5-gram models on the New York Times test set. k is the size of the hidden layer(s).

3.4 Runtime Speed

The runtime speed of the various models is shown in Table 2. These are computed on a single core of a E5-2650 2.6 GHz CPU. Consistent with Devlin et al. (2014), we see that the baseline model achieves only 230 n-gram lookups per second (LPS) at test time, while the pre-computed, self-normalized 1-layer network achieves 600,000 LPS. Adding a second stacked layer slows this down to 24,000 LPS due to the 500×500 matrix-vector multiplication that must be performed. However, the lateral configuration achieves 305,000 LPS while obtaining a better perplexity than the stacked network. In comparison, the fastest backoff LM implementation, KenLM (Heafield, 2011), achieves 1-2 million lookups per second.

In terms of memory usage, it is difficult to fairly compare backoff LMs and NNLMs because neural networks scale linearly with the vocabulary size, while backoff LMs scale linearly with the number of unique n-grams. In this case, the non-precomputed neural network model is 25 MB, and the pre-computed 2-lateral network is 136 MB.[4] The KenLM models are 1.1 GB for the Probing model and 317 MB for the Trie model. With a vocabulary of 50k, the 2-lateral network would be 425MB. In general, a pre-computed NNLM is comparable to or smaller than an equivalent backoff LM in terms of model size.

[4]The floats can be quantized to 2 bytes after training without loss of accuracy.

Condition	Lookups Per Sec.
KenLM Probing	1,923,000
KenLM Trie	950,000
1-Layer (No PC, No SN)	230
1-Layer (No PC)	13,000
1-Layer	600,000
2-Stacked	24,000
2-Stacked (Batch=128)	58,000
2-Lateral Mul	305,000

Table 2: Runtime speed of the 5-gram LM on a single CPU core. "PC" = pre-computation, "SN" = self-normalization, which are used in all but the first two experiments. The batch size is 1 except when specified. 500-dimensional hidden layers are used in all cases. "Float Ops." is the approximate number of floating point operations per lookup.

3.5 High-Order LM Perplexity

We also report results on a 10-gram LM trained on the same data, to explore whether the lateral network can achieve an even higher relative gain when a large input context window is available. Results are shown in Table 3. Although there is a large absolute improvement over the 5-gram LM, the relative improvement between the 1-layer, 3-stacked, and 3-lateral systems are similar to the 5-gram scenario.

Condition	PPL
1-Layer (k=500)	69.8
3-Stacked (k=1000)	65.8
3-Lateral Mul (k=500)	63.4
Gated Recurrent (k=1000)	55.4

Table 3: Perplexity of 10-gram models on the New York Times test set. The Gated Recurrent model uses the full word history.

As another point of comparison we report results with an gated recurrent network (Cho et al., 2014). As is consistent with the literature, the recurrent network significantly outperforms any of the feed-forward models (Sundermeyer et al., 2013).

However, recurrent models have two major downsides. First, they cannot easily be integrated into existing MT/ASR engines without significantly altering the search algorithm and search space, since they require a fully expanded target context. Second, the matrix-vector product between the previous hidden state and the hidden weight matrix cannot be pre-computed, which makes the models significantly slower than pre-computable feed-forward networks.

Condition	Test BLEU	Test PPL
Baseline	37.95	-
+NNLM 1-Layer	38.89	138.3
+NNLM 2-Stacked	39.13	136.2
+NNLM 2-Lateral	39.15	132.3
+NNJM 1-Layer	40.71	6.33
+NNJM 2-Stacked	40.82	6.25
+NNJM 2-Lateral	40.89	6.13

Table 4: Results on English-German machine translation test set.

4 Machine Translation Results

Although the lateral networks achieve a significant reduction in LM perplexity over the 1-layer network, it is not clear how much this will improved performance in a downstream task. To evaluate this, we trained two neural network models for use as additional features in a machine translation (MT) system.

The first feature is a 5-gram NNLM, which used 1000 dimensions for the stacked network and 500 for the lateral network. The second feature is a neural network joint model (NNJM), which predicts each target word using 5-gram target context *and* 7-gram source context. For evaluation, we present both the model perplexity and the BLEU score when using the model as an additional MT feature.

Results are presented on a large scale English-German speech translation task. The parallel training data consists of 600M words from a variety of sources, including OPUS (Tiedemann, 2012) and a large in-house web crawl. The baseline 4-gram Kneser-Ney smoothed LM is trained on 7B words of German data. The NNLM and NNTMs are trained only on the parallel data. Our MT decoder is a proprietary engine similar to Moses (Koehn et al., 2007). The tuning set consists of 4000 utterances from conversational and newswire data, and the test set consists of 1500 sentences of collected conversational data.

Results are show in Table 4. We can see that perplexity improvements are similar to what is

seen in the English NYT data, and that improvements in BLEU over a 1-layer model are small but consistent. There is not a significant difference in BLEU between the 2-Stacked and 2-Lateral configuration.

5 Conclusion

In this paper, we explored an alternate architecture for embedding-based neural network models which allows for a fully pre-computable network with multiple hidden layers. The resulting models achieve better perplexity than a standard multilayer network *and* is at least an order of magnitude faster at runtime.

In future work, we can assess the impact of this model on a wider array of feed-forward embedding-based neural network models, such as the DSSM (Huang et al., 2013).

References

Adam L Berger, Vincent J Della Pietra, and Stephen A Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational linguistics*.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*.

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

Jacob Devlin, Rabih Zbib, Zhongqiang Huang, Thomas Lamar, Richard Schwartz, and John Makhoul. 2014. Fast and robust neural network joint models for statistical machine translation. In *ACL*.

Ian J Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. 2013. Maxout networks. *arXiv preprint arXiv:1302.4389*.

Kenneth Heafield. 2011. Kenlm: Faster and smaller language model queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*.

Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*. ACM.

Reinhard Kneser and Hermann Ney. 1995. Improved backing-off for m-gram language modeling. In *Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proc. Assoc. for Computational Linguistics (ACL)*.

Andriy Mnih and Yee Whye Teh. 2012. A fast and simple algorithm for training neural probabilistic language models. *In Proceedings of the International Conference on Machine Learning*.

Holger Schwenk, Fethi Bougares, and Loıc Barrault. 2014. Efficient training strategies for deep neural network language models. In *NIPS Workshop on Deep Learning and Representation Learning*.

Holger Schwenk. 2007. Continuous space language models. *Computer Speech & Language*.

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

Martin Sundermeyer, Ilya Oparin, Jean-Luc Gauvain, Ben Freiberg, Ralf Schluter, and Hermann Ney. 2013. Comparison of feedforward and recurrent neural network language models. In *Proc. Conf. Acoustics, Speech and Signal Process. (ICASSP)*.

Jörg Tiedemann. 2012. Parallel data, tools and interfaces in opus. In *LREC*.

Ashish Vaswani, Yinggong Zhao, Victoria Fossum, and David Chiang. 2013. Decoding with large-scale neural language models improves translation. In *EMNLP*.

Birds of a Feather Linked Together: A Discriminative Topic Model using Link-based Priors

Weiwei Yang
Computer Science
University of Maryland
College Park, MD
wwyang@cs.umd.edu

Jordan Boyd-Graber
Computer Science
University of Colorado
Boulder, CO
Jordan.Boyd.Graber@colorado.edu

Philip Resnik
Linguistics and UMIACS
University of Maryland
College Park, MD
resnik@umd.edu

Abstract

A wide range of applications, from social media to scientific literature analysis, involve graphs in which documents are connected by links. We introduce a topic model for link prediction based on the intuition that linked documents will tend to have similar topic distributions, integrating a max-margin learning criterion and lexical term weights in the loss function. We validate our approach on the tweets from 2,000 Sina Weibo users and evaluate our model's reconstruction of the social network.

1 Introduction

Many application areas for text analysis involve documents connected by links of one or more types—for example, analysis of scientific papers (citations, co-authorship), Web pages (hyperlinks), legislation (co-sponsorship, citations), and social media (followers, mentions, etc.). In this paper we work within the widely used framework of topic modeling (Blei et al., 2003, LDA) to develop a model that is simple and intuitive, but which identifies high quality topics while also accurately predicting link structure.

Our work here is inspired by the phenomenon of *homophily*, the tendency of people to associate with others who are like themselves (McPherson et al., 2001). As manifested in social networks, the intuition is that people who are associated with one another are likely to discuss similar topics, and vice versa. The new topic model we propose therefore takes association links into account so that a document's topic distribution is influenced by the topic distributions of its neighbors. Specifically, we propose a joint model that uses link structure to define clusters (cliques) of documents and, following the intuition that documents in the same cluster are likely to have similar topic distributions, assigns each cluster its own separate Dirichlet prior over the cluster's topic distribution. This use of priors is consistent with previous work that has shown document-topic priors to be useful in encoding various types of prior knowledge and improving topic modeling performance (Mimno and McCallum, 2008). We then use distributed representations to "seed" the topic representations before getting down to modeling the documents. Our joint objective function uses a discriminative, max-margin approach (Zhu et al., 2012; Zhu et al., 2014) to both model the contents of documents and produce good predictions of links; in addition, it improves prediction by including lexical terms in the decision function (Nguyen et al., 2013).

Our baseline for comparison is the Relational Topic Model (Chang and Blei, 2010, henceforth RTM), which jointly captures topics and binary link indicators in a style similar to supervised LDA (McAuliffe and Blei, 2008, sLDA), instead of modeling links alone, e.g., as in the Latent Multi-group Membership Graph model (Kim and Leskovec, 2012, LMMG). We also compare our approach with Daumé III (2009), who uses document links to create a Markov random topic field (MRTF). Daumé does not, however, look at link prediction, as his upstream model (Mimno and McCallum, 2008) only generates documents *conditioned on* links. In contrast, our downstream model allows the prediction of links, like RTM.

Our model's primary contribution is in its novel combination of a straightforward joint modeling approach, max-margin learning, and exploitation of lexical information in both topic seeding and regression, yielding a simple but effective model for topic-informed discriminative link prediction. Like other topic models which treat binary values "probabilistically", our model can convert binary link indicators into non-zero weights, with potential application to improving models like Volkova

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 261–266,
Lisbon, Portugal, 17-21 September 2015.

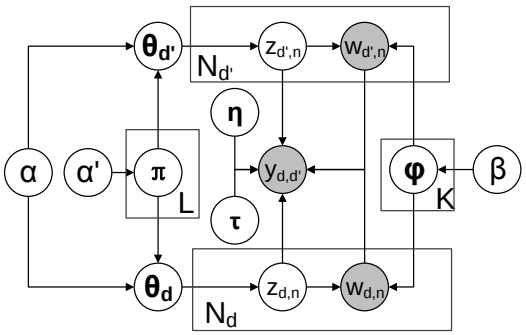


Figure 1: A graphical model of our model for two documents. The contribution of our model is the use of document clusters (π), the use of words (w) in the prediction of document links (y), and a max-margin objective.

et al. (2014), who use neighbor relationships to improve prediction of user-level attributes.

Our corpus is collected from Sina Weibo with three types of links between documents. We first conduct a reality check of our model against LDA and MRTF and then perform link prediction tasks. We demonstrate improvements in link prediction as measured by predictive link rank and provide both qualitative and quantitative perspectives on the improvements achieved by the model.

2 Discriminative Links from Topics

Figure 1 is a two-document segment of our model, which has the following generative process:

1. For each related-document cluster $l \in \{1, \ldots, L\}$
   Draw $\boldsymbol{\pi_l} \sim \mathrm{Dir}(\alpha')$
2. For each topic $k \in \{1, \ldots, K\}$
   (a) Draw word distribution $\boldsymbol{\phi_k} \sim \mathrm{Dir}(\beta)$
   (b) Draw topic regression parameter $\eta_k \sim \mathcal{N}(0, \nu^2)$
3. For each word $v \in \{1, \ldots, V\}$
   Draw lexical regression parameter $\tau_v \sim \mathcal{N}(0, \nu^2)$
4. For each document $d \in \{1, \ldots, D\}$
   (a) Draw topic proportions $\boldsymbol{\theta_d} \sim \mathrm{Dir}(\alpha \boldsymbol{\pi_{l_d}})$
   (b) For each word $t_{d,n}$ in document d
      i. Draw a topic assignment $z_{d,n} \sim \mathrm{Mult}(\boldsymbol{\theta_d})$
      ii. Draw a word $t_{d,n} \sim \mathrm{Mult}(\boldsymbol{\phi_{z_{d,n}}})$
5. For each linked pair of documents d and d'
   Draw binary link indicator
   $y_{d,d'} | \boldsymbol{z}_d, \boldsymbol{z}_{d'}, \boldsymbol{w}_d, \boldsymbol{w}_{d'} \sim \Psi(\cdot | \boldsymbol{z}_d, \boldsymbol{z}_{d'}, \boldsymbol{w}_d, \boldsymbol{w}_{d'}, \boldsymbol{\eta}, \boldsymbol{\tau})$

Step 1: Identifying birds of a feather. Prior to the generative process, given a training set of documents and document-to-document links, we begin by identifying small clusters or cliques using strongly connected components, which automatically determines the number of clusters from the link graph. Intuitively, documents in the same clique are likely to have similar topic distributions. Therefore, each of the L cliques l (the "birds of a feather" of our title) is assigned a separate Dirichlet prior $\boldsymbol{\pi_l}$ over K topics.

Step 2a: Using seed words to improve topic quality. To improve topic quality, we identify *seed words* for the K topics using distributed lexical representations: the key idea is to complement the more global information captured in LDA-style topics with representations based on local contextual information. We cluster the most frequent words' word2vec representations (Mikolov et al., 2013) into K word-clusters using the k-means algorithm, based on the training corpus.[1] We then enforce a one-to-one association between these discovered word clusters and the K topics. For any word token $w_{d,n}$ whose word type is in cluster k, the associated topic assignment $z_{d,n}$ can only be k. To choose topic k's seed words, within its word-cluster we compute each word $w_{k,i}$'s skip-gram transition probability sum $S_{k,i}$ to the other words as

$$S_{k,i} = \sum_{j=1, j \neq i}^{N_k} p(w_{k,j} \,|\, w_{k,i}), \qquad (1)$$

where N_k denotes the number of words in topic k.

We then select the three words with the highest sum of transition probabilities as the seed words for topic k. In the sampling process (Section 3), seed words are only assigned to their corresponding topics, similar to the use of hard constraints by Andrzejewski and Zhu (2009).

Steps 2b-3: Link regression parameters. Given two documents d and d', we want to predict whether they are linked by taking advantage of their topic patterns: the more similar two documents are, the more likely it is that they should be linked together. Like RTM, we will compute a regression in Step 5 using the topic distributions of d and d'; however, we follow Nguyen et al. (2013) by also including a document's word-level distribution as a regression input.[2] The regression value of document d and d' is

$$R_{d,d'} = \boldsymbol{\eta}^{\mathrm{T}}(\overline{\boldsymbol{z}}_d \circ \overline{\boldsymbol{z}}_{d'}) + \boldsymbol{\tau}^{\mathrm{T}}(\overline{\boldsymbol{w}}_d \circ \overline{\boldsymbol{w}}_{d'}), \qquad (2)$$

where $\overline{\boldsymbol{z}}_d = \frac{1}{N_d} \sum_n z_{d,n}$, and $\overline{\boldsymbol{w}}_d = \frac{1}{N_d} \sum_n w_{d,n}$; $\circ$ denotes the Hadamard product; $\boldsymbol{\eta}$ and $\boldsymbol{\tau}$ are the

[1] In the experiment, seed words must appear at least 1,000 times.

[2] Both approaches contrast with the links-only approach of Kim and Leskovec (2012).

weight vectors for topic-based and lexically-based predictions, respectively.

Step 4: Generating documents. Documents are generated as in LDA, where each document's topic distribution $\boldsymbol{\theta}$ is drawn from the cluster's topic prior (a parametric analog to the HDP of Teh et al. (2006)) and each word's topic assignment is drawn from the document's topic distribution (except for seed words, as described above).

Step 5: Generating links. Our model is a "downstream" supervised topic model, i.e., the prediction of the observable variable (here, document links) is informed by the documents' topic distributions, as in sLDA (Blei and McAuliffe, 2007). In contrast to Chang and Blei (2010), who use a sigmoid as their link prediction function Ψ, we instead use hinge loss: the probability Ψ that two documents d and d' are linked is

$$p(y_{d,d'} = 1 \mid \overline{\boldsymbol{z}}_d, \overline{\boldsymbol{z}}_{d'}, \overline{\boldsymbol{w}}_d, \overline{\boldsymbol{w}}_{d'}) = \exp(-2c\max(0, \zeta_{d,d'})),$$

where c is the regularization parameter. In the hinge loss function, $\zeta_{d,d'}$ is

$$\zeta_{d,d'} = 1 - y_{d,d'} R_{d,d'}. \quad (3)$$

3 Posterior Inference

Sampling Topics. Following Polson and Scott (2011), by introducing an auxiliary variable $\lambda_{d,d'}$, we derive the conditional probability of a topic assignment

$$p(z_{d,n} = k \mid \boldsymbol{z}_{-d,n}, \boldsymbol{w}_{-d,n}, w_{d,n} = v) \propto \frac{N_{k,v}^{-d,n} + \beta}{N_{k,\cdot}^{-d,n} + V\beta} \times (N_{d,k}^{-d,n} + \alpha\pi_{l_d,k}^{-d,n}) \times \prod_{d'} \exp\left(-\frac{(c\zeta_{d,d'} + \lambda_{d,d'})^2}{2\lambda_{d,d'}}\right), \quad (4)$$

where $N_{k,v}$ denotes the count of word v assigned to topic k; $N_{d,k}$ is the number of tokens in document d that are assigned to topic k.[3] Marginal counts are denoted by $\cdot$; $^{-d,n}$ denotes that the count excludes token n in document d; d' denotes the indexes of documents which are linked to document d; $\pi_{l_d,k}^{-d,n}$ is estimated based on the maximal path assumption (Wallach, 2008)

$$\pi_{l_d,k}^{-d,n} = \frac{\sum_{d' \in S(l_d)} N_{d',k}^{-d,n} + \alpha'}{\sum_{d' \in S(l_d)} N_{d',\cdot}^{-d,n} + K\alpha'}, \quad (5)$$

where $S(l_d)$ denotes the cluster which contains document d (Step 1 in the generative process).

[3]More details here and throughout this section appear in the supplementary materials.

Optimizing topic and lexical regression parameters. While topic regression parameters $\boldsymbol{\eta}$ and lexical regression parameters $\boldsymbol{\tau}$ can be sampled (Zhu et al., 2014), the associated covariance matrix is huge (approximately $12K \times 12K$ in our experiments). Instead, we optimize these parameters using L-BFGS.

Sampling auxiliary variables. The likelihood of auxiliary variables $\boldsymbol{\lambda}$ follows a generalized inverse Gaussian distribution $\mathrm{GIG}(\lambda_{d,d'}; \frac{1}{2}, 1, c^2\zeta_{d,d'}^2)$. Thus we sample $\lambda_{d,d'}^{-1}$ from a an inverse Gaussian distribution

$$p(\lambda_{d,d'}^{-1} \mid \boldsymbol{z}, \boldsymbol{w}, \boldsymbol{\eta}, \boldsymbol{\tau}) = \mathrm{IG}\left(\lambda_{d,d'}^{-1}; \frac{1}{c|\zeta_{d,d'}|}, 1\right). \quad (6)$$

4 Experimental Results

4.1 Dataset

We crawl data from Sina Weibo, the largest Chinese micro-blog platform. The dataset contains 2,000 randomly-selected verified users, each represented by a single document aggregating all the user's posts. We also crawl links between pairs of users when both are in our dataset. Links correspond to three types of interactions on Weibo: mentioning, retweeting and following.[4]

4.2 Perplexity Results

As an initial reality check, we first apply a simplified version of our model which only uses user interactions for topic modeling and does not predict links. This permits a direct comparison of our model's performance against LDA and Markov random topic fields (Daumé III, 2009, MRTF) by evaluating perplexity.

We set $\alpha = \alpha' = 15$ and run the models on 20 topics for all models in this and following sections. The results are the average values of five independent runs. Following Daumé, in each run, for each document, 80% of its tokens are randomly selected for training and the remaining 20% are for test. As the training corpus is generated randomly, seeding is not applied in this section. The results are given in Table 1, where I- denotes that the model incorporates user interactions.

The results confirm that our model outperforms both LDA and MRTF and that its use of user interactions holds promise.

[4]We use ICTCLAS (Zhang et al., 2003) for segmentation. After stopword and low-frequency word removal, the vocabulary includes 12,257 words, with ∼755 tokens per document and 5,404 links.

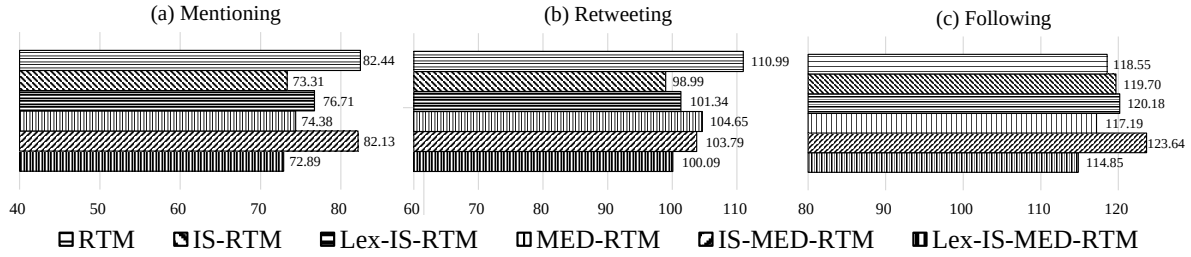


Figure 2: Lex-IS-MED-RTM, combining all three extensions, performs the best on predicting mentioning and following links, although IS-RTM achieves a close value on mentioning links and even a slightly better value on retweeting links. User interactions (denoted by "I") sometimes bring down the performance, as cluster priors are not applied in this intrinsic evaluation.

Link	Model	Perplexity
–	LDA	2605.06
Mentioning	MRTF	2582.08
	I-LDA	2522.58
Retweeting	MRTF	2588.30
	I-LDA	2519.27
Following	MRTF	2587.26
	I-LDA	2530.67

Table 1: Our simplified model I-LDA achieves lower perplexities than both LDA and MRTF, by incorporating different cliques extracted from three types of user interactions.

4.3 Link Prediction Results

In this section, we apply our model on link prediction tasks and evaluate by predictive link rank (PLR). A document's PLR is the average rank, among all documents, of the documents to which it actually links. This means that lower values of PLR are better.

Figure 2 breaks out the 5-fold cross validation results and the distinct extensions of RTM.[5] The results support the value in combining all three extensions using Lex-IS-MED-RTM, although for mentioning and retweeting, Lex-IS-MED-RTM and IS-RTM are quite close.

Applying user interactions does not always produce improvements. This is because in our intrinsic evaluation, we assume that the links on the test set are not observable and cluster priors are not applied. However, according to the training performance (extrinsic evaluations which we are still in progress), user interactions do benefit link prediction performance when links are partially available, e.g., suggesting more links based on observed links. In contrast, hinge loss and lexical term weights do not depend on metadata availability and generally produce improvements in link prediction performance.

4.4 Illustrative Example

We illustrate model behavior qualitatively by looking at two test set users, designated A and B. User A is a reporter who runs "We Media" on his account, sending news items to followers, and B is a consultant with a wide range of interests. Their tweets reveal that both are interested in social news—a topic emphasizing words like *society*, *country*, *government*, *laws*, *leaders*, *political party*, *news*, etc. Both often retweet news related to unfairness in society and local government scandals (*government*, *police*, *leaders*, *party*, *policy*, *chief secretary*). For example, User A retweeted a report that a person about to be executed was unable to take a photo with his family before his execution, writing *I feel heartbroken*. User B retweeted news that a mayor was fired and investigated because of a bribe; in his retweet, he expresses his dissatisfaction with what the mayor did when he was in power. In addition, User A follows new technology (*smart phone*, *Apple*, *Samsung*, *software*, *hardware*, etc.) and B is interested in food (*snacks*, *noodles*, *wine*, *fish*, etc.).

As ground truth, there is a *mentioning* link from A to B; Table 2 shows this link's PLR in the *mentioning* models, which generally improves with model sophistication. The mentioning tweet is a news item that is consistent with the model's

[5]IS- denotes that the model incorporates user interactions and seed words, Lex- means that lexical terms were included in the link probability function (Equation 3), and MED- denotes max-margin learning (Zhu et al., 2014; Zhu et al., 2012). Each type of link is applied separately; e.g., in Figure 2(a) results are based *only* on mentioning links, ignoring retweeting and following links.

Model		RTM	IS-RTM	Lex-IS-RTM	MED-RTM	IS-MED-RTM	Lex-IS-MED-RTM
PLR of the Link		24	10	9	74	18	26
Social News	**User A**	0.018	0.021	0.034	0.016	0.027	0.030
Topic Proportion	**User B**	0.309	0.413	0.408	0.318	0.355	0.392

Table 2: Data for Illustrative Example

Model		RTM	IS-RTM	Lex-IS-RTM	MED-RTM	IS-MED-RTM	Lex-IS-MED-RTM
Topic PMI		1.186	1.224	1.216	1.214	1.294	1.229
Average Regression Values	**Linked Pairs**	0.2403	0.3692	0.4031	0.7220	0.6321	0.7668
	All Pairs	0.06636	0.07729	0.08020	0.2482	0.2041	0.2428
	Ratio	3.621	4.777	5.026	2.909	3.097	3.158
	SD/Avg	0.9415	1.2081	1.2671	0.6364	0.7254	0.7353

Table 3: Values for Quantitative Analysis

characterization of the users' interests (particularly social news and technology): a Samsung Galaxy S4 exploded and caused a fire while charging. Consistent with intuition, the prevalence of the social news topic also generally increases as the models grow more sophisticated.[6]

4.5 Quantitative Analysis

Topic Quality. Automatic coherence detection (Lau et al., 2014) is an alternative to manual evaluations of topic quality (Chang et al., 2009). In each topic, the top n words' average pointwise mutual information (PMI)—based on a reference corpus—serves as a measure of topic coherence.[7]

Topic quality improves with user interactions and max-margin learning (Table 3). PMI drops when lexical terms are added to the link probability function, however. This is consistent with the role of lexical terms in the model; their purpose is to improve link prediction performance, not improve topic quality.

Average Regression Value. One way to assess the quality of link prediction is to compare the scores of (ground-truth) linked documents to documents in general. In Table 3, the Average Regression Values show this comparison as a ratio. The higher the ratio, the more linked document pairs differ from unlinked pairs, which means that linked documents are easier to distinguish. This ratio improves as RTM extensions are added, indicating better link modeling quality.

In the SD/Avg row of Table 3, we also compute a ratio of standard deviations to mean values. Ratios given by the models with hinge loss are lower than those not using hinge loss. This means that the regression values given by the models with hinge loss are more concentrated around the average value, suggesting that these models can better identify linked pairs, even though the ratio of linked pairs' average regression value to all pairs' average value is lower.

5 Conclusions and Future Work

We introduce a new topic model that takes advantage of document links, incorporating link information straightforwardly by deriving clusters from the link graph and assigning each cluster a separate Dirichlet prior. We also take advantage of locally-derived distributed representations to "seed" the model's latent topics in an informed way, and we integrate max-margin prediction and lexical regression to improve link prediction quality. Our quantitative results show improvements in predictive link rank, and our qualitative and quantitative analysis illustrate that the model's behavior is intuitively plausible.

In future work, we plan to engage in further model analysis and comparison, to explore alterations to model structure, e.g. introducing hierarchical topic models, to use other clustering methods to obtain priors, and to explore the value of predicted links for downstream tasks such as friend recommendation (Pennacchiotti and Gurumurthy, 2011) and inference of user attributes (Volkova et al., 2014).

[6] Numerically its proportion is consistently lower for User A, whose interests are more diverse.

[7] We set $n = 20$ and use a reference corpus of 1,143,525 news items from Sogou Lab, comprising items from June to July 2012, `http://www.sogou.com/labs/dl/ca.html`. Each averages ∼347 tokens, using the same segmentation scheme as the experimental corpus.

Acknowledgements

We thank Hal Daumé III for providing his code. This work was supported in part by NSF award 1211153. Boyd-Graber is supported by NSF Grants CCF-1409287, IIS-1320538, and NCSE-1422492. Any opinions, findings, conclusions, or recommendations expressed here are those of the authors and do not necessarily reflect the view of the sponsor.

References

David Andrzejewski and Xiaojin Zhu. 2009. Latent Dirichlet allocation with topic-in-set knowledge. In *Conference of the North American Chapter of the Association for Computational Linguistics*.

David M. Blei and Jon D. McAuliffe. 2007. Supervised topic models. In *Proceedings of Advances in Neural Information Processing Systems*.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Jonathan Chang and David M. Blei. 2010. Hierarchical relational models for document networks. *The Annals of Applied Statistics*, pages 124–150.

Jonathan Chang, Sean Gerrish, Chong Wang, Jordan L. Boyd-Graber, and David M. Blei. 2009. Reading tea leaves: How humans interpret topic models. In *Proceedings of Advances in Neural Information Processing Systems*.

Hal Daumé III. 2009. Markov random topic fields. In *Proceedings of the Association for Computational Linguistics*.

Myunghwan Kim and Jure Leskovec. 2012. Latent multi-group membership graph model. In *Proceedings of the International Conference of Machine Learning*.

Jey Han Lau, David Newman, and Timothy Baldwin. 2014. Machine reading tea leaves: Automatically evaluating topic coherence and topic model quality. In *Proceedings of the Association for Computational Linguistics*.

Jon D. McAuliffe and David M. Blei. 2008. Supervised topic models. In *Proceedings of Advances in Neural Information Processing Systems*.

Miller McPherson, Lynn Smith-Lovin, and James M. Cook. 2001. Birds of a feather: Homophily in social networks. *Annual Review of Sociology*, pages 415–444.

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

David M. Mimno and Andrew McCallum. 2008. Topic models conditioned on arbitrary features with Dirichlet-multinomial regression. In *Proceedings of Uncertainty in Artificial Intelligence*.

Viet-An Nguyen, Jordan L. Boyd-Graber, and Philip Resnik. 2013. Lexical and hierarchical topic regression. In *Proceedings of Advances in Neural Information Processing Systems*.

Marco Pennacchiotti and Siva Gurumurthy. 2011. Investigating topic models for social media user recommendation. In *Proceedings of World Wide Web Conference*.

Nicholas G. Polson and Steven L. Scott. 2011. Data augmentation for support vector machines. *Bayesian Analysis*, 6(1):1–23.

Yee Whye Teh, Michael I. Jordan, Matthew J. Beal, and David M. Blei. 2006. Hierarchical Dirichlet processes. *Journal of the American Statistical Association*, 101(476):1566–1581.

Svitlana Volkova, Glen Coppersmith, and Benjamin Van Durme. 2014. Inferring user political preferences from streaming communications. In *Proceedings of the Association for Computational Linguistics*.

Hanna M. Wallach. 2008. *Structured topic models for language*. Ph.D. thesis, University of Cambridge.

Hua-Ping Zhang, Hong-Kui Yu, De-Yi Xiong, and Qun Liu. 2003. HHMM-based Chinese lexical analyzer ICTCLAS. In *Proceedings of the second SIGHAN workshop on Chinese language processing-Volume 17*.

Jun Zhu, Amr Ahmed, and Eric P. Xing. 2012. MedLDA: Maximum margin supervised topic models. *Journal of Machine Learning Research*, 13(1):2237–2278.

Jun Zhu, Ning Chen, Hugh Perkins, and Bo Zhang. 2014. Gibbs max-margin topic models with data augmentation. *Journal of Machine Learning Research*, 15(1).

Aligning Knowledge and Text Embeddings by Entity Descriptions

Huaping Zhong[§] , **Jianwen Zhang**[†] , **Zhen Wang**[§] , **Hai Wan**[§] , **Zheng Chen**[†]
[§]{zhonghp@mail2,wangzh56@mail2,wanhai@mail}.sysu.edu.cn
[†]{jiazhan,zhengc}@microsoft.com
[§]Sun Yat-sen University [†]Microsoft Research

Abstract

We study the problem of jointly embedding a knowledge base and a text corpus. The key issue is the alignment model making sure the vectors of entities, relations and words are in the same space. Wang et al. (2014a) rely on Wikipedia anchors, making the applicable scope quite limited. In this paper we propose a new alignment model based on text descriptions of entities, without dependency on anchors. We require the embedding vector of an entity not only to fit the structured constraints in KBs but also to be equal to the embedding vector computed from the text description. Extensive experiments show that, the proposed approach consistently performs comparably or even better than the method of Wang et al. (2014a), which is encouraging as we do not use any anchor information.

1 Introduction

Knowledge base embedding has attracted surging interest recently. The aim is to learn continuous vector representations (embeddings) for entities and relations of a structured knowledge base (KB) such as Freebase. Typically it optimizes a global objective function over all the facts in the KB and hence the embedding vector of an entity / relation is expected to encode global information in the KB. It is capable of reasoning missing facts in a KB and helping facts extraction (Bordes et al., 2011; Bordes et al., 2012; Bordes et al., 2013; Socher et al., 2013; Chang et al., 2013; Wang et al., 2014b; Lin et al., 2015).

Although seeming encouraging, the approaches in the aforementioned literature suffer from two common issues: (1) Embeddings are exclusive to entities/relations within KBs. Computation between KBs and text cannot be handled, which are prevalent in practice. For example, in fact extraction, a candidate value may be just a phrase in text. (2) KB sparsity. The above approaches are only based on structured facts of KBs, and thus cannot work well on entities with few facts.

An important milestone, the approach of Wang et al. (2014a) solves issue (1) by jointly embedding entities, relations, and words into the same vector space and hence is able to deal with words/phrases beyond entities in KBs. The key component is the so-called *alignment model*, which makes sure the embeddings of entities, relations, and words are in the same space. Two alignment models are introduced there: one uses entity names and another uses Wikipedia anchors. However, both of them have drawbacks. As reported in the paper, using entity names severely pollutes the embeddings of words. Thus it is not recommended in practice. Using Wikipedia anchors completely relies on the special data source and hence the approach cannot be applied to other customer data.

To fully address the two issues, this paper proposes a new alignment method, aligning by entity descriptions. We only assume some entities in KBs have text descriptions, which almost always holds in practice. We require the embedding of an entity not only fits the structured constraints in KBs but also equals the vector computed from the text description. Meanwhile, if an entity has few facts, the description will provide information for embedding, thus the issue of KB sparsity is also well handled. We conduct extensive experiments on the tasks of triplet classification, link prediction, relational fact extraction, and analogical reasoning to compare with the previous approach (Wang et al., 2014a). Results show that our approach consistently achieves better or comparable performance.

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 267–272,
Lisbon, Portugal, 17-21 September 2015.

2 Related Work

TransE This is a representative knowledge embedding model proposed by Bordes et al. (2013). For a fact (h, r, t) in KBs, where h is the head entity, r is the relation, and t is the tail entity, TransE models the relation r as a translation vector $\mathbf{r}$ connecting the embeddings $\mathbf{h}$ and $\mathbf{t}$ of the two entities, i.e., $\mathbf{h} + \mathbf{r}$ is close to $\mathbf{t}$. The model is simple, effective and efficient. Most knowledge embedding models thereafter including this paper are variants of this model (Wang et al., 2014b; Wang et al., 2014a; Lin et al., 2015).

Skip-gram This is an efficient word embedding method proposed by Mikolov et al. (2013a), which learns word embeddings from word concurrencies in text windows. Without any supervision, it amazingly recovers the semantic relations between words in a vector space such as 'King' − 'Queen' ≈ 'Man' − 'Women'. However, as it is unsupervised, it cannot tell the exact relation between two words.

Knowledge and Text Jointly Embedding Wang et al. (2014a) combines knowledge embedding and word embedding in a joint framework so that the entities/relations and words are in the same vector space and hence operators like inner product (similarity) between them are meaningful. This brings convenience to tasks requiring computation between knowledge bases and text. Meanwhile, jointly embedding utilizes information from both structured KBs and unstructured text and hence the knowledge embedding and word embedding can be enhanced by each other. Their model is composed of three components: a *knowledge model* to embed entities and relations, a *text model* to embed words, and an *alignment model* to make sure entities/relations and words are in the same vector space. The knowledge model and text model are variants of TransE and Skip-gram respectively. The key component is the alignment model. They introduced two: alignment by entity names and alignment by Wikipedia anchors. (1) **Alignment by Entity Names** makes a replicate of KB facts but replaces each entity ID with its name string, i.e., the vector of a name phrase is encouraged to equal to the vector of the entity (identified by ID). It has problems with ambiguous entity names and observed polluting word embeddings thus it is not recommended by the authors. (2) **Alignment by Wikipedia Anchors** replaces the surface phrase v of a Wikipedia anchor with its corresponding Freebase entity e_v and defines the likelihood

$$\mathcal{L}_{AA} = \sum_{(w,v)\in\mathcal{C}, v\in\mathcal{A}} \log \Pr(w|e_v) \quad (1)$$

where $\mathcal{C}$ is the collection of observed word and context pairs and $\mathcal{A}$ refers to the set of all anchors in Wikipedia. $\Pr(w|e_v)$ is the probability of the anchor predicting its context word, which takes a form similar to Skip-gram for word embedding. Alignment by anchors works well in both improving knowledge embedding and word embeddings. However, it completely relies on the special data source of Wikipedia anchors and cannot be applied to other general data settings.

3 Alignment by Entity Descriptions

We first describe the settings and notations. Given a knowledge base, i.e., a set of facts (h, r, t), where $h, t \in \mathcal{E}$ (the set of entities) and $r \in \mathcal{R}$ (the set of relations). Some entities have text descriptions. The description of entity e is denoted as D_e. $w_{i,n}$ is the n^{th} word in the description of e_i. N_i is the length (in words) of the description of e_i. We try to learn embeddings $\mathbf{e}_i$, $\mathbf{r}_j$ and $\mathbf{w}_l$ for each entity e_i, relation r_j and word w_l respectively. The vocabulary of words is $\mathcal{V}$. The union vocabulary of entities and words together is $\mathcal{I} = \mathcal{E} \cup \mathcal{V}$. In this paper "word(s)" refers to "word(s)/phrase(s)".

We follow the jointly embedding framework of (Wang et al., 2014a), i.e., learning optimal embeddings by minimizing the following loss

$$\mathcal{L}\left(\{\mathbf{e}_i\}, \{\mathbf{r}_j\}, \{\mathbf{w}_l\}\right) = \mathcal{L}_K + \mathcal{L}_T + \mathcal{L}_A, \quad (2)$$

where $\mathcal{L}_K$, $\mathcal{L}_T$ and $\mathcal{L}_A$ are the component loss functions of the knowledge model, text model and alignment model respectively. Our focus is on a new alignment model $\mathcal{L}_A$ while the knowledge model $\mathcal{L}_K$ and text model $\mathcal{L}_T$ are the same as the counterparts in (Wang et al., 2014a). However, to make the content self-contained, we still need to briefly explain $\mathcal{L}_K$ and $\mathcal{L}_T$.

Knowledge Model Describes the plausibility of a triplet (h, r, t) by defining

$$\Pr(h|r,t) = \frac{\exp\{z(h,r,t)\}}{\sum_{\tilde{h}\in\mathcal{I}} \exp\{z(\tilde{h},r,t)\}}, \quad (3)$$

where $z(h, r, t) = b - 0.5 \cdot \|\mathbf{h} + \mathbf{r} - \mathbf{t}\|_2^2$, $b = 7$ as suggested by Wang et al. (2014a). $\Pr(r|h, t)$ and

$\Pr(t|h,r)$ are defined in the same way. The loss function of knowledge model is then defined as

$$\mathcal{L}_K = -\sum_{(h,r,t)} \big[\log \Pr(h|r,t) + \log \Pr(t|h,r) + \log \Pr(r|h,t)\big] \quad (4)$$

Text Model Defines the probability of a pair of words w and v co-occurring in a text window:

$$\Pr(w|v) = \frac{\exp\{z(w,v)\}}{\sum_{\tilde{w}\in\mathcal{V}} \exp\{z(\tilde{w},v)\}} \quad (5)$$

where $z(w,v) = b - 0.5 \cdot \|\mathbf{w} - \mathbf{v}\|_2^2$. Then the loss function of text model is

$$\mathcal{L}_T = -\sum_{(w,v)} \log \Pr(w|v) \quad (6)$$

Alignment Model This part is different from Wang et al. (2014a). For each word w in the description of entity e, we define $\Pr(w|e)$, the conditional probability of predicting w given e:

$$\Pr(w|e) = \frac{\exp\{z(e,w)\}}{\sum_{\tilde{w}\in\mathcal{V}} \exp\{z(e,\tilde{w})\}}, \quad (7)$$

where $z(e,w) = b - 0.5 \cdot \|\mathbf{e} - \mathbf{w}\|_2^2$. Notice that $\mathbf{e}$ is the same vector of entity e appearing in the knowledge model of Eq. (3).

We also define $\Pr(e|w)$ in the same way by revising the normalization term

$$\Pr(e|w) = \frac{\exp\{z(e,w)\}}{\sum_{\tilde{e}\in\mathcal{E}} \exp\{z(\tilde{e},w)\}} \quad (8)$$

Then the loss function of alignment model is

$$\mathcal{L}_A = -\sum_{e\in\mathcal{E}} \sum_{w\in D_e} [\log \Pr(w|e) + \log \Pr(e|w)] \quad (9)$$

Training We use stochastic gradient descent (SGD) to minimize the overall loss of Eq. (2), which sequentially updates the embeddings. Negative sampling is used to calculate the normalization items over large vocabularies. We implement a multi-threading version to deal with large data sets, where memory is shared and lock-free.

4 Experiments

We conduct experiments on the following tasks: link prediction (Bordes et al., 2013), triplet classification (Socher et al., 2013), relational fact extraction (Weston et al., 2013), and analogical reasoning (Mikolov et al., 2013b). The last one evaluates quality of word embeddings. We try to study whether the proposed alignment model, without using any anchor information, is able to achieve comparable or better performance than alignment by anchors. As to the methods, "Separately" denotes the method of separately embedding knowledge bases and text. "Jointly(anchor)" and "Jointly(name)" denote the jointly embedding methods based on Alignment by Wikipedia Anchors and Alignment by Entity Names in (Wang et al., 2014a) respectively. "Jointly(desp)" is the joint embedding method based on alignment by entity descriptions.

Table 1: Link prediction results.

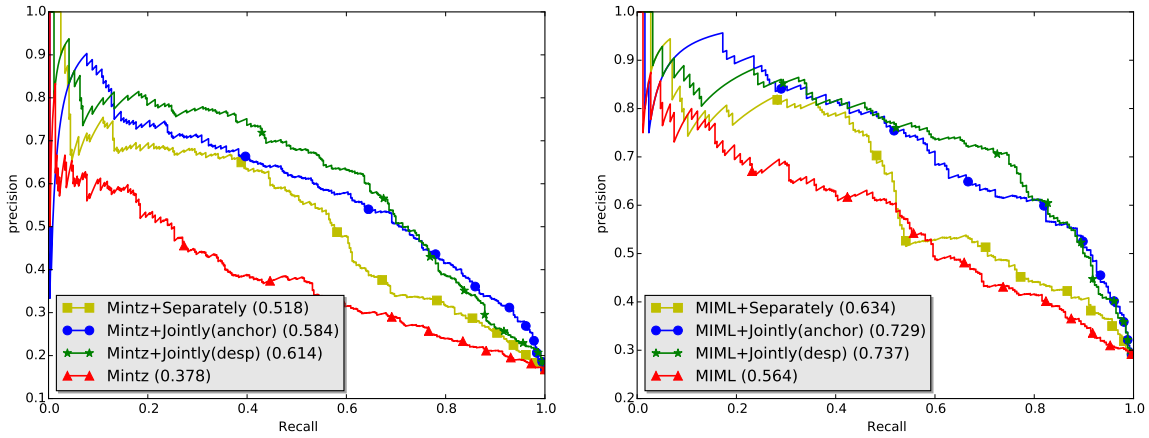
Metric	MEAN		HITS@10	
	Raw	Filtered	Raw	Filtered
TransE	243	125	34.9	47.1
Jointly(anchor)	**166**	47	49.9	72.0
Jointly(desp)	167	**39**	**51.7**	**77.3**

Table 2: Triplet classification results.

Type	e - e	w - e	e - w	w - w	all
Separately	94.0	51.7	51.0	69.0	73.6
Jointly(anchor)	95.2	65.3	65.1	76.2	79.9
Jointly(desp)	**96.1**	**66.7**	**66.1**	**76.4**	**80.9**

Data For link prediction, FB15K from (Bordes et al., 2013) is used as the knowledge base. For triplet classification, a large dataset provided by (Wang et al., 2014a) is used as the knowledge base. Both sets are subsets of Freebase. For all tasks, Wikipedia articles are used as the text corpus. As many Wikipedia articles can be mapped to Freebase entities, we regard a Wikipedia article as the description for the corresponding entity in Freebase. Following the settings in (Wang et al., 2014a), we apply the same preprocessing steps, including sentence segmentation, tokenization, and named entity recognition. We combine the consecutive tokens covered by an anchor or identically tagged as "Location/Person/Organization" and regard them as phrases.

Link Prediction This task aims to complete a fact (h,r,t) in absence of h or t, simply based on $\|\mathbf{h}+\mathbf{r}-\mathbf{t}\|$. We follow the same protocol in (Bordes et al., 2013). We directly copy the results of the baseline (TransE) from (Bordes et al., 2013) and implement "Jointly(anchor)". The results are in Table 1. "MEAN" is the average rank of the true absent entity. "HITS@10" is accuracy of the top



(a) (b)

Figure 1: Precision-recall curves for relation extraction. (a) Mintz (Mintz et al., 2009) as base extractor (b) MIML (Surdeanu et al., 2012) as base extractor.

10 predictions containing the true entity. Lower "MEAN" and higher "HITS@10" is better. "Raw" and "Filtered" are two settings on processing candidates (Bordes et al., 2013).

We train "Jointly(anchor)" and "Jointly(desp)" with the embedding dimension k among $\{50, 100, 150\}$, the learning rate α in $\{0.01, 0.025\}$, the number of negative examples per positive example c in $\{5, 10\}$, the max skip-range s in $\{5, 10\}$ and traverse the text corpus with only 1 epoch. The best configurations of "Jointly(anchor)" and "Jointly(desp)" are exactly the same: $k = 100, \alpha = 0.025, c = 10, s = 5$.

From the results, we observe that: (1) Both jointly embedding methods are much better than the baseline TransE, which demonstrates that external textual resources make entity embeddings become more discriminative. Intuitively, "Jointly(anchor)" indicates "*how to use an entity in text*", while "Jointly(desp)" shows "*what is the definition/meaning of an entity*". Both are helpful to distinguish an entity from others. (2) Under the setting of "Raw", "Jointly(desp)" and "Jointly(anchor)" are comparable. In other settings "Jointly(desp)" wins.

Triplet Classification This is a binary classification task, predicting whether a candidate triplet (h, r, t) is a correct fact or not. It is used in (Socher et al., 2013; Wang et al., 2014b; Wang et al., 2014a). We follow the same protocol in (Wang et al., 2014a).

We train their models via our own implementation on our dataset. The results are in Table 2. "e-e" means both sides of a triplet (h, r, t) are entities in KB, "e-w" means the tail side is a word out of KB entity vocabulary, similarly for "w-e" and "w-w". The best configurations of the models are: $k = 150, \alpha = 0.025, c = 10, s = 5$ and traversing the text corpus with 6 epochs.

The results reveal that: (1) Jointly embedding is indeed effective. Both jointly embedding methods can well handle the cases of "e-w", "w-e" and "w-w", which means the vector computation between entities/relations and words are really meaningful. Meanwhile, even the case of "e-e" is also improved. (2) Our method, "Jointly(desp)", outperforms "Jointly(anchor)" on all types of triplets. We believe that the good performance of "Jointly(desp)" is due to the appropriate design of the alignment mechanism. Using entity's description information is a more straightforward and effective way to align entity embeddings and word embeddings.

Relational Fact Extraction This task is to extract facts (h, r, t) from plain text. Weston et al. (2013) show that combing scores from TransE and some text side base extractor achieved much better precision-recall curve compared to the base extractor. Wang et al. (2014a) confirm this observation and show that jointly embedding brings further encouraging improvement over TransE. In this experiment, we follow the same settings as (Wang et al., 2014a) to investigate the performance of our new alignment model. We use the

Table 3: Analogical reasoning results

Metric	Words		Phrases	
	Acc.	Hits@10	Acc.	Hits@10
Skip-gram	67.4	86.7	22.0	63.6
Jointly(anchor)	**69.4**	87.7	26.2	68.1
Jointly(name)	44.5	69.7	11.5	46.0
Jointly(desp)	69.3	**88.3**	**49.0**	**86.5**

same public dataset NYT+FB, released by Riedel et al. (2010) and used in (Weston et al., 2013) and (Wang et al., 2014a). We use Mintz (Mintz et al., 2009) and MIML (Surdeanu et al., 2012) as our base extractors.

In order to combine the score of a base extractor and the score from embeddings, we only reserve the testing triplets whose entitites and relations can be mapped to the embeddings learned from the triplet classification experiment. Since both Mintz and MIML are probabilistic models, we use the same method in (Wang et al., 2014a) to linearly combine the scores.

The precision-recall curves are plot in Fig. (1). On both base extractors, the jointly embedding methods outperform separate embedding. Moreover, "Jointly(desp)" is slightly better than "Jointly(anchor)", which is in accordance with the results from the link prediction experiment and the triplet classification experiment.

Analogical Reasoning This task evaluates the quality of word embeddings (Mikolov et al., 2013b). We use the original dataset released by (Mikolov et al., 2013b) and follow the same evaluation protocol of (Wang et al., 2014a). For a true analogical pair like ("France", "Paris") and ("China", "Beijing"), we hide "Beijing" and predict it by selecting the word from the vocabulary whose vector has highest similarity with the vector of "China" + "Paris" - "France". We use the word embeddings learned for the triplet classification experiment and conduct the analogical reasoning experiment for "Skip-gram", "Jointly(anchor)", "Jointly(name)" and "Jointly(desp)".

Results are presented in Table 3. "Acc" is the accuracy of the predicted word. "HITS@10" is the accuracy of the top 10 candidates containing the ground truth. The evaluation analogical pairs are organized into two groups, "Words" and "Phrases", by whether an analogical pair contains phrases (i.e., multiple words). From the table we observe that: (1) Both "Jointly(anchor)" and "Jointly(desp)" outperform "Skip-gram". (2) "Jointly(desp)" achieves the best results, especially for the case of "Phrases". Both "Jointly(anchor)" and "Skip-gram" only consider the context of words, while "Jointly(desp)" not only consider the context but also use the whole document to disambiguate words. Intuitively, the whole document is also a valuable resource to disambiguate words. (3) We further verify that "Jointly(name)", i.e., using entity names for alignment, indeed pollutes word embeddings, which is consistent with the reports in (Wang et al., 2014a).

The above four experiments are consistent in results: without using any anchor information, alignment by entity description is able to achieve better or comparable performance, compared to alignment by Wikipedia anchors proposed by Wang et al. (2014a).

5 Conclusion

We propose a new alignment model based on entity descriptions for jointly embedding a knowledge base and a text corpus. Compared to the method of alignment using Wikipedia anchors Wang et al. (2014a), our method has no dependency on special data sources of anchors and hence can be applied to any knowledge bases with text descriptions for entities. Extensive experiments on four prevalent tasks to evaluate the quality of knowledge and word embeddings produce very consistent results: our alignment model achieves better or comparable performance.

References

Antoine Bordes, Jason Weston, Ronan Collobert, and Yoshua Bengio. 2011. Learning structured embeddings of knowledge bases. In *Proceedings of the 25th AAAI Conference on Artificial Intelligence*.

Antoine Bordes, Xavier Glorot, Jason Weston, and Yoshua Bengio. 2012. A semantic matching energy function for learning with multi-relational data. *Machine Learning*, pages 1–27.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems*, pages 2787–2795.

Kai-Wei Chang, Wen-tau Yih, and Christopher Meek. 2013. Multi-relational latent semantic analysis. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1602–1612, Seattle, Washington, USA, October. Association for Computational Linguistics.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Zheng Chen. 2015. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, pages 2181–2187.

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

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

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 1003–1011. Association for Computational Linguistics.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *Machine Learning and Knowledge Discovery in Databases*, pages 148–163. Springer.

Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *Advances in Neural Information Processing Systems*, pages 926–934.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D Manning. 2012. Multi-instance multi-label learning for relation extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 455–465. Association for Computational Linguistics.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014a. Knowledge graph and text jointly embedding. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). Association for Computational Linguistics*, pages 1591–1601.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014b. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, pages 1112–1119.

Jason Weston, Antoine Bordes, Oksana Yakhnenko, and Nicolas Usunier. 2013. Connecting language and knowledge bases with embedding models for relation extraction. *arXiv preprint arXiv:1307.7973*.

An Empirical Analysis of Optimization for Max-Margin NLP

Jonathan K. Kummerfeld, Taylor Berg-Kirkpatrick and **Dan Klein**
Computer Science Division
University of California, Berkeley
Berkeley, CA 94720, USA
{jkk,tberg,klein}@cs.berkeley.edu

Abstract

Despite the convexity of structured max-margin objectives (Taskar et al., 2004; Tsochantaridis et al., 2004), the many ways to optimize them are not equally effective in practice. We compare a range of online optimization methods over a variety of structured NLP tasks (coreference, summarization, parsing, etc) and find several broad trends. First, margin methods do tend to outperform both likelihood and the perceptron. Second, for max-margin objectives, primal optimization methods are often more robust and progress faster than dual methods. This advantage is most pronounced for tasks with dense or continuous-valued features. Overall, we argue for a particularly simple online primal subgradient descent method that, despite being rarely mentioned in the literature, is surprisingly effective in relation to its alternatives.

1 Introduction

Structured discriminative models have proven effective across a range of tasks in NLP including tagging (Lafferty et al., 2001; Collins, 2002), reranking parses (Charniak and Johnson, 2005), and many more (Taskar, 2004; Smith, 2011). Common approaches to training such models include margin methods, likelihood methods, and mistake-driven procedures like the averaged perceptron algorithm. In this paper, we primarily consider the relative empirical behavior of several online optimization methods for margin-based objectives, with secondary attention to other approaches for calibration.

It is increasingly common to train structured models using a max-margin objective that incorporates a loss function that decomposes in the same way as the dynamic program used for inference (Taskar, 2004). Fortunately, most structured margin objectives are convex, so a range of optimization methods with similar theoretical properties are available – in short, any of these methods will work in the end. However, in practice, how fast each method converges varies across tasks. Moreover, some of the most popular methods more loosely associated with the margin objective, such as the MIRA algorithm (Crammer and Singer, 2003) or even the averaged perceptron (Freund and Schapire, 1999) are not global optimizations and can have different properties.

We analyze a range of methods empirically, to understand on which tasks and with which feature types, they are most effective. We modified six existing, high-performance, systems to enable loss-augmented decoding, and trained these models with six different methods. We have released our learning code as a Java library.[1] Our results provide support for the conventional wisdom that margin-based optimization is broadly effective, frequently outperforming likelihood optimization and the perceptron algorithm. We also found that directly optimizing the primal structured margin objective based on subgradients calculated from single training instances is surprisingly effective, performing consistently well across all tasks.

2 Learning Algorithms

We implemented a range of optimization methods that are widely used in NLP; below we categorize them into margin, likelihood, and perceptron-like methods. In each case, we used a structured loss function, modified to suit each task. In general, we focus on online methods because of their substantial speed advantages, rather than algorithms such as LBFGS (Liu and Nocedal, 1989) or batch Exponentiated Gradient (Collins et al., 2008).

[1] http://nlp.cs.berkeley.edu/software.shtml

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 273–279,
Lisbon, Portugal, 17-21 September 2015.

Algorithm 1 The Online Primal Subgradient Algorithm with ℓ_1 or ℓ_2 regularization, and sparse updates

Parameters:

iters	Number of iterations
C	Regularization constant (10^{-1} to 10^{-8})
η	Learning rate (10^{0} to 10^{-4})
δ	Initializer for q (10^{-6})

$\mathbf{w} = \mathbf{0}$ Weight vector
$\mathbf{q} = \boldsymbol{\delta}$ Cumulative squared gradient
$\mathbf{u} = \mathbf{0}$ Time of last update for each weight
$n = 0$ Number of updates so far
for iter $\in [1, \text{iters}]$ **do**
  for batch $\in$ data **do**
    Sum gradients from loss-aug. decodes
    $\mathbf{g} = \mathbf{0}$
    for $(x_i, y_i) \in$ batch **do**
      $y = \underset{y' \in Y(x_i)}{\text{argmax}}[\text{SCORE}(y') + \mathbf{L}(y', y_i)]$
      $\mathbf{g}$ += $(\mathbf{f}(y) - \mathbf{f}(y_i))$
    Update the active features
    $\mathbf{q}$ += $\mathbf{g}^2$ Element-wise square
    n += 1
    for $f \in$ nonzero features in $\mathbf{g}$ **do**
      $w_f = \text{UPDATE-ACTIVE}(w_f, g_f, q_f)$
      $u_f = n$

The AdaGrad update
function UPDATE-ACTIVE(w, g, q)
  return $\frac{w\sqrt{q} - \eta g}{\eta \mathrm{C} + \sqrt{q}}$ $[\ell_2]$
  $\mathrm{d} = |w - \frac{\eta}{\sqrt{q}} g| - \frac{\eta}{\sqrt{q}} \mathrm{C}$ $[\ell_1]$
  return $\text{sign}(w - \frac{\eta}{\sqrt{q}} g) \cdot \max(0, \mathrm{d})$ $[\ell_1]$

Functions only needed for sparse updates

A single update equivalent to a series of AdaGrad updates where the weight's subgradient was zero
function UPDATE-CATCHUP(w, q, t)
  return $w \left(\frac{\sqrt{q}}{\eta \mathrm{C} + \sqrt{q}}\right)^t$ $[\ell_2]$
  return $\text{sign}(w) \cdot \max(0, |w| - \frac{\eta \mathrm{C}}{\sqrt{q}} t)$ $[\ell_1]$

Compute $\mathbf{w}^\top \mathbf{f}(y')$, but for each weight, apply an update to catch up on the steps in which the gradient for that weight was zero
function SCORE(y')
  $s = 0$
  for $f \in \mathbf{f}(y')$ **do**
    $w_f = \text{UPDATE-CATCHUP}(w_f, q_f, n - u_f)$
    $u_f = n$
    s += w_f
  return s

Note: To implement without the sparse update, use SCORE $= \mathbf{w}^\top \mathbf{f}(y')$, and run the update loop on the left over all features. Also, for comparison, to implement perceptron, remove the sparse update and use UPDATE-ACTIVE $=$ **return** $w + g$.

2.1 Margin

Cutting Plane (Tsochantaridis et al., 2004) Solves a sequence of quadratic programs (QP), each of which is an approximation to the dual formulation of the margin-based learning problem. At each iteration, the current QP is refined by adding additional active constraints. We solve each approximate QP using Sequential Minimal Optimization (Platt, 1999; Taskar et al., 2004).

Online Cutting Plane (Chang and Yih, 2013) A modified form of cutting plane that only partially solves the QP on each iteration, operating in the dual space and optimizing a single dual variable on each iteration. We use a variant of Chang and Yih (2013) for the L_1 loss margin objective.

Online Primal Subgradient (Ratliff et al., 2007) Computes the subgradient of the margin objective on each instance by performing a loss-augmented decode, then uses these instance-wise subgradients to optimize the global objective using AdaGrad (Duchi et al., 2011) with either L_1 or L_2 regularization. The simplest implementation of AdaGrad touches every weight when doing the update for a batch. To save time, we distinguish between two different types of update. When the subgradient is nonzero, we apply the usual update. When the subgradient is zero, we apply a numerically equivalent update later, at the next time the weight is queried. This saves time, as we only touch the weights corresponding to the (usually sparse) nonzero directions in the current batch's subgradient. Algorithm 1 gives pseudocode for our implementation, which was based on Dyer (2013).

2.2 Likelihood

Stochastic Gradient Descent The built-in training method for many of the systems was softmax-margin likelihood optimization (Gimpel and Smith, 2010) via subgradient descent with either AdaGrad or AdaDelta (Duchi et al., 2011; Zeiler, 2012). We include results with each system's default settings as a point of comparison.

2.3 Mistake Driven

Averaged Perceptron (Freund and Schapire, 1999; Collins, 2002) On a mistake, weights for features on the system output are decremented and weights for features on the gold output are incre-

mented. Weights are averaged over the course of training, and decoding is not loss-augmented.

Margin Infused Relaxed Algorithm (Crammer and Singer, 2003) A modified form of the perceptron that uses loss-augmented decoding and makes the smallest update necessary to give a margin at least as large as the loss of each solution. MIRA is generally presented as being related to the perceptron because it does not explicitly optimize a global objective, but it also has connections to margin methods, as explored by Chiang (2012). We consider one-best decoding, where the quadratic program for determining the magnitude of the update has a closed form.

3 Tasks and Systems

We considered tasks covering a range of structured output spaces, from sequences to non-projective trees. Most of the corresponding systems use models designed for likelihood-based structured prediction. Some use sparse indicator features, while others use dense continuous-valued features.

Named Entity Recognition This task provides a case of sequence prediction. We used the NER component of Durrett and Klein (2014)'s entity stack, training it independently of the other components. We define the loss as the number of incorrectly labelled words, and train on the CoNLL 2012 division of OntoNotes (Pradhan et al., 2007).

Coreference Resolution This gives an example of training when there are multiple gold outputs for each instance. The system we consider uses latent links between mentions in the same cluster, marginalizing over the possibilities during learning (Durrett and Klein, 2013). Since the model decomposes across mentions, we train by treating them as independent predictions with multiple gold outputs, comparing the inferred link with the gold link that is scored highest under the current model. We use the system's weighted loss function, and the same data as for NER.

Constituency Parsing We considered two different systems. The first uses only sparse indicator features (Hall et al., 2014), while the second is parameterized via a neural network and adds dense features derived from word vectors (Durrett and Klein, 2015).[2] We define the loss as the number of incorrect rule productions, and use the standard Penn Treebank division (Marcus et al., 1993).

Dependency Parsing We used the first-order MST parser in two modes, Eisner's algorithm for projective trees (Eisner, 1996; McDonald et al., 2005b), and the Chu-Liu-Edmonds algorithm for non-projective trees (Chu and Liu, 1965; Edmonds, 1967; McDonald et al., 2005a). The loss function was the number of arcs with an incorrect parent or label, and we used the standard division of the English Universal Dependencies (Agić et al., 2015). The built-in training method for MST parser is averaged, 1-best MIRA, which we include for comparison purposes.

Summarization With this task, we explore a case in which there is relatively little training data and the model uses a small number of dense features. The system uses a linear model with features considering counts of bigrams in the input document collection. The system forms the output summary by selecting a subset of the sentences in the input collection that does not exceed a fixed word-length limit (Berg-Kirkpatrick et al., 2011). Inference involves solving an integer linear program, the loss function is bigram recall, and the data is from the TAC shared tasks (Dang and Owczarzak, 2008; Dang and Owczarzak, 2009).

3.1 Tuning

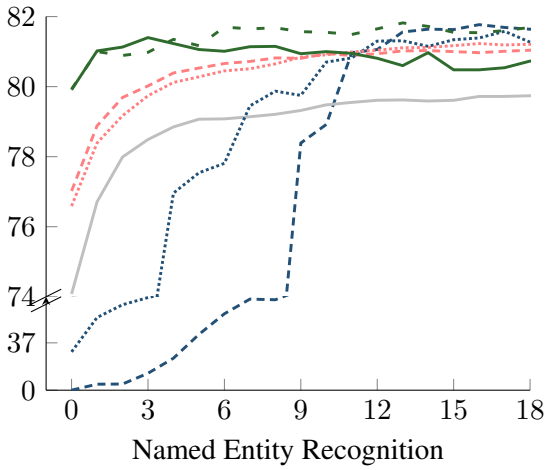
For each method we tuned hyperparameters by considering a grid of values and measuring dev set performance over five training iterations, except for constituency parsing, where we took five measurements, 4k instances apart. For the cutting plane methods we cached constraints in memory to save time, but the memory cost was too great to run batch cutting plane on constituency parsing (over 60 Gb), and so is not included in the results.

4 Observations

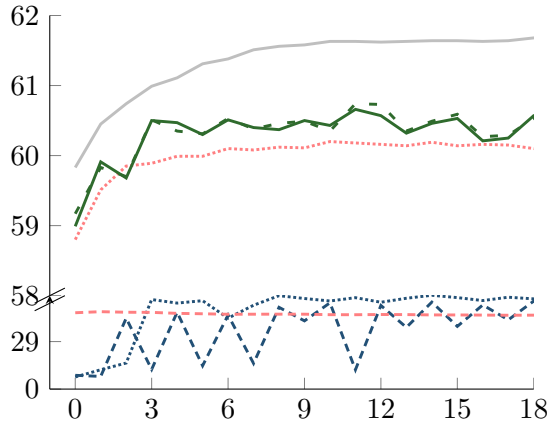
From the results in Figure 1 and during tuning, we can make several observations about these optimization methods' performance on these tasks.

Observation 1: Margin methods generally perform best As expected given prior work, margin methods equal or surpass the performance of likelihood and perceptron methods across almost all of these tasks. Coreference resolution is an exception, but that model has latent variables that likelihood may treat more effectively,

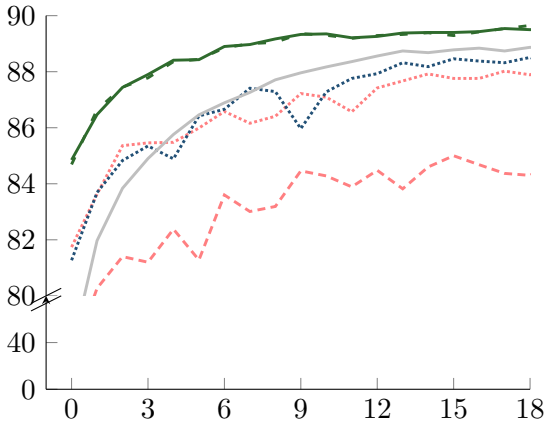
[2] Our results are slightly lower as we save time by only using the dense features and a reduced n-gram context.



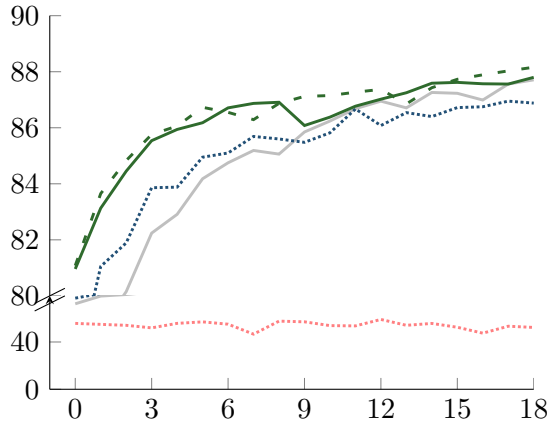
Named Entity Recognition



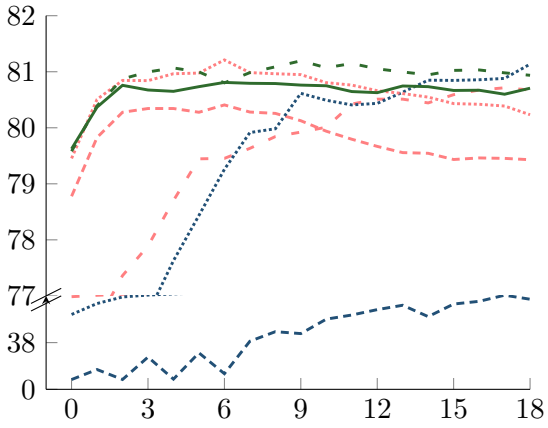
Coreference Resolution



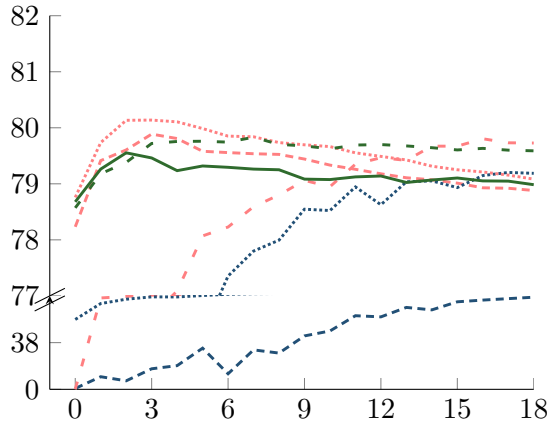
Constituency Parsing



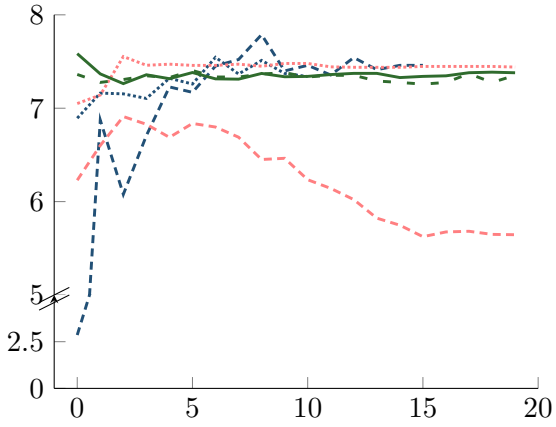
Constituency Parsing, Neural CRF



Dependency Parsing, Projective Decoder



Dependency Parsing, Non-Proj. Decoder



Summarization

Margin	Cutting Plane
	Online Cutting Plane
	Online Primal Subgradient & L_1
	Online Primal Subgradient & L_2
Mistake Driven	Averaged Perceptron
	MIRA
	Averaged MIRA (MST built-in)
Llhood	Stochastic Gradient Descent

Figure 1: Variation in dev set performance (y) across training iterations (x). To show all variation, the scale of the y-axis changes partway, as indicated. Lines that stop early had converged.

	Time per iteration relative to averaged perceptron						
Method	NER	Coref	Span Parser	Neural Parser	MST Proj.	MST Non-Proj.	Summ.
AP	1.0	1.0	1.0	-	1.0	1.0	1.0
MIRA	1.9	1.0	1.0	1.0	1.0	1.0	1.0
CP	60.8	2.7	-	-	6.8	8.4	0.6
OCP	2.7	1.7	0.9	0.9	1.5	1.6	1.1
OPS	3.9	1.3	1.1	1.0	1.8	2.0	0.9
Decoding	0.6	0.2	0.9	0.7	0.7	0.6	0.7

Table 1: Comparison of time per iteration relative to the perceptron (or MIRA for the Neural Parser). Decoding shows the time spent on inference. Times were averaged across the entire run. OPS uses batch size 10 for NER to save time, but performs just as well as with batch size 1 in Figure 1.

and has a weighted loss function tuned for likelihood (softmax-margin).

Observation 2: Dual cutting plane methods appear to learn more slowly Both cutting plane methods took more iterations to reach peak performance than the other methods. In addition, for batch cutting plane, accuracy varied so drastically that we extended tuning to ten iterations, and even then choosing the best parameters was sometimes difficult. Table 1 shows that the online cutting plane method did take slightly less time per iteration than OPS, but not enough to compensate for the slower learning rate.

Observation 3: Learning with real-valued features is difficult for perceptron methods Learning models for tasks such as NER, which are driven by sparse indicator features, often roughly amounts to tallying the features that are contrastively present in correct hypotheses. In such cases, most learning methods work fairly well. However, when models use real-valued features, learning may involve determining a more delicate balance between features. In the models we consider that have real-valued features, summarization and parsing with a neural model, we can see that perceptron methods indeed have difficulty.[3]

Observation 4: Online Primal Subgradient is robust and effective All of the margin based methods, and gradient descent on likelihood, require tuning of a regularization constant and a step size (or convergence requirements for SMO). The dual methods were particularly sensitive to these hyperparameters, performing poorly if they were not chosen carefully. In contrast, performance for the primal methods remained high over a broad range of values.

Our implementation of sparse updates for AdaGrad was crucial for high-speed performance, decreasing time by an order of magnitude on tasks with many sparse features, such as NER and dependency parsing.

Observation 5: Other minor properties We found that varying the batch size did not substantially impact performance after a given number of decodes, but did enable a speed improvement as decoding of multiple instances can occur in parallel. Increasing batch sizes leads to a further improvement to OPS, as overall there are fewer updates per iteration. For some tasks, re-tuning the step size was necessary when changing batch size.

5 Conclusion

The effectiveness of max-margin optimization methods is widely known, but the default choice of learning algorithm in NLP is often a form of the perceptron (or likelihood) instead. Our results illustrate some of the pitfalls of perceptron methods and suggest that online optimization of the max-margin objective via primal subgradients is a simple, well-behaved alternative.

6 Acknowledgments

We would like to thank Greg Durrett for assistance running his code, Adam Pauls for advice on dual methods, and the anonymous reviewers for their helpful suggestions. This work was supported by National Science Foundation grant CNS-1237265, Office of Naval Research MURI grant N000140911081, and a General Sir John Monash Fellowship to the first author. Opinions, findings, conclusions and recommendations expressed in this material are those of the authors and do not necessarily reflect the views of sponsors.

[3]For the neural parser, the perceptron took a gradient step for each mistake, but this had dismal performance.

References

Željko Agić, Maria Jesus Aranzabe, Aitziber Atutxa, Cristina Bosco, Jinho Choi, Marie-Catherine de Marneffe, Timothy Dozat, Richárd Farkas, Jennifer Foster, Filip Ginter, Iakes Goenaga, Koldo Gojenola, Yoav Goldberg, Jan Hajič, Anders Trærup Johannsen, Jenna Kanerva, Juha Kuokkala, Veronika Laippala, Alessandro Lenci, Krister Lindén, Nikola Ljubešić, Teresa Lynn, Christopher Manning, Héctor Alonso Martínez, Ryan McDonald, Anna Missilä, Simonetta Montemagni, Joakim Nivre, Hanna Nurmi, Petya Osenova, Slav Petrov, Jussi Piitulainen, Barbara Plank, Prokopis Prokopidis, Sampo Pyysalo, Wolfgang Seeker, Mojgan Seraji, Natalia Silveira, Maria Simi, Kiril Simov, Aaron Smith, Reut Tsarfaty, Veronika Vincze, and Daniel Zeman. 2015. Universal dependencies 1.1.

Taylor Berg-Kirkpatrick, Dan Gillick, and Dan Klein. 2011. Jointly learning to extract and compress. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 481–490, Portland, Oregon, USA, June.

Ming-Wei Chang and Wen-Tau Yih. 2013. Dual coordinate descent algorithms for efficient large margin structured prediction. *Transactions of the Association for Computational Linguistics*, 1:207–218.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine N-best parsing and MaxEnt discriminative reranking. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 173–180, June.

David Chiang. 2012. Hope and fear for discriminative training of statistical translation models. *Journal of Machine Learning Research*, 13(1):1159–1187, April.

Yoeng-jin Chu and Tseng-hong Liu. 1965. On the shortest arborescence of a directed graph. *Science Sinica*, pages 1396–1400.

Michael Collins, Amir Globerson, Terry Koo, Xavier Carreras, and Peter L. Bartlett. 2008. Exponentiated gradient algorithms for conditional random fields and max-margin markov networks. *Journal of Machine Learning Research*, 9:1775–1822, June.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing - Volume 10*, pages 1–8.

Koby Crammer and Yoram Singer. 2003. Ultraconservative online algorithms for multiclass problems. *Journal of Machine Learning Research*, 3:951–991, March.

Hoa Trang Dang and Karolina Owczarzak. 2008. Overview of the TAC 2008 update summarization task. In *Text Analysis Conference*.

Hoa Trang Dang and Karolina Owczarzak. 2009. Overview of the TAC 2009 summarization track. In *Text Analysis Conference*.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159, July.

Greg Durrett and Dan Klein. 2013. Easy victories and uphill battles in coreference resolution. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1971–1982, Seattle, Washington, USA, October.

Greg Durrett and Dan Klein. 2014. A joint model for entity analysis: Coreference, typing, and linking. volume 2, pages 477–490.

Greg Durrett and Dan Klein. 2015. Neural CRF parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 302–312, Beijing, China, July.

Chris Dyer. 2013. Notes on AdaGrad. Technical report, Carnegie Mellon University, June.

Jack Edmonds. 1967. Optimum branchings. *Journal of Research of the National Bureau of Standards*, 71B:233–240.

Jason Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of the 16th Conference on Computational Linguistics - Volume 1*, pages 340–345.

Yoav Freund and Robert E. Schapire. 1999. Large margin classification using the perceptron algorithm. *Machine Learning*, 37(3):277–296, December.

Kevin Gimpel and Noah A. Smith. 2010. Softmax-margin CRFs: Training log-linear models with cost functions. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 733–736.

David Hall, Greg Durrett, and Dan Klein. 2014. Less grammar, more features. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 228–237, Baltimore, Maryland, USA, June.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 282–289.

D. C. Liu and J. Nocedal. 1989. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45(3):503–528, December.

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

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005a. Online large-margin training of dependency parsers. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 91–98, Ann Arbor, Michigan, USA, June.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajic. 2005b. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 523–530, Vancouver, British Columbia, Canada, October.

John C. Platt. 1999. Fast training of support vector machines using sequential minimal optimization. In Bernhard Schölkopf, Christopher J. C. Burges, and Alexander J. Smola, editors, *Advances in Kernel Methods*, pages 185–208. MIT Press.

Sameer Pradhan, Lance Ramshaw, Ralph Weischedel, Jessica MacBride, and Linnea Micciulla. 2007. Unrestricted coreference: Identifying entities and events in OntoNotes. In *Proceedings of the International Conference on Semantic Computing*, pages 446–453, September.

Nathan Ratliff, J. Andrew (Drew) Bagnell, and Martin Zinkevich. 2007. (Online) subgradient methods for structured prediction. In *Eleventh International Conference on Artificial Intelligence and Statistics (AIStats)*, March.

Noah A. Smith. 2011. *Linguistic Structure Prediction*. Synthesis Lectures on Human Language Technologies. Morgan and Claypool.

Ben Taskar, Dan Klein, Michael Collins, Daphne Koller, and Chris Manning. 2004. Max-margin parsing. In *Proceedings of EMNLP 2004*, pages 1–8, Barcelona, Spain, July.

Ben Taskar. 2004. *Learning Structured Prediction Models: A Large Margin Approach*. Ph.D. thesis, Stanford University.

Ioannis Tsochantaridis, Thomas Hofmann, Thorsten Joachims, and Yasemin Altun. 2004. Support vector machine learning for interdependent and structured output spaces. In *Proceedings of the Twenty-first International Conference on Machine Learning*, pages 104–112, July.

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

Learning Better Embeddings for Rare Words Using Distributional Representations

Irina Sergienya and **Hinrich Schütze**
Center for Information and Language Processing
University of Munich, Germany
sergienya@cis.lmu.de

Abstract

There are two main types of word representations: low-dimensional embeddings and high-dimensional distributional vectors, in which each dimension corresponds to a context word. In this paper, we initialize an embedding-learning model with distributional vectors. Evaluation on word similarity shows that this initialization significantly increases the quality of embeddings for rare words.

1 Introduction

Standard neural network (NN) architectures for inducing embeddings have an input layer that represents each word as a *one-hot vector* (e.g., Turian et al. (2010), Collobert et al. (2011), Mikolov et al. (2013)). There is no usable information available in this input-layer representation except for the identity of the word. We call this standard initialization method *one-hot initialization*.

Distributional representations (e.g., Schütze (1992), Lund and Burgess (1996), Sahlgren (2008), Turney and Pantel (2010), Baroni and Lenci (2010)) represent a word as a high-dimensional vector in which each dimension corresponds to a context word. They have been successfully used for a wide variety of tasks in natural language processing such as phrase similarity (Mitchell and Lapata, 2010) and sentiment analysis (Turney and Littman, 2003).

In this paper, we investigate *distributional initialization*: the use of distributional vectors as representations of words at the input layer of NN architectures for embedding learning to improve the embeddings of rare words. It is difficult for one-hot initialization to learn good embeddings from only a few examples. In contrast, distributional initialization provides an additional source of information – the global distribution of the word in the corpus – that improves embeddings learned for rare words. We will demonstrate this type of improvement in the experiments reported below.

In summary, we introduce the idea of distributional initialization for embedding learning, an alternative to one-hot initialization that combines distributed representations (or embeddings) with distributional representations (or high-dimensional vectors). We show that distributional initialization significantly improves the quality of embeddings learned for rare words.

We will first describe our methods in Section 2 and the experimental setup in Section 3. Section 4 presents and discusses experimental results. We summarize related work in Section 5 and finish with conclusion in Section 6 and discussion of future work in Section 7.

2 Method

Weighting. We use two different weighting schemes for distributional vectors. Let $v_1, \ldots, v_n$ be the vocabulary of context words. In BINARY weighting, entry $1 \leq i \leq n$ in the distributional vector of target word w is set to 1 iff v_i and w cooccur at a distance of at most ten words in the corpus and to 0 otherwise.

In PPMI weighting, entry $1 \leq i \leq n$ in the distributional vector of target word w is set to the PPMI (positive pointwise mutual information, introduced by Niwa and Nitta (1994)) of w and v_i. We divide PPMI values by their maximum to ensure they are in $[0, 1]$ because we will combine one-hot vectors (whose values are 0/1) with PPMI weights and it is important that they are on the same scale.

We use two different **distributional initializations**, shown in Figure 1: *separate* (left) and *mixed* (right). Combinations of these two initializations with both BINARY and PPMI weighting will be investigated in the experiments.

Recall that n is the dimensionality of the distri-

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 280–285,
Lisbon, Portugal, 17-21 September 2015.

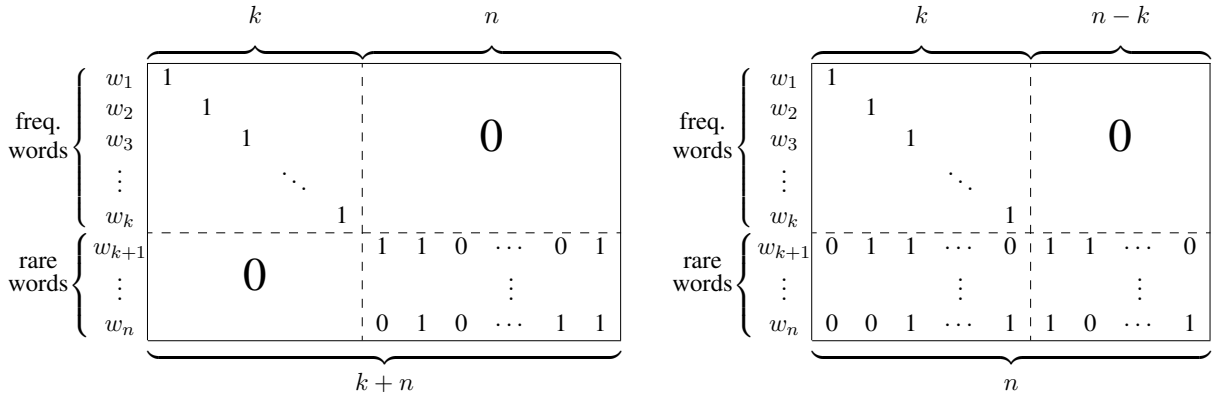


Figure 1: One-hot vectors of frequent words and distributional vectors of rare words are separate in *separate initialization* (left) and overlap in *mixed initialization* (right). This example is for BINARY weighting.

butional vectors. Let k be the number of words with frequency $> \theta$, where the frequency threshold θ is a parameter.

In *separate* initialization, the input representation for a word is the concatenation of a k-dimensional vector and an n-dimensional vector. For a word with frequency $> \theta$, the k-dimensional vector is a one-hot vector and the n-dimensional vector is zero. For a word with frequency $\leq \theta$, the k-dimensional vector is zero and the n-dimensional vector is its distributional vector.

In *mixed* initialization, the input representation for a word is an n-dimensional vector: a one-hot vector for a word with frequency $> \theta$ and a distributional vector for a word with frequency $\leq \theta$.

In summary, separate initialization uses separate representation spaces for frequent words (one-hot space) and rare words (distributional space). Mixed initialization uses the same representation space for all words; and rare words share weights with the frequent words that they cooccur with.

3 Experimental setup

We use ukWaC+WaCkypedia (Baroni et al., 2009), a corpus of 2.4 billion tokens and 6 million word types. Based on (Turian et al., 2010), we preprocess the corpus by removing sentences that are less than 90% lowercase; lowercasing; replacing URLs, email addresses and digits by special tokens; tokenization (Schmid, 2000); replacing words with frequency 1 with <unk>; and adding end-of-sentence tokens. After preprocessing, the size n of the context word vocabulary is 2.7 million.

We evaluate on six word similarity judgment data sets (number of pairs in parentheses): **RG** (Rubenstein and Goodenough (1965), 65) **MC** (Miller and Charles (1991), 30), **MEN**[1] (Bruni et al. (2012), 3000), **W**ord**S**im353[2] (Finkelstein et al. (2001), 353), Stanford **R**are **W**ord[3] (Luong et al. (2013), 2034) and **S**im**L**ex-999[4] (Hill et al. (2014), 999). We exclude from the evaluation the 16 pairs in RW that contain a word that does not occur in our corpus.

Our goal in this paper is to investigate the effect of using distributional initialization vs. one-hot initialization on the quality of embeddings of rare words.

However, except for RW, the six data sets contain only a single word with frequency $\leq$100, all other words are more frequent.

To address this issue, we artificially make all words in the six data sets rare. We do this by keeping only θ randomly chosen occurrences in the corpus (for words with frequency $>\theta$) and replacing all other occurrences with a different token (e.g., "fire" is replaced with "*fire*"). This procedure – *corpus downsampling* – ensures that all words in the six data sets are rare in the corpus and that our setup directly evaluates the impact of distributional initialization on rare words.

Note that we use θ for two different purposes: (i) θ is the frequency threshold that determines which words are classified as rare and which as frequent in Figure 1 – changing θ corresponds to moving the horizontal dashed line in separate and mixed initialization up and down; (ii) θ is the parameter that determines how many occurrences of a word are left in the corpus when we remove oc-

[1] clic.cimec.unitn.it/~elia.bruni/MEN
[2] alfonseca.org/eng/research/wordsim353.html
[3] www-nlp.stanford.edu/~lmthang/morphoNLM/
[4] cl.cam.ac.uk/~fh295/simlex.html

			A	B	C	D	E	F	G	H	I	J	K	L
			RG		MC		MEN		WS		RW		SL	
		θ	mixed	sep	mixed	sep	mixed	sep	mixed	sep	mixed	sep	mixed	sep
1	BINARY	10	***56.54**	**47.06**	**35.96**	**32.10**	***43.76**	***45.56**	34.21	***40.93**	***24.81**	**20.85**	***18.30**	***13.76**
2		20	***59.08**	**45.31**	***46.66**	**35.22**	**52.05**	***52.38**	41.44	**47.53**	***29.48**	**26.93**	***20.85**	***16.86**
3		50	***63.20**	51.07	***52.35**	37.45	58.21	53.80	43.14	44.88	31.32	29.16	***24.19**	***22.45**
4		100	**68.33**	52.50	**61.70**	35.94	61.69	55.23	48.25	44.89	33.29	30.22	***26.74**	**24.66**
5	PPMI	10	***56.87**	***51.94**	***37.31**	***46.52**	***48.05**	***50.49**	**38.41**	***47.54**	***25.53**	**23.12**	***19.70**	***15.59**
6		20	***59.08**	***50.32**	***47.51**	***45.17**	***54.88**	***56.42**	43.31	***53.19**	***29.78**	***28.51**	***21.84**	***19.23**
7		50	***64.90**	***64.36**	***55.27**	***56.75**	**60.51**	**61.04**	45.76	**55.55**	32.05	30.25	***25.11**	***21.60**
8		100	**71.08**	58.37	**68.14**	52.33	63.05	60.74	48.66	55.49	33.25	30.49	***27.13**	22.60
9	one-hot	10	38.93		16.67		40.70		35.17		20.69		8.97	
10		20	42.17		25.21		50.21		43.74		26.58		13.62	
11		50	56.01		42.35		60.22		54.10		32.16		20.01	
12		100	67.47		61.33		65.14		59.87		35.19		24.06	

Table 1: Spearman correlation coefficients $\times 100$ between human and embedding-based similarity judgments, averaged over 5 runs. Distributional initialization correlations that are higher (resp. significantly higher) than corresponding one-hot correlations are set in **bold** (resp. marked *).

currences to ensure that words from the evaluation data sets are rare in the corpus.

We covary these two parameters in the experiments below; e.g., we apply distributional initialization with $\theta = 20$ to a corpus constructed to have $\theta = 20$ occurrences of words from similarity data sets. We do this to ensure that all evaluation words are rare words for the purpose of distributional initialization and so we can exploit all pairs in the evaluation data sets for evaluating the efficacy of our method for rare words.

We modified word2vec[5] (Mikolov et al., 2013) to accommodate distributional initialization; to support distributional vectors at the input layer, we changed the implementation of activation functions and backpropagation. We use the skipgram model, hierarchical softmax, set the size of the context window to 10 (10 words to the left and 10 to the right), min-count to 1 (train on all tokens), embedding size to 100, sampling rate to 10^{-3} and train models for one epoch.

For four values of the frequency threshold, $\theta \in \{10, 20, 50, 100\}$,[6] we train word2vec models with one-hot initialization and with the four combinations of weighting (BINARY, PPMI) and distributional initialization (mixed, separate), a total of $4 \times (1 + 2 \times 2) = 20$ models. For each training run, we perform corpus downsampling and initialize the parameters of the models randomly. To get a reliable assessment of performance, we train 5 instances of each model and report averages of the 5 runs. One model takes $\sim$3 hours to train on 23 CPU cores, 2.30GHz.

4 Experimental results and discussion

Table 1 shows experimental results, averaged over 5 runs. The evaluation measure is Spearman correlation $\times 100$ between human and machine-generated pair similarity judgments.

Frequency threshold θ. The main result is that for $\theta \in \{10, 20\}$ *distributional initialization is better than one-hot initialization* (see bold numbers): compare lines 1&5 with line 9; and lines 2&6 with line 10. This is true for both mixed and separate initialization, with the exception of WS, for which mixed (column G) is better in only 1 (line 5) of 4 cases.

Looking only at results for $\theta \in \{10, 20\}$, 18 of 24 improvements are significant[7] for mixed initialization and 16 of 24 improvements are significant for separate initialization (lines 1&5 vs 9 and lines

[5] code.google.com/p/word2vec

[6] A reviewer asks whether the value of θ should depend on the size of the training corpus. Our intuition is that it is independent of corpus size. If a certain amount of information – corresponding to a certain number of contexts – is required to learn a meaningful representation of a word, then it should not matter whether that given number of contexts occurs in a small corpus or in a large corpus. However, if the contexts themselves contain many rare words (which is more likely in a small corpus), then corpus size could be an important variable to take into account.

[7] Two-sample t-test, two-tailed, assuming equal variance, $p < .05$

2&6 vs 10).

For $\theta \in \{50, 100\}$, mixed initialization does well for RG, MC and SL, but the gap between mixed and one-hot initializations is generally smaller for these larger values of θ; e.g., the difference is larger than 9 for $\theta = 10$ (A1&A5 vs A/B9, C1&C5 vs C/D9, K1&K5 vs K/L9) and less than 9 for $\theta = 100$ (A4&A8 vs A/B12, C4&C8 vs C/D12, K4&K8 vs K/L12) for these three data sets.

Recall that each value of θ effectively results in a different training corpus – a training corpus in which the number of occurrences of the words in the evaluation data sets has been reduced to $\leq \theta$ (cf. Section 3).

Our results indicate that distributional initialization is beneficial for very rare words – those that occur no more than 20 times in the corpus. Our results for medium rare words – those that occur between 50 and 100 times – are less clear: either there are no improvements or improvements are small.

Thus, our recommendation is to use $\theta = 20$.

Scalability. The time complexity of the basic version of word2vec is $O(ECWD \log V)$ (Mikolov et al., 2013) where E is the number of epochs, C is the corpus size, W is the context window size, D is the number of dimensions of the embedding space, and V is the vocabulary size. Distributional initialization adds a term I, the average number of entries in the distributional vectors, so that time complexity increases to $O(IECWD \log V)$. For rare words, I is small, so that there is no big difference in efficiency between one-hot initialization and distributional initialization of word2vec. However, for frequent words I would be large, so that distributional initialization may not be scalable in that case. So even if our experiments had shown that distributional initialization helps for both rare and frequent words, scalability would be an argument for only using it for rare words.

Binary vs. PPMI. PPMI weighting is almost always better than BINARY, with three exceptions (I8, L7, L8) where the difference between the two is small and not significant. The probable explanation is that the PPMI weights in $[0, 1]$ convey detailed, graded information about the strength of association between two words, taking into account their base frequencies. In contrast, the BINARY weights in $\{0, 1\}$ only indicate if there was any instance of cooccurrence at all – without considering frequency of cooccurrence and without normalizing for base frequencies.

Mixed vs. Separate. Mixed initialization is less variable and more predictable than separate initialization: performance for mixed initialization always goes up as θ increases, e.g., 56.54 $\rightarrow$ 59.08 $\rightarrow$ 63.20 $\rightarrow$ 68.33 (column A, lines 1–4). In contrast, separate initialization performance often decreases, e.g., from 47.06 to 45.31 (column B, lines 1–2) when θ is increased. Since more information (more occurrences of the words that similarity judgments are computed for) should generally not have a negative effect on performance, the only explanation is that separate is more variable than mixed and that this variability sometimes results in decreased performance. Figure 1 explains this difference between the two initializations: in mixed initialization (right panel), rare words are tied to frequent words, so their representations are smoothed by representations learned for frequent words. In separate initialization (left panel), no such links to frequent words exist, resulting in higher variability.

Because of its lower variability, our experiments suggest that mixed initialiation is a better choice than separate initialization.

One-hot vs. Distributional initialization. Our experiments show that distributional representation is helpful for rare words. It is difficult for one-hot initialization to learn good embeddings for such words, based on only a small number of contexts in the corpus. In such cases, distributional initialization makes the learning task easier since in addition to the *contexts of the rare word*, the learner now also has access to the *global distribution of the rare word* and can take advantage of weight sharing with other words that have similar distributional representations to smooth embeddings systematically.

Thus, distributional initialization is a form of smoothing: the embedding of a rare word is tied to the embeddings of other words via the links shown in Figure 1: the 1s in the lower "rare words" part of the illustrations for separate and mixed initialization. As is true for smoothing in general, parameter estimates for frequent events benefit less from smoothing or can even deteriorate. In contrast, smoothing is essential for rare events. Where the boundary lies between rare and frequent events depends on the specifics of the problem and the

smoothing method used and is usually an empirical question. Our results indicate that that boundary lies somewhere between 20 and 50 in our setting.[8]

Variance of results. Table 1 shows averages of five runs. The variance of results was quite high for low-performing models. For higher performing models – those with values ≥ 40 – the ratio of standard deviation divided by mean ranged from .005 to .29. The median was .044. While the variance from run to run is quite high for low-performing models and for a few high-performing models, the significance test takes this into account, so that the relatively high variability does not undermine our results.

In summary, we have shown that distributional initialization improves the quality of word embeddings for rare words. Our recommendation is to use mixed initialization with PPMI weighting and the value $\theta = 20$ of the frequency threshold.

5 Related work

An alternative to using distributional information for initialization is to use syntactic and semantic information for initialization. Approaches along these lines include Botha and Blunsom (2014) who represent a word as a sum of embedding vectors of its morphemes. Cui et al. (2014) use a weighted average of vectors of morphologically similar words. Bian et al. (2014) extend a word's vector with vectors of entity categories and POS tags. This line of work also is partially motivated by improving the embeddings of rare words. Distributional information on the one hand and syntactic/semantic information on the other hand are likely to be complementary, so that a combination of our approach with this prior work is promising.

Le et al. (2010) propose three schemes to address word embedding initialization. *Reinitialization* and *iterative reinitialization* use vectors from prediction space to initialize the context space during training. This approach is both more complex and less efficient than ours. *One-vector initialization* initializes all word embeddings with the same random vector to keep rare words close to each other. This approach is also less efficient than ours since the initial embedding is much denser than in our approach.

6 Conclusion

We have introduced distributional initialization of neural network architectures for learning better embeddings for rare words. Experimental results on a word similarity judgment task demonstrate that embeddings of rare words learned with distributional initialization perform better than embeddings learned with traditional one-hot initialization.

7 Future work

Our work is the first exploration of the utility of distributional representations as initialization for embedding learning algorithms like word2vec. There are a number of research questions we would like to investigate in the future.

First, we showed that distributional representation is beneficial for words with very low frequency. It was not beneficial in our experiments for more frequent words. A more extensive analysis of the factors that are responsible for the positive effect of distributional representation is in order.

Second, to simplify our experimental setup and make the number of runs mangeable, we used the parameter θ both for corpus processing (only θ occurrences of a particular word were left in the corpus) and as the separator between rare words that are distributionally initialized and frequent words that are not. It remains to be investigated whether there are interactions between these two properties of our model, e.g., a high rare-frequent separator may work well for words whose corpus frequency is much smaller than the separator.

Third, while we have shown that distributional initialization improves the quality of representations of rare words, we did not investigate whether distributional initialization for rare words has any adverse effect on the quality of representations of frequent words for which one-hot initialization is applied. Since rare and frequent words are linked in the mixed model, this possibility cannot be dismissed and we plan to investigate it in future work.

Acknowledgments. This work was supported by Deutsche Forschungsgemeinschaft (grant DFG SCHU 2246/10-1, FADeBaC).

[8] A reviewer asks: "If a word is rare, its distributional vector should also be sparse and less informative, which does not guarantee to be a good starting point." This is true and it suggests that it may not be possible to learn a very high-quality representation for a rare word. But this it not our goal. Our goal is simply to learn a *better* representation than the one that is learned by standard word2vec. Our explanation for our positive experimental results is that distributional initialization implements a form of smoothing.

References

Marco Baroni and Alessandro Lenci. 2010. Distributional memory: A general framework for corpus-based semantics. *Computational Linguistics*, 36(4):673–721.

Marco Baroni, Silvia Bernardini, Adriano Ferraresi, and Eros Zanchetta. 2009. The WaCky Wide Web: A Collection of Very Large Linguistically Processed Web-Crawled Corpora. *Language Resources and Evaluation*, 43(3):209–226.

Jiang Bian, Bin Gao, and Tie-Yan Liu. 2014. Knowledge-powered deep learning for word embedding. In *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD*, pages 132–148.

Jan A. Botha and Phil Blunsom. 2014. Compositional morphology for word representations and language modelling. In *Proceedings of the 31st International Conference on Machine Learning (ICML)*, Beijing, China, June.

Elia Bruni, Gemma Boleda, Marco Baroni, and Nam Khanh Tran. 2012. Distributional semantics in technicolor. In *ACL*, pages 136–145.

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

Qing Cui, Bin Gao, Jiang Bian, Siyu Qiu, and Tie-Yan Liu. 2014. Knet: A general framework for learning word embedding using morphological knowledge. *Preprint pubslished on arXiv* `arXiv:1407.1687 [cs.CL]`.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2001. Placing search in context: The concept revisited. In *WWW*, pages 406–414.

Felix Hill, Roi Reichart, and Anna Korhonen. 2014. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Preprint pubslished on arXiv* `arXiv:1408:3456 [cs.CL]`.

Hai Son Le, Alexandre Allauzen, Guillaume Wisniewski, and François Yvon. 2010. Training continuous space language models: Some practical issues. In *EMNLP*, pages 778–788.

Kevin Lund and Curt Burgess. 1996. Producing high-dimensional semantic spaces from lexical co-occurrence. *Behavior Research Methods, Instruments, & Computers*, 28(2):203–208.

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

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

George A. Miller and Walter G. Charles. 1991. Contextual correlates of semantic similarity. *Language & Cognitive Processes*, 6(1):1–28.

Jeff Mitchell and Mirella Lapata. 2010. Composition in distributional models of semantics. *Cognitive Science*, 34(8):1388–1439.

Yoshiki Niwa and Yoshihiko Nitta. 1994. Co-occurrence vectors from corpora vs. distance vectors from dictionaries. In *COLING*, volume 1, pages 304–309.

Herbert Rubenstein and John B. Goodenough. 1965. Contextual correlates of synonymy. *Commun. ACM*, 8(10):627–633, October.

Magnus Sahlgren. 2008. The distributional hypothesis. *Rivista di Linguistica (Italian Journal of Linguistics)*, 20(1):33–53.

Helmut Schmid. 2000. Unsupervised Learning of Period Disambiguation for Tokenisation. Technical report, IMS, University of Stuttgart.

Hinrich Schütze. 1992. Dimensions of Meaning. In *ACM/IEEE Conference on Supercomputing*, pages 787–796.

Joseph Turian, Lev-Arie Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *ACL*, pages 384–394.

Peter D. Turney and Michael L. Littman. 2003. Measuring praise and criticism: Inference of semantic orientation from association. *ACM TOIS*, 21(4):315–346.

Peter D. Turney and Patrick Pantel. 2010. From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research (JAIR)*, 37:141–188.

Composing Relationships with Translations

Alberto García-Durán
Sorbonne universités, UTC
CNRS, Heudiasyc 7253
60203 Compiègne, France
agarciad@utc.fr

Antoine Bordes
Facebook AI Research
770 Broadway
New York, NY. USA
abordes@fb.com

Nicolas Usunier
Facebook AI Research
112, avenue de Wagram
75017 Paris, France
usunier@fb.com

Abstract

Performing link prediction in Knowledge Bases (KBs) with embedding-based models, like with the model TransE (Bordes et al., 2013) which represents relationships as translations in the embedding space, have shown promising results in recent years. Most of these works are focused on modeling single relationships and hence do not take full advantage of the graph structure of KBs. In this paper, we propose an extension of TransE that learns to explicitly model composition of relationships via the addition of their corresponding translation vectors. We show empirically that this allows to improve performance for predicting single relationships as well as compositions of pairs of them.

1 Introduction

Performing link prediction on multi-relational data is becoming essential in order to complete the huge amount of missing information of the knowledge bases. These knowledge can be formalized as directed multi-relation graphs, whose node correspond to entities connected with edges encoding various kind of relationships. We denote these connections via triples *(head, label, tail)*. Link prediction consists in filling in incomplete triples like *(head, label, ?)* or *(?, label, tail)*.

In this context, embedding models (Wang et al., 2014; Lin et al., 2015; Jenatton et al., 2012; Socher et al., 2013) that attempt to learn low-dimensional vector or matrix representations of entities and relationships have shown promising performance in recent years. In particular, the basic model TRANSE (Bordes et al., 2013) has been proved to be very powerful. This model treats each relationship as a translation vector operating on the embedding representing the entities. Hence, for a triple *(head, label, tail)*, the vector embeddings of *head* and *tail* are learned so that they are connected through a translation parameterized by the vector associated with *label*. Many extensions have been proposed to improve the representation power of TRANSE while still keeping its simplicity, by adding some projections steps before the translation (Wang et al., 2014; Lin et al., 2015).

In this paper, we propose an extension of TRANSE [1] that focuses on improving its representation of the underlying graph of multi-relational data by trying to learn compositions of relationships as sequences of translations in the embedding space. The idea is to train the embeddings by learning simple reasonings, such as the relationship *people/nationality* should give a similar result as the composition *people/city_of_birth* and *city/country*. In our approach, called RTRANSE, the training set is augmented with relevant examples of such compositions by performing constrained walks in the knowledge graph, and training so that sequences of translations lead to the desired result. The idea of compositionality to model multi-relational data was previously introduced in (Neelakantan et al., 2015). That work composes relationships by means of recurrent neural networks (RNN) (one per relationship) with non-linearities. However, we show that there is a natural way to compose relationships by simply adding translation vectors and not requiring additional parameters, which makes it specially appealing because of its scalability.

We present experimental results that show the superiority of RTRANSE over TRANSE in terms of link prediction. A detailed evaluation, in which test examples are classified as *easy* or *hard* depending on their similarity with training data, highlights the improvement of RTRANSE on both categories. Our experiments include a new evaluation protocol, in which the model is directly

[1] Code available in https://github.com/glorotxa/SME

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 286–290,
Lisbon, Portugal, 17-21 September 2015.

asked to answer questions related to compositions of relations, such as *(head, label$_1$, label$_2$, ?)*. RTRANSE also achieves significantly better performances than TRANSE on this new dataset.

We describe RTRANSE in the next section, and present our experiments in Section 3.

2 Model

The model we propose is inspired by TRANSE (Bordes et al., 2013). In TRANSE, entities and relationships of a KB are mapped to low dimensional vectors, called embeddings. These embeddings are learnt so that for each fact (h, ℓ, t) in the KB, we have $h + \ell \approx t$ in the embedding space.

Using translations for relationships naturally leads to embed the composition of two relationships as the sum of their embeddings: on a path $(h, \ell, t), (t, \ell', t')$, we should have $h+\ell+\ell' \approx t'$ in the embedding space. The original TRANSE does not enforce that the embeddings accurately reproduce such compositions. The *recurrent* TRANSE we propose here has a modified training stage to include such compositions. This should allow to model simple reasonings in the KB, such as *people/nationality* is similar to the composition of *people/city_of_birth* and *city/country*.

2.1 Recurrent TransE

We describe in this section our model in its full generality, which allows to deal with compositions of an arbitrary number of relationships, even though in this first work we experimented only with compositions of two relationships.

Triples that are the result of a compositions are denoted by $(h, \{\ell_i\}_{i=1}^p, t)$, where p is the number of relationships that are composed to go from h to t. Such a path means that there exist entities $e_1, ..., e_{p+1}$, with $e_1 = h$ and $e_{p+1} = t$ such that for all k, (e_k, ℓ_k, e_{k+1}) is a fact in the KB. Our model, RTRANSE for *recurrent* TRANSE, represents each step $s_k(h, \{\ell_i\}_{i=1}^p, t)$ along the path in the KB with the recurrence relationship (boldface characters denote embedding vectors i.e. $\mathbf{h}$ is the embedding vector of the entity h):

$$\begin{aligned} \boldsymbol{s}_1(h, \{\ell_i\}_{i=1}^p, t) &= \boldsymbol{h} \\ \boldsymbol{s}_{k+1}(h, \{\ell_i\}_{i=1}^p, t) &= \boldsymbol{s}_k(h, \{\ell_i\}_{i=1}^p, t) + \boldsymbol{\ell}_k \,. \end{aligned}$$

Then, the energy of a triple is computed as

$$d(h, \{\ell_i\}_{i=1}^p, t) = ||\boldsymbol{s}_p(h, \{\ell_i\}_{i=1}^p, t) - \boldsymbol{t}||_2 \,.$$

2.2 Path construction and filtering

The experience of the paper is motivated by learning simple reasonings in the KB through the compositions of relationships. Therefore, we restricted our analysis to paths of length 2 created as follows.

First, for each fact (h, ℓ, t), retrieve all paths $(h, \{\ell_1, \ell_2\}, t)$ such that there is e such that both (h, ℓ_1, e) and (e, ℓ_2, t) are in the KB. Then, we filter out paths where $(h, \ell_1, e) = (h, \ell, t)$ or $(e, \ell_2, t) = (h, \ell, t)$, as well as the paths with $\ell_1 = \ell_2$ and $h = e = t$.

We focused on "unambiguous" paths, so that the reasoning might actually make sense. In particular, we considered only paths where ℓ_1 is either a 1-to-1 or a 1-to-many relationship, and where ℓ_2 is either a 1-to-1 or a many-to-1 relationship. In our experiments, the paths created for training only consider the training subset of facts.

In the remainder of the paper, such paths of length 2 are called *quadruples*.

2.3 Training and regularization

Our training objective is decomposed in two parts: the first one is the ranking criterion on triples of TRANSE, ignoring quadruples. Paths are then taken into account through additional regularization terms.

Denoting by $\mathcal{S}$ the set of facts in the KB, the first part of the training objective is the following ranking criterion that operates on triples

$$\sum_{\substack{(h,\ell,t)\in\mathcal{S} \\ (h',\ell,t')\in\mathcal{S}_{(h,\ell,t)}}} \left[\gamma + d(h,\ell,t) - d(h',\ell,t')\right]_+ ,$$

where $[x]_+ = \max(x, 0)$ is the positive part of x, γ is a margin hyperparameter and $\mathcal{S}_{(h,\ell,t)}$ is the set of corrupted triples created from (h, ℓ, t) by replacing either h or t with another KB entity.

This ranking loss effectively trains so that the embedding of the tail is the nearest neighbor of the translated head, but it does not guarantee that the distance between the tail and the translated head is small. The nearest neighbor criterion is sufficient to make inference over simple triples, but making sure that the distance is small is necessary for the composition rule to be accurate. In order to account for the compositionality of relationships, we add two additional regularization terms:

- $\lambda \sum_{(h,\ell,t)\in\mathcal{S}} d(h,\ell,t)^2$
- $\alpha \sum_{(h,\{\ell_1,\ell_2\},t)\in\mathcal{S}} N_{\ell\to\{\ell_1,\ell_2\}} d(h,\{\ell_1,\ell_2\},t)^2$.

DATA SET	FAMILY	FB15K
ENTITIES	721	14,951
RELATIONSHIPS	7	1,345
TRAINING TRIPLES	8,461	483,142
TRAINING QUAD.	–	30,252
VALIDATION TRIPLES	2,820	50,000
TEST TRIPLES	2,821	59,071
TEST QUAD.	–	1,852

Table 1: Statistics of the datasets.

MODEL	TRANSE		RTRANSE	
	MR	H@10	MR	H@10
EASY	17.7	76.8	**12.5**	**82.2**
HARD	**191.0**	48.9	205.7	**51.0**
EASY W. COMP.	16.4	78.8	**11.6**	**83.0**
EASY W/O COMP.	21.6	71.3	**16.0**	**75.3**
HARD W. COMP.	**208.1**	46.8	212.2	**49.3**
HARD W/O COMP.	**122.9**	**57.0**	123.8	**57.5**
OVERALL	50.7	71.5	**49.5**	**76.2**

Table 2: **Detailed performances on FB15k** of TRANSE and RTRANSE. H@10 are in %. W. COMP. indicates examples for which there exist quadruplets in train matching their relationship.

The first criterion only applies to original facts of the KB, while the second term applies to quadruples. $N_{\ell \to \{\ell_1,\ell_2\}}$, which involves both the relationships of the quadruple and the relationship ℓ from which it was created, is the number of paths involving relationships $\{\ell_1, \ell_2\}$ created from a fact involving ℓ, normalized by the total number of quadruples created from facts involving ℓ. This criterion puts more weight on paths that are reliable as an alternative for a relationship, for instance {*people/city_of_birth*, *city/country*} is likely a better alternative to *people/nationality* than {*people/writer_of_the_film*, *film/film_release_region*}. Finally, a regularization term $\mu||\mathbf{e}||_2^2$ is added for each entity embedding $\mathbf{e}$.

3 Experiments

This section presents experiments on the benchmark FB15K introduced in (Bordes et al., 2013) and on FAMILY, a slightly extended version of the artificial database described in (García-Durán et al., 2014). Table 1 gives their statistics.

3.1 Experimental Protocol

Data FB15K is a subset of Freebase, a very large database of generic facts gathering more than 1.2 billion triples and 80 million entities. Inspired by (Hinton, 1986), FAMILY is a database that contains triples expressing family relationships (cousin_of, has_ancestor, married_to, parent_of, related_to, sibling_of, uncle_of) among the members of 5 families along 6 generations. This dataset is artificial and each family is organized in a layered tree structure where each layer refers to a generation. Families are connected among them by marriage links between two members, randomly sampled from the same layer of different families. Interestingly on this dataset, there are obvious compositional relationships like *uncle_of* $\approx$ *sibling_of* + *parent_of* or *parent_of* $\approx$ *married_to* + *parent_of*, among others.

Setting Our main comparison is TRANSE so we followed the same experimental setting as in (Bordes et al., 2013), using ranking metrics for evaluation. For each test triple we replaced the head by each of the entities in turn, and then computed the score of each of these candidates and sorted them. Since other positive candidates (i.e. entities forming true triples) can be ranked higher than the target one, we filtered out all the positive candidates existing in either the training, validation and test set, except the target one, from the ranking and then we kept the rank of the target entity. The same procedure is repeated but removing the tail instead of the head. The filtered mean rank (mean rank in the rest) is the average of these ranks, and the filtered Hits@10 (H@10 in the rest) is the proportion of target entities in the top 10 predictions.

The embedding dimensions were set to 20 for FAMILY and 100 for FB15K. Training was performed by stochastic gradient descent, stopping after for 500 epochs. On FB15K, we used the embeddings of TRANSE to initialize RTRANSE, and we set a learning rate of 0.001 to fine-tune RTRANSE. On FAMILY, both algorithms were initialized randomly and used a learning rate of 0.01. The mean rank was used as a validation criterion, and the values of γ, λ, α and μ were chosen respectively among $\{0.25, 0.5, 1\}$, $\{1e^{-4}, 1e^{-5}, 0\}$, $\{0.1, 0.05, 0.1, 0.01, 0.005\}$ and $\{1e^{-4}, 1e^{-5}, 0\}$.

3.2 Results

Overall performances Experiments on FAMILY show a quantitative improvement of the performance of RTRANSE : where TRANSE gets a mean rank of 6.7 and a H@5 of 68.7, RTRANSE get a performance of 6.3 and 72.3 respectively.

Similarly, on FB15K, Table 2 (last row) shows that training on longer paths (length 2 here) actually consistently improves the performance while predicting heads and tails of triples only: the overall H@10 improves by almost 5% from 71.5 for

	3 Nearest entities to $h + l_1 + l_2$	
	RTRANSE	TRANSE
h: madtv	**U.S.A.**	Ireland
l_1: regular_TV_appearance	Ireland	**U.S.A.**
l_2: nationality	Japan	U.K.
h: stargate atlantis	Hawaii	Scotland
l_1: regular_TV_appearance	Scotland	Hawaii
l_2: nationality	**U.S.A.**	U.K.
h: malay	southeast asia	taiwan
l_1: language/main_country	malaysia	southeast asia
l_2: continent	**asia**	philippines
h: indiana_state_university	**the_hoosier_state**	maryland
l_1: institution/campuses	terre_haute	rhode_island
l_2: location/state_province_region	rhode_island	the_constitution_state
h: university_of_victoria	**victoria**	kelowna
l_1: institution/campuses	kurnaby	toronto
l_2: location/citytown	kelowna	ottawa

Table 3: **Examples of predictions on quadruples** of TRANSE and RTRANSE. The relation paths $\{l_1, l_2\}$ of the first two examples encode the single the relationship l tv_program/country_of_origin; the third one stands for /language/human_language/region and the last two ones for /location/location/containedby. The correct answer is in **bold**.

TRANSE to 76.2 for RTRANSE.

Detailed results In order to better understand the gains of RTRANSE, we performed a detailed evaluation on FB15K, by classifying the test triples along two axes: *easy* vs *hard* and *with composition* vs *without composition*. A test triple (h, ℓ, t) is *easy* if its head and tail are connected by a triple in the training set, i.e. if either (h, ℓ', t) or (t, ℓ', h) is seen in train for some relationship ℓ'. Otherwise, the triple is *hard*. Orthogonally, the test triple (h, ℓ, t) is *with composition* if there is at least one path $\{\ell_1, \ell_2\}$ for the relationship ℓ, regardless of the existence of that specific path between the entities h and t. If no such path exists, (h, ℓ, t) is *without composition*.

The detailed results are shown in Table 2. We can see that comparatively to TRANSE, RTANSE particularly improves performances in terms of H@10 on triples *with composition*, improving on *easy* triples by 4.2% (from 78.8% to 83,0%) and *hard* triples by 2.5% (from 46.8% to 49.3%). The main gains are still on *easy* triples, and in fact the H@10 on *easy* triples *without composition* increases by 4%, from 71.3% to 75.3%. The mean rank also considerably improves on *easy triples*, and stays somehow still on *hard* ones. All in all, the results show that considering paths during training very significantly improves performances, and the results on triples *with composition* suggest that RTRANSE is indeed capable of capturing the evidence of links that exist in longer paths.

3.3 Results on quadruples

While usual evaluations for link prediction in KBs focus on predicting a missing element of a test triple, we propose here to extend the evaluation to answering more complex questions, such as $(h, \{\ell_1, \ell_2\}, ?)$ or $(?, \{\ell_1, \ell_2\}, t)$.

Examples Table 3 presents examples of predictions of both TRANSE and our model RTRANSE on such quadruples. The two first examples try to predict the origin of two TV series from the nationality of the actors that regularly appear in them (regular_tv_appearance). In the first one, the american actor phil_lamarr is the only entity connected to the american TV show madtv through the relationship regular_tv_appearance. RTRANSE is able to correctly infer the country of origin from this information since it forces country_of_origin $\approx$ regular_tv_appearance + nationality. On the other side TRANSE is affected by the cascading error since the ranking loss does not guarantee that the distance between $h + l_1$ and phil_lamarr is small, so when summing l_2 it eventually ends up closer to Ireland rather than USA. In contrast, the second

example shows that answering that question by using that path is sometimes difficult: the members of the cast of that TV show have different nationalities, so RTRANSE lists the nationalities of these ones and the correct one is ranked third. TRANSE is again more affected than RTRANSE by the cascading error. In the third one, RTRANSE deducts the main region where malay is spoken from the continent of the country with the most number of speakers of that language. In the last two examples, our model infers the location of those universities by forcing an equivalence between their location and the location of their respective campus.

Prediction performance For a more quantitative analysis, we have generated a new test dataset of link prediction on quadruples on FB15K. This test set was created by generating the paths from the usual test set (the triple test set) and removing those quadruples that are used for training. We obtain 1,852 quadruples. The overall experimental protocol is the same as before, trying to predict the head or tail of these quadruple in turn.

On that evaluation protocol, RTRANSE has a mean rank of 114.0 and a H@10 of 68.2%, while TRANSE obtains a mean rank of 159.9 and a H@10 of 65.2% (using the same models as in the previous subsection). We can see that learning on paths improves performances on both metrics, with a gain of 3% in terms of H@10 and an important gain of about 46 in mean rank, which corresponds to a relative improvement of about 30%.

4 Conclusion

We have proposed to learn embeddings of compositions of relationships in the translation model for link prediction in KBs. Our experimental results show that this approach is promising.

We considered in this work a restricted set of small paths of length two. We leave the study of more general paths to future work.

Acknowledgments

This work was carried out in the framework of the Labex MS2T (ANR-11-IDEX-0004-02), and was funded by the French National Agency for Research (EVEREST-12-JS02-005-01).

References

Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems*, pages 2787–2795.

Alberto García-Durán, Antoine Bordes, and Nicolas Usunier. 2014. Effective blending of two and three-way interactions for modeling multi-relational data. In *ECML PKDD 2014*. Springer Berlin Heidelberg.

Geoffrey E Hinton. 1986. Learning distributed representations of concepts. In *Proceedings of the eighth annual conference of the cognitive science society*, volume 1, page 12. Amherst, MA.

Rodolphe Jenatton, Nicolas Le Roux, Antoine Bordes, and Guillaume Obozinski. 2012. A latent factor model for highly multi-relational data. In *NIPS 25*.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of AAAI'15*.

Arvind Neelakantan, Benjamin Roth, and Andrew McCallum. 2015. Compositional vector space models for knowledge base completion. *arXiv preprint arXiv:1504.06662*.

Richard Socher, Danqi Chen, Christopher D. Manning, and Andrew Y. Ng. 2013. Reasoning With Neural Tensor Networks For Knowledge Base Completion. In *Advances in Neural Information Processing Systems 26*.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, pages 1112–1119.

Noise or additional information? Leveraging crowdsource annotation item agreement for natural language tasks.

Emily K. Jamison [‡] **and Iryna Gurevych** [†‡]
[‡]Ubiquitous Knowledge Processing Lab (UKP-TUDA),
Department of Computer Science, Technische Universität Darmstadt
† Ubiquitous Knowledge Processing Lab (UKP-DIPF),
German Institute for Educational Research
http://www.ukp.tu-darmstadt.de

Abstract

In order to reduce noise in training data, most natural language crowdsourcing annotation tasks gather redundant labels and aggregate them into an integrated label, which is provided to the classifier. However, aggregation discards potentially useful information from linguistically ambiguous instances.

For five natural language tasks, we pass item agreement on to the task classifier via soft labeling and low-agreement filtering of the training dataset. We find a statistically significant benefit from low item agreement training filtering in four of our five tasks, and no systematic benefit from soft labeling.

1 Introduction

Crowdsourcing is a cheap and increasingly-utilized source of annotation labels. In a typical annotation task, five or ten labels are collected for an instance, and are aggregated together into an *integrated label*. The high number of labels is used to compensate for worker bias, task misunderstanding, lack of interest, incompetance, and malicious intent (Wauthier and Jordan, 2011).

Majority voting for label aggregation has been found effective in filtering noisy labels (Nowak and Rüger, 2010). Labels can be aggregated under weighted conditions reflecting the reliability of the annotator (Whitehill et al., 2009; Welinder et al., 2010). Certain classifiers are also robust to random (unbiased) label noise (Tibshirani and Manning, 2014; Beigman and Beigman Klebanov, 2009). However, minority label information is discarded by aggregation, and when the labels were gathered under controlled circumstances, these labels may reflect linguistic intuition and contain useful information (Plank et al., 2014b). Two alternative strategies that allow the classifier to learn from the item agreement include training instance *filtering* and *soft labeling*. Filtering training instances by item agreement removes low agreement instances from the training set. Soft labeling assigns a classifier weight to a training instance based on the item agreement.

Consider two Affect Recognition instances and their Krippendorff (1970)'s α *item agreement* :

Text: *India's Taj Mahal gets facelift*
Sadness Rating (0-100): 8.0
α **Agreement (-1.0 – 1.0):** 0.7

Figure 1: Affect Recognition Easy Case.

Text: *After Iraq trip, Clinton proposes war limits*
Sadness Rating (0-100): 12.5
α **Agreement (-1.0 – 1.0):** -0.1

Figure 2: Affect Recognition Hard Case.

In Figure 1, annotators mostly agreed that the headline expresses little sadness. But in Figure 2, the low item agreement may be caused by instance difficulty (i.e., *Is a war zone* sad *or just bad?*): a *Hard Case* (Zeman, 2010). Previous work (Beigman Klebanov and Beigman, 2014; Beigman and Beigman Klebanov, 2009) has shown that training strategy may affect Hard and Easy Case test instances differently.

In this work, for five natural language tasks, we examine the impact of passing crowdsource item agreement on to the task classifier, by means of training instance *filtering* and *soft labeling*. We construct classifiers for Biased Text Detection, Stemming Classification, Recognizing Textual Entailment, Twitter POS Tagging, and Affect Recognition, and evaluate the effect of our different training strategies on the accuracy of each

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 291–297,
Lisbon, Portugal, 17-21 September 2015.

task. These tasks represent a wide range of machine learning tasks typical in NLP: sentence-level SVM regression using n-grams; word pairs with character-based features and binary SVM classification; pairwise sentence binary SVM classification with similarity score features; CRF sequence word classification with a range of feature types; and sentence-level regression using a token-weight averaging, respectively. We use pre-existing, freely-available crowdsourced datasets and post all our experiment code on GitHub[1].

Contributions This is the first work (1) to apply item-agreement-weighted soft labeling from crowdsourced labels to multiple real natural language tasks; (2) to filter training instances by item agreement from crowdsourced labels, for multiple natural language tasks; (3) to evaluate classifier performance on high item agreement (*Easy Case*) instances and low item agreement (*Hard Case*) instances across multiple natural language tasks.

2 Related Work

Dekel and Shamir (2009) calculated integrated labels for an information retrieval crowdsourced dataset, and identified low-quality workers by deviation from the integrated label. Removal of these workers' labels improved classifier performance on data that was not similarly filtered. While much work (Dawid and Skene, 1979; Ipeirotis et al., 2010; Dalvi et al., 2013) has explored techniques to model worker ability, bias, and instance difficulty while aggregating labels, there is no evaluation comparing classifiers trained on the new integrated labels with other options, on their respective NLP tasks.

Training instance filtering aims to remove mislabeled instances from the training dataset. Sculley and Cormack (2008) learned a logistic regression classifier to identify and filter noisy labels in a spam email filtering task. They also proposed a label correcting technique that replaces identified noisy labels with "corrected" labels, at the risk of introducing noise into the corpus. Rebbapragada et al. (2009) developed a label noise detection technique to cluster training instances and remove label outliers. Raykar et al. (2010) jointly learned a classifier/regressor, annotator accuracy, and the integrated label on datasets with multiple noisy labels, outperforming Smyth et al. (1995)'s model of estimating ground truth labels.

Soft labeling, or the association of one training instance with multiple, weighted, conflicting labels, is a technique to model noisy training data. Thiel (2008) found that soft labeled training data produced more accurate classifiers than hard labeled training data, with both Radial Basis Function Networks and Fuzzy-Input Fuzzy-Output SVMs. Shen and Lapata (2007) used soft labeling to model their semantic frame structures in a question answering task, to represent that the semantic frames can bear multiple sematic roles.

Previous research has found that, for a few individual NLP tasks, training while incorporating label noise weight may produce a better model. Martínez Alonso et al. (2015) show that informing a parser of annotator disagreement via loss function reduced error in labeled attachments by 6.4%. Plank et al. (2014a) incorporate annotator disagreement in POS tags into the loss function of a POS-tag machine learner, resulting in improved performance on downstream chunking. Beigman Klebanov and Beigman (2014) observed that, on a task classifying text as semantically *old* or *new*, the inclusion of Hard Cases in training data resulted in reduced classifier performance on Easy Cases.

3 Overview of Experiments

We built systems for the five NLP tasks, and trained them using aggregation, soft labeling, and instance screening strategies. When labels were numeric, the integrated label was the average[2]. When labels were nominal, the integrated label was majority vote. Krippendorff (1970)'s α item agreement was used to filter ambiguous training instances. For soft labeling, percentage item agreement was used to assign instance weights. We followed Sheng et al. (2008)'s suggested *Multiplied Examples* procedure: for each unlabeled instance x_i and each existing label $y_i \in L_i = \{y_{ij}\}$ (as annotated by worker j), we create one replica of x_i, assign it y_i, and weight the instance according to the count of y_i in L_i (i.e., the percentage item agrement). For each training strategy (*Soft-Label*, etc), the *training* instances were changed by the strategy, but the *test* instances were unaffected. For the division of test instances into Hard

[1] github.com/EmilyKJamison/crowdsourcing

[2] We followed Yano et al. (2010) and Strapparava and Mihalcea (2007) in using *mean* as gold standard. Although another aggregation such as as *median* might be more representative, such discussion is beyond the scope of this paper.

and Easy Cases, the training instances were unaffected, but the test instances were filtered by α item agreement. Hard/Easy Case parameters were chosen to divide the corpus by item agreement into roughly equal portions[3], relative to the corpus, for post-hoc error analysis.

All systems except Affect Recognition were constructed using DKPro Text Classification (Daxenberger et al., 2014), and used Weka's *SMO* (Platt, 1999) or *SMOreg* (Shevade et al., 2000) implementations with default parameters, with 10-fold (or 5-fold, for computationally-intensive POS Tagging) cross-validation. More details are available in the Supplemental Notes document.

Agreement Parameters Training strategies *HighAgree* and *VeryHigh* utilize agreement cutoff parameters that vary per corpus. These strategies are a discretized approximation of the gradual effect of filtering low agreement instances from the training data. For any given corpus, we could not use a cutoff value equal to no filtering, or that eliminated a class. If there were only 2 remaining cutoffs, we used these. If there were more candidate cutoff values, we trained and evaluated a classifier on a development set and chose the value for *HighAgree* that maximized Hard Case performance on the development set.

Percentage Agreement In this paper, we follow Beigman Klebanov and Beigman (2014) in using the nominal agreement categories *Hard Cases* and *Easy Cases* to separate instances by item agreement. However, unlike Beigman Klebanov and Beigman (2014) who use simple *percentage agreement*, we calculate item-specific agreement via Krippendorff (1970)'s α item agreement[4], with Nominal, Ordinal, or Ratio distance metrics as appropriate. The agreement is expressed in the range (-1.0 – 1.0); 1.0 is perfect agreement.

3.1 Biased Language Detection

This task detects the use of bias in political text. The corpus (Yano et al., 2010)[5] consists of 1,041 sentences from American political blogs. For each sentence, five crowdsource annotators chose a label *no bias*, *some bias*, and *very biased*. We follow Yano et al. (2010) in representing the amount of bias on a numerical scale (1-3). Hard/Easy Case cutoffs were <-.21 and >.20. Of 1041 total instances, 161 were Hard Cases (<-.21) and 499 were Easy Cases (>.20).

We built an SVM regression task using unigrams, to predict the numerical amount of bias. The gold standard was the integrated labels. Item-specific agreement was calculated with Ordinal Distance Function (Krippendorff, 1980).

We used the following training strategies:
VeryHigh Filtered for agreement >0.4.
HighAgree Filtered for agreement >-0.2.
SoftLabel One training instance is generated for each label from a text, and weighted by how many times that label occurred with the text.
SLLimited SoftLabel, except that training instances with a label distance >1.0 from the original text label average are discarded.

3.2 Morphological Stemming

The goal of this binary classification task is to predict, given an original word and a stemmed version of the word, whether the stemmed version has been correctly stemmed. The word pair was correct if: the stemmed word contained one less affix; or if the original word was a compound, the stemmed word had a space inserted between the components; or if the original word was misspelled, the stemmed word was deleted; or if the original word had no affixes and was not a compound and was not misspelled, then the stemmed word had no changes.

This dataset was compiled by Carpenter et al. (2009)[6]. The dataset contains 6679 word pairs; most pairs have 5 labels each. In the cross-validation division, no pairs with the same original word could be split across training and test data. The gold standard was the integrated label, with 4898 positive and 1781 negative pairs. Hard/Easy Case cutoffs were <-.5 and >.5. Of 6679 total instances, 822 were Hard Cases (<-.5) and 3615 were Easy Cases (>.5). Features used are combinations of the characters after the removal of the longest common substring between the word pair, including 0-2 additional characters from the substring; word boundaries are marked.

Stemming-new training strategies include:
HighAgree Filtered for agreement >-0.1.
SLLimited MajVote with instances weighted by the frequency of the label for the text pair.

[3]Limited by the discrete nature of our agreement.
[4]From the DKPro Statistics library (Meyer et al., 2014)
[5]Available at `sites.google.com/site/amtworkshop2010/data-1`
[6]Available at `github.com/bob-carpenter/anno`

3.3 Recognising Textual Entailment

Recognizing textual entailment is the process of determining if, given two sentences *text* and *hypothesis*, the meaning of the hypothesis can be inferred from the text.

We used the dataset from the PASCAL RTE-1, which contains 800 sentence pairs. The crowdsource annotations of 10 labels per pair were obtained by Snow et al. (2008)[7]. We reproduced the basic system described in (Dagan et al., 2006) of TF-IDF weighted Cosine Similarity between lemmas of the text and hypothesis. The weight of each word$_i$ in $document_j$, n total documents, is the log-plus-one term$_i$ frequency normalized by raw term$_i$ document frequency, with Euclidean normalization.

$$\text{weight}(i,j) = \begin{cases} (1+\log(\text{tf}_{i,j}))\frac{N}{\text{df}_i} & \text{if tf}_{i,j} \geq 1 \\ 0 & \text{if tf}_{i,j} = 0 \end{cases}$$

Additionally, we used features including the difference in noun chunk character and token length, the difference in number of tokens, shared named entities, and subtask names. The gold standard was the original labels from Dagan et al. (2006). Hard/Easy Case cutoffs were <0.0 and >.3. Training strategies are from Biased Language (*VeryHigh*) and Stem (others) experiments, except the *HighAgree* cutoff was 0.0 and the *VeryHigh* cutoff was 0.3. Of 800 total instances, 230 were Hard Cases (<0.0) and 207 were Easy Cases (>.30).

3.4 POS tagging

We built a POS-tagger for Twitter posts. We used the training section of the dataset from Gimpel et al. (2011). The POS tagset was the universal tag set (Petrov et al., 2012); we converted Gimpel et al. (2011)'s tags to the universal tagset using Hovy et al. (2014)'s mapping. Crowdsource labels for this data came from Hovy et al. (2014)[8], who obtained 5 labels for each tweet. After aligning and cleaning, our dataset consisted of 953 tweets of 14,439 tokens.

We followed Hovy et al. (2014) in constructing a CRF classifier (Lafferty et al., 2001), using a list of English affixes, Hovy et al. (2014)'s set of orthographic features, and word clusters (Owoputi et al., 2013). In the cross-validation division, individual tweets were assigned to folds. The gold standard was the integrated label. Hard/Easy Case cutoffs were <0.0 and >.49. Of 14,439 tokens, 649 were Hard Cases (<0.0) and 10830 were Easy Cases (>.49).

We used the following strategies:

VeryHigh For each token t in sequence s where $agreement(t)$ <0.5, s is broken into two separate sequences s_1 and s_2 and t is deleted (i.e. filtered).

HighAgree *VeryHigh* with agreement <0.2.

SoftLabel For each proto-sequence s, we generate 5 sequences $\{s_0, s_1, ..., s_i\}$, in which each token t is assigned a crowdsource label drawn at random: $l_{t,s_i} \in L_t$.

SLLimited, Each token t in sequence s is assigned its MajVote label. Then s is given a weight representing the average item agreement for all $t \in s$.

3.5 Affect Recognition

Our Affect Recognition experiments are based on the affective text annotation task in Strapparava and Mihalcea (2007), using the *Sadness* dataset. Each headline is rated for "sadness" using a scale of 0-100. Examples are in Figures 1 and 2. We use the crowdsourced annotation for a 100-headline sample of this dataset provided by Snow et al. (2008)[9], with 10 annotations per emotion per headline. Of 100 total instances, 20 were Hard Cases (<0.0) and 49 were Easy Cases (>.30).

Our system design is identical to Snow et al. (2008), which is similar to the SWAT system (Katz et al., 2007), a top-performing system on the SemEval Affective Text task. Hard/Easy Case cutoffs were <0.0 and >.3.

Training strategies are the same as for the Biased Language experiments, except:

VeryHigh Filtered for agreement >0.3.

HighAgree Filtered for agreement >0.

SLLimited SoftLabel, except that instances with a label distance >20.0 from the original label average are discarded.

4 Results

Our results on all five tasks, using each of the training strategies and variously evaluating on all, Easy, or Hard Cases, can be seen in Table 1. Systems outputing numeric values show results in Pearson correlation, and systems outputing nominal labels show micro F_1. Soft labeling (*SoftLabel*) failed to outperform integrated labels for 4 of the 5 complete test sets. Likewise, *SLLimited*

[7] Available at sites.google.com/site/nlpannotations/

[8] Available at lowlands.ku.dk/results/

[9] Available at sites.google.com/site/nlpannotations/

	Biased Lang			Stemming			RTE			POS			Affective Text		
Training	**All**	**Hard**	**Easy**	**All**	**Hard**	**Easy**	**All**	**Hard**	**Easy**	**All**	**Hard**	**Easy**	**All**	**Hard**	**Easy**
Integrated	.236	.144	.221	.797	.568	.927	.513	.330	.831	.790	.370	.878	.446	.115	.476
VeryHigh	.140	.010	.158	–	–	–	.499	.304	.836	.771	.310	.869	.326	.059	.376
HighAgree	.231	.210	.222	.796	.569	.924	.543	.361	.831	.810	.382	.901	.453	.265	.505
SoftLabel	.223	.131	.210	.766	.539	.957	.499	.304	.836	.789	.353	.880	.450	.112	.477
SLLimited	.235	.158	.208	.799	.569	.930	.493	.291	.831	.797	.376	.882	.450	.139	.472

Table 1: Results (Pearson or micro F_1) with different training strategies and all, Hard, and Easy Cases.

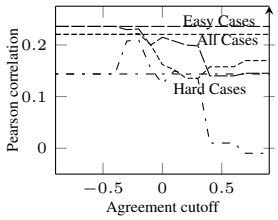


Figure 3: Biased Language.

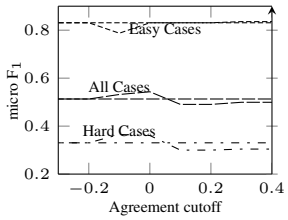


Figure 4: RTE.

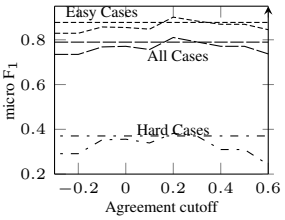


Figure 5: POS Tags.

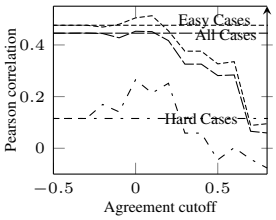


Figure 6: Affective Text.

did not significantly outperform *Integrated*. However, HighAgree does outperform *Integrated* on 4 or the 5 tasks, especially for Hard Cases: Hard Case improvements for Biased Language and POS Tagging, and Affective Text, and overall improvements for RTE, POS Tagging, and Affective Text were significant (Paired TTest, $p < 0.05$, for numerical output, or McNemar's Test[10] (McNemar, 1947), $p < 0.05$, for nominal classes). The fifth task, Stemming, had the lowest number of item agreement categories of the five tasks, preventing fine-grained agreement training filtering, which explains why filtering shows no benefit.

All training strategies used the same amount of annotated data as input, and for filtering strategies such as *HighAgree*, a reduced number of strategy-output instances are used to train the model. As a higher cutoff is used for *HighAgree*, the lack of training data results in a worse model; this can be seen in the downward curves of Figures 3 – 6, where the curved line is *HighAgree* and the matching pattern straight line is *Integrated*. (Due to the low number of item agreement categories, Stemming results are not displayed in an item agreement cutoff table.) However, Figures 4 – 6 show the overall performance boost, and Figure 3 shows the Hard Case performance boost, right before the downward curves from too little training data, using *HighAgree*.

Comparability We found the accuracy of our systems was similar to that reported in previous literature. Dagan et al. (2006) report performance of the RTE system, on a different data division, with accuracy=0.568. Hovy et al. (2014) report majority vote results (from acc=0.805 to acc=0.837 on a different data section) similar to our results of 0.790 micro-F_1. For Affective Text, Snow et al. (2008) report results on a different data section of r=0.174, a merged result from systems trained on combinations of crowdsource labels and evaluated against expert-trained systems. The SWAT system (Katz et al., 2007), which also used lexical resources and additional training data, acheived r=0.3898 on a different section of data. These results are comparable with ours, which range from r=0.326 to r=0.453.

[10] See Japkowicz and Shah (2011) for usage description.

5 Conclusions and Future Work

In this work, for five natural langauge tasks, we have examined the impact of informing the classifier of crowdsource item agreement, by means of soft labeling and removal of low-agreement training instances. We found a statistically significant benefit from low-agreement training filtering in four of our five tasks, and strongest improvements for Hard Cases. Previous work (Beigman Klebanov and Beigman, 2014) found a similar effect, but only evaluated a single task, so generalizability was unknown. We also found that soft labeling was not beneficial compared to aggregation. Our findings suggest that the best crowdsource label training strategy is to remove low item agreement instances from the training set.

Acknowledgments

This work has been supported by the Volkswagen Foundation as part of the Lichtenberg-Professorship Program under grant No. I/82806, and by the Center for Advanced Security Research (www.cased.de).

References

Eyal Beigman and Beata Beigman Klebanov. 2009. Learning with annotation noise. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 280–287, Suntec, Singapore.

Beata Beigman Klebanov and Eyal Beigman. 2014. Difficult cases: From data to learning, and back. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 390–396, Baltimore, Maryland.

Bob Carpenter, Emily Jamison, and Breck Baldwin. 2009. Building a stemming corpus: Coding standards. `http://lingpipe-blog.com/2009/02/25/stemming-morphology-corpus-coding-standards/`.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The PASCAL Recognising Textual Entailment Challenge. In *Machine learning challenges. Evaluating predictive uncertainty, visual object classification, and recognising textual entailment*, pages 177–190. Springer.

Nilesh Dalvi, Anirban Dasgupta, Ravi Kumar, and Vibhor Rastogi. 2013. Aggregating crowdsourced binary ratings. In *Proceedings of the 22nd International Conference on World Wide Web*, pages 285–294, Rio de Janeiro, Brazil.

Alexander Philip Dawid and Allan M. Skene. 1979. Maximum likelihood estimation of observer error-rates using the EM algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):20–28.

Johannes Daxenberger, Oliver Ferschke, Iryna Gurevych, and Torsten Zesch. 2014. DKPro TC: A Java-based Framework for Supervised Learning Experiments on Textual Data. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics*, pages 61–66, Baltimore, Maryland.

Ofer Dekel and Ohad Shamir. 2009. Vox populi: Collecting high-quality labels from a crowd. In *Proceedings of the Twenty-Second Annual Conference on Learning Theory*, Montreal, Canada. Online proceedings.

Kevin Gimpel, Nathan Schneider, Brendan O'Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A. Smith. 2011. Part-of-speech tagging for Twitter: Annotation, features, and experiments. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 42–47, Portland, Oregon.

Dirk Hovy, Barbara Plank, and Anders Søgaard. 2014. Experiments with crowdsourced re-annotation of a pos tagging data set. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 377–382, Baltimore, Maryland.

Panagiotis G. Ipeirotis, Foster Provost, and Jing Wang. 2010. Quality management on Amazon Mechanical Turk. In *Proceedings of the ACM SIGKDD Workshop on Human Computation*, pages 64–67, Washington DC, USA.

Nathalie Japkowicz and Mohak Shah. 2011. *Evaluating learning algorithms: a classification perspective*. Cambridge University Press.

Phil Katz, Matt Singleton, and Richard Wicentowski. 2007. SWAT-MP:The SemEval-2007 Systems for Task 5 and Task 14. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 308–313, Prague, Czech Republic.

Klaus Krippendorff. 1970. Estimating the reliability, systematic error and random error of interval data. *Educational and Psychological Measurement*, 30(1):61–70.

Klaus Krippendorff. 1980. *Content analysis: An introduction to its methodology*. Sage, Beverly Hills, California.

John Lafferty, Andrew McCallum, and Fernando C.N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning*, pages 282–289, Williamstown, Massachusetts.

Héctor Martínez Alonso, Barbara Plank, Arne Skjærholt, and Anders Søgaard. 2015. Learning to parse with IAA-weighted loss. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1357–1361, Denver, Colorado.

Quinn McNemar. 1947. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, 12(2):153–157.

Christian M. Meyer, Margot Mieskes, Christian Stab, and Iryna Gurevych. 2014. DKPro Agreement: An open-source java library for measuring inter-rater agreement. In *Proceedings of the 25th International Conference on Computational Linguistics (COLING)*, pages 105–109, Dublin, Ireland.

Stefanie Nowak and Stefan Rüger. 2010. How reliable are annotations via crowdsourcing: A study about inter-annotator agreement for multi-label image annotation. In *Proceedings of the International Conference on Multimedia Information Retrieval*, pages 557–566, Philadelphia, Pennsylvania.

Olutobi Owoputi, Brendan O'Connor, Chris Dyer, Kevin Gimpel, Nathan Schneider, and Noah A. Smith. 2013. Improved part-of-speech tagging for

online conversational text with word clusters. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 380–390, Atlanta, Georgia.

Slav Petrov, Dipanjan Das, and Ryan McDonald. 2012. A universal part-of-speech tagset. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC-2012)*, pages 2089–2096, Istanbul, Turkey.

Barbara Plank, Dirk Hovy, and Anders Søgaard. 2014a. Learning part-of-speech taggers with inter-annotator agreement loss. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 742–751, Gothenburg, Sweden.

Barbara Plank, Dirk Hovy, and Anders Søgaard. 2014b. Linguistically debatable or just plain wrong? In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 507–511, Baltimore, Maryland.

John Platt. 1999. Fast training of support vector machines using sequential minimal optimization. In *Advances in kernel methods – support vector learning*, pages 185–208. MIT Press.

Vikas C. Raykar, Shipeng Yu, Linda H. Zhao, Gerardo Hermosillo Valadez, Charles Florin, Luca Bogoni, and Linda Moy. 2010. Learning from crowds. *The Journal of Machine Learning Research*, 11:1297–1322.

Umaa Rebbapragada, Lukas Mandrake, Kiri L. Wagstaff, Damhnait Gleeson, Rebecca Castano, Steve Chien, and Carla E. Brodley. 2009. Improving onboard analysis of hyperion images by filtering mislabeled training data examples. In *Proceedings of the 2009 IEEE Aerospace Conference*, pages 1–9, Big Sky, Montana.

D. Sculley and Gordon V. Cormack. 2008. Filtering email spam in the presence of noisy user feedback. In *Proceedings of the Conference on Email and Anti-spam (CEAS)*, Mountain View, CA, USA. Online proceedings.

Dan Shen and Mirella Lapata. 2007. Using semantic roles to improve question answering. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 12–21, Prague, Czech Republic.

Victor S. Sheng, Foster Provost, and Panagiotis G. Ipeirotis. 2008. Get another label? Improving data quality and data mining using multiple, noisy labelers. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 614–622, Las Vegas, Nevada.

Shirish Krishnaj Shevade, S. Sathiya Keerthi, Chiranjib Bhattacharyya, and Karaturi Radha Krishna Murthy. 2000. Improvements to the SMO algorithm for SVM regression. *IEEE Transactions on Neural Networks*, 11(5):1188–1193.

Padhraic Smyth, Usama Fayyad, Michael Burl, Pietro Perona, and Pierre Baldi. 1995. Inferring ground truth from subjective labelling of Venus images. *Advances in Neural Information Processing Systems*, pages 1085–1092.

Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Ng. 2008. Cheap and fast – but is it good? Evaluating non-expert annotations for natural language tasks. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 254–263, Honolulu, Hawaii.

Carlo Strapparava and Rada Mihalcea. 2007. SemEval-2007 Task 14: Affective Text. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 70–74, Prague, Czech Republic.

Christian Thiel. 2008. Classification on soft labels is robust against label noise. In *Knowledge-Based Intelligent Information and Engineering Systems*, pages 65–73, Wellington, New Zealand.

Julie Tibshirani and Christopher D. Manning. 2014. Robust logistic regression using shift parameters. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 124–129, Baltimore, Maryland.

Fabian L. Wauthier and Michael I. Jordan. 2011. Bayesian bias mitigation for crowdsourcing. In *Advances in Neural Information Processing Systems*, pages 1800–1808.

Peter Welinder, Steve Branson, Pietro Perona, and Serge J. Belongie. 2010. The multidimensional wisdom of crowds. In *Advances in Neural Information Processing Systems*, pages 2424–2432.

Jacob Whitehill, Ting fan Wu, Jacob Bergsma, Javier R. Movellan, and Paul L. Ruvolo. 2009. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. In Y. Bengio, D. Schuurmans, J.D. Lafferty, C.K.I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems*, pages 2035–2043. Curran Associates, Inc.

Tae Yano, Philip Resnik, and Noah A. Smith. 2010. Shedding (a thousand points of) light on biased language. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, pages 152–158, Los Angeles, California.

Daniel Zeman. 2010. Hard problems of tagset conversion. In *Proceedings of the Second International Conference on Global Interoperability for Language Resources*, pages 181–185, Hong Kong, China.

Evaluation methods for unsupervised word embeddings

Tobias Schnabel
Cornell University
Ithaca, NY, 14853
tbs49@cornell.edu

Igor Labutov
Cornell University
Ithaca, NY, 14853
iil4@cornell.edu

David Mimno, Thorsten Joachims
Cornell University
mimno@cornell.edu,
tj@cs.cornell.edu

Abstract

We present a comprehensive study of evaluation methods for unsupervised embedding techniques that obtain meaningful representations of words from text. Different evaluations result in different orderings of embedding methods, calling into question the common assumption that there is one single optimal vector representation. We present new evaluation techniques that directly compare embeddings with respect to specific queries. These methods reduce bias, provide greater insight, and allow us to solicit data-driven relevance judgments rapidly and accurately through crowdsourcing.

1 Introduction

Neural word embeddings represent meaning via geometry. A good embedding provides vector representations of words such that the relationship between two vectors mirrors the linguistic relationship between the two words. Despite the growing interest in vector representations of semantic information, there has been relatively little work on direct evaluations of these models. In this work, we explore several approaches to measuring the quality of neural word embeddings. In particular, we perform a comprehensive analysis of evaluation methods and introduce novel methods that can be implemented through crowdsourcing, providing better insights into the relative strengths of different embeddings.

Existing schemes fall into two major categories: extrinsic and intrinsic evaluation. In extrinsic evaluation, we use word embeddings as input features to a downstream task and measure changes in performance metrics specific to that task. Examples include part-of-speech tagging and named-entity recognition (Pennington et al., 2014). Extrinsic evaluation only provides one way to specify the goodness of an embedding, and it is not clear how it connects to other measures.

Intrinsic evaluations directly test for syntactic or semantic relationships between words (Mikolov et al., 2013a; Baroni et al., 2014). These tasks typically involve a pre-selected set of query terms and semantically related target words, which we refer to as a *query inventory*. Methods are evaluated by compiling an aggregate score for each method such as a correlation coefficient, which then serves as an absolute measure of quality. Query inventories have so far been collected opportunistically from prior work in psycholinguistics, information retrieval (Finkelstein et al., 2002), and image analysis (Bruni et al., 2014). Because these inventories were not constructed for word embedding evaluation, they are often idiosyncratic, dominated by specific types of queries, and poorly calibrated to corpus statistics.

To remedy these problems, this paper makes the following contributions. First, this is the first paper to conduct a comprehensive study covering a wide range of evaluation criteria and popular embedding techniques. In particular, we study how outcomes from three different evaluation criteria are connected: word relatedness, coherence, downstream performance. We show that using different criteria results in different relative orderings of embeddings. These results indicate that embedding methods should be compared in the context of a specific task, e.g., linguistic insight or good downstream performance.

Second, we study the connections between direct evaluation with real users and pre-collected offline data. We propose a new approach to evaluation that focuses on direct comparison of embeddings with respect to individual queries rather than overall summary scores. Because we phrase all tasks as choice problems rather than ordinal relevance tasks, we can ease the burden of the an-

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 298–307,
Lisbon, Portugal, 17-21 September 2015.

notators. We show that these evaluations can be gathered efficiently from crowdsourcing. Our results also indicate that there is in fact strong correlation between the results of automated similarity evaluation and direct human evaluation. This result justifies the use of offline data, at least for the similarity task.

Third, we propose a model- and data-driven approach to constructing query inventories. Rather than picking words in an *ad hoc* fashion, we select query words to be diverse with respect to their frequency, parts-of-speech and abstractness. To facilitate systematic evaluation and comparison of new embedding models, we release a new frequency-calibrated query inventory along with all user judgments at `http://www.cs.cornell.edu/~schnabts/eval/`.

Finally, we observe that word embeddings encode a surprising degree of information about word frequency. We found this was true even in models that explicitly reserve parameters to compensate for frequency effects. This finding may explain some of the variability across embeddings and across evaluation methods. It also casts doubt on the common practice of using the vanilla cosine similarity as a similarity measure in the embedding space.

It is important to note that this work is a survey of *evaluation* methods not a survey of *embedding* methods. The specific example embeddings presented here were chosen as representative samples only, and may not be optimal.

2 Word embeddings

We refer to a word embedding as a mapping $V \rightarrow \mathbb{R}^D : w \mapsto \vec{w}$ that maps a word w from a vocabulary V to a real-valued vector $\vec{w}$ in an embedding space of dimensionality D.

Following previous work (Collobert et al., 2011; Mikolov et al., 2013a) we use the commonly employed cosine similarity, defined as $\text{similarity}(w_1, w_2) = \frac{\vec{w_1} \cdot \vec{w_2}}{\|\vec{w_1}\| \|\vec{w_2}\|}$, for all similarity computations in the embedding space. The list of nearest neighbors of a word w are all words $v \in V \setminus \{w\}$, sorted in descending order by $\text{similarity}(w, v)$. We will denote w as the *query word* in the remainder of this paper.

All experiments in this paper are carried out on six popular unsupervised embedding methods. These embeddings form a representative but incomplete subset; and since we are studying evaluation methods and not embeddings themselves, no attempt has been made to optimize these embeddings. The first two embedding models, the CBOW model of word2vec (Mikolov et al., 2013a) and C&W embeddings (Collobert et al., 2011) both are motivated by a probabilistic prediction approach. Given a number of context words around a target word w, these models formulate the embedding task as that of finding a representation that is good at predicting w from the context representations.

The second group of models, Hellinger PCA (Lebret and Collobert, 2014), GloVe (Pennington et al., 2014), TSCCA (Dhillon et al., 2012) and Sparse Random Projections (Li et al., 2006) follow a reconstruction approach: word embeddings should be able to capture as much relevant information from the original co-occurrence matrix as possible.

Training corpus. We tried to make the comparison as fair as possible. As the C&W embeddings were only available pretrained on a November 2007 snapshot of Wikipedia, we chose the closest available Wikipedia dump (2008-03-01) for training the other models. We tokenized the data using the Stanford tokenizer (Manning et al., 2014). Like Collobert et al. (2011), we lower-cased all words and replaced digits with zeros.

Details. All models embedded words into a 50-dimensional space ($D = 50$). As implemented, each method uses a different vocabulary, so we computed the intersection of the six vocabularies and used the resulting set of 103,647 words for all nearest-neighbor experiments.

3 Relatedness

We begin with intrinsic evaluation of relatedness using both pre-collected human evaluations and a novel online user study. Section 3.1 introduces the list of datasets that is commonly used as a benchmark for embedding methods. There, embeddings are evaluated individually and only their final scores are compared, hence we refer to this scenario as *absolute intrinsic* evaluation. We present a new scenario, *comparative intrinsic* evaluation, in which we ask people directly for their preferences among different embeddings. We demonstrate that we can achieve the same results as offline, absolute metrics using online, comparative metrics.

3.1 Absolute intrinsic evaluation

For the absolute intrinsic evaluation, we used the same datasets and tasks as Baroni et al. (2014). While we present results on all tasks for completeness, we will mainly focus on relatedness in this section. There are four broad categories:

- **Relatedness**: These datasets contain relatedness scores for pairs of words; the cosine similarity of the embeddings for two words should have high correlation (Spearman or Pearson) with human relatedness scores.
- **Analogy**: This task was popularized by Mikolov et al. (2013a). The goal is to find a term x for a given term y so that $x : y$ best resembles a sample relationship $a : b$.
- **Categorization**: Here, the goal is to recover a clustering of words into different categories. To do this, the corresponding word vectors of all words in a dataset are clustered and the purity of the returned clusters is computed with respect to the labeled dataset.
- **Selectional preference**: The goal is to determine how typical a noun is for a verb either as a subject or as an object (e.g., people eat, but we rarely eat people). We follow the procedure that is outlined in Baroni et al. (2014).

Several important design questions come up when designing reusable datasets for evaluating relatedness. While we focus mainly on challenges that arise in the relatedness evaluation task, many of the questions discussed also apply to other scenarios.

Query inventory. How we pick the word pairs to evaluate affects the results of the evaluation. The commonly-used WordSim-353 dataset (Finkelstein et al., 2002), for example, only tries to have word pairs with a diverse set of similarity scores. The more recent MEN dataset (Bruni et al., 2014) follows a similar strategy, but restricts queries to words that occur as annotations in an image dataset. However, there are more important criteria that should be considered in order to create a diverse dataset: (i) the frequency of the words in the English language (ii) the parts of speech of the words and (iii) abstractness vs. concreteness of the terms. Not only is frequency important because we want to test the quality of embeddings on rare words, but also because it is related with distance in the embedding space as we show later and should be explicitly considered.

Metric aggregation. The main conceptual shortcoming of using correlation-based metrics is that they aggregate scores of different pairs — even though these scores can vary greatly in the embedding space. We can view the relatedness task as the task of evaluating a set of rankings, similar to ranking evaluation in Information Retrieval. More specifically, we have one query for each unique query word w and rank all remaining words v in the vocabulary accordingly. The problem now is that we usually cannot directly compare scores from different rankings (Aslam and Montague, 2001) as their scores are not guaranteed to have the same ranges. An even worse case is the following scenario. Assume we use rank correlation as our metric. As a consequence, we need our gold ranking to define an order on all the word pairs. However, this also means that we somehow need to order completely unrelated word pairs; for example, we have to decide whether (dog, cat) is more similar than (banana, apple).

3.2 Absolute results

Table 1 presents the results on 14 different datasets for the six embedding models. We excluded examples from datasets that contained words not in our vocabulary. For the relatedness and selective preference tasks, the numbers in the table indicate the correlation coefficient of human scores and the cosine similarity times 100. The numbers for the categorization tasks reflect the purities of the resulting clusters. For the analogy task, we report accuracy.

CBOW outperforms other embeddings on 10 of 14 datasets. CBOW especially excels at the relatedness and analogy tasks, but fails to surpass other models on the selective preferences tasks. Random projection performs worst in 13 out of the 14 tasks, being followed by Hellinger PCA. C&W and TSCCA are similar on average, but differ across datasets. Moreover, although TSCCA and GloVe perform similarly on most tasks, TSCCA suffers disproportionally on the analogy tasks.

3.3 Comparative intrinsic evaluation

In comparative evaluation, users give direct feedback on the embeddings themselves, so we do not have to define a metric that compares scored word pairs. Rather than defining both query and target words, we need only choose query words since the

	relatedness						categorization			sel. prefs		analogy			
	rg	ws	wss	wsr	men	toefl	ap	esslli	batt.	up	mcrae	an	ansyn	ansem	average
CBOW	**74.0**	**64.0**	**71.5**	**56.5**	**70.7**	66.7	**65.9**	**70.5**	**85.2**	24.1	13.9	**52.2**	**47.8**	**57.6**	**58.6**
GloVe	63.7	54.8	65.8	49.6	64.6	**69.4**	64.1	65.9	77.8	27.0	**18.4**	42.2	44.2	39.7	53.4
TSCCA	57.8	54.4	64.7	43.3	56.7	58.3	57.5	**70.5**	64.2	**31.0**	14.4	15.5	19.0	11.1	44.2
C&W	48.1	49.8	60.7	40.1	57.5	66.7	60.6	61.4	80.2	28.3	16.0	10.9	12.2	9.3	43.0
H-PCA	19.8	32.9	43.6	15.1	21.3	54.2	34.1	50.0	42.0	-2.5	3.2	3.0	2.4	3.7	23.1
Rand. Proj.	17.1	19.5	24.9	16.1	11.3	51.4	21.9	38.6	29.6	-8.5	1.2	1.0	0.3	1.9	16.2

Table 1: Results on absolute intrinsic evaluation. The best result for each dataset is highlighted in bold. The second row contains the names of the corresponding datasets.

embeddings themselves will be used to define the comparable target words.

Query inventory. We compiled a diverse inventory of 100 query words that balance frequency, part of speech (POS), and concreteness. First, we selected 10 out of 45 broad categories from WordNet (Miller, 1995). We then chose an equal number of categories that mostly contained abstract concepts and categories that referred to concrete concepts. Among those categories, we had one for adjectives and adverbs each, and four for nouns and verbs each. From each category, we drew ten random words with the restriction that there be exactly three rare words (i.e., occurring fewer than 2500 times in the training corpus) among the ten.

Details. Our experiments were performed with users from Amazon Mechanical Turk (MTurk) that were native speakers of English with sufficient experience and positive feedback on the Amazon Mechanical Turk framework.

For each of the 100 query words in the dataset, the nearest neighbors at ranks $k \in \{1, 5, 50\}$ for the six embeddings were retrieved. For each query word and k, we presented the six words along with the query word to the users. Each Turker was requested to evaluate between 25 and 50 *items* per task, where an item corresponds to the query word and the set of 6 retrieved neighbor words from each of the 6 embeddings. The payment was between \$0.01 and \$0.02 per item. The users were then asked to pick the word that is most similar according to their perception (the instructions were almost identical to the WordSim-353 dataset instructions). Duplicate words were consolidated, and a click was counted for all embeddings that returned that word. An option "I don't know the meaning of one (or several) of the words" was also provided as an alternative. Table 2 shows an example instance that was given to the Turkers.

Query:	**skillfully**			
	(a)	swiftly	(b)	expertly
	(c)	cleverly	(d)	pointedly

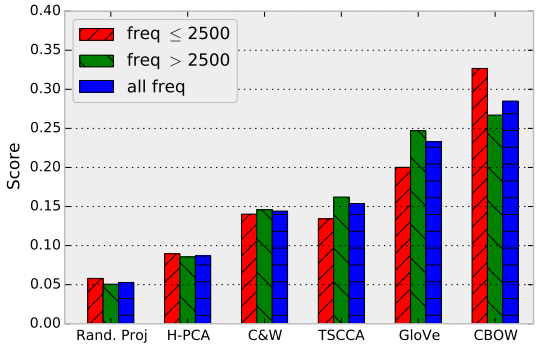
Table 2: Example instance of comparative intrinsic evaluation task. The presented options in this example are nearest neighbors to the query word according to (a) C&W, (b) CBOW, GloVe, TSCCA (c) Rand. Proj. and (d) H-PCA.

The combination of 100 query words and 3 ranks yielded 300 items on which we solicited judgements by a median of 7 Turkers (min=5, max=14). We compare embeddings by average win ratio, where the win ratio was how many times raters chose embedding e divided by the number of total ratings for item i.

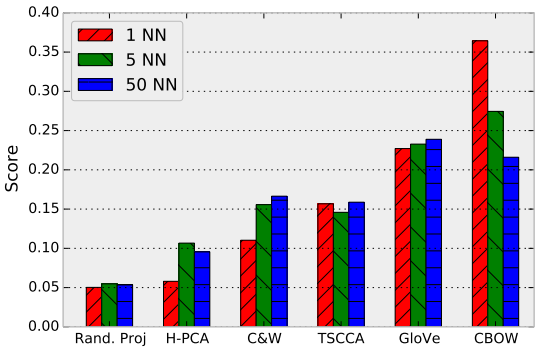
3.4 Comparative results

Overall comparative results replicate previous results. Figure 1(a) shows normalized win ratio scores for each embedding across 3 conditions corresponding to the frequency of the query word in the training corpus. The scores were normalized to sum to one in each condition to emphasize relative differences. CBOW in general performed the best and random projection the worst (p-value < 0.05 for all pairs except H-PCA and C&W in comparing un-normalized score differences for the ALL-FREQ condition with a randomized permutation test). The novel comparative evaluations correspond both in rank and in relative margins to those shown in Table 1.

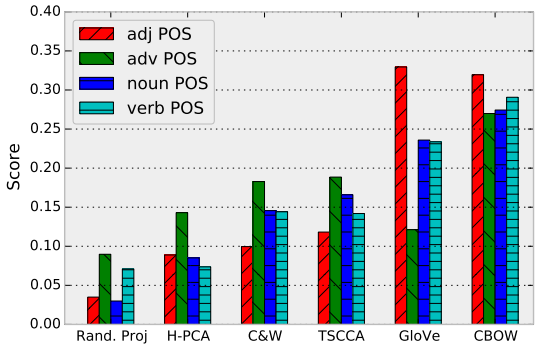
Unlike previous results, we can now show differences beyond the nearest neighbors. Figure 1(b) presents the same results, but this time



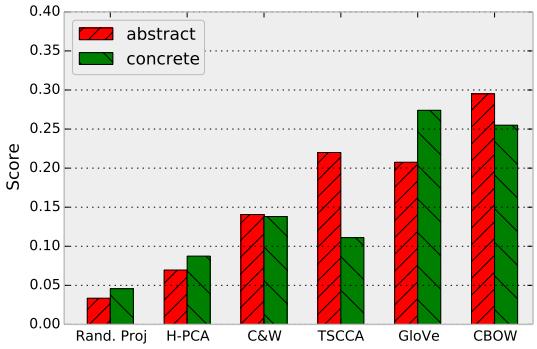
(a) Normalized scores by global word frequency.



(b) Normalized scores by nearest neighbor rank k.



(c) Normalized scores by part of speech.



(d) Normalized scores by category.

Figure 1: Direct comparison task

broken up by the rank k of the neighbors that were compared. CBOW has its strengths especially at rank $k = 1$. For neighbors that appear after that, CBOW does not necessarily produce better embeddings. In fact, it even does worse for $k = 50$ than GloVe. It is important to note, however, that we cannot make absolute statements about how performance behaves across different values of k since each assessment is always relative to the quality of all other embeddings.

We balanced our query inventory also with respect to parts of speech and abstractness vs. concreteness. Figure 1(c) shows the relative performances of all embeddings for the four POS classes (adjectives, adverbs, nouns and verbs). While most embeddings show relatively homogeneous behaviour across the four classes, GloVe suffers disproportionally on adverbs. Moving on to Figure 1(d), we can see a similar behavior for TSCCA: Its performance is much lower on concrete words than on abstract ones. This difference may be important, as recent related work finds that simply differentiating between general and specific terms explains much of the observed variation between embedding methods in hierarchical classification tasks (Levy et al., 2015b). We take the two observations above as evidence that a more fine-grained analysis is necessary in discerning different embedding methods.

As a by-product, we observed that there was no embedding method that consistently performed best on all of the four different absolute evaluation tasks. However, we would like to reiterate that our goal is not to identify one best method, but rather point out that different evaluations (e.g., changing the rank k of the nearest neighbors in the comparison task) result in different outcomes.

4 Coherence

In the relatedness task we measure whether a pair of semantically similar words are near each other in the embedding space. In this novel coherence task we assess whether *groups* of words in a small neighborhood in the embedding space are mutually related. Previous work has used this property for qualitative evaluation using visualizations of 2D projections (Turian et al., 2010), but we are not aware of any work using local neighborhoods for

quantitative evaluation. Good embeddings should have coherent neighborhoods for each word, so inserting a word not belonging to this neighborhood should be easy to spot. Similar to Chang et al. (2009), we presented Turkers with four words, three of which are close neighbors and one of which is an "intruder." For each of the 100 words in our query set of Section 3.3, we retrieved the two nearest neighbors. These words along with the query word defined the set of (supposedly) good options. Table 3 shows an example instance that was given to the Turkers.

(a)	finally	(b)	eventually
(c)	immediately	(d)	put

Table 3: Example instance of intrusion task. The query word is option (a), intruder is (d).

To normalize for frequency-based effects, we computed the average frequency avg of the three words in this set and chose the intruder word to be the first word that had a frequency of $avg \pm 500$ starting at rank 100 of the list of nearest neighbors.

Results. In total, we solicited judgments on 600 items (100 query words for each of the 6 embeddings) from a median of 7 Turkers (min=4, max=11) per item, where each Turker evaluated between 25 and 50 items per task. Figure 2 shows the results of the intrusion experiment. The evaluation measure is micro-averaged precision for an embedding across 100 query words, where per-item precision is defined as the number of raters that discovered the intruder divided the total number of raters of item i. Random guessing would achieve an average precision of 0.25.

All embeddings perform better than guessing, indicating that there is at least some coherent structure captured in all of them. However, the best performing embeddings at this task are TSCCA, CBOW and GloVe (the precision mean differences were not significant under a random permutation test), while TSCCA attains greater precision ($p < 0.05$) in relation to C&W, H-PCA and random projection embeddings. These results are in contrast to the direct comparison study, where the performance of TSCCA was found to be significantly worse than that of CBOW. However, the order of the last three embeddings remains unchanged, implying that performance on the intrusion task and performance on the direct comparison task are correlated. CBOW and C&W seem to do equally well on rare and frequent words, whereas the other models' performance suffers on rare words.

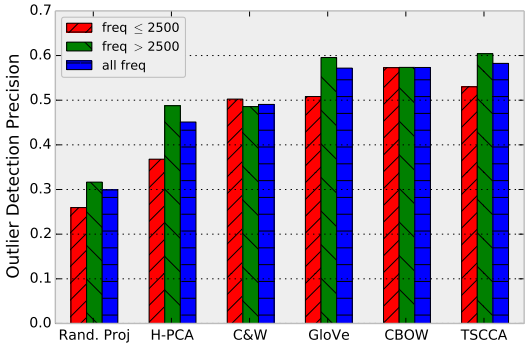


Figure 2: Intrusion task: average precision by global word frequency.

Discussion. Evaluation of set-based properties of embeddings may produce different results from item-based evaluation: rankings we got from the intrusion task did not match the rankings we obtained from the relatedness task. Pairwise similarities seem to be only part of the information that is encoded in word embeddings, so looking at more global measures is necessary for a better understanding of differences between embeddings.

We choose intruder words based on similar but lower-ranked words, so an embedding could score well on this task by doing an unusually bad job at returning less-closely related words. However, the results in Figure 1(b) suggest that there is little differences at higher ranks (rank 50) between embeddings.

5 Extrinsic Tasks

Extrinsic evaluations measure the contribution of a word embedding model to a specific task. There is an implicit assumption in the use of such evaluations that there is a consistent, global ranking of word embedding quality, and that higher quality embeddings will necessarily improve results on any downstream task. We find that this assumption does not hold: different tasks favor different embeddings. Although these evaluations are useful in characterizing the relative strengths of different models, we do not recommend that they be used as a proxy for a general notion of embedding quality.

	dev	test	p-value
Baseline	94.18	93.78	0.000
Rand. Proj.	94.33	93.90	0.006
GloVe	94.28	93.93	0.015
H-PCA	94.48	93.96	0.029
C&W	**94.53**	**94.12**	
CBOW	94.32	93.93	0.012
TSCCA	**94.53**	94.09	0.357

Table 4: F1 chunking results using different word embeddings as features. The p-values are with respect to the best performing method.

	test	p-value
BOW (baseline)	88.90	$7.45{\cdot}10^{-14}$
Rand. Proj.	62.95	$7.47{\cdot}10^{-12}$
GloVe	74.87	$5.00{\cdot}10^{-2}$
H-PCA	69.45	$6.06{\cdot}10^{-11}$
C&W	72.37	$1.29{\cdot}10^{-7}$
CBOW	**75.78**	
TSCCA	75.02	$7.28{\cdot}10^{-4}$

Table 5: F1 sentiment analysis results using different word embeddings as features. The p-values are with respect to the best performing embedding.

Noun phrase chunking. First we use a noun phrase chunking task similar to that used by Turian et al. (2010). The only difference is that we normalize all word vectors to unit length, rather than scaling them with some custom factor, before giving them to the conditional random field (CRF) model as input. We expect that this task will be more sensitive to syntactic information than to semantic information.

Sentiment classification. Second we use a recently released dataset for binary sentiment classification by Maas et al. (2011). The dataset contains 50K movie reviews with a balanced distribution of binary polarity labels. We evaluate the relative performance of word embeddings at this task as follows: we generate embedding-only features for each review by computing a linear combination of word embeddings weighted by the number of times that the word appeared in the review (using the same bag-of-words features as Maas et al. (2011)). A LIBLINEAR logistic regression model (Fan et al., 2008) with the default parameters is trained and evaluated using 10 fold cross-validation. A vanilla bag of words feature set is the baseline (denoted as BOW here). We expect that this task will be more sensitive to semantic information than syntactic information.

Results. Table 4 shows the average F1-scores for the chunking task. The p-values were computed using randomization (Yeh, 2000) on the sentence level. First, we can observe that adding word vectors as features results in performance lifts with all embeddings when compared to the baseline. The performance of C&W and TSCCA is statistically not significant, and C&W does better than all the remaining methods at the $p = 0.05$ level. Surprisingly, although the performance of Random Projections is still last, the gap to GloVe and CBOW is now very small. Table 5 shows results on the sentiment analysis task. We recover a similar order of embeddings as in the absolute intrinsic evaluation, however, the order of TSCCA and GloVe is now reversed.

Discussion. Performance on downstream tasks is not consistent across tasks, and may not be consistent with intrinsic evaluations. Comparing performance across tasks may provide insight into the information encoded by an embedding, but we should not expect any specific task to act as a proxy for abstract quality. Furthermore, if good downstream performance is really the goal of an embedding, we recommend that embeddings be trained specifically to optimize a specific objective (Lebret and Collobert, 2014).

6 Discussion

We find consistent differences between word embeddings, despite the fact that they are operating on the same input data and optimizing arguably very similar objective functions (Pennington et al., 2014; Levy and Goldberg, 2014). Recent work suggests that many apparent performance differences on specific tasks are due to a lack of hyperparameter optimization (Levy et al., 2015a). Different algorithms are, in fact, encoding surprisingly different information that may or may not align with our desired use cases. For example, we find that embeddings encode differing degrees of information about word frequency, even after length normalization. This result is surprising for two reasons. First, many algorithms reserve distinct "intercept" parameters to absorb frequency-based effects. Second, we expect that the geometry of the embedding space will be primar-

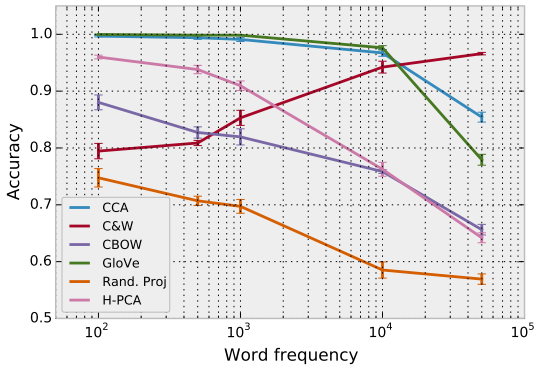


Figure 3: Embeddings can accurately predict whether a word is frequent or rare.

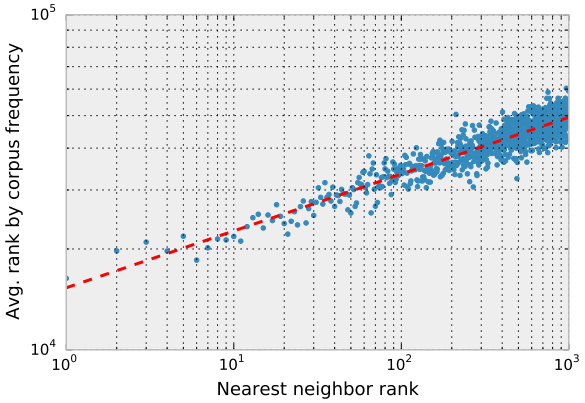


Figure 4: Avg. word rank by frequency in training corpus vs. nearest-neighbor rank in the C&W embedding space.

ily driven by semantics: the relatively small number of frequent words should be evenly distributed through the space, while large numbers of rare, specific words should cluster around related, but more frequent, words.

We trained a logistic regression model to predict word frequency categories based on word vectors. The linear classifier was trained to put words either in a frequent or rare category, with thresholds varying from 100 to 50,000. At each threshold frequency, we sampled the training sets to ensure a consistent balance of the label distribution across all frequencies. We used length-normalized embeddings, as rare words might have shorter vectors resulting from fewer updates during training (Turian et al., 2010). We report the mean accuracy and standard deviation (1σ) using five-fold cross-validation at each threshold frequency in Figure 3.

All word embeddings do better than random, suggesting that they contain some frequency information. GloVe and TSCCA achieve nearly 100% accuracy on thresholds up to 1000. Unlike all other embeddings, accuracy for C&W embeddings increases for larger threshold values. Further investigation revealed that the weight vector direction changes gradually with the threshold frequency — indicating that frequency seems to be encoded in a smooth way in the embedding space.

Although GloVe and CBOW are the two best performing embeddings on the intrinsic tasks, they differ vastly in the amount of frequency information they encode. As a consequence, we can conclude that most of the differences in frequency prediction are not due to intrinsic properties of natural language: it is not the case that frequent words naturally have only frequent neighbors.

Word frequency information in the embedding space also affects cosine similarity. For each of the words in the WordSim-353 dataset, we queried for the $k = 1000$ nearest neighbors. We then looked up their frequency ranks in the training corpus and averaged those ranks over all the query words. We found a strong correlation between the frequency of a word and its position in the ranking of nearest neighbors in our experiments. Figure 4 shows a power law relationship for C&W embeddings between a word's nearest neighbor rank (w.r.t. a query) and the word's frequency rank in the training corpus (nn-rank $\sim$ 1000 $\cdot$ corpus-rank$^{0.17}$). This is a concern: the frequency of a word in the language plays a critical role in word processing of humans as well (Cattell, 1886). As a consequence, we need to explicitly consider word frequency as a factor in the experiment design. Also, the above results mean that the commonly-used cosine similarity in the embedding space for the intrinsic tasks gets polluted by frequency-based effects. We believe that further research should address how to better measure linguistic relationships between words in the embedding space, e.g., by learning a custom metric.

7 Related work

Mikolov et al. (2013b) demonstrate that certain linguistic regularities exist in the embedding space. The authors show that by doing simple vector arithmetic in the embedding space, one can solve various syntactic and semantic analogy tasks. This is different to previous work, which phrased the analogy task as a classification problem (Turney, 2008). Surprisingly, word embed-

dings seem to capture even more complex linguistic properties. Chen et al. (2013) show that word embeddings even contain information about regional spellings (UK vs. US), noun gender and sentiment polarity.

Previous work in evaluation for word embeddings can be divided into intrinsic and extrinsic evaluations. Intrinsic evaluations measure the quality of word vectors by directly measuring correlation between semantic relatedness and geometric relatedness, usually through inventories of query terms. Focusing on intrinsic measures, Baroni et al. (2014) compare word embeddings against distributional word vectors on a variety of query inventories and tasks. Faruqui and Dyer (2014) provide a website that allows the automatic evaluation of embeddings on a number of query inventories. Gao et al. (2014) publish an improved query inventory for the analogical reasoning task. Finally, Tsvetkov et al. (2015) propose a new intrinsic measure that better correlates with extrinsic performance. However, all these evaluations are done on precollected inventories and mostly limited to local metrics like relatedness.

Extrinsic evaluations use embeddings as features in models for other tasks, such as semantic role labeling or part-of-speech tagging (Collobert et al., 2011), and improve the performance of existing systems (Turian et al., 2010). However, they have been less successful at other tasks such as parsing (Andreas and Klein, 2014).

More work has been done in unsupervised semantic modeling in the context of topic models. One example is the *word intrusion* task (Chang et al., 2009), in which annotators are asked to identify a random word inserted into the set of high probability words for a given topic. Word embeddings do not produce interpretable dimensions, so we cannot directly use this method, but we present a related task based on nearest neighbors. Manual evaluation is expensive and time-consuming, but other work establishes that automated evaluations can closely model human intuitions (Newman et al., 2010).

8 Conclusions

There are many factors that affect word embedding quality. Standard aggregate evaluations, while useful, do not present a complete or consistent picture. Factors such as word frequency play a significant and previously unacknowledged role. Word frequency also interferes with the commonly-used cosine similarity measure. We present a novel evaluation framework based on direct comparisons between embeddings that provides more fine-grained analysis and supports simple, crowdsourced relevance judgments. We also present a novel *Coherence* task that measures our intuition that neighborhoods in the embedding space should be semantically or syntactically related. We find that extrinsic evaluations, although useful for highlighting specific aspects of embedding performance, should not be used as a proxy for generic quality.

Acknowledgments

This research was funded in part through NSF Award IIS-1513692. We would like to thank Alexandra Schofield, Adith Swaminathan and all other members of the NLP seminar for their helpful feedback.

References

Jacob Andreas and Dan Klein. 2014. How much do word embeddings encode about syntax? In *ACL: Short Papers*, pages 822–827.

Javed Aslam and Mark Montague. 2001. Models for metasearch. In *SIGIR*, pages 276–284.

Marco Baroni, Georgiana Dinu, and Germán Kruszewski. 2014. Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *ACL*, pages 238–247.

Elia Bruni, Nam-Khanh Tran, and Marco Baroni. 2014. Multimodal distributional semantics. *JAIR*, 49:1–47.

James McKeen Cattell. 1886. The time taken up by cerebral operations. *Mind,*, (42):220–242.

Jonathan Chang, Jordan Boyd-Graber, Sean Gerrish, Chong Wang, and David Blei. 2009. Reading tea leaves: How humans interpret topic models. In *NIPS*, pages 288–296.

Yanqing Chen, Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2013. The expressive power of word embeddings. *arXiv preprint: 1408.3456*.

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

Paramveer S. Dhillon, Jordan Rodu, Dean P. Foster, and Lyle H. Ungar. 2012. Two step CCA: A new spectral method for estimating vector models of words. In *ICML*, pages 1551–1558.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A library for large linear classification. *JLMR*, 9:1871–1874.

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

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

Bin Gao, Jiang Bian, and Tie-Yan Liu. 2014. WordRep: A benchmark for research on learning word representations. *ICML Workshop on Knowledge-Powered Deep Learning for Text Mining*.

Rémi Lebret and Ronan Collobert. 2014. Word embeddings through Hellinger PCA. In *EACL*, pages 482–490.

Omer Levy and Yoav Goldberg. 2014. Neural word embedding as implicit matrix factorization. In *NIPS*, pages 2177–2185.

Omer Levy, Yoav Goldberg, and Ido Dagan. 2015a. Improving distributional similarity with lessons learned from word embeddings. *TACL*.

Omer Levy, Steffen Remus, Chris Biemann, and Ido Dagan. 2015b. Do supervised distributional methods really learn lexical inference relations? In *NAACL*.

Ping Li, Trevor J Hastie, and Kenneth W Church. 2006. Very sparse random projections. In *KDD*, pages 287–296.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *HLT-ACL*, pages 142–150.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *ACL: System Demonstrations*, pages 55–60.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013a. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119.

Tomas Mikolov, Wen-Tau Yih, and Geoffrey Zweig. 2013b. Linguistic regularities in continuous space word representations. In *HLT-NAACL*, pages 746–751.

George A Miller. 1995. WordNet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

David Newman, Jey Han Lau, Karl Grieser, and Timothy Baldwin. 2010. Automatic evaluation of topic coherence. In *HLT-NAACL*, pages 100–108.

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

Yulia Tsvetkov, Manaal Faruqui, Wang Ling, Guillaume Lample, and Chris Dyer. 2015. Evaluation of word vector representations by subspace alignment. In *EMNLP*.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *ACL*, pages 384–394.

Peter D. Turney. 2008. A uniform approach to analogies, synonyms, antonyms, and associations. In *COLING*, pages 905–912.

Alexander Yeh. 2000. More accurate tests for the statistical significance of result differences. In *ACL*, pages 947–953.

Efficient Methods for Incorporating Knowledge into Topic Models

Yi Yang, Doug Downey
Electrical Engineering and Computer Science
Northwestern University
Evanston, IL
yiyang@u.northwestern.edu
ddowney@eecs.northwetsern.edu

Jordan Boyd-Graber
Computer Science
University of Colorado
Boulder, CO
Jordan.Boyd.Graber
@colorado.edu

Abstract

Latent Dirichlet allocation (LDA) is a popular topic modeling technique for exploring hidden topics in text corpora. Increasingly, topic modeling needs to scale to larger topic spaces and use richer forms of prior knowledge, such as word correlations or document labels. However, inference is cumbersome for LDA models with prior knowledge. As a result, LDA models that use prior knowledge only work in small-scale scenarios. In this work, we propose a factor graph framework, Sparse Constrained LDA (SC-LDA), for efficiently incorporating prior knowledge into LDA. We evaluate SC-LDA's ability to incorporate word correlation knowledge and document label knowledge on three benchmark datasets. Compared to several baseline methods, SC-LDA achieves comparable performance but is significantly faster.

1 Challenge: Leveraging Prior Knowledge in Large-scale Topic Models

Topic models, such as Latent Dirichlet Allocation (Blei et al., 2003, LDA), have been successfully used for discovering hidden topics in text collections. LDA is an unsupervised model—it requires no annotation—and discovers, without any supervision, the thematic trends in a text collection. However, LDA's lack of supervision can lead to disappointing results. Often, the hidden topics learned by LDA fail to make sense to end users. Part of the problem is that the objective function of topic models does not always correlate with human judgments of topic quality (Chang et al., 2009). Therefore, it's often necessary to incorporate prior knowledge into topic models to improve the model's performance. Recent work has also shown that by interactive human feedback can improve the quality and stability of topics (Hu and Boyd-Graber, 2012; Yang et al., 2015). Information about documents (Ramage et al., 2009) or words (Boyd-Graber et al., 2007) can improve LDA's topics.

In addition to its occasional inscrutability, scalability can also hamper LDA's adoption. Conventional Gibbs sampling—the most widely used inference for LDA—scales linearly with the number of topics. Moreover, accurate training usually takes many sampling passes over the dataset. Therefore, for large datasets with millions or even billions of tokens, conventional Gibbs sampling takes too long to finish. For standard LDA, recently introduced fast sampling methods (Yao et al., 2009; Li et al., 2014; Yuan et al., 2015) enable industrial applications of topic modeling to search engines and online advertising, where capturing the "long tail" of infrequently used topics requires large topic spaces. For example, while typical LDA models in academic papers have up to 10^3 topics, industrial applications with 10^5–10^6 topics are common (Wang et al., 2014). Moreover, scaling topic models to many topics can also reveal the hierarchical structure of topics (Downey et al., 2015).

Thus, there is a need for topic models that can both benefit from rich prior information and that can scale to large datasets. However, existing methods for improving scalability focus on topic models without prior information. To rectify this, we propose a factor graph model that encodes a potential function over the hidden topic variables, encouraging topics consistent with prior knowledge. The factor model representation admits an efficient sampling algorithm that takes advantage of the model's sparsity. We show that our method achieves comparable performance but runs significantly faster than baseline methods, enabling mod-

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 308–317,
Lisbon, Portugal, 17-21 September 2015.

els to discover models with many topics enriched by prior knowledge.

2 Efficient Algorithm for Incorporating Knowledge into LDA

In this section, we introduce the factor model for incorporating prior knowledge and show how to efficiently use Gibbs sampling for inference.

2.1 Background: LDA and SparseLDA

A statistical topic model represents words in documents in a collection D as mixtures of T topics, which are multinomials over a vocabulary of size V. In LDA, each document d is associated with a multinomial distribution over topics, θ_d. The probability of a word type w given topic z is $\phi_{w|z}$. The multinomial distributions θ_d and ϕ_z are drawn from Dirichlet distributions: α and β are the hyperparameters for θ and ϕ. We represent the document collection D as a sequence of words $\mathbf{w}$, and topic assignments as $\mathbf{z}$. We use symmetric priors α and β in the model and experiment, but asymmetric priors are easily encoded in the models (Wallach et al., 2009).

Discovering the latent topic assignments $\mathbf{z}$ from observed words $\mathbf{w}$ requires inferring the the posterior distribution $P(\mathbf{z}|\mathbf{w})$. Griffiths and Steyvers (2004) propose using collapsed Gibbs sampling. The probability of a topic assignment $z = t$ in document d given an observed word type w and the other topic assignments z_- is

$$P(z = t|\mathbf{z}_-, w) \propto (n_{d,t} + \alpha)\frac{n_{w,t} + \beta}{n_t + V\beta} \quad (1)$$

where $\mathbf{z}_-$ are the topic assignments of all other tokens. This conditional probability is based on cumulative counts of topic assignments: $n_{d,t}$ is the number of times topic t is used in document d, $n_{w,t}$ is the number of times word type w is used in topic t, and n_t is the marginal count of the number of tokens assigned to topic t.

Unfortunately, explicitly computing the conditional probability is quite for models with many topics. The time complexity of drawing a sample by Equation 1 is linear to the number of topics. Yao et al. (2009) propose a clever factorization of Equation 1 so that the complexity is typically sublinear by breaking the conditional probability into three "buckets":

$$\sum_t P(z = t|\mathbf{z}_-, w) = \underbrace{\sum_t \frac{\alpha\beta}{n_t + V\beta}}_{\text{s}} + \underbrace{\sum_{t,n_{d,t}>0} \frac{n_{d,t}\beta}{n_t + V\beta}}_{\text{r}} + \underbrace{\sum_{t,n_{w,t}>0} \frac{(n_{d,t} + \alpha)n_{w,t}}{n_t + V\beta}}_{\text{q}}. \quad (2)$$

The first term s is the "smoothing only" bucket—constant for all documents. The second term r is the "document only" bucket that is shared by a document's tokens. Both s and r have simple constant time updates. The last term q has to be computed specifically for each token, only for the few types with non-zero counts in a topic, due to the sparsity of word-topic count. Since q often has the largest mass and few non-zero terms, we start the sampling from bucket q.

2.2 A Factor Model for Incorporating Prior Knowledge

With SparseLDA, inferring LDA models over large topic spaces becomes tractable. However, existing methods for incorporating prior knowledge use conventional Gibbs sampling, which hinders inference. We address this limitation in this section by adding a factor graph to encode prior knowledge.

LDA assumes that the hidden topic assignment of a word is independent from other hidden topics, given the document's topic distribution θ. While this assumption facilitates computational efficiency, it loses the rich correlation between words. In many scenarios, users have external knowledge regarding word correlation, document labels, or document relations, which can reshape topic models and improve coherence.

Prior knowledge can constrain what models discover. A correlation between two words v and w indicates that they have a similar topic distribution, i.e., $p(z|v) \approx p(z|w)$.[1] Therefore, the posterior topic assignments v and w will be correlated. In contrast, if v and w are uncorrelated, nothing—other than the Dirichlet's rich get richer effect—prevents the topics from diverging. Similarly, if two documents share a label, then it is reasonable

[1] In (Andrzejewski et al., 2009) two correlated words are taken to indicate that $p(v|z) \approx p(w|z)$. However, for word types that have very different frequencies, these two quantities would never be close, and thus $p(z|v) \approx p(z|w)$ is a more intuitive constraint.

to assume that they are more likely than two random documents to share topics.

We denote the set of prior knowledge as M. Each prior knowledge $m \in M$ defines a potential function $f_m(z, w, d)$ of the hidden topic z of word type w in document d with which m is associated. Therefore, the complete prior knowledge M defines a score on the current topic assignments $\mathbf{z}$:

$$\psi(\mathbf{z}, M) = \prod_{z \in \mathbf{z}} \exp f_m(z, w, d) \tag{3}$$

If m is knowledge about word type w, then $f_m(z, w, d)$ applies to all hidden topics of word w. If m is knowledge about document d, then $f_m(z, w, d)$ applies to all topics that are in document d. The potential function assigns large values to the topics that accord with prior knowledge but penalizes the topic assignments that disagree with the prior knowledge. In an extreme case, if a prior knowledge m says word type w in document d is Topic 3, then the potential function $f_m(z, w, d)$ is zero for all topics but Topic 3.

Since the potential function ψ is a function of $\mathbf{z}$, and it is only a real-value score of current topic assignments, the potential can be factored out of the marginalized joint:

$$\begin{aligned} &P(\mathbf{w}, \mathbf{z}|\alpha, \beta, M) = P(\mathbf{w}|\mathbf{z}, \beta) P(\mathbf{z}|\alpha) \psi(\mathbf{z}, M) \qquad (4) \\ &= \int_\theta \int_\phi p(\mathbf{w}|\mathbf{z}, \phi) p(\phi|\beta) p(\mathbf{z}|\theta) p(\theta|\alpha) \psi(\mathbf{z}, M) d\theta d\phi \\ &= \psi(\mathbf{z}, M) \int_\theta \int_\phi p(\mathbf{w}|\mathbf{z}, \phi) p(\phi|\beta) p(\mathbf{z}|\theta) p(\theta|\alpha) d\theta d\phi. \end{aligned}$$

Given the joint likelihood and observed data, the goal is evaluate the posterior $P(\mathbf{z}|\mathbf{w})$. Computing $P(\mathbf{z}|\mathbf{w})$ involves evaluating a probability distribution on a large discrete state space: $P(\mathbf{z}|\mathbf{w}) = P(\mathbf{z}, \mathbf{w}) / \sum_{\mathbf{z}} P(\mathbf{z}, \mathbf{w})$. Griffiths and Steyvers (2004)—mirroring the original inspirations for Gibbs sampling (Geman and Geman, 1990)—draw an analogy to statistical physics, viewing standard LDA as a system that favors configurations $\mathbf{z}$ that compromise between having few topics per document and having few words per topic, with the terms of this compromise being set by the hyperparameters α and β. Our factor model representation of prior knowledge adds a further constraint that asks the model to also consider ensembles of topic assignments $\mathbf{z}$ that are compatible with a standard LDA model *and* the given prior knowledge.

The collapsed Gibbs Sampling for inferring topic assignment z of word w in document d is:

$$\begin{aligned} &P(z = t|w, \mathbf{z}_-, M) \qquad (5) \\ &= \frac{P(\mathbf{w}, \mathbf{z}_-, z = t|\alpha, \beta, M)}{P(\mathbf{w}, \mathbf{z}_-|\alpha, \beta, M)} \\ &= \frac{P(\mathbf{w}, \mathbf{z}_-, z = t)}{P(\mathbf{w}, \mathbf{z}_-)} \frac{\psi(\mathbf{z}_-, z = t, M)}{\psi(\mathbf{z}_-, M)} \\ &\propto \left\{ (n_{d,t} + \alpha) \frac{n_{w,t} + \beta}{n_t + W\beta} \right\} \frac{\psi(\mathbf{z}_-, z = t, M)}{\psi(\mathbf{z}_-, M)} \\ &\propto \left\{ (n_{d,t} + \alpha) \frac{n_{w,t} + \beta}{n_t + W\beta} \right\} \exp f_m(z = t, w, d). \end{aligned}$$

The first term is identical to standard LDA, and admits efficient computation using SparseLDA. However, if the second term, $\exp f_m(z, w, d)$, is dense, we still need to compute it explicitly T times (once for each topic) because we need the summation of $P(z = t)$ for sampling. Therefore, the critical part of speeding up the sampler is finding a sparse representation of the second term. In the following sections, we show that natural, sparse prior knowledge representations are possible. We first present an efficient sparse representation of word correlation prior knowledge and then one for document-label knowledge.

2.3 Word Correlation Prior Knowledge

We now illustrate how we can encode word correlation knowledge as a set of sparse constraints $f_m(z, w, d)$ in our model. In previous work (Andrzejewski et al., 2009; Hu et al., 2011; Xie et al., 2015), word correlation prior knowledge is represented as word must-link constraints and cannot-link constraints. A must-link relation between two words indicates that the two words tend to be related to the same topics, i.e. their topic probabilities are correlated. In contrast, a cannot-link relation between two words indicates that these two words are not topically similar, and they should not both be prominent within the same topic. For example, "quarterback" and "fumble" are both related to American football, so they can share a must-link relation. But "fumble" and "bank" imply two different topics, so they share a cannot-link.

Let us say word w is associated with a set of prior knowledge correlations M_w. Each prior knowledge $m \in M_w$ is a word pair (w, w'), and it has "topic preference" of w given its correlation word w'. The must-link set of w is M_w^m, and the cannot-link set of w is M_w^c, i.e., $M_w = M_w^c \bigcup M_w^m$. In the example above, $M_{fumble}^m =$

$\{quarterback\}$, and $M^c_{fumble} = \{bank\}$, so $M_{fumble} = \{quarterback, bank\}$. The topic assignment of word "fumble" has higher conditional probability for the same topics as "quarterback" but lower probability for topics containing "bank".

The potential score of sampling topic t for word type w—if M_w is not empty—is

$$f_m(z, w, d) = \sum_{u \in M^m_w} \log \max(\lambda, n_{u,z}) + \sum_{v \in M^c_w} \log \frac{1}{\max(\lambda, n_{v,z})}. \quad (6)$$

where λ is a hyperparameter, which we call the correlation strength. The intuitive explanation of Equation 6 is that the prior knowledge about the word type w will make an impact on the conditional probability of sampling the hidden topic z. Unlike standard LDA where every word's hidden topic is independent of other words given θ, Equation 6 instead increases the probability that a word w will be drawn from the same topics as those of w's must-link word set, and decreases its probability of being drawn from the same topics as those of w's cannot-link word set.

The hyperparameter λ controls the strength of each piece of prior knowledge. The smaller λ is, the stronger this correlation is. For large λ, the constraint is inactive for topics except those with the large counts. As λ decreases, the constraint becomes active for topics with lesser counts. We can adjust the value of λ for each piece of prior knowledge based on our confidence. In our experiments, for simplicity, we use the same value λ for all knowledge and set $\lambda = 1$.

From Equation 6 and Equation 5, the conditional probability of a topic z in document d given an observed word type w is:

$$P(z = t|w, \mathbf{z}_-, M) \propto \left\{ \frac{\alpha\beta}{n_t + V\beta} + \frac{n_{d,t}\beta}{n_t + V\beta} + \frac{(n_{d,t} + \alpha)n_{w,t}}{n_t + V\beta} \right\} \left\{ \prod_{u \in M^m_w} \max(\lambda, n_{u,t}) \prod_{v \in M^c_w} \frac{1}{\max(\lambda, n_{v,t})} \right\} \quad (7)$$

As explained above, λ controls the "strength" of the prior knowledge term. If λ is large, the prior knowledge has little impact on the conditional probability of topic assignments.

Let's return to the question whether Equation 6 is sparse, allowing efficient computation of Equation 7. Fortunately, $n_{u,t}$ and $n_{v,t}$, which are the topic counts for must-link word u and cannot-link word v, are often sparse. For example, in a 100-topic model trained on the NIPS dataset, 87.2% of word types have fewer than ten topics with nonzero counts. In a 500-topic model trained on a larger dataset like the New York Times News (Sandhaus, 2008), 81.9% of word types have fewer than 50 topics with nonzero counts. Moreover, the model becomes increasingly sparse with additional Gibbs iterations. Figure 1 shows the word frequency histogram of nonzero topic counts of NYT-News dataset.

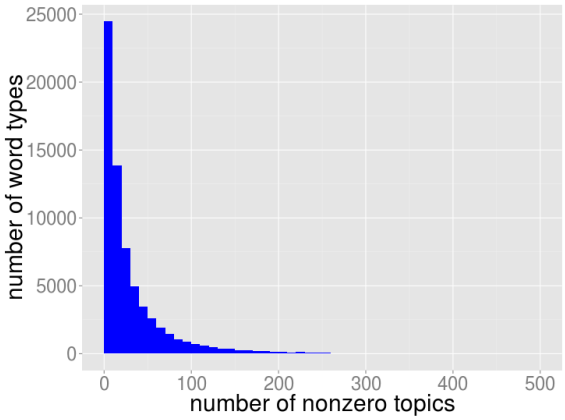


Figure 1: Histogram of nonzero topic counts for word types in NYT-News dataset after inference. 81.9% word types have fewer than 50 topics with nonzero counts. This sparsity allows our sparse constraints to speed inference.

Therefore, the computational cost of Equation 7 can be reduced. SparseLDA efficiently computes the s, r, q bins as in Equation 3. Then for words that are associated with prior knowledge, we update s, r, q with an additional potential term. We only need to compute the potential term for the topics whose counts are greater than λ. The collapsed Gibbs sampling procedure is summarized in Algorithm 1.

Algorithm 1 Gibbs Sampling for word type w in document d, given w's correlation set M_w

1: compute s_t, r_t, q_t with SparseLDA, (see Eq. 3)
2: **for** $t \leftarrow 0$ **to** T **do**
3: update s_t, r_t, q_t. $\forall u \in M_w$ if $n_{u,t} > \lambda$
4: **end for**
5: $p(t) = s_t + r_t + q_t$
6: sample new topic assignment for w from $p(t)$

2.4 Other Types of Prior Knowledge

The factor model framework can also handle other types of prior knowledge, such as document labels, sentence labels, and document link relations. We briefly describe document labels here.

Ramage et al. (2009) propose Labeled-LDA, which improves LDA with document labels. It assumes that there is a one-to-one mapping between topics and labels, and it restricts each document's topics to be sampled only from those allowed by the documents label set. Therefore, Labeled-LDA can be expressed in our model. We define

$$f_m(z, w, d) = \begin{cases} 1, & \text{if } z \in m_d \\ -\infty, & \text{else} \end{cases} \quad (8)$$

where m_d specifies document d's label set converted to corresponding topic labels. Since $f_m(z, w, d)$ is sparse, we can speed up the training as well. Sentence-level prior knowledge (e.g., for sentiment or aspect models (Paul and Girju, 2010)) can be defined in a similar way.

Documents can be associated with other useful metadata. For example, a scientific paper and the prior work it cites might have similar topics (Dietz et al., 2007) or friends in a social network might talk about the same topics (Chang and Blei, 2009). To model link relations, we can use Equation 6 and replace the word-topic counts $n_{v,z}$ with document-topic counts $n_{d,z}$. By doing so, we encourage related documents to have similar topic structures. Moreover, the document-topic count is also sparse, which fits into the efficient learning framework.

Therefore, for different types of prior knowledge, as long as we can define $\psi(\mathbf{z}, M)$ appropriately so that $f(z, w, d)$ are sparse, we are able to speed up learning.

3 Experiments

In this section, we demonstrate the effectiveness of our SC-LDA by comparing it with several baseline methods on three benchmark datasets. We first evaluate the convergence rate of each method and then evaluate the learned model parameter ϕ—the topic-word distribution—in terms of topic coherence. We show that SC-LDA can achieve results comparable to the baseline models but is significantly faster. We set up all experiments on a 8-Core 2.8GHz CPU, 16GB RAM machine.[2]

[2]Our implementation of SC-LDA is available at `https://github.com/yya518/sparse-constrained-lda`.

DATASET	DOCS	TYPE	TOKEN(APPROX)
NIPS	1,500	12,419	1,900,000
NYT-NEWS	3,000,000	102,660	100,000,000
20NG	18,828	21,514	1,946,000

Table 1: Characteristics of benchmark datasets. We use NIPS and NYT for word correlation experiments and 20NG for document label experiments.

3.1 Dataset

We use the NIPS and NYT-News datasets from the UCI bag of words data collections.[3] These two datasets have no document labels, and we use them for word correlation experiments. We also use the 20Newsgroup (20NG) dataset,[4] which has document labels, for document label experiments. Table 1 shows the characteristics of each dataset. Since NIPS and NYT-News have already been preprocessed, to ensure repeatability, we use the data "as they are" from the sources. For 20NG, we perform tokenization and stopword removal using Mallet (McCallum, 2002) and remove words that appear fewer than 10 times.

3.2 Prior Knowledge Generation

Word Correlation Prior Knowledge Previous work proposes two methods to automatically generate prior word correlation knowledge from external sources. Hu and Boyd-Graber (2012) use WordNet 3.0 to obtain synsets for word types, and then if a synset is also in the vocabulary, they add a must-link correlation between the word type and the synset. Xie et al. (2015) use a different method that takes advantage of an existing pretrained word embedding. Each word embedding is a real-valued vector capturing the word's semantic meaning based on distributional similarity. If the similarity between the embeddings of two word types in the vocabulary exceeds a threshold, they generate a must-link between the two words.

In our experiments, we adopt a hybrid method that combines the above two methods. For a noun word type, we first obtain its synsets from WordNet 3.0. We also obtain the embeddings of each word from word2vec (Mikolov et al., 2013). If the synset is also in the vocabulary, and the similarity between the synset and the word is higher than a threshold, which in our experiment is 0.2, we generate a must-link between thee words. Empir-

[3]https://archive.ics.uci.edu/ml/datasets/Bag+of+Words

[4]http://qwone.com/ jason/20Newsgroups/

ically, this hybrid method is able to obtain high quality correlated words. For example, for the NIPS dataset, the must-links we obtain for *randomness* are {noise, entropy, stochasticity}.

Document Label Prior Knowledge Since documents in the 20NG dataset are associated with labels, we use the labels directly as prior knowledge.

3.3 Baselines

The baseline methods for incorporating word correlation prior knowledge in our experiments are as follows:

DF-LDA: incorporates word must-links and cannot-links using a Dirichlet Forest prior in LDA (Andrzejewski et al., 2009). Here we use Hu and Boyd-Graber (2012)'s efficient implementation *FAST-RB-SDW* for DF-LDA.

Logic-LDA: encodes general domain knowledge as first-order logic and incorporates it in LDA (Andrzejewski et al., 2011). Logic-LDA has been used for word correlations and document label knowledge.

MRF-LDA: encodes word correlations in LDA as a Markov random field (Xie et al., 2015).

We also use Mallet's SparseLDA implementation for vanilla LDA in the topic coherence experiment. We use a symmetric Dirichlet prior for all models. We set $\alpha = 1.0$, $\beta = 0.01$. For DF-LDA, $\eta = 100$. For Logic-LDA, we use the default parameter setting in the package: a sample rate of 1.0 and step rate of 10.0. For MRF-LDA, we use the default setting with $\gamma = 1.0$. (Parameter semantics can be found in the original papers.)

3.4 Convergence

The main advantage of our method over other existing methods is efficiency. In this experiment, we show the change of our model's log likelihood over time. In topic models, the log likelihood change is a good indicator of whether a model has converged or not. Figure 2 shows the log likelihood change over time for SC-LDA and three baseline methods on NIPS and NYT-News dataset. SC-LDA converges faster than all the other methods.

We also conduct experiments on SC-LDA with varying numbers of word correlations. Table 2 shows the Gibbs sampling iteration time on the 1st, 50th, 100th and the 200th iteration. We also incorporate different numbers of word correlations

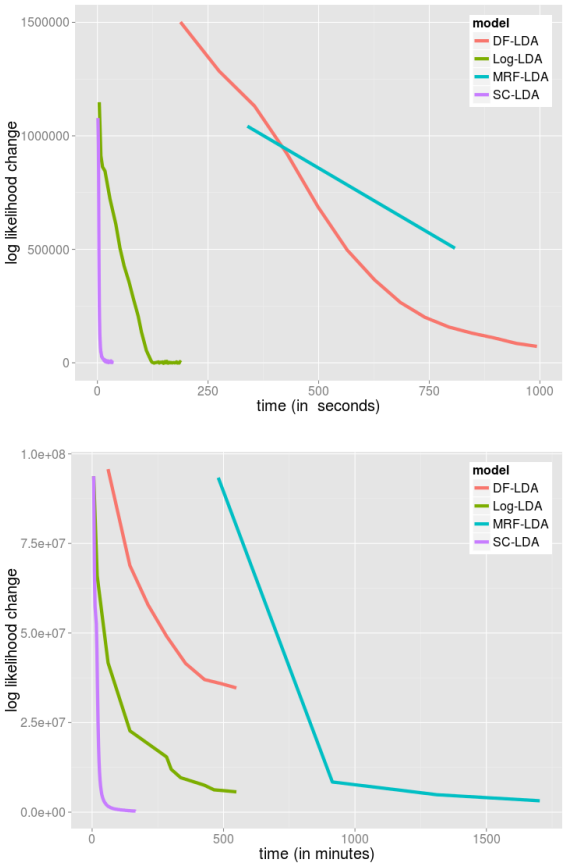


Figure 2: Models' log likelihood convergence on NIPS dataset (above) and NYT-News dataset (below). For NIPS, a 100-topic model with 100 must-links is trained. For NYT-News, a 500-topic model with 100 must-links is trained. SC-LDA reaches likelihood convergence much more rapidly than the other methods.

	# Word Correlations			
round	C0	C100	C500	C1000
1st iteration	2.02	2.14	2.30	2.50
50th iteration	0.53	0.56	0.58	0.62
100th iteration	0.48	0.50	0.53	0.56
200th iteration	0.48	0.49	0.52	0.56

Table 2: SC-LDA runtime (in seconds) in the 1st, 50th, 100th, and 200th iteration with different numbers of correlations.

in SC-LDA. SC-LDA runs faster as sampling proceeds as the sparsity increases, but additional correlations slow the model.

3.5 Topic Coherence

Topic models are often evaluated using perplexity on held-out test data, but this evaluation is of-

ten at odds with human evaluations (Chang et al., 2009). Following Mimno et al. (2011), we employ Topic Coherence—a metric that is consistent with human judgment—to measure a topic model's quality. Topic t's coherence is defined as $C(t : V^{(t)}) = \sum_{m=2}^{M} \sum_{l=1}^{m-1} \log \frac{F(v_m^{(t)}, v_l^{(t)}) + \epsilon}{F(v_l^{(t)})}$, where $F(v)$ is the document frequency of word type v, $F(v, v')$ is the co-document frequency of word type v and v', and $V^{(t)} = (v_1^{(t)}, ..., v_M^{(t)})$ is a list of the M most probable words in topic t. In our experiments, we choose the ten words with highest probability in the topic to compute topic coherence, i.e., $M = 10$. Mimno et al. (2011) use $\epsilon = 1$, but Röder et al. (2015) show smaller ϵ (such as 10^{-12}) improves coherence stability, so we set $\epsilon = 10^{-12}$. Larger topic coherence scores imply more coherent topics.

We train a 500-topic model on the NIPS dataset with different methods and compare the average topic coherence score and the average of the top twenty topic coherence scores. Since the topics learned by topic model often contain "bad" topics (Mimno et al., 2011) which do not make sense to end users, evaluating the top twenty topics reflects the model's performance. We let each model train for one hour. Figure 3 shows the topic coherence of each method. SC-LDA has about the same average topic coherence with LDA but has higher coherence score (-36.6) for the top 20 topics than LDA (-39.1). This is because incorporating word correlation knowledge encourages correlated words to have high probability under the same topic, thus improving the coherence score. For the other methods, however, because they cannot converge within an hour, their topic coherence scores are much worse than SC-LDA and LDA. This again demonstrates the efficiency of SC-LDA over other baselines.

3.6 Document Label Prior Knowledge

SC-LDA can also handle other types of prior knowledge. We compare it with Labeled-LDA (Ramage et al., 2009). Labeled-LDA also uses Gibbs sampling for inference, allowing direct computation time comparisons.

Table 3 shows the average running time per iteration for Labeled-LDA and SC-LDA. Because document labels apply sparsity to the document-topic counts, the average running time per iteration decreases as the number of labeled document increases. SC-LDA exhibits greater speedup with more topics; when $T = 500$,[5] SC-LDA runs more than ten times faster than Labeled-LDA.

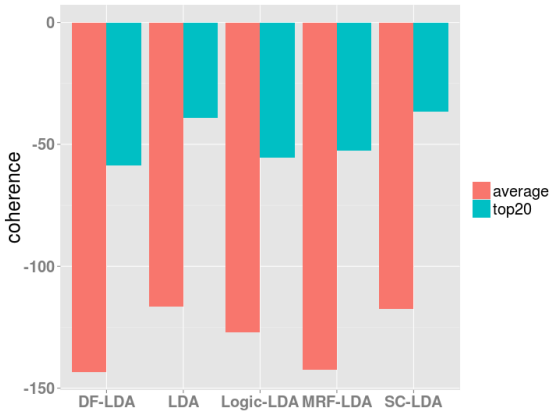


Figure 3: Average topic coherence and average top 20 topic coherence. The models are trained on NIPS dataset with 500-topic and 100 word correlations. SC-LDA achieves higher topic coherence than other methods.

# Topics				
	T50	T100	T200	T500
Labeled-LDA	0.93	1.89	3.60	8.05
SC-LDA	0.38	0.45	0.51	0.72
# Labeled Documents				
	C500	C1000	C2000	C5000
Labeled-LDA	1.95	1.88	1.75	1.48
SC-LDA	0.51	0.45	0.41	0.31

Table 3: The average running time per iteration over 100 iterations, averaged over 5 seeds, on 20NG dataset. Experiments begin with 100 topics, 1000 labeled documents, and then vary one dimension: number of topics (top), or number of labeled documents (bottom).

4 Related Work

This works brings together two lines of research: incorporating rich knowledge into probabilistic models and efficient inference of probabilistic models on large datasets. Both are common areas of interest across many machine learning formalisms: probabilistic logic (Bach et al., 2015), graph algorithms (Low et al., 2012), and probabilistic grammars (Cohen et al., 2008). However, our focus in this paper is the intersection of these lines of research with topic models.

[5]For 20NG dataset, it may overfit the data with 500 topics, but here we use it to demonstrate the scalability.

Adding knowledge and metadata to topic models makes the models richer, more understandable, and more domain-specific. A common distinction is upstream (conditioning on metadata) vs. downstream models (conditioning on variables already present in a topic model to predict metadata) (Mimno et al., 2008). Downstream models are typically better at prediction tasks such as predicting sentiment (Blei and McAuliffe, 2007), ideology (Nguyen et al., 2014a), or links in a social network (Chang and Blei, 2009). In contrast, our approach—an upstream model—is often easier to implement and leads to more interpretable topics. Upstream models at the document level have been used to understand the labels in large document collections (Ramage et al., 2009; Nguyen et al., 2014b) and capture relationships in document networks using Markov random fields (Daumé III, 2009). At the word level, Xie et al. (2015) incorporate word correlation to LDA by building a Markov Random Field regularization, similar to Newman et al. (2011), who use regularization to improve topic coherence. However, despite these exciting applications, the experiments in the above work are typically on small datasets.

In contrast, there is a huge interest in improving the scalability of topic models to large numbers of documents, numbers of topics, and vocabularies. Attempts to scale inference for topic models have started from both variational inference and Gibbs sampling—two popular learning inference techniques for topic modeling. Gibbs sampling is a popular technique because of its simplicitly and low latency. However, for large numbers of topics, Gibbs sampling can become unwieldy. Porteous et al. (2008) address this issue by creating an upper bound approximation that produces accurate results, while SparseLDA (Yao et al., 2009) present an effective factorization that speeds inference without sacrificing accuracy. Just as our model builds on SparseLDA's insights, SparseLDA has been incorporated into commercial deployments (Wang et al., 2014) and improved using alias tables (Li et al., 2014). Yuan et al. (2015) also presents an efficient constant time sampling algorithm for building big topic models. Variational inference can easily be parallelized (Nallapati et al., 2007; Zhai et al., 2012), but has high latency, which has been addressed by performing online updates (Hoffman et al., 2010) and taking stochastic gradients estimated by MCMC inference (Mimno et al., 2012). In this paper, we only focus on single-processor learning, but existing parallelization techniques (Newman et al., 2009) are applicable to our model.

At the intersection lies models that improve the scalability of upstream topic model inference. In addition to our SC-LDA, Hu and Boyd-Graber (2012) speed Gibbs sampling in tree-based topic models using SparseLDA's factorization strategy, and Hu et al. (2014) extend this approach by parallelizing global parameter updates using variational inference. Our work is more general (also encompassing document-based constraints) and is faster. In contrast to these upstream models, Zhu et al. (2013) and Nguyen et al. (2015) improve inference of downstream models.

5 Conclusion

We present a factor graph framework for incorporating prior knowledge into topic models. By expressing the prior knowledge as sparse constraints on the hidden topic variables, we are able to take advantage of the sparsity to speed up training. We demonstrate in experiments that our model runs significantly faster than the other alternative models and achieves comparable performance in terms of topic coherence. Efficient algorithms for incorporating prior knowledge with large topic models will benefit several downstream applications. For example, interactive topic modeling becomes feasible because fast model updates reduce the user's waiting time and thus improve the user experience. Personalized topic modeling is also an interesting future direction in which the model will generate a personalized topic structure based on the user's preferences or interests. For all these applications, an efficient learning algorithm is a crucial prerequisite.

Acknowledgments

We thank the anonymous reviews for their helpful comments. This research was supported in part by NSF grant IIS-1351029 and DARPA contract D11AP00268. Boyd-Graber is supported by NSF Grants CCF-1409287, IIS-1320538, and NCSE-1422492. Any opinions, findings, conclusions, or recommendations expressed here are those of the authors and do not necessarily reflect the view of the sponsor.

References

David Andrzejewski, Xiaojin Zhu, and Mark Craven. 2009. Incorporating domain knowledge into topic modeling via Dirichlet forest priors. In *Proceedings of the International Conference of Machine Learning*.

David Andrzejewski, Xiaojin Zhu, Mark Craven, and Benjamin Recht. 2011. A framework for incorporating general domain knowledge into latent Dirichlet allocation using first-order logic. In *International Joint Conference on Artificial Intelligence*.

Stephen H. Bach, Bert Huang, Jordan Boyd-Graber, and Lise Getoor. 2015. Paired-dual learning for fast training of latent variable hinge-loss mrfs. In *Proceedings of the International Conference of Machine Learning*.

David M. Blei and Jon D. McAuliffe. 2007. Supervised topic models. In *Proceedings of Advances in Neural Information Processing Systems*.

David M. Blei, Andrew Ng, and Michael Jordan. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3.

Jordan Boyd-Graber, David M. Blei, and Xiaojin Zhu. 2007. A topic model for word sense disambiguation. In *Proceedings of Empirical Methods in Natural Language Processing*.

Jonathan Chang and David M. Blei. 2009. Relational topic models for document networks. In *Proceedings of Artificial Intelligence and Statistics*.

Jonathan Chang, Jordan Boyd-Graber, Chong Wang, Sean Gerrish, and David M. Blei. 2009. Reading tea leaves: How humans interpret topic models. In *Proceedings of Advances in Neural Information Processing Systems*.

Shay B. Cohen, Kevin Gimpel, and Noah A. Smith. 2008. Logistic normal priors for unsupervised probabilistic grammar induction. In *Proceedings of Advances in Neural Information Processing Systems*.

Hal Daumé III. 2009. Markov random topic fields. In *Proceedings of Artificial Intelligence and Statistics*.

Laura Dietz, Steffen Bickel, and Tobias Scheffer. 2007. Unsupervised prediction of citation influences. In *Proceedings of the International Conference of Machine Learning*.

Doug Downey, Chandra Bhagavatula, and Yi Yang. 2015. Efficient methods for inferring large sparse topic hierarchies. In *Proceedings of the Association for Computational Linguistics*.

S. Geman and D. Geman, 1990. *Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images*, pages 452–472. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Thomas L. Griffiths and Mark Steyvers. 2004. Finding scientific topics. *Proceedings of the National Academy of Sciences*, 101(Suppl 1):5228–5235.

Matthew Hoffman, David M. Blei, and Francis Bach. 2010. Online learning for latent Dirichlet allocation. In *Proceedings of Advances in Neural Information Processing Systems*.

Yuening Hu and Jordan Boyd-Graber. 2012. Efficient tree-based topic modeling. In *Proceedings of the Association for Computational Linguistics*.

Yuening Hu, Jordan Boyd-Graber, and Brianna Satinoff. 2011. Interactive topic modeling. In *Proceedings of the Association for Computational Linguistics*.

Yuening Hu, Ke Zhai, Vlad Eidelman, and Jordan Boyd-Graber. 2014. Polylingual tree-based topic models for translation domain adaptation. In *Association for Computational Linguistics*.

Aaron Q. Li, Amr Ahmed, Sujith Ravi, and Alexander J. Smola. 2014. Reducing the sampling complexity of topic models. In *Knowledge Discovery and Data Mining*.

Yucheng Low, Danny Bickson, Joseph Gonzalez, Carlos Guestrin, Aapo Kyrola, and Joseph M. Hellerstein. 2012. Distributed graphlab: A framework for machine learning and data mining in the cloud. *Proceedings of the VLDB Endowment*, 5(8):716–727, April.

Andrew Kachites McCallum. 2002. Mallet: A machine learning for language toolkit. http://www.cs.umass.edu/ mccallum/mallet.

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

David Mimno, Hanna Wallach, and Andrew McCallum. 2008. Gibbs sampling for logistic normal topic models with graph-based priors. In *NIPS 2008 Workshop on Analyzing Graphs: Theory and Applications*.

David Mimno, Hanna Wallach, Edmund Talley, Miriam Leenders, and Andrew McCallum. 2011. Optimizing semantic coherence in topic models. In *Proceedings of Empirical Methods in Natural Language Processing*.

David Mimno, Matthew Hoffman, and David Blei. 2012. Sparse stochastic inference for latent Dirichlet allocation. In *Proceedings of the International Conference of Machine Learning*.

Ramesh Nallapati, William Cohen, and John Lafferty. 2007. Parallelized variational EM for latent Dirichlet allocation: An experimental evaluation of speed and scalability. In *International Conference on Data Mining Workshops*.

David Newman, Arthur Asuncion, Padhraic Smyth, and Max Welling. 2009. Distributed Algorithms for Topic Models. *Journal of Machine Learning Research*, pages 1801–1828.

David Newman, Edwin Bonilla, and Wray Buntine. 2011. Improving topic coherence with regularized topic models. In *Proceedings of Advances in Neural Information Processing Systems*, December.

Viet-An Nguyen, Jordan Boyd-Graber, Philip Resnik, Deborah Cai, Jennifer Midberry, and Yuanxin Wang. 2014a. Modeling topic control to detect influence in conversations using nonparametric topic models. *Machine Learning*, 95:381–421.

Viet-An Nguyen, Jordan Boyd-Graber, Philip Resnik, and Jonathan Chang. 2014b. Learning a concept hierarchy from multi-labeled documents. In *Neural Information Processing Systems*.

Thang Nguyen, Jordan Boyd-Graber, Jeff Lund, Kevin Seppi, and Eric Ringger. 2015. Is your anchor going up or down? Fast and accurate supervised topic models. In *North American Association for Computational Linguistics*.

Michael Paul and Roxana Girju. 2010. A two-dimensional topic-aspect model for discovering multi-faceted topics. In *Association for the Advancement of Artificial Intelligence*.

Ian Porteous, David Newman, Alexander Ihler, Arthur Asuncion, Padhraic Smyth, and Max Welling. 2008. Fast collapsed gibbs sampling for latent dirichlet allocation. In *Knowledge Discovery and Data Mining*.

Daniel Ramage, David Hall, Ramesh Nallapati, and Christopher Manning. 2009. Labeled LDA: A supervised topic model for credit attribution in multi-labeled corpora. In *Proceedings of Empirical Methods in Natural Language Processing*.

Michael Röder, Andreas Both, and Alexander Hinneburg. 2015. Exploring the space of topic coherence measures. In *Proceedings of ACM International Conference on Web Search and Data Mining*.

Evan Sandhaus. 2008. The New York Times annotated corpus. http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2008T19.

Hanna Wallach, David Mimno, and Andrew McCallum. 2009. Rethinking LDA: Why priors matter. In *Proceedings of Advances in Neural Information Processing Systems*.

Yi Wang, Xuemin Zhao, Zhenlong Sun, Hao Yan, Lifeng Wang, Zhihui Jin, Liubin Wang, Yang Gao, Jia Zeng, Qiang Yang, and Ching Law. 2014. Towards topic modeling for big data. *CoRR*, abs/1405.4402.

Pengtao Xie, Diyi Yang, and Eric P Xing. 2015. Incorporating word correlation knowledge into topic modeling. In *Conference of the North American Chapter of the Association for Computational Linguistics*.

Yi Yang, Shimei Pan, Yangqiu Song, Jie Lu, and Mercan Topkara. 2015. User-directed non-disruptive topic model update for effective exploration of dynamic content. In *Proceedings of the 20th International Conference on Intelligent User Interfaces*.

Limin Yao, David Mimno, and Andrew McCallum. 2009. Efficient methods for topic model inference on streaming document collections. In *Knowledge Discovery and Data Mining*.

Jinhui Yuan, Fei Gao, Qirong Ho, Wei Dai, Jinliang Wei, Xun Zheng, Eric Po Xing, Tie-Yan Liu, and Wei-Ying Ma. 2015. Lightlda: Big topic models on modest computer clusters. In *Proceedings of the World Wide Web Conference*.

Ke Zhai, Jordan Boyd-Graber, Nima Asadi, and Mohamad Alkhouja. 2012. Mr. LDA: A flexible large scale topic modeling package using variational inference in mapreduce. In *Proceedings of the World Wide Web Conference*.

Jun Zhu, Ning Chen, Hugh Perkins, and Bo Zhang. 2013. Gibbs max-margin topic models with fast sampling algorithms. In *Proceedings of the International Conference of Machine Learning*.

Traversing Knowledge Graphs in Vector Space

Kelvin Guu
Stanford University
kguu@stanford.edu

John Miller
Stanford University
millerjp@stanford.edu

Percy Liang
Stanford University
pliang@cs.stanford.edu

Abstract

Path queries on a knowledge graph can be used to answer compositional questions such as "*What languages are spoken by people living in Lisbon?*". However, knowledge graphs often have missing facts (edges) which disrupts path queries. Recent models for knowledge base completion impute missing facts by embedding knowledge graphs in vector spaces. We show that these models can be recursively applied to answer path queries, but that they suffer from cascading errors. This motivates a new "compositional" training objective, which dramatically improves all models' ability to answer path queries, in some cases more than doubling accuracy. On a standard knowledge base completion task, we also demonstrate that compositional training acts as a novel form of structural regularization, reliably improving performance across all base models (reducing errors by up to 43%) and achieving new state-of-the-art results.

1 Introduction

Broad-coverage knowledge bases such as Freebase (Bollacker et al., 2008) support a rich array of reasoning and question answering applications, but they are known to suffer from incomplete coverage (Min et al., 2013). For example, as of May 2015, Freebase has an entity Tad Lincoln (Abraham Lincoln's son), but does not have his ethnicity. An elegant solution to incompleteness is using vector space representations: Controlling the dimensionality of the vector space forces generalization to new facts (Nickel et al., 2011; Nickel et al., 2012; Socher et al., 2013; Riedel et al., 2013; Neelakantan et al., 2015). In the example, we would hope to infer Tad's ethnicity from the ethnicity of his parents.

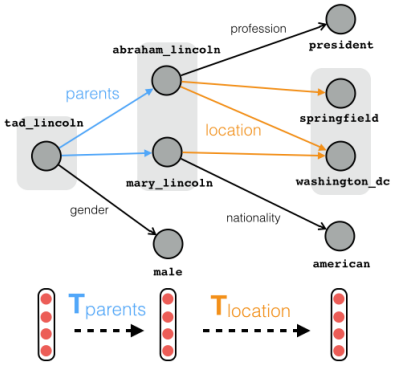


Figure 1: We propose performing path queries such as `tad_lincoln/parents/location` ("*Where are Tad Lincoln's parents located?*") in a parallel low-dimensional vector space. Here, entity sets (boxed) are represented as real vectors, and edge traversal is driven by vector-to-vector transformations (e.g., matrix multiplication).

However, what is missing from these vector space models is the original strength of knowledge bases: the ability to support compositional queries (Ullman, 1985). For example, we might ask what the ethnicity of Abraham Lincoln's daughter would be. This can be formulated as a path query on the knowledge graph, and we would like a method that can answer this efficiently, while generalizing over missing facts and even missing or hypothetical entities (Abraham Lincoln did not in fact have a daughter).

In this paper, we present a scheme to answer path queries on knowledge bases by "compositionalizing" a broad class of vector space models that have been used for knowledge base completion (see Figure 1). At a high level, we interpret the base vector space model as implementing a soft edge traversal operator. This operator can then be recursively applied to predict paths. Our interpretation suggests a new *compositional training* objective that encourages better modeling of

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 318–327,
Lisbon, Portugal, 17-21 September 2015.

paths. Our technique is applicable to a broad class of *composable* models that includes the bilinear model (Nickel et al., 2011) and TransE (Bordes et al., 2013).

We have two key empirical findings: First, we show that compositional training enables us to answer path queries up to at least length 5 by substantially reducing cascading errors present in the base vector space model. Second, we find that somewhat surprisingly, compositional training also improves upon state-of-the-art performance for knowledge base completion, which is a special case of answering unit length path queries. Therefore, compositional training can also be seen as a new form of structural regularization for existing models.

2 Task

We now give a formal definition of the task of answering path queries on a knowledge base. Let $\mathcal{E}$ be a set of entities and $\mathcal{R}$ be a set of binary relations. A knowledge graph $\mathcal{G}$ is defined as a set of triples of the form (s, r, t) where $s, t \in \mathcal{E}$ and $r \in \mathcal{R}$. An example of a triple in Freebase is $(\texttt{tad\_lincoln}, \texttt{parents}, \texttt{abraham\_lincoln})$.

A path query q consists of an initial *anchor entity,* s, followed by a sequence of relations to be traversed, $p = (r_1, \ldots, r_k)$. The answer or denotation of the query, $[\![q]\!]$, is the set of all entities that can be reached from s by traversing p. Formally, this can be defined recursively:

$$[\![s]\!] \stackrel{\text{def}}{=} \{s\}, \tag{1}$$

$$[\![q/r]\!] \stackrel{\text{def}}{=} \{t : \exists s \in [\![q]\!], (s, r, t) \in \mathcal{G}\}. \tag{2}$$

For example, `tad_lincoln/parents/location` is a query q that asks: "*Where did Tad Lincoln's parents live?*".

For evaluation (see Section 5 for details), we define the set of candidate answers to a query $\mathcal{C}(q)$ as the set of all entities that "type match", namely those that participate in the final relation of q at least once; and let $\mathcal{N}(q)$ be the incorrect answers:

$$\mathcal{C}\left(s/r_1/\cdots/r_k\right) \stackrel{\text{def}}{=} \{t \mid \exists e, (e, r_k, t) \in \mathcal{G}\} \tag{3}$$

$$\mathcal{N}\left(q\right) \stackrel{\text{def}}{=} \mathcal{C}\left(q\right) \setminus [\![q]\!]. \tag{4}$$

Knowledge base completion. Knowledge base completion (KBC) is the task of predicting whether a given edge (s, r, t) belongs in the graph or not. This can be formulated as a path query $q = s/r$ with candidate answer t.

3 Compositionalization

In this section, we show how to compositionalize existing KBC models to answer path queries. We start with a motivating example in Section 3.1, then present the general technique in Section 3.2. This suggests a new compositional training objective, described in Section 3.3. Finally, we illustrate the technique for several more models in Section 3.4, which we use in our experiments.

3.1 Example

A common vector space model for knowledge base completion is the *bilinear model* (Nickel et al., 2011). In this model, we learn a vector $x_e \in \mathbb{R}^d$ for each entity $e \in \mathcal{E}$ and a matrix $W_r \in \mathbb{R}^{d \times d}$ for each relation $r \in \mathcal{R}$. Given a query s/r (asking for the set of entities connected to s via relation r), the bilinear model scores how likely $t \in [\![s/r]\!]$ holds using

$$\text{score}(s/r, t) = x_s^\top W_r x_t. \tag{5}$$

To motivate our compositionalization technique, take $d = |\mathcal{E}|$ and suppose W_r is the adjacency matrix for relation r and entity vector x_e is the indicator vector with a 1 in the entry corresponding to entity e. Then, to answer a path query $q = s/r_1/\ldots/r_k$, we would then compute

$$\text{score}(q, t) = x_s^\top W_{r_1} \ldots W_{r_k} x_t. \tag{6}$$

It is easy to verify that the score counts the number of unique paths between s and t following relations $r_1/\ldots/r_k$. Hence, any t with positive score is a correct answer ($[\![q]\!] = \{t : \text{score}(q, t) > 0\}$).

Let us interpret (6) recursively. The model begins with an entity vector x_s, and sequentially applies *traversal operators* $\mathbb{T}_{r_i}(v) = v^\top W_{r_i}$ for each r_i. Each traversal operation results in a new "set vector" representing the entities reached at that point in traversal (corresponding to the nonzero entries of the set vector). Finally, it applies the *membership operator* $\mathbb{M}(v, x_t) = v^\top x_t$ to check if $t \in [\![s/r_1/\ldots/r_k]\!]$. Writing graph traversal in this way immediately suggests a useful generalization: take d much smaller than $|\mathcal{E}|$ and learn the parameters W_r and x_e.

3.2 General technique

The strategy used to extend the bilinear model of (5) to the compositional model in (6) can be applied to any *composable* model: namely, one that

has a scoring function of the form:

$$\text{score}(s/r, t) = \mathbb{M}(\mathbb{T}_r(x_s), x_t) \tag{7}$$

for some choice of membership operator $\mathbb{M} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ and traversal operator $\mathbb{T}_r : \mathbb{R}^d \to \mathbb{R}^d$.

We can now define the *vector denotation* of a query $[\![q]\!]_{\text{V}}$ analogous to the definition of $[\![q]\!]$ in (1) and (2):

$$[\![s]\!]_{\text{V}} \stackrel{\text{def}}{=} x_s, \tag{8}$$

$$[\![q/r]\!]_{\text{V}} \stackrel{\text{def}}{=} \mathbb{T}_r\left([\![q]\!]_{\text{V}}\right). \tag{9}$$

The score function for a compositionalized model is then

$$\text{score}(q, t) = \mathbb{M}([\![q]\!]_{\text{V}}, [\![t]\!]_{\text{V}}). \tag{10}$$

We would like $[\![q]\!]_{\text{V}}$ to approximately represent the set $[\![q]\!]$ in the sense that for every $e \in [\![q]\!]$, $\mathbb{M}\left([\![q]\!]_{\text{V}}, [\![e]\!]_{\text{V}}\right)$ is larger than the values for $e \notin [\![q]\!]$. Of course it is not possible to represent all sets perfectly, but in the next section, we present a training objective that explicitly optimizes $\mathbb{T}$ and $\mathbb{M}$ to preserve path information.

3.3 Compositional training

The score function in (10) naturally suggests a new compositional training objective. Let $\{(q_i, t_i)\}_{i=1}^N$ denote a set of path query training examples with path lengths ranging from 1 to L. We minimize the following max-margin objective:

$$J(\Theta) = \sum_{i=1}^{N} \sum_{t' \in \mathcal{N}(q_i)} \left[1 - \text{margin}(q_i, t_i, t')\right]_+,$$

$$\text{margin}(q, t, t') = \text{score}(q, t) - \text{score}(q, t'),$$

where the parameters are the membership operator, the traversal operators, and the entity vectors:

$$\Theta = \{\mathbb{M}\} \cup \{\mathbb{T}_r : r \in \mathcal{R}\} \cup \left\{x_e \in \mathbb{R}^d : e \in \mathcal{E}\right\}.$$

This objective encourages the construction of "set vectors": because there are path queries of different lengths and types, the model must learn to produce an accurate set vector $[\![q]\!]_{\text{V}}$ after any sequence of traversals. Another perspective is that each traversal operator is trained such that its transformation preserves information in the set vector which might be needed in subsequent traversal steps.

In contrast, previously proposed training objectives for knowledge base completion only train on queries of path length 1. We will refer to this special case as *single-edge training*.

In Section 5, we show that compositional training leads to substantially better results for both path query answering and knowledge base completion. In Section 6, we provide insight into why.

3.4 Other composable models

There are many possible candidates for $\mathbb{T}$ and $\mathbb{M}$. For example, $\mathbb{T}$ could be one's favorite neural network mapping from $\mathbb{R}^d$ to $\mathbb{R}^d$. Here, we focus on two composable models that were both recently shown to achieve state-of-the-art performance on knowledge base completion.

TransE. The TransE model of Bordes et al. (2013) uses the scoring function

$$\text{score}(s/r, t) = -\|x_s + w_r - x_t\|_2^2. \tag{11}$$

where x_s, w_r and x_t are all d-dimensional vectors.

In this case, the model can be expressed using membership operator

$$\mathbb{M}(v, x_t) = -\|v - x_t\|_2^2 \tag{12}$$

and traversal operator $\mathbb{T}_r(x_s) = x_s + w_r$. Hence, TransE can handle a path query $q = s/r_1/r_2/\cdots/r_k$ using

$$\text{score}(q, t) = -\|x_s + w_{r_1} + \cdots + w_{r_k} - x_t\|_2^2.$$

We visualize the compositional TransE model in Figure 2.

Bilinear-Diag. The Bilinear-Diag model of Yang et al. (2015) is a special case of the bilinear model with the relation matrices constrained to be diagonal. Alternatively, the model can be viewed as a variant of TransE with multiplicative interactions between entity and relation vectors.

Not all models can be compositionalized. It is important to point out that some models are not naturally composable—for example, the latent feature model of Riedel et al. (2013) and the neural tensor network of Socher et al. (2013). These approaches have scoring functions which combine s, r and t in a way that does not involve an intermediate vector representing s/r alone without t, so they do not decompose according to (7).

		WordNet	Freebase
Relations		11	13
Entities		38,696	75,043
Base	Train	112,581	316,232
	Test	10,544	23,733
Paths	Train	2,129,539	6,266,058
	Test	46,577	109,557

Table 1: WordNet and Freebase statistics for base and path query datasets.

3.5 Implementation

We use AdaGrad (Duchi et al., 2010) to optimize $J(\Theta)$, which is in general non-convex. Initialization scale, mini-batch size and step size were cross-validated for all models. We initialize all parameters with i.i.d. Gaussians of variance 0.1 in every entry, use a mini-batch size of 300 examples, and a step size in $[0.001, 0.1]$ (chosen via cross-validation) for all of the models. For each example q, we sample 10 negative entities $t' \in \mathcal{N}(q)$. During training, all of the entity vectors are constrained to lie on the unit ball, and we clipped the gradients to the median of the observed gradients if the update exceeded 3 times the median.

We first train on path queries of length 1 until convergence and then train on all path queries until convergence. This guarantees that the model masters basic edges before composing them to form paths. When training on path queries, we explicitly parameterize inverse relations. For the bilinear model, we initialize $W_{r^{-1}}$ with $W_r^\top$. For TransE, we initialize $w_{r^{-1}}$ with $-w_r$. For Bilinear-Diag, we found initializing $w_{r^{-1}}$ with the exact inverse $1/w_r$ is numerically unstable, so we instead randomly initialize $w_{r^{-1}}$ with i.i.d Gaussians of variance 0.1 in every entry. Additionally, for the bilinear model, we replaced the sum over $\mathcal{N}(q_i)$ in the objective with a max since it yielded slightly higher accuracy. Our models are implemented using Theano (Bastien et al., 2012; Bergstra et al., 2010).

4 Datasets

In Section 4.1, we describe two standard knowledge base completion datasets. These consist of single-edge queries, so we call them *base datasets*. In Section 4.2, we generate path query datasets from these base datasets.

4.1 Base datasets

Our experiments are conducted using the subsets of WordNet and Freebase from Socher et al. (2013). The statistics of these datasets and their splits are given in Table 1.

The WordNet and Freebase subsets exhibit substantial differences that can influence model performance. The Freebase subset is almost bipartite with most of the edges taking the form (s, r, t) for some *person* s, relation r and property t. In WordNet, both the source and target entities are arbitrary words.

Both the raw WordNet and Freebase contain many relations that are almost perfectly correlated with an inverse relation. For example, WordNet contains both `has_part` and `part_of`, and Freebase contains both `parents` and `children`. At test time, a query on an edge (s, r, t) is easy to answer if the inverse triple (t, r^{-1}, s) was observed in the training set. Following Socher et al. (2013), we account for this by excluding such "trivial" queries from the test set.

4.2 Path query datasets

Given a base knowledge graph, we generate path queries by performing random walks on the graph. If we view compositional training as a form of regularization, this approach allows us to generate extremely large amounts of auxiliary training data. The procedure is given below.

Let $\mathcal{G}_{\text{train}}$ be the training graph, which consists only of the edges in the training set of the base dataset. We then repeatedly generate training examples with the following procedure:

1. Uniformly sample a path length $L \in \{1, \ldots, L_{\max}\}$, and uniformly sample a starting entity $s \in \mathcal{E}$.

2. Perform a random walk beginning at entity s and continuing L steps.

   (a) At step i of the walk, choose a relation r_i uniformly from the set of relations incident on the current entity e.

   (b) Choose the next entity uniformly from the set of entities reachable via r_i.

3. Output a query-answer pair, (q, t), where $q = s/r_1/\cdots/r_L$ and t is the final entity of the random walk.

In practice, we do not sample paths of length 1 and instead directly add all of the edges from $\mathcal{G}_{\text{train}}$ to the path query dataset.

To generate a path query test set, we repeat the above procedure except using the graph $\mathcal{G}_{\text{full}}$, which is $\mathcal{G}_{\text{train}}$ plus all of the test edges from the base dataset. Then we remove any queries from the test set that also appeared in the training set. The statistics for the path query datasets are presented in Table 1.

5 Main results

We evaluate the models derived in Section 3 on two tasks: path query answering and knowledge base completion. On both tasks, we show that the compositional training strategy proposed in Section 3.3 leads to substantial performance gains over standard single-edge training. We also compare directly against the KBC results of Socher et al. (2013), demonstrating that previously inferior models now match or outperform state-of-the-art models after compositional training.

Evaluation metric. Numerous metrics have been used to evaluate knowledge base queries, including hits at 10 (percentage of correct answers ranked in the top 10) and mean rank. We evaluate on hits at 10, as well as a normalized version of mean rank, *mean quantile*, which better accounts for the total number of candidates. For a query q, the quantile of a correct answer t is the fraction of incorrect answers ranked after t:

$$\frac{|\{t' \in \mathcal{N}(q) : \text{score}(q, t') < \text{score}(q, t)\}|}{|\mathcal{N}(q)|} \quad (13)$$

The quantile ranges from 0 to 1, with 1 being optimal. Mean quantile is then defined to be the average quantile score over all examples in the dataset. To illustrate why normalization is important, consider a set of queries on the relation `gender`. A model that predicts the incorrect gender on every query would receive a mean rank of 2 (since there are only 2 candidate answers), which is fairly good in absolute terms, whereas the mean quantile would be 0, rightfully penalizing the model.

As a final note, several of the queries in the Freebase path dataset are "type-match trivial" in the sense that all of the type matching candidates $\mathcal{C}(q)$ are correct answers to the query. In this case, mean quantile is undefined and we exclude such queries from evaluation.

Overview. The upper half of Table 2 shows that compositional training improves path querying performance across all models and metrics on both datasets, reducing error by up to 76.2%.

The lower half of Table 2 shows that surprisingly, compositional training also improves performance on knowledge base completion across almost all models, metrics and datasets. On WordNet, TransE benefits the most, with a 43.3% reduction in error. On Freebase, Bilinear benefits the most, with a 38.8% reduction in error.

In terms of mean quantile, the best overall model is TransE (COMP). In terms of hits at 10, the best model on WordNet is Bilinear (COMP), while the best model on Freebase is TransE (COMP).

Deduction and Induction. Table 3 takes a deeper look at performance on path query answering. We divided path queries into two subsets: *deduction* and *induction*. The *deduction* subset consists of queries $q = s/p$ where the source and target entities $[\![q]\!]$ are connected via relations p in the training graph $\mathcal{G}_{\text{train}}$, but the specific query q was never seen during training. Such queries can be answered by performing explicit traversal on the training graph, so this subset tests a model's ability to approximate the underlying training graph and predict the existence of a path from a collection of single edges. The *induction* subset consists of all other queries. This means that at least one edge was missing on all paths following p from source to target in the training graph. Hence, this subset tests a model's generalization ability and its robustness to missing edges.

Performance on the *deduction* subset of the dataset is disappointingly low for models trained with single-edge training: they struggle to answer path queries even when *all edges in the path query have been seen at training time.* Compositional training dramatically reduces these errors, sometimes doubling mean quantile. In Section 6, we analyze how this might be possible. After compositional training, performance on the harder *induction* subset is also much stronger. Even when edges are missing along a path, the models are able to infer them.

Interpretable queries. Although our path datasets consists of random queries, both datasets contain a large number of useful, interpretable queries. Results on a few illustrative examples are shown in Table 4.

		Bilinear			Bilinear-Diag			TransE		
Path query task		SINGLE	COMP	(%red)	SINGLE	COMP	(%red)	SINGLE	COMP	(%red)
WordNet	MQ	84.7	89.4	**30.7**	59.7	90.4	**76.2**	83.7	93.3	**58.9**
	H@10	43.6	54.3	**19.0**	7.9	31.1	**25.4**	13.8	43.5	**34.5**
Freebase	MQ	58.0	83.5	**60.7**	57.9	84.8	**63.9**	86.2	88	**13.0**
	H@10	25.9	42.1	**21.9**	23.1	38.6	**20.2**	45.4	50.5	**9.3**
KBC task		SINGLE	COMP	(%red)	SINGLE	COMP	(%red)	SINGLE	COMP	(%red)
WordNet	MQ	76.1	82.0	**24.7**	76.5	84.3	**33.2**	75.5	86.1	**43.3**
	H@10	19.2	27.3	**10.0**	12.9	14.4	**1.72**	4.6	16.5	**12.5**
Freebase	MQ	85.3	91.0	**38.8**	84.6	89.1	**29.2**	92.7	92.8	**1.37**
	H@10	70.2	76.4	**20.8**	63.2	67.0	**10.3**	78.8	78.6	-0.9

Table 2: **Path query answering and knowledge base completion.** We compare the performance of single-edge training (SINGLE) vs compositional training (COMP). MQ: mean quantile, H@10: hits at 10, %red: percentage reduction in error.

Interpretable Queries	Bilinear SINGLE	Bilinear COMP
`X/institution/institution`$^{-1}$`/profession`	50.0	**93.6**
`X/parents/religion`	81.9	**97.1**
`X/nationality/nationality`$^{-1}$`/ethnicity`$^{-1}$	68.0	**87.0**
`X/has_part/has_instance`$^{-1}$	92.6	**95.1**
`X/type_of/type_of/type_of`	72.8	**79.4**

Table 4: Path query performance (mean quantile) on a selection of interpretable queries. We compare Bilinear SINGLE and Bilinear COMP. Meanings of each query (descending): "What professions are there at X's institution?"; "What is the religion of X's parents?"; "What are the ethnicities of people from the same country as X?"; "What types of parts does X have?"; and the transitive "What is X a type of?". (Note that a relation r and its inverse r^{-1} do not necessarily cancel out if r is not a one-to-one mapping. For example, `X/institution/institution`$^{-1}$ denotes the set of all people who work at the institution X works at, which is not just X.)

Path query task		**WordNet**		**Freebase**	
		Ded.	Ind.	Ded.	Ind.
Bilinear	SINGLE	96.9	66.0	49.3	49.4
	COMP	**98.9**	**75.6**	**82.1**	**70.6**
Bi-Diag	SINGLE	56.3	51.6	49.3	50.2
	COMP	**98.5**	**78.2**	**84.5**	**72.8**
TransE	SINGLE	92.6	71.7	85.3	72.4
	COMP	**99.0**	**87.4**	**87.5**	**76.3**

Table 3: **Deduction and induction.** We compare mean quantile performance of single-edge training (SINGLE) vs compositional training (COMP). Length 1 queries are excluded.

Comparison with Socher et al. (2013). Here, we measure performance on the KBC task in terms of the accuracy metric of Socher et al. (2013). This evaluation involves sampled negatives, and is hence noisier than mean quantile, but makes our results directly comparable to Socher et al. (2013). Our results show that previously inferior models such as the bilinear model can outperform state-of-the-art models after compositional training.

Socher et al. (2013) proposed parametrizing each entity vector as the average of vectors of words in the entity ($w_{\texttt{tad\_lincoln}} = \frac{1}{2}(w_{\texttt{tad}} + w_{\texttt{lincoln}})$, and pretraining these word vectors using the method of Turian et al. (2010). Table 5 reports results when using this approach in conjunction with compositional training. We initialized all models with word vectors from Pennington et al. (2014). We found that compositionally trained models outperform the neural tensor network (NTN) on WordNet, while being only slightly behind on Freebase. (We did not use word vectors in any of our other experiments.)

When the strategy of averaging word vectors to form entity vectors is not applied, our compositional models are significantly better on WordNet and slightly better on Freebase. It is worth noting that in many domains, entity names are not lexically meaningful, so word vector averaging is not

Accuracy	WordNet		Freebase	
	EV	WV	EV	WV
NTN	70.6	86.2	87.2	**90.0**
Bilinear COMP	77.6	**87.6**	86.1	89.4
TransE COMP	**80.3**	84.9	**87.6**	89.6

Table 5: Model performance in terms of accuracy. EV: entity vectors are separate (initialized randomly); WV: entity vectors are average of word vectors (initialized with pretrained word vectors).

always meaningful.

6 Analysis

In this section, we try to understand why compositional training is effective. For concreteness, everything is described in terms of the bilinear model. We will refer to the compositionally trained model as COMP, and the model trained with single-edge training as SINGLE.

6.1 Why does compositional training improve path query answering?

It is tempting to think that if SINGLE has accurately modeled individual edges in a graph, it should accurately model the paths that result from those edges. This intuition turns out to be incorrect, as revealed by SINGLE's relatively weak performance on the path query dataset. We hypothesize that this is due to cascading errors along the path. For a given edge (s, r, t) on the path, single-edge training encourages x_t to be closer to $x_s^\top W_r$ than any other incorrect $x_{t'}$. However, once this is achieved by a margin of 1, it does not push x_t any closer to $x_s^\top W_r$. The remaining discrepancy is noise which gets added at each step of path traversal. This is illustrated schematically in Figure 2.

To observe this phenomenon empirically, we examine how well a model handles each *intermediate* step of a path query. We can do this by measuring the *reconstruction quality* (RQ) of the set vector produced after each traversal operation. Since each intermediate stage is itself a valid path query, we define RQ to be the average quantile over all entities that belong in $[\![q]\!]$:

$$\text{RQ}(q) = \frac{1}{|[\![q]\!]|} \sum_{t \in [\![q]\!]} \text{quantile}(q, t) \qquad (14)$$

When all entities in $[\![q]\!]$ are ranked above all incorrect entities, RQ is 1. In Figure 3, we illustrate how RQ changes over the course of a query.

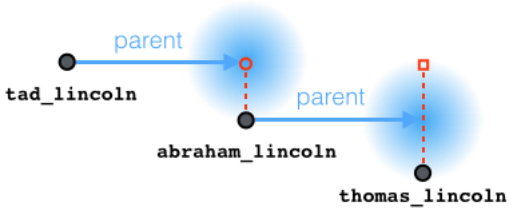


Figure 2: **Cascading errors visualized for TransE.** Each node represents the position of an entity in vector space. The relation `parent` is ideally a simple horizontal translation, but each traversal introduces noise. The red circle is where we expect Tad's parent to be. The red square is where we expect Tad's grandparent to be. Dotted red lines show that error grows larger as we traverse farther away from Tad. Compositional training pulls the entity vectors closer to the ideal arrangement.

Given the nature of cascading errors, it might seem reasonable to address the problem by adding a term to our objective which explicitly encourages $x_s^\top W_r$ to be as close as possible to x_t. With this motivation, we tried adding $\lambda \|x_s^\top W_r - x_t\|_2^2$ term to the objective of the bilinear model and a $\lambda \|x_s + w_r - x_t\|_2^2$ term to the objective of TransE. We experimented with different settings of λ over the range $[0.001, 100]$. In no case did this additional ℓ_2 term improve SINGLE's performance on the path query or single edge dataset. These results suggest that compositional training is a more effective way to combat cascading errors.

6.2 Why does compositional training improve knowledge base completion?

Table 2 reveals that COMP also performs better on the single-edge task of knowledge base completion. This is somewhat surprising, since SINGLE is trained on a training set which distributionally matches the test set, whereas COMP is not. However, COMP's better performance on path queries suggests that there must be another factor at play. At a high level, training on paths must be providing some form of structural regularization which reduces cascading errors. Indeed, paths in a knowledge graph have proven to be important features for predicting the existence of single edges (Lao et al., 2011; Neelakantan et al., 2015). For example, consider the following Horn clause:

$$\text{parents}(x, y) \wedge \text{location}(y, z) \Rightarrow \text{place\_of\_birth}(x, z),$$

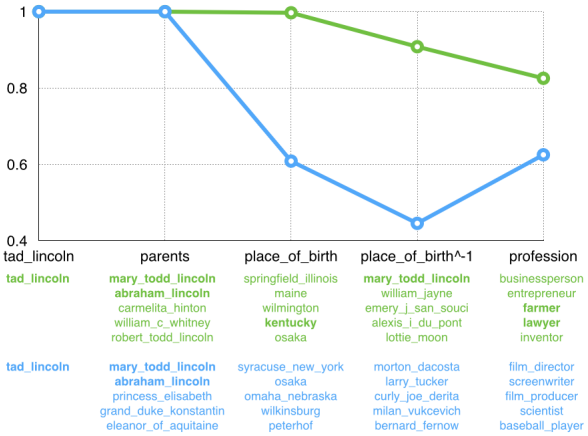


Figure 3: Reconstruction quality (RQ) at each step of the query `tad_lincoln/parents/place_of_birth/place_of_birth`$^{-1}$`/profession`. COMP experiences significantly less degradation in RQ as path length increases. Correspondingly, the set of 5 highest scoring entities computed at each step using COMP (green) is significiantly more accurate than the set given by SINGLE (blue). Correct entities are bolded.

which states that if x has a parent with location z, then x has place of birth z. The body of the Horn clause expresses a path from x to z. If COMP models the path better, then it should be better able to use that knowledge to infer the head of the Horn clause.

More generally, consider Horn clauses of the form $p \Rightarrow r$, where $p = r_1/\ldots/r_k$ is a path type and r is the relation being predicted. Let us focus on Horn clauses with high precision as defined by:

$$\text{prec}(p) = \frac{|[\![p]\!] \cap [\![r]\!]|}{|[\![p]\!]|}, \tag{15}$$

where $[\![p]\!]$ is the set of entity pairs connected by p, and similarly for $[\![r]\!]$.

Intuitively, one way for the model to implicitly learn and exploit such a Horn clause would be to satisfy the following two criteria:

1. The model should ensure a *consistent* spatial relationship between entity pairs that are related by the path type p; that is, keeping $x_s^\top W_{r_1} \ldots W_{r_k}$ close to x_t for *all* valid (s,t) pairs.

2. The model's representation of the path type p and relation r should capture that spatial relationship; that is, $x_s^\top W_{r_1} \ldots W_{r_k} \approx x_t$ implies $x_s^\top W_r \approx x_t$, or simply $W_{r_1} \ldots W_{r_k} \approx W_r$.

We have already seen empirically that SINGLE does not meet criterion 1, because cascading errors cause it to put incorrect entity vectors $x_{t'}$ closer to $x_s^\top W_{r_1} \ldots W_{r_k}$ than the correct entity. COMP mitigates these errors.

To empirically verify that COMP also does a better job of meeting criterion 2, we perform the following: for a path type p and relation r, define $\text{dist}(p,r)$ to be the angle between their corresponding matrices (treated as vectors in $\mathbb{R}^{d^2}$). This is a natural measure because $x_s^\top W_r x_t$ computes the matrix inner product between W_r and $x_s x_t^\top$. Hence, any matrix with small distance from W_r will produce nearly the same scores as W_r for the same entity pairs.

If COMP is better at capturing the correlation between p and r, then we would expect that when $\text{prec}(p)$ is high, compositional training should shrink $\text{dist}(p,r)$ more. To confirm this hypothesis, we enumerated over all 676 possible paths of length 2 (including inverted relations), and examined the proportional reduction in $\text{dist}(p,r)$ caused by compositional training,

$$\Delta\text{dist}(p,r) = \frac{\text{dist}_{\text{COMP}}(p,r) - \text{dist}_{\text{SINGLE}}(p,r)}{\text{dist}_{\text{SINGLE}}(p,r)}. \tag{16}$$

Figure 4 shows that higher precision paths indeed correspond to larger reductions in $\text{dist}(p,r)$.

7 Related work

Knowledge base completion with vector space models. Many models have been proposed for knowledge base completion, including those reviewed in Section 3.4 (Nickel et al., 2011; Bordes et al., 2013; Yang et al., 2015; Socher et al., 2013). Dong et al. (2014) demonstrated that KBC models can improve the quality of relation extraction by serving as graph-based priors. Riedel et al. (2013) showed that such models can be also be directly used for open-domain relation extraction. Our compositional training technique is an orthogonal improvement that could help any composable model.

Distributional compositional semantics. Previous works have explored compositional vector space representations in the context of logic and sentence interpretation. In Socher et al. (2012), a matrix is associated with each word of a sentence, and can be used to recursively modify the meaning of nearby constituents. Grefenstette (2013) ex-

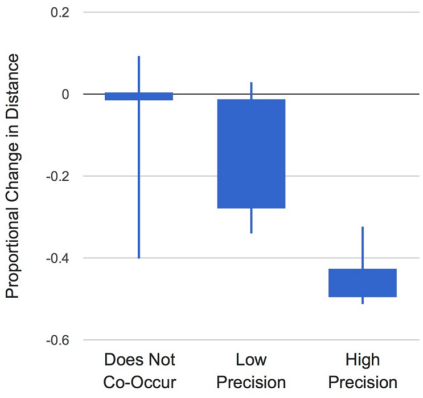


Figure 4: We divide paths of length 2 into high precision (> 0.3), low precision (≤ 0.3), and not co-occuring with r. Here $r =$ `nationality`. Each box plot shows the min, max, and first and third quartiles of $\Delta\text{dist}(p, r)$. As hypothesized, compositional training results in large decreases in $\text{dist}(p, r)$ for high precision paths p, modest decreases for low precision paths, and little to no decreases for irrelevant paths.

plored the ability of tensors to simulate logical calculi. Bowman et al. (2014) showed that recursive neural networks can learn to distinguish important semantic relations. Socher et al. (2014) found that compositional models were powerful enough to describe and retrieve images.

We demonstrate that compositional representations are also useful in the context of knowledge base querying and completion. In the aforementioned work, compositional models produce vectors which represent truth values, sentiment or image features. In our approach, vectors represent sets of entities constituting the denotation of a knowledge base query.

Path modeling. Numerous methods have been proposed to leverage path information for knowledge base completion and question answering. Nickel et al. (2014) proposed combining low-rank models with sparse path features. Lao and Cohen (2010) used random walks as features and Gardner et al. (2014) extended this approach by using vector space similarity to govern random walk probabilities. Neelakantan et al. (2015) addressed the problem of path sparsity by embedding paths using a recurrent neural network. Perozzi et al. (2014) sampled random walks on social networks as training examples, with a different goal to classify nodes in the network. Bordes et al. (2014) embed paths as a sum of relation vectors for question answering. Our approach is unique in modeling the denotation of each intermediate step of a path query, and using this information to regularize the spatial arrangement of entity vectors.

8 Discussion

We introduced the task of answering path queries on an incomplete knowledge base, and presented a general technique for compositionalizing a broad class of vector space models. Our experiments show that compositional training leads to state-of-the-art performance on both path query answering and knowledge base completion.

There are several key ideas from this paper: regularization by augmenting the dataset with paths, representing sets as low-dimensional vectors in a context-sensitive way, and performing function composition using vectors. We believe these three could all have greater applicability in the development of vector space models for knowledge representation and inference.

Reproducibility Our code, data, and experiments are available on the CodaLab platform at `https://www.codalab.org/worksheets/0xfcace41fdeec45f3bc6ddf31107b829f`.

Acknowledgments We would like to thank Gabor Angeli for fruitful discussions and the anonymous reviewers for their valuable feedback. We gratefully acknowledge the support of the Google Natural Language Understanding Focused Program and the National Science Foundation Graduate Research Fellowship under Grant No. DGE-114747.

References

F. Bastien, P. Lamblin, R. Pascanu, J. Bergstra, I. J. Goodfellow, A. Bergeron, N. Bouchard, and Y. Bengio. 2012. Theano: new features and speed improvements. Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop.

J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, and Y. Bengio. 2010. Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*.

K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In

International Conference on Management of Data (SIGMOD), pages 1247–1250.

A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2787–2795.

A. Bordes, S. Chopra, and J. Weston. 2014. Question answering with subgraph embeddings. In *Empirical Methods in Natural Language Processing (EMNLP)*.

S. R. Bowman, C. Potts, and C. D. Manning. 2014. Can recursive neural tensor networks learn logical reasoning? In *International Conference on Learning Representations (ICLR)*.

X. Dong, E. Gabrilovich, G. Heitz, W. Horn, N. Lao, K. Murphy, T. Strohmann, S. Sun, and W. Zhang. 2014. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 601–610.

J. Duchi, E. Hazan, and Y. Singer. 2010. Adaptive subgradient methods for online learning and stochastic optimization. In *Conference on Learning Theory (COLT)*.

M. Gardner, P. Talukdar, J. Krishnamurthy, and T. Mitchell. 2014. Incorporating vector space similarity in random walk inference over knowledge bases. In *Empirical Methods in Natural Language Processing (EMNLP)*.

E. Grefenstette. 2013. Towards a formal distributional semantics: Simulating logical calculi with tensors. *arXiv preprint arXiv:1304.5823*.

N. Lao and W. W. Cohen. 2010. Relational retrieval using a combination of path-constrained random walks. *Machine learning*, 81(1):53–67.

N. Lao, T. Mitchell, and W. W. Cohen. 2011. Random walk inference and learning in a large scale knowledge base. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 529–539.

B. Min, R. Grishman, L. Wan, C. Wang, and D. Gondek. 2013. Distant supervision for relation extraction with an incomplete knowledge base. In *North American Association for Computational Linguistics (NAACL)*, pages 777–782.

A. Neelakantan, B. Roth, and A. McCallum. 2015. Compositional vector space models for knowledge base completion. In *Association for Computational Linguistics (ACL)*.

M. Nickel, V. Tresp, and H. Kriegel. 2011. A three-way model for collective learning on multi-relational data. In *International Conference on Machine Learning (ICML)*, pages 809–816.

M. Nickel, V. Tresp, and H. Kriegel. 2012. Factorizing YAGO. In *World Wide Web (WWW)*.

M. Nickel, X. Jiang, and V. Tresp. 2014. Reducing the rank in relational factorization models by including observable patterns. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1179–1187.

J. Pennington, R. Socher, and C. D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*.

B. Perozzi, R. Al-Rfou, and S. Skiena. 2014. Deepwalk: Online learning of social representations. In *International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 701–710.

S. Riedel, L. Yao, and A. McCallum. 2013. Relation extraction with matrix factorization and universal schemas. In *North American Association for Computational Linguistics (NAACL)*.

R. Socher, B. Huval, C. D. Manning, and A. Y. Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP/CoNLL)*, pages 1201–1211.

R. Socher, D. Chen, C. D. Manning, and A. Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *Advances in Neural Information Processing Systems (NIPS)*, pages 926–934.

R. Socher, A. Karpathy, Q. V. Le, C. D. Manning, and A. Y. Ng. 2014. Grounded compositional semantics for finding and describing images with sentences. *Transactions of the Association for Computational Linguistics (TACL)*, 2:207–218.

J. Turian, L. Ratinov, and Y. Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 384–394.

J. D. Ullman. 1985. Implementation of logical query languages for databases. *ACM Transactions on Database Systems (TODS)*, 10(3):289–321.

B. Yang, W. Yih, X. He, J. Gao, and L. Deng. 2015. Embedding entities and relations for learning and inference in knowledge bases. *arXiv preprint arXiv:1412.6575*.

Density-Driven Cross-Lingual Transfer of Dependency Parsers

Mohammad Sadegh Rasooli and **Michael Collins***
Department of Computer Science, Columbia University
New York, NY 10027, USA
{rasooli,mcollins}@cs.columbia.edu

Abstract

We present a novel method for the cross-lingual transfer of dependency parsers. Our goal is to induce a dependency parser in a target language of interest without any direct supervision: instead we assume access to parallel translations between the target and one or more source languages, and to supervised parsers in the source language(s). Our key contributions are to show the utility of *dense* projected structures when training the target language parser, and to introduce a novel learning algorithm that makes use of dense structures. Results on several languages show an absolute improvement of 5.51% in average dependency accuracy over the state-of-the-art method of (Ma and Xia, 2014). Our average dependency accuracy of 82.18% compares favourably to the accuracy of fully supervised methods.

1 Introduction

In recent years there has been a great deal of interest in dependency parsing models for natural languages. Supervised learning methods have been shown to produce highly accurate dependency-parsing models; unfortunately, these methods rely on human-annotated data, which is expensive to obtain, leading to a significant barrier to the development of dependency parsers for new languages. Recent work has considered unsupervised methods (e.g. (Klein and Manning, 2004; Headden III et al., 2009; Gillenwater et al., 2011; Mareček and Straka, 2013; Spitkovsky et al., 2013; Le and Zuidema, 2015; Grave and Elhadad, 2015)), or methods that transfer linguistic structures across languages (e.g. (Cohen et al., 2011; McDonald et al., 2011; Ma and Xia, 2014; Tiedemann, 2015; Guo et al., 2015; Zhang and Barzilay, 2015; Xiao and Guo, 2015)), in an effort to reduce or eliminate the need for annotated training examples. Unfortunately the accuracy of these methods generally lags quite substantially behind the performance of fully supervised approaches.

This paper describes novel methods for the transfer of syntactic information between languages. As in previous work (Hwa et al., 2005; Ganchev et al., 2009; McDonald et al., 2011; Ma and Xia, 2014), our goal is to induce a dependency parser in a target language of interest without any direct supervision (i.e., a treebank) in the target language: instead we assume access to parallel translations between the target and one or more source languages, and to supervised parsers in the source languages. We can then use alignments induced using tools such as GIZA++ (Och and Ney, 2000), to transfer dependencies from the source language(s) to the target language (example projections are shown in Figure 1). A target language parser is then trained on the projected dependencies.

Our contributions are as follows:

- We demonstrate the utility of *dense* projected structures when training the target-language parser. In the most extreme case, a "dense" structure is a sentence in the target language where the projected dependencies form a fully projective tree that includes all words in the sentence (we will refer to these structures as "full" trees). In more relaxed definitions, we might include sentences where at least some proportion (e.g., 80%) of the words participate as a modifier in some dependency, or where long sequences (e.g., 7 words or more) of words all participate as modifiers in some dependency. We give empirical evidence that dense structures give particularly high accuracy for their projected dependencies.

*Currently on leave at Google Inc. New York.

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 328–338,
Lisbon, Portugal, 17-21 September 2015.

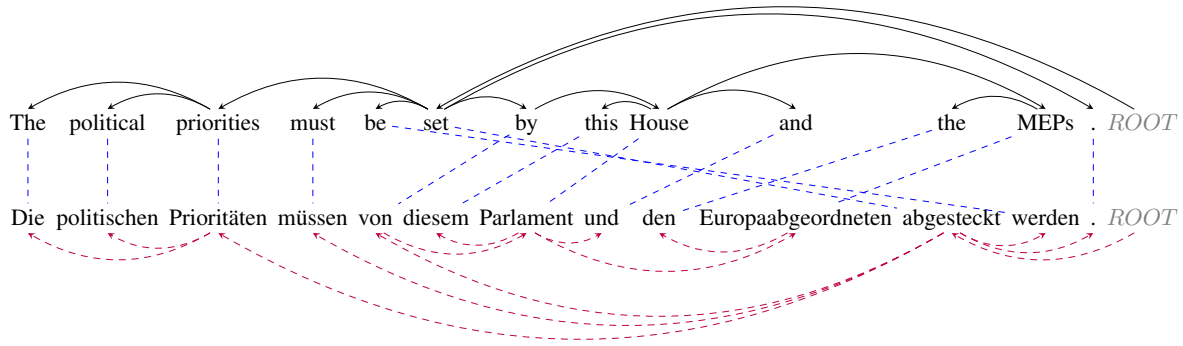


Figure 1: An example projection from English to German in the EuroParl data (Koehn, 2005). The English parse tree is the output from a supervised parser, while the German parse tree is projected from the English parse tree using translation alignments from GIZA++.

- We describe a training algorithm that builds on the definitions of dense structures. The algorithm initially trains the model on full trees, then iteratively introduces increasingly relaxed definitions of density. The algorithm makes use of a training method that can leverage partial (incomplete) dependency structures, and also makes use of confidence scores from a perceptron-trained model.

In spite of the simplicity of our approach, our experiments demonstrate significant improvements in accuracy over previous work. In experiments on transfer from a single source language (English) to a single target language (German, French, Spanish, Italian, Portuguese, and Swedish), our average dependency accuracy is 78.89%. When using multiple source languages, average accuracy is improved to 82.18%. This is a 5.51% absolute improvement over the previous best results reported on this data set, 76.67% for the approach of (Ma and Xia, 2014). To give another perspective, our accuracy is close to that of the fully supervised approach of (McDonald et al., 2005), which gives 84.29% accuracy on this data. To the best of our knowledge these are the highest accuracy parsing results for an approach that makes no use of treebank data for the language of interest.

2 Related Work

A number of researchers have considered the problem of projecting linguistic annotations from the source to the target language in a parallel corpus (Yarowsky et al., 2001; Hwa et al., 2005; Ganchev et al., 2009; Spreyer and Kuhn, 2009; McDonald et al., 2011; Ma and Xia, 2014). The projected annotations are then used to train a model in the target language. This prior work involves various innovations such as the use of posterior regularization (Ganchev et al., 2009), the use of entropy regularization and parallel guidance (Ma and Xia, 2014), the use of a simple method to transfer delexicalized parsers across languages (McDonald et al., 2011), and a method for training on partial annotations that are projected from source to target language (Spreyer and Kuhn, 2009). There is also recent work on treebank translation via a machine translation system (Tiedemann et al., 2014; Tiedemann, 2015). The work of (McDonald et al., 2011) and (Ma and Xia, 2014) is most relevant to our own work, for two reasons: first, these papers consider dependency parsing, and as in our work use the latest version of the Google universal treebank for evaluation;[1] second, these papers represent the state of the art in accuracy. The results in (Ma and Xia, 2014) dominate the accuracies for all other papers discussed in this related work section: they report an average accuracy of 76.67% on the languages German, Italian, Spanish, French, Swedish and Portuguese; this evaluation includes all sentence lengths.

Other work on unsupervised parsing has considered various methods that transfer information from source to target languages, where parsers are available in the source languages, but without the use of parallel corpora (Cohen et al., 2011; Dur-

[1]The original paper of (McDonald et al., 2011) does not use the Google universal treebank, however (Ma and Xia, 2014) reimplemented the model and report results on the Google universal treebank.

rett et al., 2012; Naseem et al., 2012; Täckström et al., 2013; Duong et al., 2015; Zhang and Barzilay, 2015). These results are somewhat below the performance of (Ma and Xia, 2014).[2]

3 Our Approach

This section describes our approach, giving definitions of parallel data and of dense projected structures; describing preliminary exploratory experiments on transfer from German to English; describing the iterative training algorithm used in our work; and finally describing a generalization of the method to transfer from multiple languages.

3.1 Parallel Data Definitions

We assume that we have parallel data in two languages. The source language, for which we have a supervised parser, is assumed to be English. The target language, for which our goal is to learn a parser, will be referred to as the "foreign" language. We describe the generalization to more than two languages in §3.5.

We use the following notation. Our parallel data is a set of examples $(e^{(k)}, f^{(k)})$ for $k = 1 \ldots n$, where each $e^{(k)}$ is an English sentence, and each $f^{(k)}$ is a foreign sentence. Each $e^{(k)} = e_1^{(k)} \ldots e_{s_k}^{(k)}$ where $e_i^{(k)}$ is a word, and s_k is the length of k'th source sentence. Similarly, $f^{(k)} = f_1^{(k)} \ldots f_{t_k}^{(k)}$ where $f_j^{(k)}$ is a word, and t_k is the length of k'th foreign sentence.

A **dependency** is a four-tuple (l, k, h, m) where $l \in \{e, f\}$ is the language, k is the sentence number, h is the head index, m is the modifier index. Note that if $l = e$ then we have $0 \leq h \leq s_k$ and $1 \leq m \leq s_k$, conversely if $l = f$ then $0 \leq h \leq t_k$ and $1 \leq m \leq t_k$. We use $h = 0$ when h is the root of the sentence.

For any $k \in \{1 \ldots n\}$, $j \in \{0 \ldots t_k\}$, $A_{k,j}$ is an integer specifying which word in $e_1^{(k)} \ldots e_{s_k}^{(k)}$, word $f_j^{(k)}$ is aligned to. It is `NULL` if $f_j^{(k)}$ is not aligned to anything. We have $A_{k,0} = 0$ for all k: that is, the root in one language is always aligned to the root in the other language.

In our experiments we use intersected alignments from GIZA++ (Och and Ney, 2000) to provide the $A_{k,j}$ values.

[2]With one exception: on Spanish, using the CoNLL definition of dependencies. The good results from (Ma and Xia, 2014) on the universal dependencies for Spanish may show that the result on the CONLL data is an anomaly, perhaps due to the annotation scheme in Spanish being different from other languages.

3.2 Projected Dependencies

We now describe various sets of projected dependencies. We use $\mathcal{D}$ to denote the set of all dependencies in the source language: these dependencies are the result of parsing the English side of the translation data using a supervised parser. Each dependency $(l, k, h, m) \in \mathcal{D}$ is a four-tuple as described above, with $l = e$. We will use $\mathcal{P}$ to denote the set of all projected dependencies from the source to target language. The set $\mathcal{P}$ is constructed from $\mathcal{D}$ and the alignment variables $A_{k,j}$ as follows:

$$\mathcal{P} = \{(l, k, h, m) : l = f \wedge (e, k, A_{k,h}, A_{k,m}) \in \mathcal{D}\}$$

We say the k'th sentence receives a *full* parse under the dependencies $\mathcal{P}$ if the dependencies (f, k, h, m) for k form a projective tree over the entire sentence: that is, each word has exactly one head, the root symbol is the head of the entire structure, and the resulting structure is a projective tree. We use $\mathcal{T}_{100} \subseteq \{1 \ldots n\}$ to denote the set of all sentences that receive a full parse under $\mathcal{P}$. We then define the following set,

$$\mathcal{P}_{100} = \{(l, k, h, m) \in \mathcal{P} : k \in \mathcal{T}_{100}\}$$

We say the k'th sentence receives a *dense* parse under the dependencies $\mathcal{P}$ if the dependencies of the form (f, k, h, m) for k form a projective tree over at least 80% of the words in the sentence. We use $\mathcal{T}_{80} \subseteq \{1 \ldots n\}$ to denote the set of all sentences that receive a dense parse under $\mathcal{P}$. We then define the following set,

$$\mathcal{P}_{80} = \{(l, k, h, m) \in \mathcal{P} : k \in \mathcal{T}_{80}\}$$

We say the k'th sentence receives a *span-s* parse where s is an integer if there is a sequence of at least s consecutive words in the target language that are all seen as a modifier in the set $\mathcal{P}$. We use $\mathcal{S}_s$ to refer to the set of all sentences with a span-s parse. We define the sets

$$\mathcal{P}_{\geq 7} = \{(l, k, h, m) \in \mathcal{P} : k \in \mathcal{S}_7\}$$

$$\mathcal{P}_{\geq 5} = \{(l, k, h, m) \in \mathcal{P} : k \in \mathcal{S}_5\}$$

$$\mathcal{P}_{\geq 1} = \{(l, k, h, m) \in \mathcal{P} : k \in \mathcal{S}_1\}$$

Finally, we also create datasets that only include projected dependencies that are consistent with respect to part-of-speech (POS) tags for the head and

modifier words in source and target data. We assume a function $\text{POS}(k, j, i)$ which returns TRUE if the POS tags for words $f_j^{(k)}$ and $e_i^{(k)}$ are consistent. The definition of POS-consistent projected dependencies is then as follows:

$$\bar{\mathcal{P}} = \{(l, k, h, m) \in \mathcal{P} : \text{POS}(k, h, A_{k,h}) \wedge \text{POS}(k, m, A_{k,m})\}$$

We experiment with two definitions for the POS function. The first imposes a hard constraint, that the POS tags in the two languages must be identical. The second imposes a soft constraint, that the two POS tags must fall into the same equivalance class: the equivalence classes used are listed in §4.1.

Given this definition of $\bar{\mathcal{P}}$, we can create sets $\bar{\mathcal{P}}_{100}$, $\bar{\mathcal{P}}_{80}$, $\bar{\mathcal{P}}_{\geq 7}$, $\bar{\mathcal{P}}_{\geq 5}$, and $\bar{\mathcal{P}}_{\geq 1}$, using analogous definitions to those given above.

3.3 Preliminary Experiments with Transfer from English to German

Throughout the experiments in this paper, we used German as the target language for development of our approach. Table 1 shows some preliminary results on transferring dependencies from English to German. We can estimate the accuracy of dependency subsets such as $\mathcal{P}_{100}$, $\mathcal{P}_{80}$, $\mathcal{P}_{\geq 7}$ and so on by comparing these dependencies to the dependencies from a supervised German parser on the same data. That is, we use a supervised parser to provide gold standard annotations. The full set of dependencies $\mathcal{P}$ give 74.0% accuracy under this measure; results for $\mathcal{P}_{100}$ are considerably higher in accuracy, ranging from 83.0% to 90.1% depending on how POS constraints are used.

As a second evaluation method, we can test the accuracy of a model trained on the $\mathcal{P}_{100}$ data. The benefit of the soft-matching POS definition is clear. The hard match definition harms performance, presumably because it reduces the number of sentences used to train the model.

Throughout the rest of this paper, we use the soft POS constraints in all projection algorithms.[3]

3.4 The Training Procedure

We now describe the training procedure used in our experiments. We use a perceptron-trained shift-reduce parser, similar to that of (Zhang and Nivre, 2011). We assume that the parser is able to operate in a "constrained" mode, where it returns the highest scoring parse that is consistent with a given subset of dependencies. This can be achieved via zero-cost dynamic oracles (Goldberg and Nivre, 2013).

Inputs: Sets $\mathcal{P}_{100}$, $\mathcal{P}_{80}$, $\mathcal{P}_{\geq 7}$, $\mathcal{P}_{\geq 5}$, $\mathcal{P}_{\geq 1}$ as defined in §3.2.

Definitions: Functions TRAIN, CDECODE, TOP as defined in §3.4.

Algorithm:

1. $\theta^1 = \text{TRAIN}(\mathcal{P}_{100})$
2. $\mathcal{P}^1_{100} = \text{CDECODE}(\mathcal{P}_{80} \cup \mathcal{P}_{\geq 7}, \theta^1)$
3. $\theta^2 = \text{TRAIN}(\mathcal{P}_{100} \cup \text{TOP}(\mathcal{P}^1_{100}, \theta^1))$
4. $\mathcal{P}^2_{100} = \text{CDECODE}(\mathcal{P}_{80} \cup \mathcal{P}_{\geq 5}, \theta^2)$
5. $\theta^3 = \text{TRAIN}(\mathcal{P}_{100} \cup \text{TOP}(\mathcal{P}^2_{100}, \theta^2))$
6. $\mathcal{P}^3_{100} = \text{CDECODE}(\mathcal{P}_{\geq 1}, \theta^3)$
7. $\theta^4 = \text{TRAIN}(\mathcal{P}_{100} \cup \text{TOP}(\mathcal{P}^3_{100}, \theta^3))$

Output: Parameter vectors $\theta^1, \theta^2, \theta^3, \theta^4$.

Figure 2: The learning algorithm.

We assume the following definitions:

- $\text{TRAIN}(\mathcal{D})$ is a function that takes a set of dependency structures $\mathcal{D}$ as input, and returns a model θ as its output. The dependency structures are assumed to be full trees: that is, they correspond to fully projected trees with the root symbol as their root.
- $\text{CDECODE}(\mathcal{P}, \theta)$ is a function that takes a set of partial dependency structures $\mathcal{P}$, and a model θ as input, and as output returns a set of full trees $\mathcal{D}$. It achieves this by constrained decoding of the sentences in $\mathcal{P}$ under the model θ, where for each sentence we use beam search to search for the highest scoring projective full tree that is consistent with the dependencies in $\mathcal{P}$.
- $\text{TOP}(\mathcal{D}, \theta)$ takes as input a set of full trees $\mathcal{D}$, and a model θ. It returns the top m highest scoring trees in $\mathcal{D}$ (in our experiments we used $m = 200,000$), where the score for each tree is the perceptron-based score normalized by the sentence length. Thus we return the

[3] The hard constraint is also used by Ma and Xia (2014).

POS Constraints	$\mathcal{P}$		$dense$		$\mathcal{P}_{100}$		Train on $\mathcal{P}_{100}$
	#sen	Acc.	#sen	Acc.	#sen	Acc.	
No Restriction	$968k$	74.0	$65k$	81.4	$23k$	83.0	69.5
Hard match	$927k$	80.1	$26k$	88.0	$8k$	90.1	68.0
Soft match	$904k$	80.0	$52k$	84.9	$18k$	85.8	70.6

Table 1: Statistics showing the accuracy for various definitions of projected trees: see §3.2 for definitions of $\mathcal{P}$, $\mathcal{P}_{100}$ etc. Columns labeled "Acc." show accuracy when the output of a supervised German parser is used as gold standard data. Columns labeled "#sen" show number of sentences. "dense" shows $\mathcal{P}_{100} \cup \mathcal{P}_{80} \cup \mathcal{P}_{\geq 7}$ and "Train" shows accuracy on test data of a model trained on the $\mathcal{P}_{100}$ trees.

200,000 trees that the perceptron is most confident on.[4]

Figure 2 shows the learning algorithm. It generates a sequence of parsing models, $\theta^1 \ldots \theta^4$. In the first stage of learning, the model is initialized by training on $\mathcal{P}_{100}$. The method then uses this model to fill in the missing dependencies on $\mathcal{P}_{80} \cup \mathcal{P}_{\geq 7}$ using the CDECODE method; this data is added to $\mathcal{P}_{100}$ and the model is retrained. The method is iterated, at each point adding in additional partial structures (note that $\mathcal{P}_{\geq 7} \subseteq \mathcal{P}_{\geq 5} \subseteq \mathcal{P}_{\geq 1}$, hence at each stage we expand the set of training data that is parsed using CDECODE).

3.5 Generalization to Multiple Languages

We now consider the generalization to learning from multiple languages. We again assume that the task is to learn a parser in a single target language, for example German. We assume that we now have multiple source languages. For example, in our experiments with German as the target, we used English, French, Spanish, Portuguese, Swedish, and Italian as source languages. We assume that we have fully supervised parsers for all source languages. We will consider two methods for combining information from the different languages:

Method 1: Concatenation In this approach, we form sets $\mathcal{P}$, $\mathcal{P}_{100}$, $\mathcal{P}_{80}$, $\mathcal{P}_{\geq 7}$ etc. from each of the languages separately, and then concatenate[5] the data to give new definitions of $\mathcal{P}$, $\mathcal{P}_{100}$,$\mathcal{P}_{80}$, $\mathcal{P}_{\geq 7}$ etc.

Method 2: Voting In this case, we assume that each target language sentence is aligned to a source language sentence in each of the source languages. This is the case, for example, in the Europarl data, where we have translations of the same material into multiple languages. We can then create the set $\mathcal{P}$ of projected dependencies using a voting scheme. For any word (k, j) seen in the target language, each source language will identify a headword (this headword may be NULL if there is no alignment giving a dependency). We simply take the most frequent headword chosen by the languages. After creating the set $\mathcal{P}$, we can create subsets such as $\mathcal{P}_{100}$, $\mathcal{P}_{80}$, $\mathcal{P}_{\geq 7}$ in exactly the same way as before.

Once the various projected dependency training sets have been created, we train the dependency parsing model using the algorithm given in §3.4.

4 Experiments

We now describe experiments using our approach. We first describe data and tools used in the experiments, and then describe results.

4.1 Data and Tools

Data We use the EuroParl data (Koehn, 2005) as our parallel data and the Google universal treebank (v2; standard data) (McDonald et al., 2013) as our evaluation data, and as our training data for the supervised source-language parsers. We use seven languages that are present in both Europarl and the Google universal treebank: English (used only as the source language), and German, Spanish, French, Italian, Portuguese and Swedish.

Word Alignments We use Giza++[6] (Och and Ney, 2000) to induce word alignments. Sentences with length greater than 100 and single-word sentences are removed from the parallel data. We follow common practice in training Giza++ for both translation directions, and taking the intersection of the two sets as our final alignment. Giza++ de-

[4] In cases where $|\mathcal{D}| < m$, the entire set $\mathcal{D}$ is returned.

[5] That is, dependency structures projected from different languages are taken to be entirely separate from each other.

[6] `http://www.statmt.org/moses/giza/GIZA++.html`

L	en→trgt				concat→trgt				voting→trgt			
	θ^1	θ^2	θ^3	θ^4	θ^1	θ^2	θ^3	θ^4	θ^1	θ^2	θ^3	θ^4
de	70.56	72.86	73.74	74.32	73.47	75.17	75.59	76.34	78.17	79.29	79.36	79.68
es	75.69	77.27	77.29	78.17	79.53	79.57	79.67	80.28	79.82	80.76	81.16	80.86
fr	77.03	78.54	78.70	79.91	81.23	81.79	82.30	82.24	82.17	82.75	82.47	82.72
it	77.35	78.64	79.06	79.46	81.49	82.25	82.02	82.49	82.58	82.95	83.45	83.67
pt	75.98	77.96	78.29	79.38	80.29	81.73	81.53	82.23	80.12	81.70	81.69	82.07
sv	78.68	80.28	80.81	82.11	82.53	83.78	83.83	83.80	82.85	83.76	83.85	84.06
avg	75.88	77.59	77.98	78.89	79.76	80.72	80.82	81.23	80.95	81.87	82.00	82.18

Table 2: Parsing accuracies of different methods on the test data using the gold standard POS tags. The models $\theta^1 \ldots \theta^4$ are described in §3.4. "en→trgt" is the single-source setting with English as the source language. "concat→trgt" and "voting→trgt" are results with multiple source languages for the concatenation and voting methods

fault alignment model is used in all of our experiments.

The Parsing Model For all parsing experiments we use the Yara parser[7] (Rasooli and Tetreault, 2015), a reimplementation of the k-beam arc-eager parser of Zhang and Nivre (2011). We use a beam size of 64, and Brown clustering features[8] (Brown et al., 1992; Liang, 2005). The parser gives performance close to the state of the art: for example on section 23 of the Penn WSJ treebank (Marcus et al., 1993), it achieves 93.32% accuracy, compared to 92.9% accuracy for the parser of (Zhang and Nivre, 2011).

POS Consistency As mentioned in §3.2, we define a soft POS consistency constraint to prune some projected dependencies. A source/target language word pair satisifies this constraint if one of the following conditions hold: 1) the POS tags for the two words are identical; 2) the word forms for the two words are identical (this occurs frequently for numbers, for example); 3) both tags are in one of the following equivalence classes: {ADV ↔ ADJ} {ADV ↔ PRT} {ADJ ↔ PRON} {DET ↔ NUM} {DET ↔ PRON} {DET ↔ NOUN} {PRON ↔ NOUN} {NUM ↔ X} {X ↔ .}. These rules were developed primarily on German, with some additional validation on Spanish. These rules required a small amount of human engineering, but we view this as relatively negligible.

Parameter Tuning We used German as a target language in the development of our approach, and in setting hyper-parameters. The parser is trained using the averaged structured perceptron algorithm (Collins, 2002) with max-violation updates (Huang et al., 2012). The number of iterations over the training data is 5 when training model θ^1 in any setting, and 2, 1 and 4 when training models $\theta^2, \theta^3, \theta^4$ respectively. These values are chosen by observing the performance on German. We use θ^4 as the final output from the training process: this is found to be optimal in English to German projections.

4.2 Results

This section gives results of our approach for the single source, multi-source (concatenation) and multi-source (voting) methods. Following previous work (Ma and Xia, 2014) we use gold-standard part-of-speech (POS) tags on test data. We also provide results with automatic POS tags.

Results with a Single Source Language The first set of results are with a single source language; we use English as the source in all of these experiments. Table 2 shows the accuracy of parameters $\theta^1 \ldots \theta^4$ for transfer into German, Spanish, French, Italian, Portuguese, and Swedish. Even the lowest performing model, θ^1, which is trained only on full trees, has a performance of 75.88%, close to the 76.15% accuracy for the method of (Ma and Xia, 2014). There are clear gains as we move from θ^1 to θ^4, on all languages. The average accuracy for θ^4 is 78.89%.

Results with Multiple Source Languages, using Concatenation Table 2 shows results using multiple source languages, using the concatenation method. In these experiments for a given target language we use all other languages in our

[7] https://github.com/yahoo/YaraParser

[8] https://github.com/percyliang/brown-cluster

Model	$en \rightarrow trgt$	$concat$	$voting$	$sup(1st)$	$sup(ae)$
de	73.01	74.70	78.77	80.29	84.25
es	76.31	78.33	79.17	82.17	84.66
fr	77.54	79.71	80.77	81.33	84.95
it	78.14	80.82	82.03	83.90	87.03
pt	78.14	80.81	80.67	84.80	88.08
sv	79.31	80.81	82.03	81.12	84.87
avg	77.08	79.20	80.57	82.27	85.64

Table 3: Parsing results with automatic part of speech tags on the test data. Sup (1st) is the supervised first-order dependency parser (McDonald et al., 2005) and sup (ae) is the Yara arc-eager parser (Rasooli and Tetreault, 2015).

Model	ge15	zb15	zb_s15	mph11	mx14	$en \rightarrow trgt$	$concat$	$voting$	$sup(1st)$	$sup(ae)$
de	51.0	62.5	74.2	69.77	74.30	$74.32_{(+0.02)}$	$76.34_{(+2.04)}$	$79.68_{(+5.38)}$	81.65	85.34
es	59.2	78.0	78.4	68.72	75.53	$78.17_{(+2.64)}$	$80.28_{(+4.75)}$	$80.86_{(+5.33)}$	83.92	86.69
fr	59.0	78.9	79.6	73.13	76.53	$79.91_{(+3.38)}$	$82.24_{(+5.71)}$	$82.72_{(+6.19)}$	83.51	86.24
it	55.6	79.3	80.9	70.74	77.74	$79.46_{(+1.72)}$	$82.49_{(+4.75)}$	$83.67_{(+5.93)}$	85.47	88.83
pt	57.0	78.6	79.3	69.82	76.65	$79.38_{(+2.73)}$	$82.23_{(+5.58)}$	$82.07_{(+5.42)}$	85.67	89.44
sv	54.8	75.0	78.3	75.87	79.27	$82.11_{(+2.84)}$	$83.80_{(+4.53)}$	$84.06_{(+4.79)}$	85.59	88.06
avg	56.1	75.4	78.4	71.34	76.67	$78.89_{(+2.22)}$	$81.23_{(+4.56)}$	$82.18_{(+5.51)}$	84.29	87.50

Table 4: Comparison to previous work: ge15 (Grave and Elhadad, 2015, Figure 4), zb15 (Zhang and Barzilay, 2015), zb_s15 (Zhang and Barzilay, 2015, semi-supervised with 50 annotated sentences), mph11 (McDonald et al., 2011) and mx14 (Ma and Xia, 2014) on the Google universal treebank v2. The mph11 results are copied from (Ma and Xia, 2014, Table 4). All results are reported on gold part of speech tags. The numbers in parentheses are absolute improvements over (Ma and Xia, 2014). Sup (1st) is the supervised first-order dependency parser used by (Ma and Xia, 2014) and sup(ae) is the Yara arc-eager supervised parser (Rasooli and Tetreault, 2015).

data as source languages. The performance of θ^1 improves from an average of 75.88% for a single source language, to 79.76% for multiple languages. The performance of θ^4 gives an additional improvement to 81.23%.

Results with Multiple Source Languages, using Voting The final set of results in Table 2 are for multiple languages using the voting strategy. There are further improvements: model θ^1 has average accuracy of 80.95%, and model θ^4 has average accuracy of 82.18%.

Results with Automatic POS Tags We use our final θ^4 models to parse the treebank with automatic tags provided by the same POS tagger used for tagging the parallel data. Table 3 shows the results for the transfer methods and the supervised parsing models of (McDonald et al., 2011) and (Rasooli and Tetreault, 2015). The first-order supervised method of (McDonald et al., 2005) gives only a 1.7% average absolute improvement in accuracy over the voting method. For one language (Swedish), our method actually gives improved accuracy over the 1st order parser.

Comparison to Previous Results Table 4 gives a comparison of the accuracy on the six languages, using the single source and multiple source methods, to previous work. As shown in the table, our model outperforms all models: among them, the results of (McDonald et al., 2011) and (Ma and Xia, 2014) are directly comparable to us because they use the same training and evaluation data. The recent work of (Xiao and Guo, 2015) uses the same parallel data but evaluates on CoNLL treebanks but their results are lower than Ma and Xia (2014). The recent work of (Guo et al., 2015) evaluates on the same data as ours but uses different parallel corpora. They only reported on three languages (German: 60.35, Spanish: 71.90 and French: 72.93) which are all far bellow our results. The work of (Grave and Elhadad, 2015) is the state-of-the-art fully unsupervised model with

L	$en \rightarrow trg$							concat							voting						
	$\mathcal{P}_{80} \cup \mathcal{P}_{\geq 7}$				$\mathcal{P}_{100}$			$\mathcal{P}_{80} \cup \mathcal{P}_{\geq 7}$				$\mathcal{P}_{100}$			$\mathcal{P}_{80} \cup \mathcal{P}_{\geq 7}$				$\mathcal{P}_{100}$		
	sen#	dep#	len	acc.	sen#	len	acc.	sen#	dep#	len	acc.	sen#	len	acc.	sen#	dep#	len	acc.	sen#	dep#	acc.
de	$34k$	9.6	28.3	84.7	$18k$	6.8	85.8	$98k$	9.4	28.8	84.1	$51k$	6.3	88.0	$75k$	10.8	23.5	84.5	$47k$	8.2	91.4
es	$108k$	10.9	31.4	87.3	$20k$	7.4	89.4	$536k$	11.0	31.8	86.3	$89k$	7.5	89.8	$346k$	17.0	28.5	86.1	$109k$	12.1	89.2
fr	$70k$	10.1	32.8	85.8	$13k$	6.7	84.1	$342k$	10.5	33.0	87.5	$47k$	6.9	89.5	$303k$	14.9	29.9	87.4	$78k$	11.7	91.2
it	$57k$	10.0	31.2	84.4	$9k$	6.3	76.9	$434k$	11.1	31.3	84.7	$70k$	7.4	87.2	$301k$	15.2	28.5	84.5	$101k$	12.4	87.9
pt	$489k$	10.0	31.0	85.2	$10k$	6.0	84.0	$462k$	11.1	31.3	81.4	$77k$	7.3	85.4	$222k$	12.4	30.3	81.3	$39k$	8.8	85.8
sv	$81k$	10.4	25.8	83.1	$30k$	7.4	87.8	$255k$	9.5	23.6	84.6	$79k$	6.8	89.7	$211k$	12.2	25.2	84.2	$86k$	9.5	88.8
avg	$140k$	10.2	30.1	85.1	$17k$	6.8	84.7	$354k$	10.4	30.0	84.8	$69k$	7.0	88.3	$243k$	13.7	27.6	84.7	$77k$	10.4	89.0

Table 5: Table showing statistics on projected dependencies for the target languages, for the single-source, multi-source (concat) and multi-source (voting) methods. "sen#" is the number of sentences. "dep#" is the average number of dependencies per sentence. "len" is the average sentence length. "acc." is the percentage of projected dependencies that agree with the output from a supervised parser.

minimal linguistic prior knowledge. The model of (Zhang and Barzilay, 2015) does not use any parallel data but uses linguistic information across languages. Their semi-supervised model selectively samples 50 annotated sentences but our model outperforms their model.

Compared to the results of (McDonald et al., 2011) and (Ma and Xia, 2014) which are directly comparable, there are clear improvements across all languages; the highest accuracy, 82.18%, is a 5.51% absolute improvement over the average accuracy for (Ma and Xia, 2014).

5 Analysis

We conclude with some analysis of the accuracy of the projected dependencies for the different languages, for different definitions ($\mathcal{P}_{100}$, $\mathcal{P}_{80}$ etc.), and for different projection methods. Table 5 gives a summary of statistics for the various languages. Recall that German is used as the development language in our experiments; the other languages can be considered to be test languages. In all cases the accuracy reported is the percentage match to a supervised parser used to parse the same data.

There are some clear trends. The accuracy of the $\mathcal{P}_{100}$ datasets is high, with an average accuracy of 84.7% for the single source method, 88.3% for the concatenation method, and 89.0% for the voting method. The voting method not only increases accuracy over the single source method, but also increases the number of sentences (from an average 17k to 77k) and the average number of dependencies per sentence (from 6.8 to 10.4).

The accuracy of the $\mathcal{P}_{80} \cup \mathcal{P}_{\geq 7}$ datasets is slightly lower, with around 83-87% accuracy for the single source, concatenation and voting methods. The voting method gives a significant increase in the number of sentences—from an average of 140k to 243k. The average sentence length for this data is around 28 words, considerably longer than the $\mathcal{P}_{100}$ data; the addition of longer sentences is very likely beneficial to the model. For the voting method the average number of dependencies is 13.7, giving an average density of 50% on these sentences.

The accuracy for the different languages, in particular for the voting data, is surprisingly uniform, with a range of 85.8-91.4% for the $\mathcal{P}_{100}$ data, and 81.3-87.4% for the $\mathcal{P}_{80} \cup \mathcal{P}_{\geq 7}$ data. The number of sentences for each language, the average length of those sentences, and average number of dependencies per sentence is also quite uniform, with the exception of German, which is a clear outlier. German has fewer sentences, and fewer dependencies per sentence: this may account for it having the lowest accuracy for our models. Future work should investigate why this is the case: one hypothesis is that German has quite different word order from the other languages (it is V2, and verb final), which may lead to a degradation in the quality of the alignments from GIZA++, or in the projection process.

Finally, figure 3 shows some randomly selected examples from the $\mathcal{P}_{100}$ data for Spanish, giving a qualitative feel for the data obtained using the voting method.

6 Conclusions

We have described a density-driven method for the induction of dependency parsers using parallel data and source-language parsers. The key ideas are a series of increasingly relaxed definitions of density, together with an iterative training procedure that makes use of these definitions. The method gives a significant gain over previous methods, with dependency accuracies approach-

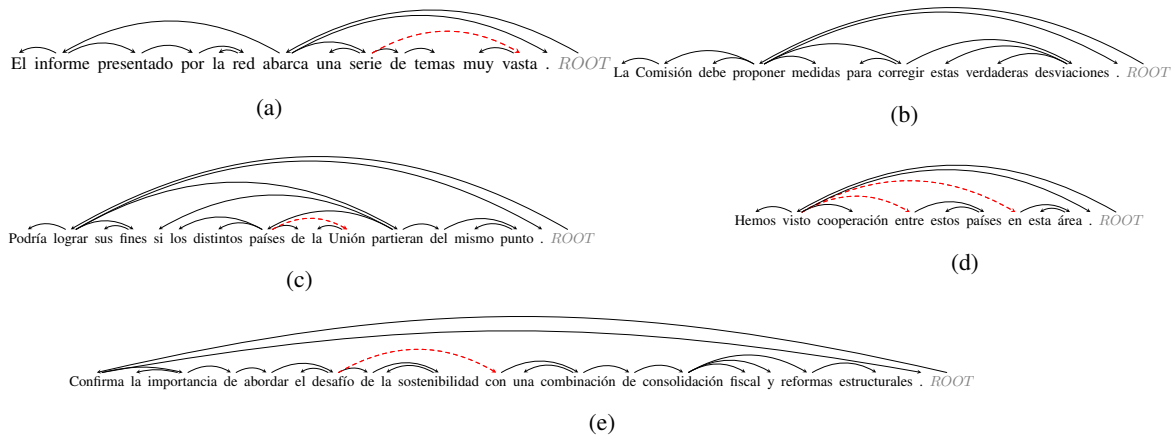


Figure 3: Randomly selected examples of Spanish dependency structures derived using the *voting* method. Dashed/red dependencies are mismatches with the output of a supervised Spanish parser; all other dependencies match the supervised parser. In these examples, 92.4% of dependencies match the supervised parser; this is close to the average match rate on Spanish of 89.2% for the voting method.

ing the level of fully supervised methods. Future work should consider application of the method to a broader set of languages, and application of the method to transfer of information other than dependency structures.

Acknowledgement

We thank Avner May and anonymous reviewers for their useful comments. Mohammad Sadegh Rasooli was supported by a grant from Bloomberg's Knowledge Engineering team.

References

Peter F Brown, Peter V Desouza, Robert L Mercer, Vincent J Della Pietra, and Jenifer C Lai. 1992. Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467–479.

Shay B. Cohen, Dipanjan Das, and Noah A. Smith. 2011. Unsupervised structure prediction with non-parallel multilingual guidance. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 50–61, Edinburgh, Scotland, UK., July. Association for Computational Linguistics.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*, pages 1–8. Association for Computational Linguistics, July.

Long Duong, Trevor Cohn, Steven Bird, and Paul Cook. 2015. Cross-lingual transfer for unsupervised dependency parsing without parallel data. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*, pages 113–122, Beijing, China, July. Association for Computational Linguistics.

Greg Durrett, Adam Pauls, and Dan Klein. 2012. Syntactic transfer using a bilingual lexicon. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1–11, Jeju Island, Korea, July. Association for Computational Linguistics.

Kuzman Ganchev, Jennifer Gillenwater, and Ben Taskar. 2009. Dependency grammar induction via bitext projection constraints. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 369–377, Suntec, Singapore, August. Association for Computational Linguistics.

Jennifer Gillenwater, Kuzman Ganchev, João Graça, Fernando Pereira, and Ben Taskar. 2011. Posterior sparsity in unsupervised dependency parsing. *The Journal of Machine Learning Research*, 12:455–490.

Yoav Goldberg and Joakim Nivre. 2013. Training deterministic parsers with non-deterministic oracles. *TACL*, 1:403–414.

Edouard Grave and Noémie Elhadad. 2015. A convex and feature-rich discriminative approach to dependency grammar induction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1375–1384, Beijing, China, July. Association for Computational Linguistics.

Jiang Guo, Wanxiang Che, David Yarowsky, Haifeng Wang, and Ting Liu. 2015. Cross-lingual dependency parsing based on distributed representations. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1234–1244, Beijing, China, July. Association for Computational Linguistics.

William P. Headden III, Mark Johnson, and David McClosky. 2009. Improving unsupervised dependency parsing with richer contexts and smoothing. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 101–109, Boulder, Colorado, June. Association for Computational Linguistics.

Liang Huang, Suphan Fayong, and Yang Guo. 2012. Structured perceptron with inexact search. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 142–151, Montréal, Canada, June. Association for Computational Linguistics.

Rebecca Hwa, Philip Resnik, Amy Weinberg, Clara Cabezas, and Okan Kolak. 2005. Bootstrapping parsers via syntactic projection across parallel texts. *Natural language engineering*, 11(03):311–325.

Dan Klein and Christopher D. Manning. 2004. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proceedings of the 42Nd Annual Meeting on Association for Computational Linguistics*, ACL '04, Stroudsburg, PA, USA. Association for Computational Linguistics.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *MT summit*, volume 5, pages 79–86.

Phong Le and Willem Zuidema. 2015. Unsupervised dependency parsing: Let's use supervised parsers. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 651–661, Denver, Colorado, May–June. Association for Computational Linguistics.

Percy Liang. 2005. *Semi-supervised learning for natural language*. Ph.D. thesis, Massachusetts Institute of Technology.

Xuezhe Ma and Fei Xia. 2014. Unsupervised dependency parsing with transferring distribution via parallel guidance and entropy regularization. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1337–1348, Baltimore, Maryland, June. Association for Computational Linguistics.

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

David Mareček and Milan Straka. 2013. Stop-probability estimates computed on a large corpus improve unsupervised dependency parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 281–290, Sofia, Bulgaria, August. Association for Computational Linguistics.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pages 523–530, Stroudsburg, PA, USA. Association for Computational Linguistics.

Ryan McDonald, Slav Petrov, and Keith Hall. 2011. Multi-source transfer of delexicalized dependency parsers. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 62–72, Edinburgh, Scotland, UK., July. Association for Computational Linguistics.

Ryan McDonald, Joakim Nivre, Yvonne Quirmbach-Brundage, Yoav Goldberg, Dipanjan Das, Kuzman Ganchev, Keith Hall, Slav Petrov, Hao Zhang, Oscar Täckström, Claudia Bedini, Núria Bertomeu Castelló, and Jungmee Lee. 2013. Universal dependency annotation for multilingual parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 92–97, Sofia, Bulgaria, August. Association for Computational Linguistics.

Tahira Naseem, Regina Barzilay, and Amir Globerson. 2012. Selective sharing for multilingual dependency parsing. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 629–637. Association for Computational Linguistics.

Franz Josef Och and Hermann Ney. 2000. Giza++: Training of statistical translation models.

Mohammad Sadegh Rasooli and Joel Tetreault. 2015. Yara parser: A fast and accurate dependency parser. *arXiv preprint arXiv:1503.06733*.

Valentin I. Spitkovsky, Hiyan Alshawi, and Daniel Jurafsky. 2013. Breaking out of local optima with count transforms and model recombination: A study in grammar induction. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1983–1995, Seattle, Washington, USA, October. Association for Computational Linguistics.

Kathrin Spreyer and Jonas Kuhn. 2009. Data-driven dependency parsing of new languages using incomplete and noisy training data. In *Proceedings of*

the Thirteenth Conference on Computational Natural Language Learning (CoNLL-2009), pages 12–20, Boulder, Colorado, June. Association for Computational Linguistics.

Oscar Täckström, Ryan McDonald, and Joakim Nivre. 2013. Target language adaptation of discriminative transfer parsers. *Transactions for ACL*.

Jörg Tiedemann, Željko Agić, and Joakim Nivre. 2014. Treebank translation for cross-lingual parser induction. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*, pages 130–140, Ann Arbor, Michigan, June. Association for Computational Linguistics.

Jörg Tiedemann. 2015. Improving the cross-lingual projection of syntactic dependencies. In *Nordic Conference of Computational Linguistics NODALIDA 2015*, pages 191–199.

Min Xiao and Yuhong Guo. 2015. Annotation projection-based representation learning for cross-lingual dependency parsing. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*, pages 73–82, Beijing, China, July. Association for Computational Linguistics.

David Yarowsky, Grace Ngai, and Richard Wicentowski. 2001. Inducing multilingual text analysis tools via robust projection across aligned corpora. In *Proceedings of the First International Conference on Human Language Technology Research*, HLT '01, pages 1–8, Stroudsburg, PA, USA. Association for Computational Linguistics.

Yuan Zhang and Regina Barzilay. 2015. Hierarchical low-rank tensors for multilingual transfer parsing. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Lisbon, Portugal, September.

Yue Zhang and Joakim Nivre. 2011. Transition-based dependency parsing with rich non-local features. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 188–193, Portland, Oregon, USA, June. Association for Computational Linguistics.

A Neural Network Model for Low-Resource Universal Dependency Parsing

Long Duong,[12] **Trevor Cohn,**[1] **Steven Bird,**[1] and **Paul Cook**[3]
[1]Department of Computing and Information Systems, University of Melbourne
[2]National ICT Australia, Victoria Research Laboratory
[3]Faculty of Computer Science, University of New Brunswick
lduong@student.unimelb.edu.au {t.cohn,sbird}@unimelb.edu.au paul.cook@unb.ca

Abstract

Accurate dependency parsing requires large treebanks, which are only available for a few languages. We propose a method that takes advantage of shared structure across languages to build a mature parser using less training data. We propose a model for learning a shared "universal" parser that operates over an inter-lingual continuous representation of language, along with language-specific mapping components. Compared with supervised learning, our methods give a consistent 8-10% improvement across several treebanks in low-resource simulations.

1 Introduction

Dependency parsing is an important task for Natural Language Processing (NLP) with application to text classification (Özgür and Güngör, 2010), relation extraction (Bunescu and Mooney, 2005), question answering (Cui et al., 2005), statistical machine translation (Xu et al., 2009), and sentiment analysis (Socher et al., 2013). A mature parser normally requires a large treebank for training, yet such resources are rarely available and are costly to build. Ideally, we would be able to construct a high quality parser with less training data, thereby enabling accurate parsing for low-resource languages.

In this paper we formalize the dependency parsing task for a low-resource language as a domain adaptation task, in which a *target* resource-poor language treebank is treated as *in-domain*, while a much larger treebank in a high-resource language forms the *out-of-domain* data. In this way, we can apply well-understood domain adaptation techniques to the dependency parsing task. However, a crucial requirement for domain adaptation is that the in-domain and out-of-domain data have compatible representations. In applying our approach to data from several languages, we must learn such a cross-lingual representation. Here we frame this representation learning as part of a neural network training. The underlying hypothesis for the joint learning is that there are some shared-structures across languages that we can exploit. This hypothesis is motivated by the excellent results of the cross-lingual application of unlexicalised parsing (McDonald et al., 2011), whereby a delexicalized parser constructed on one language is applied directly to another language.

Our approach works by jointly training a neural network dependency parser to model the syntax in both a source and target language. Many of the parameters of the source and target language parsers are shared, except for a small handful of language-specific parameters. In this way, the information can flow back and forth between languages, allowing for the learning of a compatible cross-lingual syntactic representation, while also allowing the parsers to mutually correct one another's errors. We include some language-specific components, in order to better model the lexicon of each language and allow learning of the syntactic idiosyncrasies of each language. Our experiments show that this outperforms a purely supervised setting, on both small and large data conditions, with a gain as high as 10% for small training sets. Our proposed joint training method also out-performs the conventional cascade approach where the parameters between source and target languages are related together through a regularization term (Duong et al., 2015).

Our model is flexible, allowing easy incorporation of peripheral information. For example, assuming the presence of a small bilingual dictionary is befitting of a low-resource setting, as this is prototypically one of the first artifacts generated by field linguists. We incorporate a bilingual dictionary as a set of soft constraints on the

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 339–348,
Lisbon, Portugal, 17-21 September 2015.

model, such that it learns similar representations for each word and its translation(s). For example, the representation of *house* in English should be close to *haus* in German. We empirically show that adding a bilingual dictionary improves parser performance, particularly when target data is limited.

The final contribution of the paper concerns the learned word embeddings. We demonstrate that these encode meaningful syntactic phenomena, both in terms of the observable clusters and through a verb classification task. The code for this paper is published as an open source project.[1]

2 Related Work

This work is motivated by the idea of delexicalized parsing, in which a parser is built without any lexical features and trained on a treebank for a resource-rich source language (Zeman et al., 2008). It is then applied directly to parse sentences in the target resource-poor languages. Delexicalized parsing relies on the fact that identical part-of-speech (POS) inventories are highly informative of dependency relations, and that there exists shared dependency structures across languages.

Building a dependency parser for a resource-poor language usually starts with the delexicalized parser and then uses other resources to refine the model. McDonald et al. (2011) and Ma and Xia (2014) exploited parallel data as the bridge to transfer constraints from the source resource-rich language to the target resource-poor languages. Täckström et al. (2012) also used parallel data to induce cross-lingual word clusters which added as features for their delexicalized parser. Durrett et al. (2012) constructed the set of language-independent features and used a bilingual dictionary as the bridge to transfer these features from source to target language. Täckström et al. (2013) additionally used high-level linguistic features extracted from the World Atlas of Language Structures (WALS) (Dryer and Haspelmath, 2013).

For low-resource languages, no large parallel corpus is available. Some linguists are dependency-annotating small amounts of field data, e.g. for Karuk, a nearly-extinct language of Northwest California (Garrett et al., 2013). Accordingly, we adopt a different resource requirement: a small treebank in the target low-resource language.

Domain adaptation or joint-training is a different branch of research, and falls outside the scope of this paper. Nevertheless, we would like to contrast our work with Senna (Collobert et al., 2011), a neural network framework to perform a variety of NLP tasks such as part-of-speech (POS) tagging, named entity recognition (NER), chunking, and so forth. Both approaches exploit common linguistic properties of the data through joint learning. However, Collobert et al's goal is to find a single input representation that can work well for many tasks. Our goal is different: we allow the joint-training inputs to be different but constrain the parameter weights in the upper layer to be identical. Consequently, our method applies to the task where inputs are different, possibly from different languages or domains. Their method applies for different tasks in the same language/domain where the inputs are fairly similar.

2.1 Supervised Neural Network Parser

This section describes the monolingual neural network dependency parser structure of Chen and Manning (2014). This parser achieves excellent performance, and has a highly flexible formulation allowing auxilliary inputs. The model is based on a transition-based dependency parser (Nivre, 2006) formulated as a neural-network classifier to decide which transition to apply to each parsing state configuration.[2] That is, for each configuration, the selected list of words, POS tags and labels from the Stack, Queue and Arcs are extracted. Each word, POS and label is mapped into a low-dimension vector representation using an embedding matrix, which is then fed into a two-layer neural network classifier to predict the next parsing action. The set of parameters for the model is $E = \{E^{word}, E^{\text{pos}}, E^{\text{arc}}\}$ for the embedding layer, W_1 for the fully connected cubic hidden layer and W_2 for the softmax output layer. The model prediction function is

$$P(Y|X = \vec{x}, W_1, W_2, E) = \text{softmax}\Big(W_2 \times \text{cube}(W_1 \times \Phi\left[\vec{x}, E\right])\Big) \quad (1)$$

[1] http://github.com/longdt219/universal_dependency_parser

[2] Our approach is focused on a technique for transfer learning which can be more widely applied to other types of dependency parser (and models, generally) regardless of whether they are transition-based or graph-based.

where cube is a non-linear activation function, Φ is the embedding function that returns a vector representation of parsing state x using an embedding matrix E. We refer the reader to Chen and Manning (2014) for a more detailed description.

3 A Joint Interlingual Model

We assume a small treebank in a target resource-poor language, as well as a larger treebank in the source language. Our objective is to learn a model of both languages, subject to the constraint that both models are similar overall, while allowing for some limited language variability. Instead of just training two different parsers on source and then on target, we train them jointly, in order to learn an interlingual parser. This allows the method to take maximum advantage of the limited treebank data available, resulting in highly accurate predicted parses.

Training a monolingual parser as described in section 2.1 requires optimizing the simple cross-entropy learning objective, $\mathcal{L} = -\sum_{i=1}^{|D|} \log P(Y = \vec{y}^{(i)} | X = \vec{x}^{(i)})$, where $P(Y|X)$ is given by equation 1 and $D = \{\vec{x}^{(i)}, \vec{y}^{(i)}\}_{i=1}^{n}$ is the training data. Joint training of a parser over the source and target languages can be achieved by simply adding two such cross-entropy objectives, i.e.,

$$\mathcal{L}_{\text{joint}} = -\sum_{i=1}^{|D_s|} \log P(Y_s = \vec{y}_s^{(i)} | X_s = \vec{x}_s^{(i)}) - \sum_{i=1}^{|D_t|} \log P(Y_t = \vec{y}_t^{(i)} | X_t = \vec{x}_t^{(i)}) \,, \quad (2)$$

where the training data, $D = D_s \cup D_t$, comprises data in both the source and target language. However training the model according to equation 2 will result in two independent parsers. To enforce similarity between the two parsers, we adopt parameter sharing: the neural network parameters, W_1 and W_2, are identical in both parsers. Thereby

$$P(Y_\alpha | X_\alpha = \vec{x}) = P(Y | X = \vec{x}, W_1, W_2, E_\alpha) \,,$$

where the subscript $\alpha \in \{s, t\}$ denotes the source or target language. We allow the embedding matrix E_α to differ in order to accommodate language-specific features, in terms of the representations of lexical types, E_s^{word}, part-of-speech, E_s^{pos} and dependency arc labels E_s^{arc}. This reflects the fact that different languages have different lexicon, parts-of-speech often exhibit different roles, and dependency edges serve different functions, e.g. in Korean a *static verb* can serve as an *adjective* (Kim, 2001). During training, the language-specific errors are back propagated through different branches according to the language, guiding learning towards an interlingual representation that informs parsing decisions in both languages. The set of parameters for the model is W_1, W_2, E_s, E_t where E_s, E_t are the embedding matrices for the source and target languages.

Generally speaking, we can understand the model as building the universal dependency parser that parses the universal language. Specifically, the model is the combination of two parts: the universal part (W_1, W_2) that is shared between the languages, and the conversion part (E_s, E_t) that maps a language-specific representation into the universal language. Naturally, we could stack several non-linear layers in the conversion components such that the model can better transform the input into the universal representation; we leave this exploration for future work. Currently, our cross-lingual word embeddings are meaningful for a pair of source and target languages. However, our model can easily be used for joint training over $k > 2$ languages. We also leave this avenue of enquiry for future work

One concern from equation 2 is that when the source language treebank D_s is much bigger than the target language treebank D_t, it is likely to dominate, and consequently, learning will mainly focus on optimizing the source language parser. We adjust for this disparity by balancing the two datasets, D_s and D_t, during training. When selecting mini-batches for online gradient updates, we select an equal number of classification instances from the source and target languages. Thus, for each step $|D_s| = |D_t|$, effectively reweighting the cross-entropy components in (2) to ensure parity between the languages.

The other concern is over-fitting, especially when we only have a small treebank in the target language. As suggested by Chen and Manning (2014), we apply drop-out, a form of regularization for both source and target language. That is, we randomly drop some of the activation units from both hidden layer and input layer. Following Srivastava et al. (2014), we randomly dropout 20% of the input layer and 50% of the hid-

den layer. Empirically, we observe a substantial improvement applying dropout to the model over MLE or l_2 regularization.

3.1 Incorporating a Dictionary

Our model is flexible, enabling us to freely add additional components. In this section, we assume the presence of a bilingual dictionary between the source and target language. We seek to incorporate this dictionary as a part of model learning, to encode the intuition that if two lexical items are translations of one another, the parser should treat them similarly.[3] Recall that the mapping layer is the combination of word, pos and arc embeddings, i.e., $E_\alpha = \{E_\alpha^{\text{word}}, E_\alpha^{\text{pos}}, E_\alpha^{\text{arc}}\}$. We can easily add bilingual dictionary constraints to the model in the form of regularization to minimize the l_2 distance between word representations, i.e., $\sum_{\langle i,j\rangle \in \mathcal{D}} \|E_s^{\text{word}(i)} - E_t^{\text{word}(j)}\|_F^2$, where $\mathcal{D}$ comprises translation pairs, word(i) and word(j).

When the languages share the same POS tagset and arc set,[4] we can also add further constraints such as their language-specific embeddings be close together. This results a regularised training objective,

$$\mathcal{L}_{\text{dict}} = \mathcal{L}_{\text{joint}} - \lambda \Bigg(\sum_{\langle i,j\rangle \in \mathcal{D}} \|E_s^{\text{word}(i)} - E_t^{\text{word}(j)}\|_F^2 + \|E_s^{\text{pos}} - E_t^{\text{pos}}\|_F^2 + \|E_s^{\text{arc}} - E_t^{\text{arc}}\|_F^2 \Bigg), \quad (3)$$

where $\lambda \in [0, \infty]$ controls to what degree we bind these words or pos tags or arc labels together, with high λ tying the parameters and small λ allowing independent learning. We expect the best value of λ to fall somewhere between these extremes. Finally, we use a mini-batch size of 1000 instance pairs and adaptive learning rate trainer, `adagrad` (Duchi et al., 2011) to build our two separate models corresponding to equations 2 and 3.

4 Experiments

In this section, we compare our joint training approach with baseline methods of supervised learning in the target language, and cascaded learning of source and target parsers.

[3] However, this is not always the case. For example, modal or auxiliary verbs in English often have no translations in different languages or map to words with different syntactic functions.

[4] As was the case for our experiments.

4.1 Dataset

We experiment with the Universal Dependency Treebank (UDT) V1.0 (Nivre et al., 2015), simulating low resource settings.[5] This treebank has many desirable properties for our model: the dependency types (arc labels set) and coarse POS tagset are the same across languages. This removes the need for mapping the source and target language tagsets to a common tagset. Moreover, the dependency types are also common across languages allowing evaluation of the labelled attachment score (LAS). The treebank covers 10 languages,[6] with some languages very highly resourced—Czech, French and Spanish have 400k tokens—and only modest amounts of data for other languages—Hungarian and Irish have only around $25k$ tokens. Cross-lingual models assume English as the source language, for which we have a large treebank, and only a small treebank of 3k tokens exists in each target language, simulated by subsampling the corpus.

4.2 Baseline Cascade Model

We compare our approach to a baseline interlingual model based on the same parsing algorithm as presented in section 2.1, but with cascaded training (Duong et al., 2015). This works by first learning the source language parser, and then training the target language parser using a regularization term to minimise the distance between the parameters of the target parser and the source parser (which is fixed). In this way, some structural information from the source parser can be used in the target parser, however it is likely that the representation will be overly biased towards the source language and consequently may not prove as useful for modelling the target.

4.3 Monolingual Word Embeddings

While the E^{pos} and E^{arc} are randomly initialized, we initialize both the source and target language word embeddings $E_s^{\text{word}}, E_t^{\text{word}}$ of our neural network models with pre-trained embeddings. This is an advantage since we can incorporate the monolingual data which is often available, even for

[5] Evaluating on truly resource-poor languages would be preferable to simulation. However for ease of training and evaluation, which requires a small treebank in the target language, we simulate the low-resource setting using a small part of the UDT.

[6] Czech (cs), English (en), Finnish (fi), French (fr), German (de), Hungarian (hu), Irish (ga), Italian (it), Spanish (es), Swedish (sv).

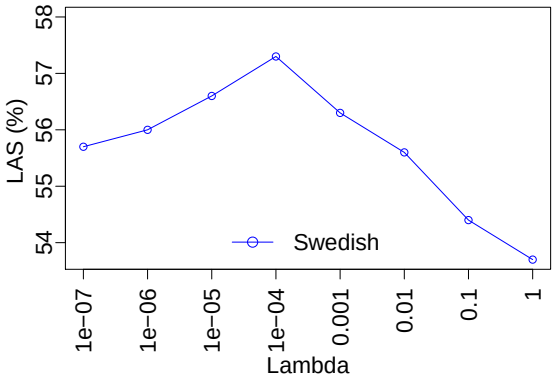


Figure 1: Sensitivity of regularization parameter λ against the LAS measured on the Swedish development set trained on 1000 (tokens).

resource-poor languages. We collect monolingual data for each language from the Machine Translation Workshop (WMT) data,[7] Europarl (Koehn, 2005) and EU Bookshop Corpus (Skadiņš et al., 2014). The size of monolingual data also varies significantly, with as much as 400 million tokens for English and German, and as few as 4 million tokens for Irish. We use the skip-gram model (Mikolov et al., 2013b) to induce 50-dimensional word embeddings.

4.4 Bilingual Dictionary

For the extended model as described in section 3.1, we also need a bilingual dictionary. We extract dictionaries from PanLex (Kamholz et al., 2014) which currently covers around 1300 language varieties and about 12 million expressions. This dataset is growing and aims at covering all languages in the world and up to 350 million expressions. The translations in PanLex come from various sources such as glossaries, dictionaries, automatic inference from other languages, etc. Naturally, the bilingual dictionary size varies greatly among resource-poor and resource-rich languages.

4.5 Regularization Parameter Tuning

Joint training with a dictionary (see equation 3) includes a regularization sensitivity parameter λ. This parameter controls to what extent we should bind the source words and their target translation, common POS tags and arcs together. In this section we measure the sensitivity of our approach with respect to this parameter. In a real world scenario, getting development data to tune this parameter is difficult. Thus, we want a parameter that can work well cross-lingually. To simulate this, we only tune the parameter on one language and apply it directly to different languages. We trained on a small Swedish treebank with 1k tokens, testing several different values of λ. We evaluated on the Swedish development dataset. Figure 1 shows the labelled attachment score (LAS) for different λ. It's clearly visible that $\lambda = 0.0001$ gives the maximum LAS on the development set. Thus, we use this value for all the experiments involving a dictionary hereafter.

4.6 Results

For our initial experiments we assume that we have only a small target treebank with 3000 tokens (around 200 sentences). Ideally the much larger source language (English) treebank should be able to improve parser performance versus simple supervised learning on such a small collection. We apply the joint model (equation 2) and joint model with the dictionary constraints (equation 3) for each target language,

The results are reported in Table 1. The supervised neural network dependency parser performed worst, as expected, and the baseline cascade model consistently outperformed the supervised model on all languages by an average margin of 5.6% (absolute).[8] The joint model also consistently out-performed both baselines giving a further 1.9% average improvement over the cascade. This was despite the fact that the cascaded model had the benefit of tuning for the regularization parameters on a development corpus, while the joint model had no parameter tuning. Note that the improvement varies substantially across languages, and is largest for Czech but is only minor for Swedish. The joint model with the bilingual dictionary outperforms the joint model, however, the improvement is modest (0.7%). Nevertheless, this model gives substantial improvements compared with the cascaded and the supervised model (2.6% and 8.2%).

5 Analysis

5.1 Learning Curve

In section 4.6, we used a 3k token treebank in the target language. What if we have more or less

[7] http://www.statmt.org/wmt14/

[8] We use absolute percentage comparisons herein.

	cs	de	es	fi	fr	ga	hu	it	sv	μ
Supervised	43.1	47.3	60.3	46.4	56.2	59.4	48.4	65.4	52.6	53.2
Baseline Cascaded	49.6	59.2	66.4	49.5	63.2	59.5	50.5	69.9	61.4	58.8
Joint	55.2	61.2	69.1	51.4	65.3	60.6	51.2	71.2	61.4	60.7
Joint + Dict	55.7	61.8	70.5	51.5	67.2	61.1	51.0	71.3	62.5	61.4

Table 1: Labelled attachment score (LAS) for each model type trained on 3000 tokens for each target language (columns). All bar the supervised model also use a large English treebank.

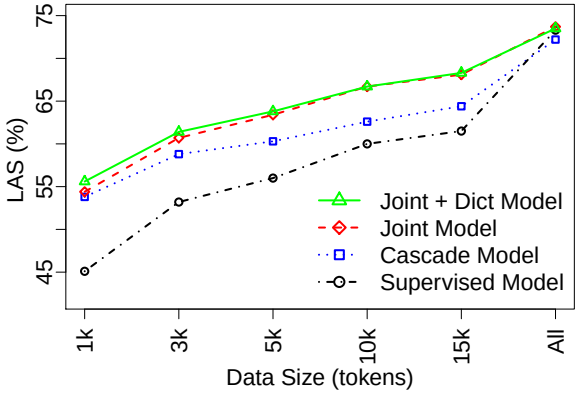


Figure 2: Learning curve for Joint model, Joint + Dict model, Baseline cascaded and Supervised model: the x-axis is the size of data (number of tokens); the y-axis is the average LAS measured on 9 languages (except English).

target language data? Figure 2 shows the learning curve with respect to various models on different data sizes averaged over all target languages. For small datasets of 1k training tokens, the cascaded model, joint model and joint + dict model performed similarly well, out-performing the supervised model by about 10% (absolute). With more training data, we see interesting changes to the relative performance of the different models. While the baseline cascade model still outperforms the supervised model, the improvement is diminishing, and by 15k, the difference is only 2.9%. On the other hand, compared with the supervised model, the joint and joint + dict models perform consistently well at all sizes, maintaining an 8% lead at 15k. This shows the superiority of joint training compared with single language training.

To understand this pattern of performance differences for the cascade versus the joint model, one needs to consider the cascade model formulation. In this approach, the target language parameters are tied (softly) with the source language parameters through regularization. This is a benefit for small datasets, providing a smoothing function to limit overtraining. However, when we have more training data, these constraints limit the capacity of the model to describe the target data. This is compounded by the problem that the source representation may not be appropriate for modelling the target language, and there is no way to correct for this. In contrast the joint model learns a mutually compatible representation automatically during joint training.

The performance results for the joint model with and without the dictionary are similar overall. Only on small datasets (1k, 3k), is the difference notable. From 5k tokens, the bilingual dictionary doesn't confer additional information, presumably as there is sufficient data for learning syntactic word representations. Moreover, translation entries exist between syntactically related word types as well as semantically related pairs, with the latter potentially limiting the beneficial effect of the dictionary.

When training on all the target language data, the supervised model does well, surpassing the cascade model. Surprisingly, the joint models outperform slightly, yielding a 0.4% improvement. This is an interesting observation suggesting that our method has potential for use not only for low resource problems, but also high resource settings.

5.2 Different Tagsets

In the above experiments, we used the universal POS tagset for all the languages in the corpus. However, for some languages,[9] the UDT also provides language specific POS tags. We use this data to test the relative performance of the model using a universal tagset cf. language specific tagsets. In this experiment, we applied the same joint model (see §3) but with a language specific tagset instead of UPOS for these languages. We expect the joint

[9] en, cs, fi, ga, it and sv.

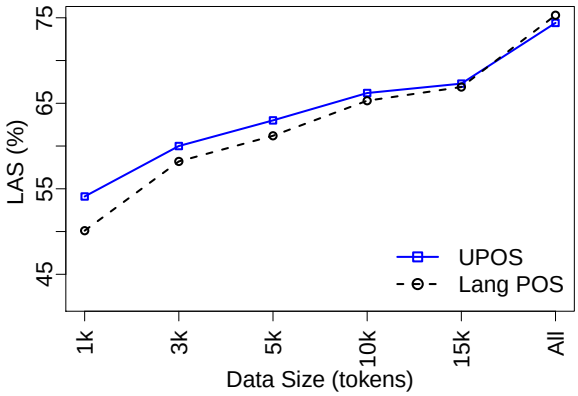


Figure 3: Learning curve for joint model using the UPOS tagset or language specific POS tagset: the x-axis is the size of data (number of tokens); the y-axis is the average LAS measured on 5 languages (except English).

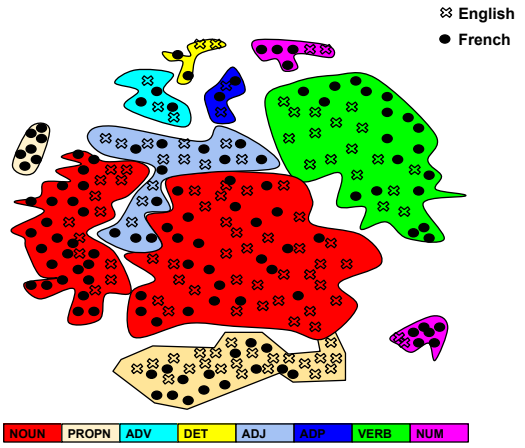


Figure 4: Universal Language visualization according to language and POS. (This should be viewed in colour.)

model to automatically learn to project the different tagsets into a common space, i.e., implicitly learn a tagset mapping between languages. Figure 3 shows the learning curve comparing the joint model with the two types of POS tagsets. For the small dataset, it is clear that the data is insufficient for the model to learn a good tagset mapping, especially for a morphologically rich language like Czech. However, with more data, the model is better able to learn the tagset mapping as part of joint training. Beyond 15k tokens, the joint model using the language specific POS tagset outperforms UPOS. Clearly there is some information lost in the UPOS tagset, although the UPOS mapping simultaneously provides implicit linguistic supervision. This explains why the UPOS might be useful in small data scenarios, but detrimental at scale. Using all the target data ("All") the language specific POS provides a 1% (absolute) gain over UPOS.

5.3 Universal Representation

As described in section 3, we can consider our joint model as the combination of two parts: a universal parser and a language-specific embedding E_s or E_t that converts the source and target language into the universal representation. We now seek to analyse qualitatively this universal representation through visualization. For this purpose we use a joint model of English and French, using all the available French treebank (more than 350k tokens) as well as a bilingual dictionary.[10] Figure 4 shows the t-SNE (Van Der Maaten, 2014) projection of the 50 dimensional word embeddings in both languages. We can see that English and French are mixed nicely together. The colouring denotes the POS tag, showing clearly that the words with similar POS tags are grouped together regardless of languages. This is partially understandable since word embeddings for dependency parsing need to convey the dependency context rather than surrounding words, as in most distributional embedding models. Words having similar dependency relation should be grouped together as they are treated similarly by the parser.

Some of the learned cross-lingual word-embeddings are shown in Table 2, which includes the five nearest neighbours to selected English words according to the monolingual word embedding (section 4.3) and our cross-lingual dependency word embeddings, trained using PanLex. The monolingual sets appear to be strongly characterised by distributional similarity. The cross-lingual embeddings display greater semantic similarity, while being more variable morphosyntactically. In many cases, the top five words of English and French are translations of each other, but with varying inflectional endings in the French forms. For example, "*buy*" vs "*vendez*" or "*invest*" vs "*in-*

[10] We also visualized the cross-lingual word embeddings without the dictionary, however the results were rather odd. Although we saw coherent POS clusters, the two languages were largely disjoint. We speculate that many components of the embeddings are use for only one language, and these outnumber the shared components, and thus more careful projection is needed for meaningful visualisation.

Words	Mono	Cross lingual embedding En	Cross lingual embedding Fr
sell	buy	buy	revendre
	eat	invest	vendez
	produce	integrate	acheter
	compete	guide	achètent
	burn	eat	investir
playing	serving	sailing	jouait
	acting	play	navigue
	paying	moving	jouent
	pursuing	faces	pièce
	running	ran	jouer
hard	difficult	crazy	dur
	harder	strange	dures
	easy	beautiful	hard
	magnificent	friendly	fou
	painful	difficult	folles
initially	originally	originally	réellement
	previously	previously	déjà
	officially	officially	récemment
	basically	actually	dernièrement
	already	already	surroît
university	teachers	school	universitaire
	student	education	université
	teacher	student	école
	student	medicine	scolaire
	training	participant	school
mobile	wireless	computers	mobile
	goods	Web	mobiles
	online	Internet	ordinateurs
	freight	computer	Web
	broadband	web	internet

Table 2: Examples of 5 nearest neighbours with the target English word using the original monolingual word embedding and our cross-lingual dependency based word embedding.

vestir". This is a direct consequence of incorporating the bilingual lexicon. Moreover, the top five closest words of both English and French mostly have the same part of speech. This is consistent with the finding in Figure 4.

Levin (1993) has shown that there is a strong connection between a verb's meaning and its syntactic behaviour. We compare the English side of our cross-lingual dependency based word embeddings with various other pre-trained monolingual English word embeddings and our monolingual embedding (section 4.3) on Verb-143 dataset (Baker et al., 2014). This dataset contains 143 pairs of verbs that are manually given score from 1 to 10 according to the meaning similarity. Table 3 shows the Pearson correlation

	Correlation
Senna (Collobert et al., 2011)	0.36
Skip-gram (Mikolov et al., 2013a)	0.27
RNN (Mikolov et al., 2011)	0.31
Our monolingual embedding	0.39
Our crosslingual embedding	0.44

Table 3: Compare the English side of our cross-lingual embeddings with various other embeddings evaluated on Verb-143 dataset (Baker et al., 2014). We directly use the pre-trained models from corresponding papers.

with human judgment for our embeddings and other pre-trained embeddings. As expected, our cross-lingual embeddings out-perform others embeddings on this dataset. This is partly because the syntactic behaviour is well encoded in our word embeddings through dependency relation.

Our embeddings encode not just cross-lingual correspondences, but also capture dependency relations which we expect might be beneficial for other NLP tasks based on dependency parsing, e.g., cross-lingual semantic role labelling where long-distance relationship can be captured by word embedding.

6 Conclusion

In this paper, we present a training method for building a dependency parser for a resource-poor language using a larger treebank in a high-resource language. Our approach takes advantage of the shared structure among languages to learn a universal parser and language-specific mappings to the lexicon, parts of speech and dependency arcs. Compared with supervised learning, our joint model gives a consistent 8-10% improvement over several different datasets in simulation low-resource scenarios. Interestingly, some small but consistent gains are still realised by joint cross-lingual training even on large complete treebanks. This suggests that our approach has utility not just in low resource settings. Our joint model is flexible, allowing the incorporation of a bilingual dictionary, which results in small improvements particularly for tiny training scenarios.

As the side-effect of training our joint model, we obtain cross-lingual word embeddings specialized for dependency parsing. We expect these embeddings to be beneficial to other syntatic and se-

mantic tasks. In future work, we plan to extend joint training to several languages, and further explore the idea of learning and exploiting cross-lingual embeddings.

Acknowledgments

This work was supported by the University of Melbourne and National ICT Australia (NICTA). Trevor Cohn is the recipient of an Australian Research Council Future Fellowship (project number FT130101105).

References

Simon Baker, Roi Reichart, and Anna Korhonen. 2014. An unsupervised model for instance level subcategorization acquisition. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 278–289.

Razvan C. Bunescu and Raymond J. Mooney. 2005. A shortest path dependency kernel for relation extraction. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pages 724–731, Stroudsburg, PA, USA. Association for Computational Linguistics.

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 740–750, Doha, Qatar, October. Association for Computational Linguistics.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.*, 12:2493–2537, November.

Hang Cui, Renxu Sun, Keya Li, Min-Yen Kan, and Tat-Seng Chua. 2005. Question answering passage retrieval using dependency relations. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '05, pages 400–407, New York, NY, USA. ACM.

Matthew S. Dryer and Martin Haspelmath, editors. 2013. *WALS Online*. Max Planck Institute for Evolutionary Anthropology, Leipzig.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.*, 12:2121–2159, July.

Long Duong, Trevor Cohn, Steven Bird, and Paul Cook. 2015. Low resource dependency parsing: Cross-lingual parameter sharing in a neural network parser. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 845–850, Beijing, China, July. Association for Computational Linguistics.

Greg Durrett, Adam Pauls, and Dan Klein. 2012. Syntactic transfer using a bilingual lexicon. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 1–11, Stroudsburg, PA, USA. Association for Computational Linguistics.

Andrew Garrett, Clare Sandy, Erik Maier, Line Mikkelsen, and Patrick Davidson. 2013. Developing the Karuk Treebank. Fieldwork Forum, Department of Linguistics, UC Berkeley.

David Kamholz, Jonathan Pool, and Susan Colowick. 2014. Panlex: Building a resource for panlingual lexical translation. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 3145–50, Reykjavik, Iceland. European Language Resources Association (ELRA).

Min-joo Kim. 2001. Does korean have adjectives. In *MIT Working Papers 43. Proceedings of HUMIT 2001*, pages 71–89. MIT Working Papers.

Philipp Koehn. 2005. Europarl: A Parallel Corpus for Statistical Machine Translation. In *Proceedings of the Tenth Machine Translation Summit (MT Summit X)*, pages 79–86, Phuket, Thailand.

B. Levin. 1993. *English Verb Classes and Alternations: A Preliminary Investigation*. University of Chicago Press.

Xuezhe Ma and Fei Xia. 2014. Unsupervised dependency parsing with transferring distribution via parallel guidance and entropy regularization. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1337–1348. Association for Computational Linguistics.

Ryan McDonald, Slav Petrov, and Keith Hall. 2011. Multi-source transfer of delexicalized dependency parsers. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 62–72.

Tomas Mikolov, Stefan Kombrink, Anoop Deoras, Lukar Burget, and Jan Honza Cernocky. 2011. Rnnlm – recurrent neural network language modeling toolkit. In *Proc. IEEE Automatic Speech Recognition and Understanding Workshop*, December.

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

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26*, pages 3111–3119.

Joakim Nivre, Cristina Bosco, Jinho Choi, Marie-Catherine de Marneffe, Timothy Dozat, Richárd Farkas, Jennifer Foster, Filip Ginter, Yoav Goldberg, Jan Hajič, Jenna Kanerva, Veronika Laippala, Alessandro Lenci, Teresa Lynn, Christopher Manning, Ryan McDonald, Anna Missilä, Simonetta Montemagni, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Maria Simi, Aaron Smith, Reut Tsarfaty, Veronika Vincze, and Daniel Zeman. 2015. Universal dependencies 1.0.

Joakim Nivre. 2006. *Inductive Dependency Parsing (Text, Speech and Language Technology)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Levent Özgür and Tunga Güngör. 2010. Text classification with the support of pruned dependency patterns. *Pattern Recogn. Lett.*, 31(12):1598–1607, September.

Raivis Skadiņš, Jörg Tiedemann, Roberts Rozis, and Daiga Deksne. 2014. Billions of parallel words for free: Building and using the eu bookshop corpus. In *Proceedings of the 9th International Conference on Language Resources and Evaluation (LREC-2014)*, Reykjavik, Iceland, May. European Language Resources Association (ELRA).

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

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

Oscar Täckström, Ryan McDonald, and Jakob Uszkoreit. 2012. Cross-lingual word clusters for direct transfer of linguistic structure. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL HLT '12, pages 477–487. Association for Computational Linguistics.

Oscar Täckström, Ryan McDonald, and Joakim Nivre. 2013. Target language adaptation of discriminative transfer parsers. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1061–1071, Atlanta, Georgia, June. Association for Computational Linguistics.

Laurens Van Der Maaten. 2014. Accelerating t-sne using tree-based algorithms. *J. Mach. Learn. Res.*, 15(1):3221–3245, January.

Peng Xu, Jaeho Kang, Michael Ringgaard, and Franz Och. 2009. Using a dependency parser to improve smt for subject-object-verb languages. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 245–253, Boulder, Colorado, June. Association for Computational Linguistics.

Daniel Zeman, Univerzita Karlova, and Philip Resnik. 2008. Cross-language parser adaptation between related languages. In *In IJCNLP-08 Workshop on NLP for Less Privileged Languages*, pages 35–42.

Improved Transition-Based Parsing by Modeling Characters instead of Words with LSTMs

Miguel Ballesteros[◇♠] **Chris Dyer**[♣♠] **Noah A. Smith**[♡]
[◇]NLP Group, Pompeu Fabra University, Barcelona, Spain
[♠]School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA
[♣]Marianas Labs, Pittsburgh, PA, USA
[♡]Computer Science & Engineering, University of Washington, Seattle, WA, USA
miguel.ballesteros@upf.edu, chris@marianaslabs.com, nasmith@cs.washington.edu

Abstract

We present extensions to a continuous-state dependency parsing method that makes it applicable to morphologically rich languages. Starting with a high-performance transition-based parser that uses long short-term memory (LSTM) recurrent neural networks to learn representations of the parser state, we replace lookup-based word representations with representations constructed from the orthographic representations of the words, also using LSTMs. This allows statistical sharing across word forms that are similar on the surface. Experiments for morphologically rich languages show that the parsing model benefits from incorporating the character-based encodings of words.

1 Introduction

At the heart of natural language parsing is the challenge of representing the "state" of an algorithm—what parts of a parse have been built and what parts of the input string are not yet accounted for—as it incrementally constructs a parse. Traditional approaches rely on independence assumptions, decomposition of scoring functions, and/or greedy approximations to keep this space manageable. **Continuous-state** parsers have been proposed, in which the state is embedded as a vector (Titov and Henderson, 2007; Stenetorp, 2013; Chen and Manning, 2014; Dyer et al., 2015; Zhou et al., 2015; Weiss et al., 2015). Dyer et al. reported state-of-the-art performance on English and Chinese benchmarks using a transition-based parser whose continuous-state embeddings were constructed using LSTM recurrent neural networks (RNNs) whose parameters were estimated to maximize the probability of a gold-standard sequence of parse actions.

The primary contribution made in this work is to take the idea of continuous-state parsing a step further by making the word embeddings that are used to construct the parse state sensitive to the morphology of the words.[1] Since it it is well known that a word's form often provides strong evidence regarding its grammatical role in morphologically rich languages (Ballesteros, 2013, *inter alia*), this has promise to improve accuracy and statistical efficiency relative to traditional approaches that treat each word type as opaque and independently modeled. In the traditional parameterization, words with similar grammatical roles will only be embedded near each other if they are observed in similar contexts with sufficient frequency. Our approach reparameterizes word embeddings using the same RNN machinery used in the parser: a word's vector is calculated based on the sequence of orthographic symbols representing it (§3).

Although our model is provided no supervision in the form of explicit morphological annotation, we find that it gives a large performance increase when parsing morphologically rich languages in the SPMRL datasets (Seddah et al., 2013; Seddah and Tsarfaty, 2014), especially in agglutinative languages and the ones that present extensive case systems (§4). In languages that show little morphology, performance remains good, showing that the RNN composition strategy is capable of capturing both morphological regularities and arbitrariness in the sense of Saussure (1916). Finally, a particularly noteworthy result is that we find that character-based word embeddings in some cases obviate explicit POS information, which is usually found to be indispensable for accurate parsing.

A secondary contribution of this work is to show that the continuous-state parser of Dyer et al. (2015) can learn to generate nonprojective trees. We do this by augmenting its transition operations

[1]Software for replicating the experiments is available from `https://github.com/clab/lstm-parser`.

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 349–359,
Lisbon, Portugal, 17-21 September 2015.

with a SWAP operation (Nivre, 2009) (§2.4), enabling the parser to produce nonprojective dependencies which are often found in morphologically rich languages.

2 An LSTM Dependency Parser

We begin by reviewing the parsing approach of Dyer et al. (2015) on which our work is based.

Like most transition-based parsers, Dyer et al.'s parser can be understood as the sequential manipulation of three data structures: a buffer B initialized with the sequence of words to be parsed, a stack S containing partially-built parses, and a list A of actions previously taken by the parser. In particular, the parser implements the arc-standard parsing algorithm (Nivre, 2004).

At each time step t, a transition action is applied that alters these data structures by pushing or popping words from the stack and the buffer; the operations are listed in Figure 1.

Along with the discrete transitions above, the parser calculates a vector representation of the states of B, S, and A; at time step t these are denoted by $\mathbf{b}_t$, $\mathbf{s}_t$, and $\mathbf{a}_t$, respectively. The total parser state at t is given by

$$\mathbf{p}_t = \max\{\mathbf{0}, \mathbf{W}[\mathbf{s}_t; \mathbf{b}_t; \mathbf{a}_t] + \mathbf{d}\} \quad (1)$$

where the matrix $\mathbf{W}$ and the vector $\mathbf{d}$ are learned parameters. This continuous-state representation $\mathbf{p}_t$ is used to decide which operation to apply next, updating B, S, and A (Figure 1).

We elaborate on the design of $\mathbf{b}_t$, $\mathbf{s}_t$, and $\mathbf{a}_t$ using RNNs in §2.1, on the representation of partial parses in S in §2.2, and on the parser's decision mechanism in §2.3. We discuss the inclusion of SWAP in §2.4.

2.1 Stack LSTMs

RNNs are functions that read a sequence of vectors incrementally; at time step t the vector $\mathbf{x}_t$ is read in and the hidden state $\mathbf{h}_t$ computed using $\mathbf{x}_t$ and the previous hidden state $\mathbf{h}_{t-1}$. In principle, this allows retaining information from time steps in the distant past, but the nonlinear "squashing" functions applied in the calcluation of each $\mathbf{h}_t$ result in a decay of the error signal used in training with backpropagation. LSTMs are a variant of RNNs designed to cope with this "vanishing gradient" problem using an extra memory "cell" (Hochreiter and Schmidhuber, 1997; Graves, 2013).

Past work explains the computation within an LSTM through the metaphors of deciding how much of the current input to pass into memory ($\mathbf{i}_t$) or forget ($\mathbf{f}_t$). We refer interested readers to the original papers and present only the recursive equations updating the memory cell $\mathbf{c}_t$ and hidden state $\mathbf{h}_t$ given $\mathbf{x}_t$, the previous hidden state $\mathbf{h}_{t-1}$, and the memory cell $\mathbf{c}_{t-1}$:

$$\begin{aligned}
\mathbf{i}_t &= \sigma(\mathbf{W}_{ix}\mathbf{x}_t + \mathbf{W}_{ih}\mathbf{h}_{t-1} + \mathbf{W}_{ic}\mathbf{c}_{t-1} + \mathbf{b}_i) \\
\mathbf{f}_t &= \mathbf{1} - \mathbf{i}_t \\
\mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \\
&\qquad \mathbf{i}_t \odot \tanh(\mathbf{W}_{cx}\mathbf{x}_t + \mathbf{W}_{ch}\mathbf{h}_{t-1} + \mathbf{b}_c) \\
\mathbf{o}_t &= \sigma(\mathbf{W}_{ox}\mathbf{x}_t + \mathbf{W}_{oh}\mathbf{h}_{t-1} + \mathbf{W}_{oc}\mathbf{c}_t + \mathbf{b}_o) \\
\mathbf{h}_t &= \mathbf{o}_t \odot \tanh(\mathbf{c}_t),
\end{aligned}$$

where σ is the component-wise logistic sigmoid function and $\odot$ is the component-wise (Hadamard) product. Parameters are all represented using $\mathbf{W}$ and $\mathbf{b}$. This formulation differs slightly from the classic LSTM formulation in that it makes use of "peephole connections" (Gers et al., 2002) and defines the forget gate so that it sums with the input gate to $\mathbf{1}$ (Greff et al., 2015). To improve the representational capacity of LSTMs (and RNNs generally), they can be stacked in "layers." In these architectures, the input LSTM at higher layers at time t is the value of $\mathbf{h}_t$ computed by the lower layer (and $\mathbf{x}_t$ is the input at the lowest layer).

The **stack LSTM** augments the left-to-right sequential model of the conventional LSTM with a stack pointer. As in the LSTM, new inputs are added in the right-most position, but the stack pointer indicates which LSTM cell provides $\mathbf{c}_{t-1}$ and $\mathbf{h}_{t-1}$ for the computation of the next iterate. Further, the stack LSTM provides a pop operation that moves the stack pointer to the previous element. Hence each of the parser data structures (B, S, and A) is implemented with its own stack LSTM, each with its own parameters. The values of $\mathbf{b}_t$, $\mathbf{s}_t$, and $\mathbf{a}_t$ are the $\mathbf{h}_t$ vectors from their respective stack LSTMs.

2.2 Composition Functions

Whenever a REDUCE operation is selected, two tree fragments are popped off of S and combined to form a new tree fragment, which is then popped back onto S (see Figure 1). This tree must be embedded as an input vector $\mathbf{x}_t$.

To do this, Dyer et al. (2015) use a recursive neural network g_r (for relation r) that composes

$\textbf{Stack}_t$	$\textbf{Buffer}_t$	Action	$\textbf{Stack}_{t+1}$	$\textbf{Buffer}_{t+1}$	Dependency
$(\mathbf{u}, u), (\mathbf{v}, v), S$	B	REDUCE-RIGHT(r)	$(g_r(\mathbf{u}, \mathbf{v}), u), S$	B	$u \xrightarrow{r} v$
$(\mathbf{u}, u), (\mathbf{v}, v), S$	B	REDUCE-LEFT(r)	$(g_r(\mathbf{v}, \mathbf{u}), v), S$	B	$u \xleftarrow{r} v$
S	$(\mathbf{u}, u), B$	SHIFT	$(\mathbf{u}, u), S$	B	—
$(\mathbf{u}, u), (\mathbf{v}, v), S$	B	SWAP	$(\mathbf{u}, u), S$	$(\mathbf{v}, v), B$	—

Figure 1: Parser transitions indicating the action applied to the stack and buffer and the resulting stack and buffer states. Bold symbols indicate (learned) embeddings of words and relations, script symbols indicate the corresponding words and relations. Dyer et al. (2015) used the SHIFT and REDUCE operations in their continuous-state parser; we add SWAP.

the representations of the two subtrees popped from S (we denote these by $\mathbf{u}$ and $\mathbf{v}$), resulting in a new vector $g_r(\mathbf{u}, \mathbf{v})$ or $g_r(\mathbf{v}, \mathbf{u})$, depending on the direction of attachment. The resulting vector embeds the tree fragment in the same space as the words and other tree fragments. This kind of composition was thoroughly explored in prior work (Socher et al., 2011; Socher et al., 2013b; Hermann and Blunsom, 2013; Socher et al., 2013a); for details, see Dyer et al. (2015).

2.3 Predicting Parser Decisions

The parser uses a probabilistic model of parser decisions at each time step t. Letting $\mathcal{A}(S, B)$ denote the set of allowed transitions given the stack S and buffer S (i.e., those where preconditions are met; see Figure 1), the probability of action $z \in \mathcal{A}(S, B)$ defined using a log-linear distribution:

$$p(z \mid \mathbf{p}_t) = \frac{\exp\left(\mathbf{g}_z^\top \mathbf{p}_t + q_z\right)}{\sum_{z' \in \mathcal{A}(S,B)} \exp\left(\mathbf{g}_{z'}^\top \mathbf{p}_t + q_{z'}\right)} \quad (2)$$

(where $\mathbf{g}_z$ and q_z are parameters associated with each action type z).

Parsing proceeds by always choosing the most probable action from $\mathcal{A}(S, B)$. The probabilistic definition allows parameter estimation for all of the parameters ($\mathbf{W}_*$, $\mathbf{b}_*$ in all three stack LSTMs, as well as $\mathbf{W}$, $\mathbf{d}$, $\mathbf{g}_*$, and q_*) by maximizing the conditional likelihood of each correct parser decisions given the state.

2.4 Adding the SWAP Operation

Dyer et al. (2015)'s parser implemented the most basic version of the arc-standard algorithm, which is capable of producing only projective parse trees. In order to deal with nonprojective trees, we also add the SWAP operation which allows nonprojective trees to be produced.

The SWAP operation, first introduced by Nivre (2009), allows a transition-based parser to produce nonprojective trees. Here, the inclusion of the SWAP operation requires breaking the linearity of the stack by removing tokens that are not at the top of the stack. This is easily handled with the stack LSTM. Figure 1 shows how the parser is capable of moving words from the stack (S) to the buffer (B), breaking the linear order of words. Since a node that is swapped may have already been assigned as the head of a dependent, the buffer (B) can now also contain tree fragments.

3 Word Representations

The main contribution of this paper is to change the word representations. In this section, we present the standard word embeddings as in Dyer et al. (2015), and the improvements we made generating word embeddings designed to capture morphology based on orthographic strings.

3.1 Baseline: Standard Word Embeddings

Dyer et al.'s parser generates a word representation for each input token by concatenating two vectors: a vector representation for each word type ($\mathbf{w}$) and a representation ($\mathbf{t}$) of the POS tag of the token (if it is used), provided as auxiliary input to the parser.[2] A linear map ($\mathbf{V}$) is applied to the resulting vector and passed through a component-wise ReLU:

$$\mathbf{x} = \max\{\mathbf{0}, \mathbf{V}[\mathbf{w}; \mathbf{t}] + \mathbf{b}\}$$

For out-of-vocabulary words, the parser uses an "UNK" token that is handled as a separate word during parsing time. This mapping can be shown schematically as in Figure 2.

[2] Dyer et al. (2015), included a third input representation learned from a neural language model ($\tilde{\mathbf{w}}_{\text{LM}}$). We do not include these pretrained representations in our experiments, focusing instead on character-based representations.

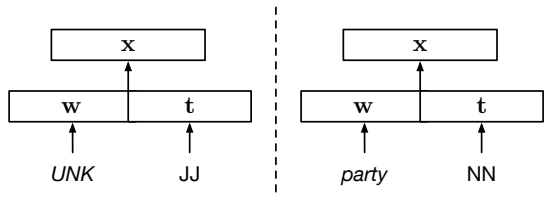


Figure 2: Baseline model word embeddings for an in-vocabulary word that is tagged with POS tag NN (right) and an out-of-vocabulary word with POS tag JJ (left).

3.2 Character-Based Embeddings of Words

Following Ling et al. (2015), we compute character-based continuous-space vector embeddings of words using bidirectional LSTMs (Graves and Schmidhuber, 2005). When the parser initiates the learning process and populates the buffer with all the words from the sentence, it reads the words character by character from left to right and computes a continuous-space vector embedding the character sequence, which is the $\mathbf{h}$ vector of the LSTM; we denote it by $\overrightarrow{\mathbf{w}}$. The same process is also applied in reverse (albeit with different parameters), computing a similar continuous-space vector embedding starting from the last character and finishing at the first ($\overleftarrow{\mathbf{w}}$); again each character is represented with an LSTM cell. After that, we concatenate these vectors and a (learned) representation of their tag to produce the representation $\mathbf{w}$. As in §3.1, a linear map ($\mathbf{V}$) is applied and passed through a component-wise ReLU.

$$\mathbf{x} = \max\left\{\mathbf{0}, \mathbf{V}[\overrightarrow{\mathbf{w}}; \overleftarrow{\mathbf{w}}; \mathbf{t}] + \mathbf{b}\right\}$$

This process is shown schematically in Figure 3.

Note that under this representation, out-of-vocabulary words are treated as bidirectional LSTM encodings and thus they will be "close" to other words that the parser has seen during training, ideally close to their more frequent, syntactically similar morphological relatives. We conjecture that this will give a clear advantage over a single "UNK" token for all the words that the parser does not see during training, as done by Dyer et al. (2015) and other parsers without additional resources. In §4 we confirm this hypothesis.

4 Experiments

We applied our parsing model and several variations of it to several parsing tasks and report results below.

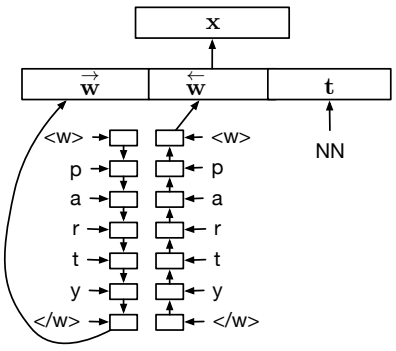


Figure 3: Character-based word embedding of the word *party*. This representation is used for both in-vocabulary and out-of-vocabulary words.

4.1 Data

In order to find out whether the character-based representations are capable of learning the morphology of words, we applied the parser to morphologically rich languages specifically the treebanks of the SPMRL shared task (Seddah et al., 2013; Seddah and Tsarfaty, 2014): Arabic (Maamouri et al., 2004), Basque (Aduriz et al., 2003), French (Abeillé et al., 2003), German (Seeker and Kuhn, 2012), Hebrew (Sima'an et al., 2001), Hungarian (Vincze et al., 2010), Korean (Choi, 2013), Polish (Świdziński and Woliński, 2010) and Swedish (Nivre et al., 2006b). For all the corpora of the SPMRL Shared Task we used predicted POS tags as provided by the shared task organizers.[3] For these datasets, evaluation is calculated using `eval07.pl`, which includes punctuation.

We also experimented with the Turkish dependency treebank[4] (Oflazer et al., 2003) of the CoNLL-X Shared Task (Buchholz and Marsi, 2006). We used gold POS tags, as is common with the CoNLL-X data sets.

To put our results in context with the most recent neural network transition-based parsers, we run the parser in the same Chinese and English

[3]The POS tags were calculated with the MarMot tagger (Müller et al., 2013) by the best performing system of the SPMRL Shared Task (Björkelund et al., 2013). Arabic: 97.38. Basque: 97.02. French: 97.61. German: 98.10. Hebrew: 97.09. Hungarian: 98.72. Korean: 94.03. Polish: 98.12. Swedish: 97.27.

[4]Since the Turkish dependency treebank does not have a development set, we extracted the last 150 sentences from the 4996 sentences of the training set as a development set.

setups as Chen and Manning (2014) and Dyer et al. (2015). For Chinese, we use the Penn Chinese Treebank 5.1 (CTB5) following Zhang and Clark (2008b),[5] with gold POS tags. For English, we used the Stanford Dependency (SD) representation of the Penn Treebank[6] (Marcus et al., 1993; Marneffe et al., 2006).[7]. Results for Turkish, Chinese, and English are calculated using the CoNLL-X `eval.pl` script, which ignores punctuation symbols.

4.2 Experimental Configurations

In order to isolate the improvements provided by the LSTM encodings of characters, we run the stack LSTM parser in the following configurations:

- **Words**: words only, as in §3.1 (but without POS tags)
- **Chars**: character-based representations of words with bidirectional LSTMs, as in §3.2 (but without POS tags)
- **Words + POS**: words and POS tags (§3.1)
- **Chars + POS**: character-based representations of words with bidirectional LSTMs plus POS tags (§3.2)

None of the experimental configurations include pretrained word-embeddings or any additional data resources. All experiments include the SWAP transition, meaning that nonprojective trees can be produced in any language.

Dimensionality. The full version of our parsing model sets dimensionalities as follows. LSTM hidden states are of size 100, and we use two layers of LSTMs for each stack. Embeddings of the parser actions used in the composition functions have 20 dimensions, and the output embedding size is 20 dimensions. The learned word representations embeddings have 32 dimensions when used, while the character-based representations have 100 dimensions, when used. Part of speech embeddings have 12 dimensions. These dimensionalities were chosen after running several tests with different values, but a more careful selection of these values would probably further improve results.

[5]Training: 001–815, 1001–1136. Development: 886–931, 1148–1151. Test: 816–885, 1137–1147.

[6]Training: 02–21. Development: 22. Test: 23.

[7]The POS tags are predicted by using the Stanford Tagger (Toutanova et al., 2003) with an accuracy of 97.3%.

4.3 Training Procedure

Parameters are initialized randomly—refer to Dyer et al. (2015) for specifics—and optimized using stochastic gradient descent (without minibatches) using derivatives of the negative log likelihood of the sequence of parsing actions computed using backpropagation. Training is stopped when the learned model's UAS stops improving on the development set, and this model is used to parse the test set. No pretraining of any parameters is done.

4.4 Results and Discussion

Tables 1 and 2 show the results of the parsers for the development sets and the final test sets, respectively. Most notable are improvements for agglutinative languages—Basque, Hungarian, Korean, and Turkish—both when POS tags are included and when they are not. Consistently, across all languages, **Chars** outperforms **Words**, suggesting that the character-level LSTMs are learning representations that capture similar information to parts of speech. On average, **Chars** is on par with **Words + POS**, and the best average of labeled attachment scores is achieved with **Chars + POS**.

It is common practice to encode morphological information in treebank POS tags; for instance, the Penn Treebank includes English number and tense (e.g., NNS is plural noun and VBD is verb in past tense). Even if our character-based representations are capable of encoding the same kind of information, existing POS tags suffice for high accuracy. However, the POS tags in treebanks for morphologically rich languages do not seem to be enough.

Swedish, English, and French use suffixes for the verb tenses and number,[8] while Hebrew uses prepositional particles rather than grammatical case. Tsarfaty (2006) and Cohen and Smith (2007) argued that, for Hebrew, determining the correct morphological segmentation is dependent on syntactic context. Our approach sidesteps this step, capturing the same kind of information in the vectors, and learning it from syntactic context. Even for Chinese, which is not morphologically rich, **Chars** shows a benefit over **Words**, perhaps by capturing regularities in syllable structure within words.

[8]Tense and number features provide little improvement in a transition-based parser, compared with other features such as case, when the POS tags are included (Ballesteros, 2013).

UAS

Language	Words	Chars	Words + POS	Chars + POS
Arabic	86.14	**87.20**	**87.44**	87.07
Basque	78.42	**84.97**	83.49	**85.58**
French	84.84	**86.21**	**87.00**	86.33
German	88.14	**90.94**	91.16	**91.23**
Hebrew	79.73	**79.92**	**81.99**	80.76
Hungarian	72.38	**80.16**	78.47	**80.85**
Korean	78.98	**88.98**	87.36	**89.14**
Polish	73.29	**85.69**	**89.32**	88.54
Swedish	73.44	**75.03**	**80.02**	78.85
Turkish	71.10	**74.91**	77.13	**77.96**
Chinese	79.43	**80.36**	**85.98**	85.81
English	91.64	**91.98**	**92.94**	92.49
Average	79.79	**83.86**	85.19	**85.38**

LAS

Language	Words	Chars	Words + POS	Chars + POS
Arabic	82.73	**84.34**	**84.81**	84.36
Basque	67.08	**78.22**	74.31	**79.52**
French	80.32	**81.70**	**82.71**	81.51
German	85.36	**88.68**	**89.04**	88.83
Hebrew	69.42	**70.58**	**74.11**	72.18
Hungarian	62.14	**75.61**	69.50	**76.16**
Korean	67.48	**86.80**	83.80	**86.88**
Polish	65.13	**78.23**	**81.84**	80.97
Swedish	64.77	**66.74**	**72.09**	69.88
Turkish	53.98	**62.91**	62.30	**62.87**
Chinese	75.64	**77.06**	**84.36**	84.10
English	88.60	**89.58**	**90.63**	90.08
Average	71.89	**78.37**	79.13	**79.78**

Table 1: Unlabeled attachment scores (left) and labeled attachment scores (right) on the **development** sets (not a standard development set for Turkish). In each table, the first two columns show the results of the parser with word lookup (**Words**) vs. character-based (**Chars**) representations. The last two columns add POS tags. Boldface shows the better result comparing **Words** vs. **Chars** and comparing **Words + POS** vs. **Chars + POS**.

UAS

Language	Words	Chars	Words + POS	Chars + POS
Arabic	85.21	**86.08**	86.05	**86.07**
Basque	77.06	**85.19**	82.92	**85.22**
French	83.74	**85.34**	**86.15**	85.78
German	82.75	**86.80**	**87.33**	87.26
Hebrew	77.62	**79.93**	**80.68**	80.17
Hungarian	72.78	**80.35**	78.64	**80.92**
Korean	78.70	**88.39**	86.85	**88.30**
Polish	72.01	**83.44**	**87.06**	85.97
Swedish	76.39	**79.18**	**83.43**	83.24
Turkish	71.70	**76.32**	75.32	**76.34**
Chinese	79.01	**79.94**	**85.96**	85.30
English	91.16	**91.47**	**92.57**	91.63
Average	79.01	**85.36**	84.41	**84.68**

LAS

Language	Words	Chars	Words + POS	Chars + POS
Arabic	82.05	**83.41**	**83.46**	83.40
Basque	66.61	**79.09**	73.56	**78.61**
French	79.22	**80.92**	**82.03**	81.08
German	79.15	**84.04**	**84.62**	84.49
Hebrew	68.71	**71.26**	**72.70**	72.26
Hungarian	61.93	**75.19**	69.31	**76.34**
Korean	67.50	**86.27**	83.37	**86.21**
Polish	63.96	**76.84**	**79.83**	78.24
Swedish	67.69	**71.19**	**76.40**	74.47
Turkish	54.55	**64.34**	61.22	**62.28**
Chinese	74.79	**76.29**	**84.40**	83.72
English	88.42	**88.94**	**90.31**	89.44
Average	71.22	**78.15**	78.43	**79.21**

Table 2: Unlabeled attachment scores (left) and labeled attachment scores (right) on the **test** sets. In each table, the first two columns show the results of the parser with word lookup (**Words**) vs. character-based (**Chars**) representations. The last two columns add POS tags. Boldface shows the better result comparing **Words** vs. **Chars** and comparing **Words + POS** vs. **Chars + POS**.

4.4.1 Learned Word Representations

Figure 4 visualizes a sample of the character-based bidirectional LSTMs's learned representations (**Chars**). Clear clusters of past tense verbs, gerunds, and other syntactic classes are visible. The colors in the figure represent the most common POS tag for each word.

4.4.2 Out-of-Vocabulary Words

The character-based representation for words is notably beneficial for out-of-vocabulary (OOV) words. We tested this specifically by comparing **Chars** to a model in which all OOVs are replaced by the string "UNK" during parsing. This always has a negative effect on LAS (average -4.5 points, -2.8 UAS). Figure 5 shows how this drop varies with the development OOV rate across treebanks; most extreme is Korean, which drops 15.5 LAS. A similar, but less pronounced pattern, was observed for models that include POS.

Interestingly, this artificially impoverished model is still consistently better than **Words** for all languages (e.g., for Korean, by 4 LAS). This implies that not all of the improvement is due to OOV words; statistical sharing across orthographically close words is beneficial, as well.

4.4.3 Computational Requirements

The character-based representations make the parser slower, since they require composing the character-based bidirectional LSTMs for each

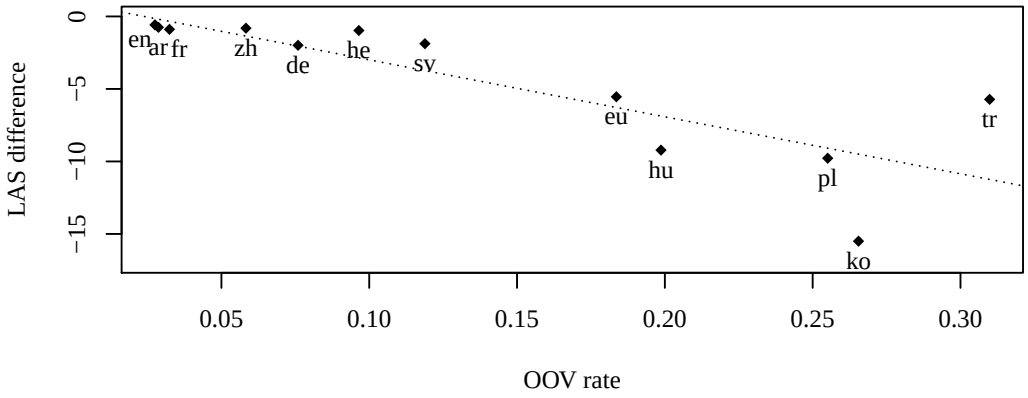


Figure 5: On the x-axis is the OOV rate in development data, by treebank; on the y-axis is the difference in development-set LAS between **Chars** model as described in §3.2 and one in which all OOV words are given a single representation.

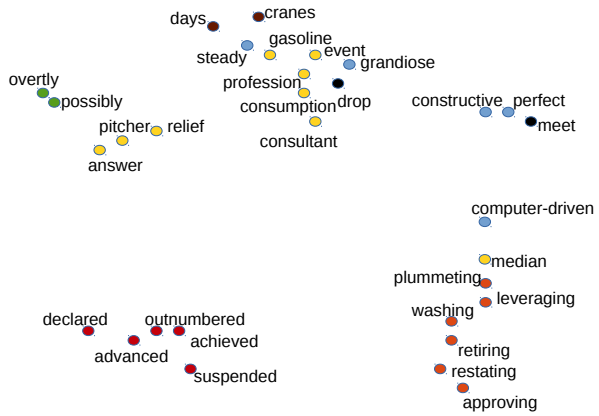


Figure 4: Character-based word representations of 30 random words from the English development set (**Chars**). Dots in red represent past tense verbs; dots in orange represent gerund verbs; dots in black represent present tense verbs; dots in blue represent adjectives; dots in green represent adverbs; dots in yellow represent singular nouns; dots in brown represent plural nouns. The visualization was produced using t-SNE; see `http://lvdmaaten.github.io/tsne/`.

word of the input sentence; however, at test time these results could be cached. On average, **Words** parses a sentence in 44 ms, while**Chars** needs 130 ms.[9] Training time is affected by the same constant, needing some hours to have a competitive model. In terms of memory, **Words** requires on average 300 MB of main memory for both training and parsing, while **Chars** requires 450 MB.

4.4.4 Comparison with State-of-the-Art

Table 3 shows a comparison with state-of-the-art parsers. We include greedy transition-based parsers that, like ours, do not apply a beam search (Zhang and Clark, 2008b) or a dynamic oracle (Goldberg and Nivre, 2013). For all the SPMRL languages we show the results of Ballesteros (2013), who reported results after carrying out a careful automatic morphological feature selection experiment. For Turkish, we show the results of Nivre et al. (2006a) which also carried out a careful manual morphological feature selection. Our parser outperforms these in most cases. Since those systems rely on morphological features, we believe that this comparison shows even more that the character-based representations are capturing morphological information, though without explicit morphological features. For English and Chinese, we report (Dyer et al., 2015) which is **Words + POS** but with pretrained word embeddings.

We also show the best reported results on these datasets. For the SPMRL data sets, the best performing system of the shared task is either Björkelund et al. (2013) or Björkelund et al. (2014), which are consistently better than our sys-

[9] We are using a machine with 32 Intel Xeon CPU E5-2650 at 2.00GHz; the parser runs on a single core.

	This Work			Best Greedy Result			Best Published Result		
Language	**UAS**	**LAS**	**System**	**UAS**	**LAS**	**System**	**UAS**	**LAS**	**System**
Arabic	86.08	83.41	**Chars**	84.57	81.90	B'13	88.32	86.21	B+'13
Basque	85.22	78.61	**Chars + POS**	84.33	78.58	B'13	89.96	85.70	B+'14
French	86.15	82.03	**Words + POS**	83.35	77.98	B'13	89.02	85.66	B+'14
German	87.33	84.62	**Words + POS**	85.38	82.75	B'13	91.64	89.65	B+'13
Hebrew	80.68	72.70	**Words + POS**	79.89	73.01	B'13	87.41	81.65	B+'14
Hungarian	80.92	76.34	**Chars + POS**	83.71	79.63	B'13	89.81	86.13	B+'13
Korean	88.39	86.27	**Chars**	85.72	82.06	B'13	89.10	87.27	B+'14
Polish	87.06	79.83	**Words + POS**	85.80	79.89	B'13	91.75	87.07	B+'13
Swedish	83.43	76.40	**Words + POS**	83.20	75.82	B'13	88.48	82.75	B+'14
Turkish	76.32	64.34	**Chars**	75.82	65.68	N+'06a	77.55	n/a	K+'10
Chinese	85.96	84.40	**Words + POS**	87.20	85.70	D+'15	87.20	85.70	D+'15
English	92.57	90.31	**Words + POS**	93.10	90.90	D+'15	94.08	92.19	W+'15

Table 3: Test-set performance of our best results (according to UAS or LAS, whichever has the larger difference), compared to state-of-the-art greedy transition-based parsers ("Best Greedy Result") and best results reported ("Best Published Result"). All of the systems we compare against use explicit morphological features and/or one of the following: pretrained word embeddings, unlabeled data and a combination of parsers; our models do not. B'13 is Ballesteros (2013); N+'06a is Nivre et al. (2006a); D+'15 is Dyer et al. (2015); B+'13 is Björkelund et al. (2013); B+'14 is Björkelund et al. (2014); K+'10 is Koo et al. (2010); W+'15 is Weiss et al. (2015).

tem for all languages. Note that the comparison is harsh to our system, which does not use unlabeled data or explicit morphological features nor any combination of different parsers. For Turkish, we report the results of Koo et al. (2010), which only reported unlabeled attachment scores. For English, we report (Weiss et al., 2015) and for Chinese, we report (Dyer et al., 2015) which is **Words + POS** but with pretrained word embeddings.

5 Related Work

Character-based representations have been explored in other NLP tasks; for instance, dos Santos and Zadrozny (2014) and dos Santos and Guimarães (2015) learned character-level neural representations for POS tagging and named entity recognition, getting a large error reduction in both tasks. Our approach is similar to theirs. Others have used character-based models as features to improve existing models. For instance, Chrupała (2014) used character-based recurrent neural networks to normalize tweets.

Botha and Blunsom (2014) show that stems, prefixes and suffixes can be used to learn useful word representations but relying on an external morphological analyzer. That is, they learn the morpheme-meaning relationship with an additive model, whereas we do not need a morphological analyzer. Similarly, Chen et al. (2015) proposed joint learning of character and word embeddings for Chinese, claiming that characters contain rich information.

Methods for joint morphological disambiguation and parsing have been widely explored Tsarfaty (2006; Cohen and Smith (2007; Goldberg and Tsarfaty (2008; Goldberg and Elhadad (2011). More recently, Bohnet et al. (2013) presented an arc-standard transition-based parser that performs competitively for joint morphological tagging and dependency parsing for richly inflected languages, such as Czech, Finnish, German, Hungarian, and Russian. Our model seeks to achieve a similar benefit to parsing without explicitly reasoning about the internal structure of words.

Zhang et al. (2013) presented efforts on Chinese parsing with characters showing that Chinese can be parsed at the character level, and that Chinese word segmentation is useful for predicting the correct POS tags (Zhang and Clark, 2008a).

To the best of our knowledge, previous work has not used character-based embeddings to improve dependency parsers, as done in this paper.

6 Conclusion

We have presented several interesting findings. First, we add new evidence that character-based representations are useful for NLP tasks. In this paper, we demonstrate that they are useful for transition-based dependency parsing, since they are capable of capturing morphological information crucial for analyzing syntax.

The improvements provided by the character-based representations using bidirectional LSTMs are strong for agglutinative languages, such as

Basque, Hungarian, Korean, and Turkish, comparing favorably to POS tags as encoded in those languages' currently available treebanks. This outcome is important, since annotating morphological information for a treebank is expensive. Our finding suggests that the best investment of annotation effort may be in dependencies, leaving morphological features to be learned implicitly from strings.

The character-based representations are also a way of overcoming the out-of-vocabulary problem; without any additional resources, they enable the parser to substantially improve the performance when OOV rates are high. We expect that, in conjunction with a pretraing regime, or in conjunction with distributional word embeddings, further improvements could be realized.

Acknowledgments

MB was supported by the European Commission under the contract numbers FP7-ICT-610411 (project MULTISENSOR) and H2020-RIA-645012 (project KRISTINA). This research was supported by the U.S. Army Research Laboratory and the U.S. Army Research Office under contract/grant number W911NF-10-1-0533 and NSF IIS-1054319. This work was completed while NAS was at CMU. Thanks to Joakim Nivre, Bernd Bohnet, Fei Liu and Swabha Swayamdipta for useful comments.

References

Anne Abeillé, Lionel Clément, and François Toussenel. 2003. Building a treebank for French. In *Treebanks*. Springer.

Itziar Aduriz, María Jesús Aranzabe, Jose Mari Arriola, Aitziber Atutxa, Arantza Díaz de Ilarraza, Aitzpea Garmendia, and Maite Oronoz. 2003. Construction of a Basque dependency treebank. In *Proc of TLT*.

Miguel Ballesteros. 2013. Effective morphological feature selection with maltoptimizer at the SPMRL 2013 shared task. In *Proc. of SPMRL-EMNLP*.

Anders Björkelund, Ozlem Cetinoglu, Richárd Farkas, Thomas Mueller, and Wolfgang Seeker. 2013. (Re)ranking Meets Morphosyntax: State-of-the-art Results from the SPMRL 2013 Shared Task. In *SPMRL-EMNLP*.

Anders Björkelund, Özlem Çetinoğlu, Agnieszka Faleńska, Richárd Farkas, Thomas Mueller, Wolfgang Seeker, and Zsolt Szántó. 2014. Introducing the IMS-Wrocław-Szeged-CIS entry at the SPMRL 2014 Shared Task: Reranking and Morpho-syntax meet Unlabeled Data. In *SPMRL-SANCL*.

Bernd Bohnet, Joakim Nivre, Igor Boguslavsky, Richard Farkas, Filip Ginter, and Jan Hajič. 2013. Joint morphological and syntactic analysis for richly inflected languages. *TACL*, 1.

Jan A. Botha and Phil Blunsom. 2014. Compositional Morphology for Word Representations and Language Modelling. In *ICML*.

Sabine Buchholz and Erwin Marsi. 2006. CoNLL-X. In *Proc of CoNLL*.

Danqi Chen and Christopher D. Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proc. EMNLP*.

Xinxiong Chen, Lei Xu, Zhiyuan Liu, Maosong Sun, and Huanbo Luan. 2015. Joint learning of character and word embeddings. In *Proc. IJCAI*.

Jinho D. Choi. 2013. Preparing Korean Data for the Shared Task on Parsing Morphologically Rich Languages. *ArXiv e-prints*, September.

Grzegorz Chrupała. 2014. Normalizing tweets with edit scripts and recurrent neural embeddings. In *Proc of ACL*.

Shay B. Cohen and Noah A. Smith. 2007. Joint morphological and syntactic disambiguation. In *Proc. EMNLP-CoNLL*.

Cicero Nogueira dos Santos and Victor Guimarães. 2015. Boosting named entity recognition with neural character embeddings. *Arxiv*.

Cicero dos Santos and Bianca Zadrozny. 2014. Learning character-level representations for part-of-speech tagging. In *Proc of ICML-14*.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proc of ACL*.

Felix A. Gers, Nicol N. Schraudolph, and Jürgen Schmidhuber. 2002. Learning precise timing with LSTM recurrent networks. *JMLR*.

Yoav Goldberg and Michael Elhadad. 2011. Joint Hebrew segmentation and parsing using a PCFG-LA lattice parser. In *Proc of ACL*.

Yoav Goldberg and Joakim Nivre. 2013. Training deterministic parsers with non-deterministic oracles. *TACL*.

Yoav Goldberg and Reut Tsarfaty. 2008. A single generative model for joint morphological segmentation and syntactic parsing. In *Proc of ACL*.

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

Alex Graves. 2013. Generating sequences with recurrent neural networks. *CoRR*, abs/1308.0850.

Klaus Greff, Rupesh Kumar Srivastava, Jan Koutník, Bas R. Steunebrink, and Jürgen Schmidhuber. 2015. LSTM: A search space odyssey. *CoRR*, abs/1503.04069.

Karl Moritz Hermann and Phil Blunsom. 2013. The role of syntax in vector space models of compositional semantics. In *Proc. ACL*.

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

Terry Koo, Alexander M. Rush, Michael Collins, Tommi Jaakkola, and David Sontag. 2010. Dual decomposition for parsing with non-projective head automata. In *Proc of EMNLP*.

Wang Ling, Tiago Luís, Luís Marujo, Ramón Fernandez Astudillo, Silvio Amir, Chris Dyer, Alan W Black, and Isabel Trancoso. 2015. Finding function in form: Compositional character models for open vocabulary word representation. In *Proc. EMNLP*.

Mohamed Maamouri, Ann Bies, Tim Buckwalter, and Wigdan Mekki. 2004. The Penn Arabic Treebank: Building a Large-Scale Annotated Arabic Corpus. In *NEMLAR Conference on Arabic Language Resources and Tools*.

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

Marie-Catherine De Marneffe, Bill Maccartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proc of LREC*.

Thomas Müller, Helmut Schmid, and Hinrich Schütze. 2013. Efficient higher-order CRFs for morphological tagging. In *Proc of EMNLP*.

Joakim Nivre, Johan Hall, Jens Nilsson, Gülsen Eryiğit, and Svetoslav Marinov. 2006a. Labeled pseudo-projective dependency parsing with support vector machines. In *Proc of CoNLL*.

Joakim Nivre, Jens Nilsson, and Johan Hall. 2006b. Talbanken05: A Swedish treebank with phrase structure and dependency annotation. In *Proc of LREC*, Genoa, Italy.

Joakim Nivre. 2004. Incrementality in deterministic dependency parsing. In *Proc of the Workshop on Incremental Parsing: Bringing Engineering and Cognition Together*.

Joakim Nivre. 2009. Non-projective dependency parsing in expected linear time. In *Proc of ACL*.

Kemal Oflazer, Bilge Say, Dilek Zeynep Hakkani-Tür, and Gökhan Tür. 2003. Building a Turkish treebank. In *Treebanks*, pages 261–277. Springer.

Ferdinand Saussure. 1916. Nature of the linguistic sign. In *Course in General Linguistics*.

Djamé Seddah and Reut Tsarfaty. 2014. Introducing the SPMRL 2014 shared task on parsing morphologically-rich languages. *SPMRL-SANCL 2014*.

Djamé Seddah, Reut Tsarfaty, Sandra Kübler, Marie Candito, Jinho D. Choi, Richárd Farkas, Jennifer Foster, Iakes Goenaga, Koldo Gojenola Galletebeitia, Yoav Goldberg, Spence Green, Nizar Habash, Marco Kuhlmann, Wolfgang Maier, Joakim Nivre, Adam Przepiórkowski, Ryan Roth, Wolfgang Seeker, Yannick Versley, Veronika Vincze, Marcin Woliński, Alina Wróblewska, and Eric Villemonte de la Clergerie. 2013. Overview of the SPMRL 2013 shared task: cross-framework evaluation of parsing morphologically rich languages. In *SPMRL-EMNLP 2013*.

Wolfgang Seeker and Jonas Kuhn. 2012. Making Ellipses Explicit in Dependency Conversion for a German Treebank. In *Proc of LREC*.

Khalil Sima'an, Alon Itai, Yoad Winter, Alon Altman, and Noa Nativ. 2001. Building a Tree-Bank for Modern Hebrew Text. In *Traitement Automatique des Langues*.

Richard Socher, Eric H. Huang, Jeffrey Pennington, Andrew Y. Ng, and Christopher D. Manning. 2011. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Proc of NIPS*.

Richard Socher, Andrej Karpathy, Quoc V. Le, Christopher D. Manning, and Andrew Y. Ng. 2013a. Grounded compositional semantics for finding and describing images with sentences. *TACL*.

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

Pontus Stenetorp. 2013. Transition-based dependency parsing using recursive neural networks. In *Proc of NIPS Deep Learning Workshop*.

Marek Świdziński and Marcin Woliński. 2010. Towards a bank of constituent parse trees for Polish. In *Proc of TSD*.

Ivan. Titov and James. Henderson. 2007. A latent variable model for generative dependency parsing. In *Proc of IWPT*.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proc of NAACL*.

Reut Tsarfaty. 2006. Integrated morphological and syntactic disambiguation for Modern Hebrew. In *Proc of ACL Student Research Workshop*.

Veronika Vincze, Dóra Szauter, Attila Almási, György Móra, Zoltán Alexin, and János Csirik. 2010. Hungarian dependency treebank. In *Proc of LREC*.

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

Yue Zhang and Stephen Clark. 2008a. Joint word segmentation and POS tagging using a single perceptron. In *Proc of ACL*.

Yue Zhang and Stephen Clark. 2008b. A tale of two parsers: Investigating and combining graph-based and transition-based dependency parsing. In *Proc of EMNLP*.

Meishan Zhang, Yue Zhang, Wanxiang Che, and Ting Liu. 2013. Chinese parsing exploiting characters. In *Proc of ACL*.

Hao Zhou, Yue Zhang, Shujian Huang, and Jiajun Chen. 2015. A Neural Probabilistic Structured-Prediction Model for Transition-Based Dependency Parsing. In *Proc of ACL*.

Sentence Compression by Deletion with LSTMs

Katja Filippova, Enrique Alfonseca, Carlos A. Colmenares, Lukasz Kaiser, Oriol Vinyals
Google Research
{katjaf,ealfonseca,crcarlos,lukaszkaiser,vinyals}@google.com

Abstract

We present an LSTM approach to deletion-based sentence compression where the task is to translate a sentence into a sequence of zeros and ones, corresponding to token deletion decisions. We demonstrate that even the most basic version of the system, which is given no syntactic information (no PoS or NE tags, or dependencies) or desired compression length, performs surprisingly well: around 30% of the compressions from a large test set could be regenerated. We compare the LSTM system with a competitive baseline which is trained on the same amount of data but is additionally provided with all kinds of linguistic features. In an experiment with human raters the LSTM-based model outperforms the baseline achieving 4.5 in readability and 3.8 in informativeness.

1 Introduction

Sentence compression is a standard NLP task where the goal is to generate a shorter paraphrase of a sentence. Dozens of systems have been introduced in the past two decades and most of them are deletion-based: generated compressions are token subsequences of the input sentences (Jing, 2000; Knight & Marcu, 2000; McDonald, 2006; Clarke & Lapata, 2008; Berg-Kirkpatrick et al., 2011, to name a few).

Existing compression systems heavily use syntactic information to minimize chances of introducing grammatical mistakes in the output. A common approach is to use only some syntactic information (Jing, 2000; Clarke & Lapata, 2008, among others) or use syntactic features as signals in a statistical model (McDonald, 2006). It is probably even more common to operate on syntactic trees directly (dependency or constituency) and generate compressions by pruning them (Knight & Marcu, 2000; Berg-Kirkpatrick et al., 2011; Filippova & Altun, 2013, among others). Unfortunately, this makes such systems vulnerable to error propagation as there is no way to recover from an incorrect parse tree. With the state-of-the-art parsing systems achieving about 91 points in labeled attachment accuracy (Zhang & McDonald, 2014), the problem is not a negligible one. To our knowledge, there is no competitive compression system so far which does not require any linguistic preprocessing but tokenization.

In this paper we research the following question: can a robust compression model be built which only uses tokens and has no access to syntactic or other linguistic information? While phenomena like long-distance relations may seem to make generation of grammatically correct compressions impossible, we are going to present an evidence to the contrary. In particular, we will present a model which benefits from the very recent advances in deep learning and uses word embeddings and Long Short Term Memory models (LSTMs) to output surprisingly readable and informative compressions. Trained on a corpus of less than two million automatically extracted parallel sentences and using a standard tool to obtain word embeddings, in its best and most simple configuration it achieves 4.5 points out of 5 in readability and 3.8 points in informativeness in an extensive evaluation with human judges. We believe that this is an important result as it may suggest a new direction for sentence compression research which is less tied to modeling linguistic

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 360–368,
Lisbon, Portugal, 17-21 September 2015.

structures, especially syntactic ones, than the compression work so far.

The paper is organized as follows: Section 3 presents a competitive baseline which implements the system of McDonald (2006) for large training sets. The LSTM model and its three configurations are introduced in Section 4. The evaluation set-up and a discussion on wins and losses with examples are presented in Section 5 which is followed by the conclusions.

2 Related Work

The problem formulation we adopt in this paper is very simple: for every token in the input sentence we ask whether it should be kept or dropped, which translates into a sequence labeling problem with just two labels: one and zero. The deletion approach is a standard one in compression research, although the problem is often formulated over the syntactic structure and not the raw token sequence. That is, one usually drops constituents or prunes dependency edges (Jing, 2000; Knight & Marcu, 2000; McDonald, 2006; Clarke & Lapata, 2008; Berg-Kirkpatrick et al., 2011; Filippova & Altun, 2013). Thus, the relation to existing compression work is that we also use the deletion approach.

Recent advances in machine learning made it possible to escape the typical paradigm of mapping a fixed dimensional input to a fixed dimensional output to mapping an input sequence onto an output sequence. Even though many of these models were proposed more than a decade ago, it is not until recently that they have empirically been shown to perform well. Indeed, core problems in natural language processing such as translation (Cho et al., 2014; Sutskever et al., 2014; Luong et al., 2014), parsing (Vinyals et al., 2014), image captioning (Vinyals et al., 2015; Xu et al., 2015), or learning to execute small programs (Zaremba & Sutskever, 2014) employed virtually the same principles—the use of Recurrent Neural Networks (RNNs). Thus, with regard to this line of research, our work comes closest to the recent machine translation work. An important difference is that we do not aim at building a model that generates compressions directly but rather a model which generates a sequence of deletion decisions.

A more complex translation model is also conceivable and may significantly advance work on compression by paraphrasing, of which there have not been many examples yet (Cohn & Lapata, 2008). However, in this paper our goal is to demonstrate that a simple but robust deletion-based system can be built without using any linguistic features other than token boundaries. We leave experiments with paraphrasing models to future work.

3 Baseline

We compare our model against the system of McDonald (2006) which also formulates sentence compression as a binary sequence labeling problem. In contrast to our proposal, it makes use of a large set of syntactic features which are treated as soft evidence. The presence or absence of these features is treated as signals which do not condition the output that the model can produce. Therefore the model is robust against noise present in the precomputed syntactic structures of the input sentences.

The system was implemented based on the description by McDonald (2006) with two changes which were necessary due to the large size of the training data set used for model fitting. The first change was related to the learning procedure and the second one to the family of features used.

Regarding the learning procedure, the original model uses a large-margin learning framework, namely MIRA (Crammer & Singer, 2003), but with some minor changes as presented by McDonald et al. (2005). In this set-up, online learning is performed, and at each step an optimization procedure is made where K constraints are included, which correspond to the top-K solutions for a given training observation. This optimization step is equivalent to a Quadratic Programming problem if $K > 1$, which is time-costly to solve, and therefore not adequate for the large amount of data we used for training the model. Furthermore, in his publication McDonald states clearly that different values of K did not actually have a major impact on the final performance of the model. Consequently, and for the sake of being able to successfully train the model with large-scale data, the learning procedure is implemented as a distributed structured perceptron with iterative parameter mixing (McDonald et al., 2010), where each shard is processed with MIRA and K is set to 1.

Setting $K = 1$ will only affect the weight update described on line 4 of Figure 3 of McDonald

(2006), which is now expressed as:

$$\boldsymbol{w}^{(i+1)} \leftarrow \boldsymbol{w}^{(i)} + \tau \times \boldsymbol{e}_{\boldsymbol{y}_t,\boldsymbol{y}'}$$
$$\text{where } \tau = \max\left(0, \frac{L(\boldsymbol{y}_t, \boldsymbol{y}') - \boldsymbol{w} \cdot \boldsymbol{e}_{\boldsymbol{y}_t,\boldsymbol{y}'}}{||\boldsymbol{e}_{\boldsymbol{y}_t,\boldsymbol{y}'}||^2}\right)$$
$$\boldsymbol{e}_{\boldsymbol{y}_t,\boldsymbol{y}'} = \boldsymbol{F}(\boldsymbol{x}_t, \boldsymbol{y}_t) - \boldsymbol{F}(\boldsymbol{x}_t, \boldsymbol{y}')$$
$$\boldsymbol{y}' = \text{best}(\boldsymbol{x}; \boldsymbol{w}^{(i)})$$
$$\boldsymbol{F}(\boldsymbol{x}, \boldsymbol{y}) = \sum_{j=2}^{|\boldsymbol{y}|} \boldsymbol{f}(\boldsymbol{x}, I(y_{j-1}), I(y_j))$$

The second change concerns the feature set used. While McDonald's original model contains deep syntactic features coming from both dependency and constituency parse trees, we use only dependency-based features. Additionally, and to better compare the baseline with the LSTM models, we have included as an optional feature a 256-dimension embedding-vector representation of each input word and its syntactic parent. The vectors are pre-trained using the Skipgram model[1] (Mikolov et al., 2013). Ultimately, our implementation of McDonald's model contained 463,614 individual features, summarized in three categories:

- **PoS features:** Joint PoS tags of selected tokens. Unigram, bigram and trigram PoS context of selected and dropped tokens. All the previous features conjoined with one indicating if the last two selected tokens are adjacent.
- **Deep syntactic features:** Dependency labels of taken and dropped tokens and their parent dependencies. Boolean features indicating syntactic relations between selected tokens (*i.e.*, siblings, parents, leaves, etc.). Dependency label of the least common ancestor in the dependency tree between a batch of dropped tokens. All the previous features conjoined with the PoS tag of the involved tokens.
- **Word features:** Boolean features indicating if a group of dropped nodes contain a complete or incomplete parenthesization. Word-embedding vectors of selected and dropped tokens and their syntactic parents.

The model is fitted over ten epochs on the whole training data, and for model selection a small development set consisting of 5,000 previously unseen sentences is used (none of them belonging to the evaluation set). The automated metric used for this selection was accuracy@1 which is the proportion of golden compressions which could be fully reproduced. The performance on the development set plateaus when getting close to the last epoch.

[1] https://code.google.com/p/word2vec/

4 The LSTM model

Our approach is largely based on the sequence to sequence paradigm proposed in Sutskever et al. (2014). We train a model that maximizes the probability of the correct output given the input sentence. Concretely, for each training pair (X, Y), we will learn a parametric model (with parameters θ), by solving the following optimization problem:

$$\theta^* = \arg\max_{\theta} \sum_{X,Y} \log p(Y|X;\theta) \quad (1)$$

where the sum is assumed to be over all training examples. To model the probability p, we use the same architecture described by Sutskever et al. (2014). In particular, we use a RNN based on the Long Short Term Memory (LSTM) unit (Hochreiter & Schmidhuber, 1997), designed to avoid vanishing gradients and to remember some long-distance dependences from the input sequence. Figure 1 shows a basic LSTM architecture. The RNN is fed with input words X_i (one at a time), until we feed a special symbol "GO". It is now a common practice (Sutskever et al., 2014; Li & Jurafsky, 2015) to start feeding the input in reversed order, as it has been shown to perform better empirically. During the first pass over the input, the network is expected to learn a compact, distributed representation of the input sentence, which will allow it to start generating the right predictions when the second pass starts, after the "GO" symbol is read.

We can apply the chain rule to decompose Equation (1) as follows:

$$p(Y|X;\theta) = \prod_{t=1}^{T} p(Y_t|Y_1, \ldots, Y_{t-1}, X; \theta) \quad (2)$$

noting that we made no independence assumptions. Once we find the optimal θ^*, we construct our estimated compression $\hat{Y}$ as:

$$\hat{Y} = \arg\max_{Y} p(Y|X;\theta^*) \quad (3)$$

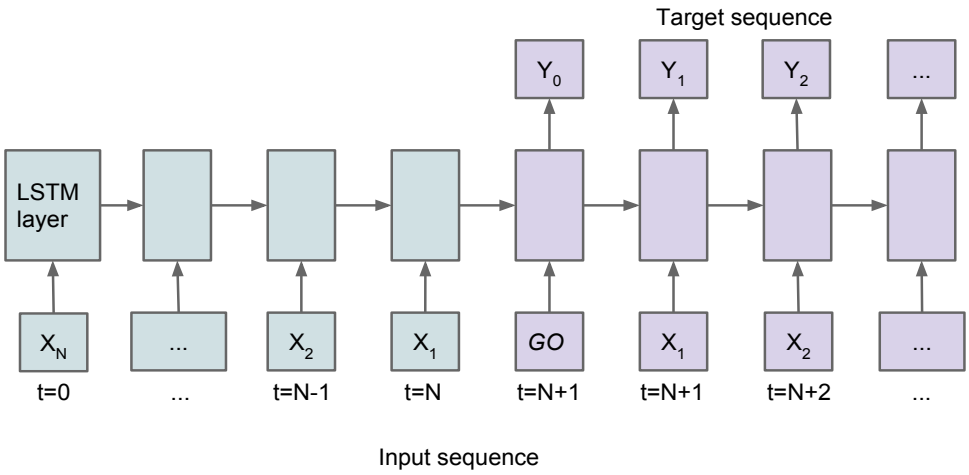


Figure 1: High-level overview of an LSTM unrolled through time.

LSTM cell: Let us review the sequence-to-sequence LSTM model. The Long Short Term Memory model of Hochreiter & Schmidhuber (1997) is defined as follows. Let x_t, h_t, and m_t be the input, control state, and memory state at timestep t. Then, given a sequence of inputs $(x_1, \ldots, x_T)$, the LSTM computes the h-sequence $(h_1, \ldots, h_T)$ and the m-sequence $(m_1, \ldots, m_T)$ as follows

$$\begin{aligned} i_t &= \text{sigm}(W_1 x_t + W_2 h_{t-1}) \\ i'_t &= \tanh(W_3 x_t + W_4 h_{t-1}) \\ f_t &= \text{sigm}(W_5 x_t + W_6 h_{t-1}) \\ o_t &= \text{sigm}(W_7 x_t + W_8 h_{t-1}) \\ m_t &= m_{t-1} \odot f_t + i_t \odot i'_t \\ h_t &= m_t \odot o_t \end{aligned}$$

The operator $\odot$ denotes element-wise multiplication, the matrices $W_1, \ldots, W_8$ and the vector h_0 are the parameters of the model, and all the non-linearities are computed element-wise.

Stochastic gradient descent is used to maximize the training objective (Eq. (1)) w.r.t. all the LSTM parameters.

Network architecture: In these experiments we have used the architecture depicted in Figure 3. Following Vinyals et al. (2014), we have used three stacked LSTM layers to allow the upper layers to learn higher-order representations of the input, interleaved with dropout layers to prevent overfitting (Srivastava et al., 2014). The output layer is a SoftMax classifier that predicts, after the "GO" symbol is read, one of the following three

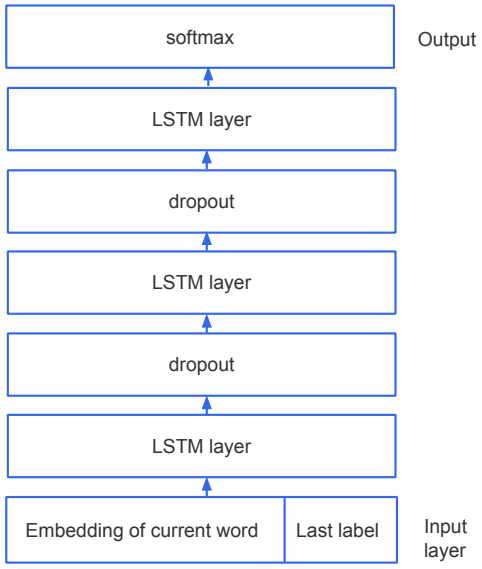


Figure 3: Architecture of the network used for sentence compression. Note that this basic structure is then unrolled 120 times, with the standard dependences from LSTM networks (Hochreiter & Schmidhuber, 1997).

labels: **1**, if a word is to be retained in the compression, **0** if a word is to be deleted, or **EOS**, which is the output label used for the "GO" input and the end-of-sentence final period.

Input representation: In the simplest implementation, that we call LSTM, the input layer has 259 dimensions. The first 256 contain the embedding-vector representation of the current in-

```
function DECODE(X)
                                          ▷ Initialize and feed the reversed input.
    Lstm ← CREATELSTM
    LayersState ← INITIALIZELAYERS(Lstm)
    for all X_i ∈ REVERSE(X) do
        LayersState ← ONESTEP(Lstm, LayersState, X_i)
    end for
    LayersState ← ONESTEP(Lstm, LayersState, GO)
      ▷ Create the beam vector. Each item contains the state of the layers, the labels predicted so far, and probability.
    Beam ← {(LayersState, (), 1.0)}
                                                          ▷ Beam search
    for all X_i ∈ X do
        NextBeam ← {}
        for all (LayersState, Labels, Prob) ∈ Beam do
            (NextLayersState, Outputs) ← ONESTEP(Lstm, LayersState, X_i)
            for all Output ∈ Outputs do
                NextBeam ← NextBeam∪{(NextLayerState, Labels+Output.label, Prob∗Output.prob)}
            end for
        end for
        Beam ← TOPN(NextBeam)
    end for
    return TOP(Beam)
end function
```

Figure 2: Pseudocode of the beam-search algorithm for compressing an input sentence.

put word, pre-trained using the Skipgram model[2] (Mikolov et al., 2013). The final three dimensions contain a one-hot-spot representation of the gold-standard label of the previous word (during training), or the generated label of the previous word (during decoding).

For the LSTM+PAR architecture we first parse the input sentence, and then we provide as input, for each input word, the embedding-vector representation of that word and its parent word in the dependency tree. If the current input is the root node, then a special parent embedding is constructed with all nodes set to zero except for one node. In these settings we want to test the hypothesis whether knowledge about the parent node can be useful to decide if the current constituent is relevant or not for the compression. The dimensionality of the input layer in this case is 515. Similarly to McDonald (2006), syntax is used here as a soft feature in the model.

For the LSTM+PAR+PRES architecture, we again parse the input sentence, and use a 518-sized embedding vector, that includes:

- The embedding vector for the current word (256 dimensions).
- The embedding vector for the parent word (256 dimensions).
- The label predicted for the last word (3 dimensions).
- A bit indicating whether the parent word has already been seen and kept in the compression (1 dimension).
- A bit indicating whether the parent word has already been seen but discarded (1 dimension).
- A bit indicating whether the parent word comes later in the input (1 dimension).

Decoding: Eq. (3) involves searching through all possible output sequences (given X). Contrary to the baseline, in the case of LSTMs the complete previous history is taken into account for each prediction and we cannot simplify Eq. (2) with a Markov assumption. Therefore, the search space at decoding time is exponential on the length of the input, and we have used a beam-search procedure as described in Figure 2.

Fixed parameters: For training, we unfold the network 120 times and make sure that none of our training instances is longer than that. The learning rate is initialized at 2, with a decay factor of 0.96 every 300,000 traning steps. The dropping probability for the dropout layers is 0.2. The number of nodes in each LSTM layer is always identical to the number of nodes in the input layer. We have not tuned these parameters nor the number of stacked layers.

[2] https://code.google.com/p/word2vec/

5 Evaluation

5.1 Data

Both the LSTM systems we introduced and the baseline require a training set of a considerable size. In particular, the LSTM model uses 256-dimensional embeddings of token sequences and cannot be expected to perform well if trained on a thousand parallel sentences, which is the size of the commonly used data sets (Knight & Marcu, 2000; Clarke & Lapata, 2006). Following the method of Filippova & Altun (2013), we collect a much larger corpus of about two million parallel sentence-compression instances from the news where every compression is a subsequence of tokens from the input. For testing, we use the publicly released set of 10,000 sentence-compression pairs[3]. We take the first 200 sentences from this set for the manual evaluation with human raters, and the first 1,000 sentences for the automatic evaluation.

5.2 Experiments

We evaluate the baseline and our systems on the 200-sentence test set in an experiment with human raters. The raters were asked to rate *readability* and *informativeness* of compressions given the input which are the standard evaluation metrics for compression. The former covers the grammatical correctness, comprehensibility and fluency of the output while the latter measures the amount of important content preserved in the compression.

Additionally, for experiments on the development set, we used two metrics for automatic evaluation: *per-sentence accuracy* (i.e., how many compressions could be fully reproduced) and *word-based F1-score*. The latter differs from the RASP-based relation F-score by Riezler et al. (2003) in that we simply compute the recall and precision in terms of tokens kept in the golden and the generated compressions. We report these results for completeness although it is the results of the human evaluation from which we draw our conclusions.

Compression ratio: The three versions of our system (LSTM*) and the baseline (MIRA) have comparable compression ratios (CR) which are defined as the length of the compression in characters divided over the sentence length. Since the ratios are very close, a comparison of the systems' scores is justified (Napoles et al., 2011).

Automatic evaluation: A total of 1,000 sentence pairs from the test set[4] were used in the automatic evaluation. The results are summarized in Table 1.

	F1	*Acc*	*CR*
MIRA	0.75	0.21	0.37
LSTM	0.80	0.30	0.39
LSTM+PAR	0.81	0.31	0.38
LSTM+PAR+PRES	0.82	0.34	0.38

Table 1: F1-score, per-sentence accuracy and compression ratio for the baseline and the systems

There is a significant difference in performance of the MIRA baseline and the LSTM models, both in terms of F1-score and in accuracy. More than 30% of golden compressions could be fully regenerated by the LSTM systems which is in sharp contrast with the 20% of MIRA. The differences in F-score between the three versions of LSTM are not significant, all scores are close to 0.81.

Evaluation with humans: The first 200 sentences from the set of 1,000 used in the automatic evaluation were compressed by each of the four systems. Every sentence-compression pair was rated by three raters who were asked to select a rating on a five-point Likert scale, ranging from one to five. In very few cases (around 1%) the ratings were inconclusive (i.e., *1, 3, 5* were given to the same pair) and had to be skipped. Table 2 summarizes the results.

	read	*info*
MIRA	4.31	3.55
LSTM	4.51†	3.78†
LSTM+PAR	4.40	3.73
LSTM+PAR+PRES	4.37	3.79†

Table 2: Readability and informativeness for the baseline and the systems: † stands for *significantly better than MIRA with 0.95 confidence*.

The results indicate that the LSTM models produce more readable and more informative compressions. Interestingly, there is no benefit in using the syntactic information, at least not with

[3] http://storage.googleapis.com/sentencecomp/compressiondata.json

[4] We used the very first 1,000 instances.

Sentence & LSTM *Compression*	difficulty
A Virginia state senator and one-time candidate for governor stabbed by his son said Friday that he is "alive so must live," his first public statement since the assault and his son's suicide shortly thereafter. *State senator alive so must live*.	quotes
Gwyneth Paltrow, 41 and husband Chris Martin, 37 are to separate after more than 10 years of marriage, the actress announced on her website GOOP. *Gwyneth Paltrow are to separate*.	commas
Chris Hemsworth and the crew of his new movie 'In the Heart of the Sea' were forced to flee flash floods in the Canary Islands yesterday. *Chris Hemsworth were forced to flee flash floods*.	quotes
Police in Deltona, Fla., are trying to sniff out the identity of a man who allegedly attempted to pay his water bill with cocaine. *Police are trying to sniff out the identity.*	nothing to remove
Just a week after a CISF trooper foiled a suicide bid by a woman in the Delhi metro, another woman trooper from the same force prevented two women commuters from ending their lives, an official said Monday. *Another woman trooper prevented two women commuters.*	important context
Whatever the crisis or embarrassment to his administration, Pres. Obama don't know nuttin' about it. *Pres. Obama don't know nuttin.*	nothing to remove
TRADE and Industry Minister Rob Davies defended the government's economic record in Parliament on Tuesday, saying it had implemented structural reforms and countercyclical infrastructure projects to help shore up the economy. *Rob Davies defended the government's economic record.*	
Social activist Medha Patkar on Monday extended her "complete" support to Arvind Kejriwal-led Aam Aadmi Party in Maharashtra. *Medha Patkar extended her support to Aam Aadmi Party.*	
State Sen. Stewart Greenleaf discusses his proposed human trafficking bill at Calvery Baptist Church in Willow Grove Thursday night. *Stewart Greenleaf discusses his human trafficking bill.*	
Alan Turing, known as the father of computer science, the codebreaker that helped win World War 2, and the man tortured by the state for being gay, is to receive a pardon nearly 60 years after his death. *Alan Turing is to receive a pardon.*	
Robert Levinson, an American who disappeared in Iran in 2007, was in the country working for the CIA, according to a report from the Associated Press's Matt Apuzzo and Adam Goldman. *Robert Levinson was working for the CIA.*	

Figure 4: Example sentences and compressions.

the amount of parallel data we had at our disposal. The simple LSTM model which only uses token embeddings to generate a sequence of deletion decisions significantly outperforms the baseline which was given not only embeddings but also syntactic and other features.

Discussion: What are the wins and losses of the LSTM systems? Figure 4 presents some of the evaluated sentence-compression pairs. In terms of readability, the basic LSTM system performed surprisingly well. Only in a few cases (out of 200) did it get an average score of two or three. Sentences which pose difficulty to the model are the ones with quotes, intervening commas, or other uncommon punctuation patterns. For example, in the second sentence in Figure 4, if one removes from the input the age modifiers and the preceding commas, the words *and Chris Martin* are not dropped and the output compression is grammatical, preserving both conjoined elements.

With regard to informativeness, the difficult cases are those where there is very little to be removed and where the model still removed more than a half to achieve the compression ratio it observed in the training data. For example, the only part that can be removed from the fourth sentence in Figure 4 is the modifier of *police*, everything else being important content. Similarly, in the fifth sentence the context of the event must be retained in the compression for the event to be interpreted correctly.

Arguably, such cases would also be difficult for other systems. In particular, recognizing when the context is crucial is a problem that can be solved only by including deep semantic and discourse features which has not been attempted yet. And

sentences with quotes (direct speech, a song or a book title, etc.) are challenging for parsers which in turn provide important signals for most compression systems.

The bottom of Figure 4 contains examples of good compressions. Even though for a significant number of input sentences the compression was a continuous subsequence of tokens, there are many discontinuous compressions. In particular, the LSTM model learned to drop appositions, no matter how long they are, temporal expressions, optional modifiers, introductory clauses, etc.

Our understanding of why the extended model (LSTM+PAR+PRES) performed worse in the human evlauation than the base model is that, in the absence of syntactic features, the basic LSTM learned a model of syntax useful for compression, while LSTM++, which was given syntactic information, learned to optimize for the particular way the "golden" set was created (tree pruning). While the automatic evaluation penalized all deviations from the single golden variant, in human evals there was no penalty for readable alternatives.

6 Conclusions

We presented, to our knowledge, a first attempt at building a competitive compression system which is given no linguistic features from the input. The two important components of the system are (1) word embeddings, which can be obtained by anyone either pre-trained, or by running `word2vec` on a large corpus, and (2) an LSTM model which draws on the very recent advances in research on RNNs. The training data of about two million sentence-compression pairs was collected automatically from the Internet.

Our results clearly indicate that a compression model which is not given syntactic information explicitly in the form of features may still achieve competitive performance. The high readability and informativeness scores assigned by human raters support this claim. In the future, we are planning to experiment with more "interesting" paraphrasing models which translate the input not into a zero-one sequence but into words.

References

Berg-Kirkpatrick, T., D. Gillick & D. Klein (2011). Jointly learning to extract and compress. In *Proc. of ACL-11*.

Cho, K., B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk & Y. Bengio (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing,* Doha, Qatar, 25–29 October 2014.

Clarke, J. & M. Lapata (2006). Models for sentence compression: A comparison across domains, training requirements and evaluation measures. In *Proc. of COLING-ACL-06*, pp. 377–385.

Clarke, J. & M. Lapata (2008). Global inference for sentence compression: An integer linear programming approach. *Journal of Artificial Intelligence Research*, 31:399–429.

Cohn, T. & M. Lapata (2008). Sentence compression beyond word deletion. In *Proc. of COLING-08*, pp. 137–144.

Crammer, K. & Y. Singer (2003). Ultraconservative online algorithms for multiclass problems. *Journal of Machine Learning Research*, 3:951–991.

Filippova, K. & Y. Altun (2013). Overcoming the lack of parallel data in sentence compression. In *Proc. of EMNLP-13*, pp. 1481–1491.

Hochreiter, S. & J. Schmidhuber (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Jing, H. (2000). Sentence reduction for automatic text summarization. In *Proc. of ANLP-00*, pp. 310–315.

Knight, K. & D. Marcu (2000). Statistics-based summarization – step one: Sentence compression. In *Proc. of AAAI-00*, pp. 703–711.

Li, J. & D. Jurafsky (2015). A hierarchical LSTM autoencoder for paragraphs and documents. In *Proc. of ACL-15*.

Luong, M.-T., I. Sutskever, Q. V. Le, O. Vinyals & W. Zaremba (2014). Addressing the rare word problem in neural machine translation. *CoRR*, abs/1410.8206.

McDonald, R. (2006). Discriminative sentence compression with soft syntactic evidence. In *Proc. of EACL-06*, pp. 297–304.

McDonald, R., K. Crammer & F. Pereira (2005). Online large-margin training of dependency parsers. In *Proc. of ACL-05*, pp. 91–98.

McDonald, R., K. Hall & G. Mann (2010). Distributed training strategies for the structured perceptron. In *Proc. of NAACL-HLT-10*, pp. 456–464.

Mikolov, T., I. Sutskever, K. Chen, G. S. Corrado & J. Dean (2013). Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pp. 3111–3119.

Napoles, C., C. Callison-Burch & B. Van Durme (2011). Evaluating sentence compression: Pitfalls and suggested remedies. In *Proceedings of the Workshop on Monolingual Text-to-text Generation,* Portland, OR, June 24 2011, pp. 91–97.

Riezler, S., T. H. King, R. Crouch & A. Zaenen (2003). Statistical sentence condensation using ambiguity packing and stochastic disambiguation methods for Lexical-Functional Grammar. In *Proc. of HLT-NAACL-03*, pp. 118–125.

Srivastava, N., G. Hinton, A. Krizhevsky, I. Sutskever & R. Salakhutdinov (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

Sutskever, I., O. Vinyals & Q. V. Le (2014). Sequence to sequence learning with neural networks. In *Proc. of NIPS-2014*.

Vinyals, O., A. Toshev, S. Bengio & D. Erhan (2015). Show and tell: A neural image caption generator. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR-2015*.

Vinyals, O., L. Kaiser, T. Koo, S. Petrov, I. Sutskever & G. E. Hinton (2014). Grammar as a foreign language. *CoRR*, abs/1412.7449.

Xu, K., J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhutdinov, R. Zemel & Y. Bengio (2015). Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of ICML-2015*.

Zaremba, W. & I. Sutskever (2014). Learning to execute. *CoRR*, abs/1410.4615.

Zhang, H. & R. McDonald (2014). Enforcing structural diversity in cube-pruned dependency parsing. In *Proc. of ACL-14*, pp. 656–661.

An Empirical Comparison Between *N*-gram and Syntactic Language Models for Word Ordering

Jiangming Liu and **Yue Zhang**
Singapore University of Technology and Design,
8 Somapah Road, Singapore, 487372
{jiangming_liu, yue_zhang}@sutd.edu.sg

Abstract

Syntactic language models and *N*-gram language models have both been used in word ordering. In this paper, we give an empirical comparison between *N*-gram and syntactic language models on word order task. Our results show that the quality of automatically-parsed training data has a relatively small impact on syntactic models. Both of syntactic and *N*-gram models can benefit from large-scale raw text. Compared with *N*-gram models, syntactic models give overall better performance, but they require much more training time. In addition, the two models lead to different error distributions in word ordering. A combination of the two models integrates the advantages of each model, achieving the best result in a standard benchmark.

1 Introduction

N-gram language models have been used in a wide range of the generation tasks, such as machine translation (Koehn et al., 2003; Chiang, 2007; Galley et al., 2004), text summarization (Barzilay and McKeown, 2005) and realization (Guo et al., 2011). Such models are trained from large-scale raw text, capturing distributions of local word *N*-grams, which can be used to improve the fluency of synthesized text.

More recently, syntactic language models have been used as a complement or alternative to *N*-gram language models for machine translation (Charniak et al., 2003; Shen et al., 2008; Schwartz et al., 2011), syntactic analysis (Chen et al., 2012) and tree linearization (Song et al., 2014). Compared with *N*-gram models, syntactic models capture rich structural information, and can be more effective in improving the fluency of large constituents, long-range dependencies and overall sentential grammaticality. However, Syntactic models require annotated syntactic structures for training, which are expensive to obtain manually. In addition, they can be slower compared to *N*-gram models.

In this paper, we make an empirical comparison between syntactic and *N*-gram language models on the task of word ordering (Wan et al., 2009; Zhang and Clark, 2011a; De Gispert et al., 2014), which is to order a set of input words into a grammatical and fluent sentence. The task can be regarded as an abstract language modeling problem, although methods have been explored extending it for tree linearization (Zhang, 2013), broader text generation (Song et al., 2014) and machine translation (Zhang et al., 2014).

We choose the model of Liu et al.(2015) as the syntactic language model. There has been two main types of syntactic language models in the literature, the first being relatively more oriented to syntactic structure, without an explicit emphasis on word orders (Shen et al., 2008; Chen et al., 2012). As a result, this type of syntactic language models are typically used jointly with *N*-gram model for text-to-text tasks. The second type models syntactic structures incrementally, thereby can be used to directly score surface orders (Schwartz et al., 2011; Liu et al., 2015). We choose the discriminative model of Liu et al. (2015), which gives state-of-the-art results for word ordering.

We try to answer the following research questions by comparing the syntactic model and the *N*-gram model using the same search algorithm.

• **What is the influence of automatically-parsed training data on the performance of syntactic models.** Because manual syntactic annotations are relatively limited and highly expensive, it is necessary to use large-scale automatically-parsed sentences for training syntactic language models. As a result, the syntactic structures that a word ordering system learns can be inaccurate. However, this might not affect

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 369–378,
Lisbon, Portugal, 17-21 September 2015.

$$\text{Initial State} \quad ([], set(1...n), \emptyset)$$

$$\text{Final State} \quad ([], \emptyset, A)$$

Induction Rules:

$$\text{SHIFT} \quad \frac{(\sigma,\rho,A)}{([\sigma|i],\rho-\{i\},A)}$$

$$\text{L-ARC} \quad \frac{([\sigma|j\ i],\rho,A)}{([\sigma|i],\rho,A\cup\{j\leftarrow i\})}$$

$$\text{R-ARC} \quad \frac{([\sigma|j\ i],\rho,A)}{([\sigma|i],\rho,A\cup\{j\rightarrow i\})}$$

Figure 1: Deduction system for transition-based linearization.

step	action	σ	ρ	A
init		[]	(0 1 2)	$\emptyset$
0	shift	[1]	(0 2)	
1	shift	[1 2]	(0)	
2	L-arc	[2]	(0)	$A \cup \{1 \leftarrow 2\}$
3	shift	[2 0]	()	
4	R-arc	[2]	()	$A \cup \{2 \rightarrow 0\}$

Figure 2: Transition-based process for ordering {"*potatoes*$_0$", "*Tom*$_1$", "*likes*$_2$"}.

the quality of the synthesized output, which is a string only. We quantitatively study the influence of parsing accuracy of syntactic training data on word ordering output.

• **What is the influence of data scale on the performance.** *N*-gram language models can be trained efficiently over large numbers of raw sentences. In contrast, syntactic language models can be much slower to train due to rich features. We compare the output quality of the two models on different scales of training data, and also on different amounts of training time.

• **What are the errors characteristics of each model.** Syntactic language models can potentially be better in capturing larger constituents and overall sentence structures. However, compared with *N*-gram models, little work has been done to quantify the difference between the two models. We characterise the outputs using a set of different measures, and show empirically the relative strength and weakness of each model.

• **What is the effect of model combination.** Finally, because the two models make different types of errors, they can be combined to give better outputs. We develop a combined model by discretizing probability from *N*-gram model, and using them as features in the syntactic model. The combined model gives the best results in a standard benchmark.

2 Systems

2.1 Syntactic word ordering

Syntactic word ordering algorithms take a multi-set of input words constructing an output sentence and its syntactic derivation simultaneously. Transition-based syntactic word ordering can be modelled as an extension to transition-based parsing (Liu et al., 2015), with the main difference being that the order of words is not given in the input, which leads to a much larger search space.

We take the system of Liu, et al.[1], which gives state-of-the-art performance and efficiencies in standard word ordering benchmark. It maintains outputs in stack σ, and orders the unprocessed incoming words in a set ρ. Given an input bag of words, ρ is initialized to the input and σ is initialized as empty. The system repeatedly applies transition actions to consume words from ρ and construct output on σ.

Figure 1 shows the deduction system, where ρ is unordered and any word in ρ can be shifted onto the stack σ. The set of actions are SHIFT, L-ARC and R-ARC. The SHIFT actions add a word to the stack. For the L-ARC and R-ARC actions, new arcs $\{j \leftarrow i\}$ and $\{j \rightarrow i\}$ are constructed respectively. Under these possible actions, the unordered word set "*potatoes*$_0$ *Tom*$_1$ *likes*$_2$" is generated as shown in Figure 2, and the result is "*Tom*$_1$ ←*likes*$_2$→*potatoes*$_0$".

We apply the learning and search framework of Zhang and Clark (2011a). Pseudocode of the search algorithm is shown in Algorithm 1. [] refers to an empty stack, and $set(1...n)$ represents the full set of input words W and n is the number of distinct words. $candidates$ stores possible states, and $agenda$ stores temporary states transited from possible actions. GETACTIONS generates a set of possible actions depending on the current state s. APPLY generates a new state by applying action on the current state s. N-BEST produces the top k candidates in $agenda$. Finally, the algorithm returns the highest-score state $best$ in the $agenda$.

A global linear model is used to score search hypotheses. Given a hypothesis h, its score is calculated by:

$$Score(h) = \Phi(h) \cdot \vec{\theta},$$

[1] http://sourceforge.net/projects/zgen/

Algorithm 1 Transition-based linearisation

Input: *W*, a set of input word
Output: the highest-scored final state

$candidates \leftarrow ([], set(1..n), \emptyset)$
$agenda \leftarrow \emptyset$
$N \leftarrow 2n$
for $i \leftarrow 1..N$ **do**
  for s **in** $candidates$ **do**
    for $action$ **in** GETACTIONS(s) **do**
      $agenda \leftarrow$ APPLY($s, action$)
    end for
  end for
  $candidates \leftarrow$ N-BEST($agenda$)
  $agenda \leftarrow \emptyset$
end for
$best \leftarrow$ BEST($candidates$)
return $best$

where $\Phi(h)$ is the feature vector of h, extracted by using the same feature templates as Liu et al.(2015), which are shown in Table 1 and $\vec{\theta}$ is the parameter vector of the model. The feature templates essentially represents a syntactic language model. As shown in Figure 2, from the hypotheses produced in steps 2 and 4, the features "$Tom_1 \leftarrow likes_2$" and "$likes_2 \rightarrow potatoes_0$" are extracted, which corresponds to $P(Tom_1|likes_2)$ and $P(potatoes_0|likes_2)$ respectively in the dependency language model of Chen et al.,(2012).

Training. We apply perceptron with early-update (Collins and Roark, 2004), and iteratively tune related parameters on a set of development data. For each iteration, we measure the performance on the development data, and choose best parameters for final tests.

2.2 *N*-gram word ordering

We build an *N*-gram word ordering system under the same beam-search framework as the syntactic word ordering system. In particular, search is performed incrementally, from left to right, adding one word at each step. The decoding process can be regarded as a simplified version of Algorithm 1, with only SHIFT being returned by GETACTIONS, and the score of each transition is given by a standard *N*-gram language model. We use the same beam size for both *N*-gram and the syntactic word ordering. Compared with the syntactic model, the *N*-gram model has less information for disambiguation, but also has less structural ambiguities, and therefore a smaller search space.

Unigram
S_0w; S_0p; $S_{0,l}w$; $S_{0,l}p$; $S_{0,r}w$; $S_{0,r}p$;
$S_{0,l2}w$; $S_{0,l2}p$; $S_{0,r2}w$; $S_{0,r2}p$;
S_1w; S_1p; $S_{1,l}w$; $S_{1,l}p$; $S_{1,r}w$; $S_{1,r}p$;
$S_{1,l2}w$; $S_{1,l2}p$; $S_{1,r2}w$; $S_{1,r2}p$;
Bigram
$S_0wS_{0,l}w$; $S_0wS_{0,l}p$; $S_0pS_{0,l}w$; $S_0pS_{0,l}pS_{0,l}p$;
$S_0wS_{0,r}w$; $S_0wS_{0,r}p$; $S_0pS_{0,r}w$; $S_0pS_{0,r}pS_{0,r}p$;
$S_1wS_{1,l}w$; $S_1wS_{1,l}p$; $S_1pS_{1,l}w$; $S_1pS_{1,l}pS_{1,l}p$;
$S_1wS_{1,r}w$; $S_1wS_{1,r}p$; $S_1pS_{1,r}w$; $S_1pS_{1,r}pS_{1,r}p$;
S_0wS_1w; S_0wS_1p; S_0pS_1w; S_0pS_1p;
Trigram
$S_0wS_0pS_{0,l}w$; $S_0wS_{0,l}wS_{0,l}p$; $S_0wS_0pS_{0,l}p$;
$S_0pS_{0,l}wS_{0,l}p$; $S_0wS_0pS_{0,r}w$; $S_0wS_{0,l}wS_{0,r}p$;
$S_0wS_0pS_{0,r}p$; $S_0pS_{0,r}wS_{0,r}p$;
$S_1wS_1pS_{1,l}w$; $S_1wS_{1,l}wS_{1,l}p$; $S_1wS_1pS_{1,l}p$;
$S_1pS_{1,l}wS_{1,l}p$; $S_1wS_1pS_{1,r}w$; $S_1wS_{1,l}wS_{1,r}p$;
$S_1wS_1pS_{1,r}p$; $S_1pS_{1,r}wS_{1,r}p$;
Linearization
w_0; p_0; $w_{-1}w_0$; $p_{-1}p_0$; $w_{-2}w_{-1}w_0$; $p_{-2}p_{-1}p_0$;
$S_{0,l}S_{0,l2}w$; $S_{0,l}pS_{0,l2}p$; $S_{0,r2}wS_{0,r}w$; $S_{0,r2}pS_{0,r}p$;
$S_{1,l}S_{1,l2}w$; $S_{1,l}pS_{1,l2}p$; $S_{1,r2}wS_{1,r}w$; $S_{1,r2}pS_{1,r}p$;

Table 1: Feature templates.

name	domain	# of sents	# of tokens
training data			
AFP	News	35,390,025	844,395,322
XIN	News	18,095,371	401,769,616
WSJ	Finance	39,832	950,028
testing data			
WSJ	Finance	2,416	56,684
WPB	News	2,000	43,712
SANCL	Blog	1,015	20,356

Table 2: Data.

Training. We train *N*-gram language models from raw text using modified Kneser-Ney smoothing without pruning. The text is true-case tokenized, and we train 4-gram language modes using KenLM[2], which gives high efficiencies in standard *N*-gram language model construction.

3 Experimental settings

3.1 Data

For training data, we use the Wall Street Journal (WSJ) sections 1-22 of the Penn Treebank (Mar-

[2]https://kheafield.com/code/kenlm/

domain	sentence example
Finance	The $ 409 million bid includes the assumption of an estimated $ 300 million in secured liabilities on those properties , according to those making the bid.
News	But after rising steadily during the quarter-century following World War II , wages have stagnated since the manufacturing sector began to contract .
Blog	The freaky thing here is that these bozos are seriously claiming the moral high ground ?

Table 3: Domain examples.

cus et al., 1993), and the Agence France-Presse (AFP) and Xinhua News Agency (XIN) subsets of the English Giga Word Fifth Edition (Parker et al., 2011). As the development data, we use WSJ section 0 for parameter tuning. For testing, we use data from various domain, which consist of WSJ section 23, Washington Post/Bloomberg(WPB) subsets of the English Giga Word Fifth Edition and SANCL blog data, as shown in Table 2. Example sentence in various test domains are shown in Table 3.

3.2 Evaluation metrics

We follow previous work and use the BLEU metric (Papineni et al., 2002) for evaluation. Since BLEU only scores *N*-gram precisions, it can be in favour of *N*-gram language models. We additionally use METEOR[3](Denkowski and Lavie, 2010) to evaluate the system performances. The BLEU metric measures the fluency of generated sentence without considering long range ordering. The METEOR metric can potentially fix this problem using a set of mapping between generated sentences and references to evaluate distortion. The following example illustrates the difference between BLEU and METEOR on long range reordering, where the reference is

(1) *[The document is necessary for developer ,]$_0$ [so you can not follow this document to get right options .]$_1$*

and the generated output sentence is

(2) *[so you can not follow this document to get right options .]$_1$ [The document is necessary for developer ,]$_0$.*

There is a big distortion in the output. The BLEU metric gives a score of 90.09 out of 100, while the METEOR gives a score of 61.34 out of 100. This is because that METEOR is based on explicit word-to-word matches over the whole sentence. For word ordering, word-to-word matches are unique, which facilitates METEOR evaluation between generated sentences and references. As can bee seen from the example, long range distortion can highly influence the METEOR scores making the METEOR metric more suitable for evaluating word ordering distortions.

[3]http://www.cs.cmu.edu/∼alavie/METEOR/

ID	# training sent	# iter	Avg F1
set57	900	1	57.31
set66	1800	1	66.82
set78	9000	1	78.73
set83	all	1	83.93
set88	all	30	88.10

Table 4: Parsing accuracy settings.

3.3 Data preparation

For all the experiments, we assume that the input is a bag of words without order, and the output is a fully ordered sentence. Following previous work (Wan et al., 2009; Zhang, 2013; Liu et al., 2015), we treat base noun phrases (i.e. noun phrases do not contains other noun phrases, such as '*Pierre Vinken*' and '*a big cat*') as a single word. This avoids unnecessary ambiguities in combination between their subcomponents.

The syntactic model requires that the training sentences have syntactic dependency structure. However, only the WSJ data contains gold-standard annotations. In order to obtain automatically annotated dependency trees, we train a constituent parser using the gold-standard bracketed sentences from WSJ, and automatically parse the Giga Word data. The results are turned into dependency trees using Penn2Malt[4], after base noun phrases are extracted. In our experiments, we use ZPar[5] (Zhu et al., 2013) for automatic constituent parsing.

In order to study the influence of parsing accuracy of the training data, we also use ten-fold jackknifing to construct WSJ training data with different accuracies. The data is randomly split into ten equal-size subsets, and each subset is automatically parsed with a parser trained on the other

[4]http://stp.lingfil.uu.se/∼nivre/research/Penn2Malt.html

[5]http://people.sutd.edu.sg/∼yue_zhang/doc/doc/conparser.html

	in-domain on WSJ test		cross-domain on WPB test		cross-domain on SANCL test	
	BLEU (%)	METEOR (%)	BLEU (%)	METEOR (%)	BLEU (%)	METEOR (%)
syntax-set57	48.76	48.98	37.31	46.78	37.60	46.79
syntax-set66	48.79	48.98	37.52	46.81	38.28	46.90
syntax-set78	49.27	49.08	38.10	46.89	38.76	46.96
syntax-set83	49.74	49.16	37.68	46.84	38.67	46.93
syntax-set88	49.73	49.17	38.27	46.92	38.52	46.93
syntax-gold	50.82	49.33	37.76	46.84	39.97	47.26

Table 5: Influence result of parsing accuracy.

nine subset. In order to obtain datasets with different parsing accuracies, we randomly sample a small number of sentences from each training subset, as shown in Table 4. The dependency trees of each set are derived from these bracketed sentences using Penn2Malt after base noun phrase are extracted as a single word.

4 Influence of parsing accuracy

4.1 In-domain word ordering

We train the syntactic models on the WSJ training parsing data with different accuracies. The WSJ development data are used to find out the optimal number of training iterations for each experiments, and the WSJ test results are shown in Table 5.

Table 5 shows that the parsing accuracy can affect the performance of the syntactic model. A higher parsing accuracy can lead to a better syntactic language model. It conforms to the intuition that syntactic quality affects the fluency of surface texts. On the other hand, the influence is not huge, the BLEU scores decrease by 1.0 points as the parsing accuracy decreases from 88.10% to 57.31%

4.2 Cross-domain word ordering

The influence of parsing accuracy of the training data on cross-domain word ordering is measured by using the same training settings, but testing on the WPB and SANCL test sets. Table 5 shows that the performance on cross-domain word ordering cannot reach that of in-domain word ordering using the syntactic models. Compared with the cross-domain experiments, the influence of parsing accuracy becomes smaller. In the WPB test, the fluctuation of performance decline to about 0.9 BLEU points, and in the SANCL test, the fluctuation is about 1.1 BLEU points.

In conclusion, the experiments show that parsing accuracies have a relatively small influence on the syntactic models. This suggests that it is possible to use large automatically-parsed data to train syntactic models. On the other hand, when the training data scale increases, syntactic models can become much slower to train compared with *N*-gram models. The influence on data scale, which includes output quality and training time, is further studied in the next section.

5 Influence of data scale

We use the AFP news data as the training data for the experiments of this section. The syntactic models are trained using automatically-parsed trees derived from ZPar, as described in Section 3.3. The WPB test data is used to measure in-domain performance, and the SANCL blog data is used to measure cross-domain performance.

5.1 Influence on BLEU and METEOR

The Figure 3 and 4 shows that using both the BLEU and the METEOR metrics, the performance of the syntactic model is better than that of the *N*-gram models. It suggests that sentences generated by the syntactic model have both better fluency and better ordering. The performance of the syntactic models is not highly weakened in cross-domain tests.

The grey dot in each figure shows the performance of the syntactic model trained on the gold WSJ training data, and evaluated on the same WPB and SANCL test data sets. A comparison between the grey dots and the dashed lines shows that the syntactic model trained on the WSJ data perform better than the syntactic model trained on similar amounts of AFP data. This again shows the effect of syntactic quality of the training data.

On the other hand, as the scale of automatically-parsed AFP data increases, the performance of the

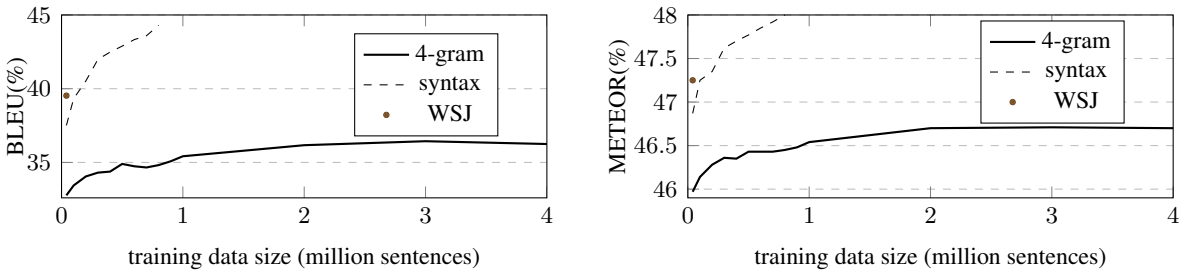


Figure 3: In-domain results on different training data size.

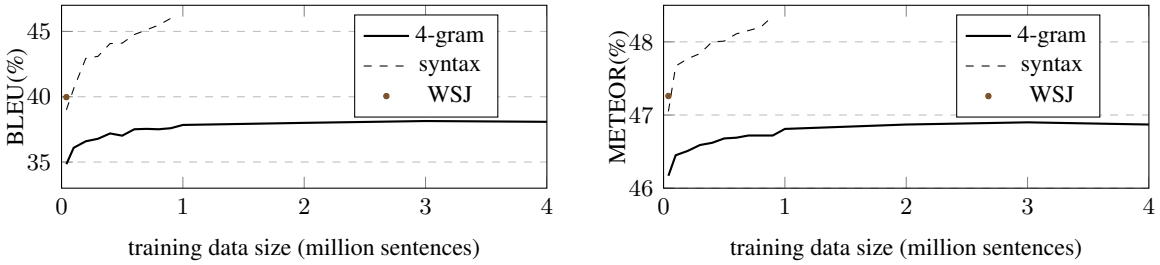


Figure 4: Cross-domain results on different training data sizes.

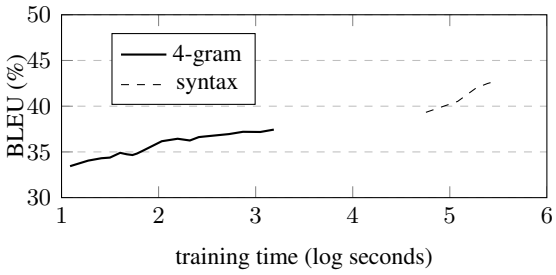


Figure 5: BLEU on different training times.

syntactic model rapidly increases, surpassing the syntactic model trained on the high-quality WSJ data. This observation is important, showing that large-scale data can be used to alleviate the problem of lower syntactic quality in automatically-parsed data, which can be leveraged to address the scarcity issue of manually annotated data in both in-domain and cross-domain settings.

5.2 Influence on training time

The training time of both syntactic models and *N*-gram models increases as the size of training data increases. Figure 5 shows the BLEU of the two systems under different amounts of training time. There is no result reported for the syntactic model beyond 1 million training sentences, because training becomes infeasibly slow [6]. On the other hand, the *N*-gram model can be trained using all the WSJ, AFP, XIN training sentences, which are 53 millions, within $10^{3.2}$ seconds. As a result, there is no overlap between the syntactic model and the *N*-gram model curves.

As can be seen from the figure, the syntactic model is much slower to train. However, it benefits more from the scale of the training data, with the slope of the dashed curve being steeper than that of the solid curve. The *N*-gram model can be trained with more data thanks to the fast training speed. However, the performance of the *N*-gram model flattens when the training data size reaches beyond 3 million. Projection of the solid curve suggests that the performance of the *N*-gram model may not surpass that of the syntactic model even if sufficiently large data is available for training the *N*-gram model in more time.

6 Error analysis

Although giving overall better performance, the syntactic model does not perform better than the *N*-gram model in all cases. Here we analyze the strength of each model via more fine-grained comparison.

In this set of experiments, the syntactic model is trained using gold-standard annotated WSJ training parse trees, and the *N*-gram model is trained using the data containing WSJ training data, AFP and XIN. The WSJ test data, which contains

[6] Our experiments are carried on a single thread of 3.60GHz CPU. If the training time is over 90 hours for a model, we consider it infeasible.

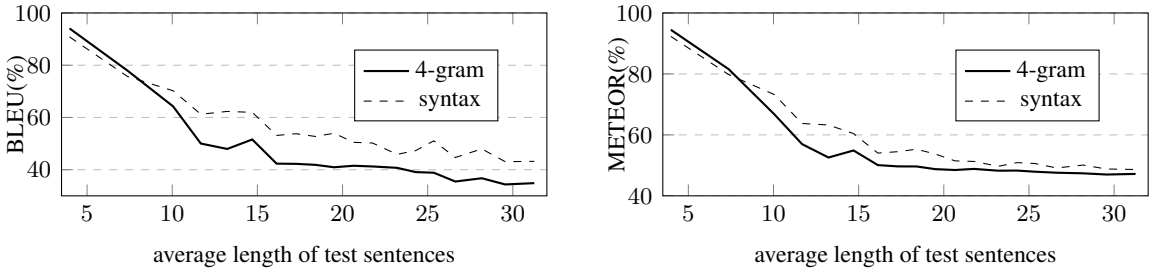


Figure 6: Performance on sentences with different length.

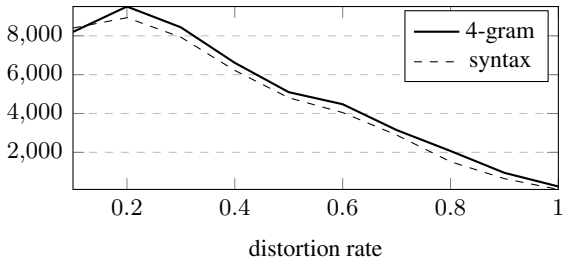


Figure 7: The distribution of distortion.

Template	distribution
NLM-LOW	set 1 if $p < e^{-12.5}$, else 0
NLM-20	use 20 bins to scatter probability
NLM-10	use 10 bins to scatter probability
NLM-5	use 5 bins to scatter probability
NLM-2	use 2 bins to scatter probability

Table 6: NLM feature templates.

golden constituent trees, is used to analyze errors in different aspects.

6.1 Sentence length

The BLEU and METEOR scores of the two systems on various sentence lengths are shown in Figure 6. The results are measured by binning sentences according to their lengths, so that each bin contains about the same number of sentences. As shown by the figure, the *N*-gram model performs better on short sentences (less than 8 tokens), and the syntactic model performs better on longer sentences. This can be explained by the fact that longer sentences have richer underlying syntactic structures, which can better captured by the syntactic model. In contrast, for shorter sentences, the syntactic structure is relatively simple, and therefore the *N*-gram model can give better performance based on string patterns, which form smaller search spaces.

6.2 Distortion range

We measure the average distortion rate of output word w using the following metric:

$$distortion(w) = \frac{|i_w - i'_w|}{len(S_w)},$$

where i_w is index of word w in the output sentence S_w, i'_w is the index of the word w in the reference sentence. $len(S_w)$ is the number of tokens in sentence S_w. Figure 7 shows distributions of distortion respectively by the syntactic and *N*-gram model. The *N*-gram model makes relatively fewer short-range distortions, but more long-range distortions. This can be explained by the local scoring nature of the *N*-gram model. In contrast, the syntactic model makes less long-range distortions, which can suggest better sentence structure.

6.3 Constituent span

We further evaluate sentence structure correctness by evaluating the recalls of discovered constituent span in output two systems, respectively. As shown in Figure 8. The syntactic model performs better in most constituent labels. However, the *N*-gram model performs better in WHPP, SBARQ and WHNP.

In the test data, WHPP, SBARQ and WHNP are much less than PP, NP, VP, ADJP, ADVP and CONJP, on which the syntactic model gives better recalls. WHNP spans are small and most of them consist of a question word (WP$) and one or two nouns (e.g. “whose (WP$) parents (NNS)”). WHPP spans are also small and usually consist of a preposition (IN) and a WHNP span (e.g “at (IN) what level (WHNP)”). The *N*-gram model performs better on these small spans. The syntactic model also performs better on S, which covers the whole sentence structure. This verifies the hypothesis introduce that syntactic language models better capture overall sentential grammaticality.

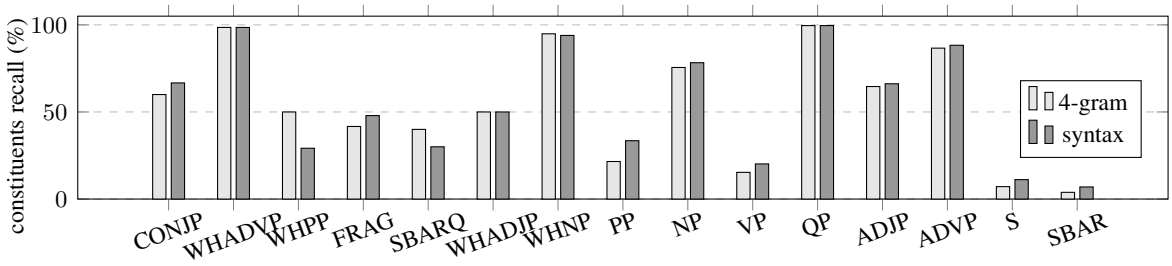


Figure 8: Recalls of different constituents.

	in-domain on WSJ test		cross-domain on WPB test		cross-domain on SANCL test		# of
	BLEU (%)	METEOR (%)	BLEU (%)	METEOR (%)	BLEU (%)	METEOR (%)	sent/s
syntax	50.82	49.33	37.76	46.84	39.97	47.26	17.9
4-gram	42.26	48.00	37.71	46.90	39.72	47.08	**177.0**
combined	**52.38**	**49.66**	**39.12**	**47.07**	**40.60**	**47.38**	15.4

Table 7: Final results on various domains.

7 Combining the syntactic and *N*-gram models

The results above show the respective error characteristics of each model, which are complimentary. This suggests that better results can be achieved by model combination.

7.1 *N*-gram language model feature

We integrate the two types of models by using *N*-gram language model probabilities as features in the syntactic model. *N*-gram language model probabilities, which ranges from 0 to 1. Direct use of real value probabilities as features does not work well in our experiments, and we use discretized features instead. For the L-ARC and R-ARC actions, because no words are pushed onto the stack, The NLM feature is set to NULL by default. For the SHIFT action, different feature values are extracted depending on the NLM from 0 to 1.

In order to measure the *N*-gram probabilities on our data, we train the 4-gram language model WSJ, AFP and XIN data, and randomly sample 4-gram probabilities from the syntactic model output on the WSJ development data, finding that most of 4-gram probabilities p are larger than $10^{-12.5}$. In this way, if p lower than $10^{-12.5}$, NLM feature value is set to LOW. As for p larger than $10^{-12.5}$, we extract the discrete features by assigning them into different bins. We bin the 4-gram probabilities with different granularities without overlap features. As shown in Table 6, NLM-20, NLM-10, NLM-5 and NLM-2 respectively use 20, 10, 5 and 2 bins to capture NLM feature values.

	BLEU (%) on WSJ test
Wan et al. (2009)	33.70*
Zhang and Clark (2011b)	40.10*
Zhang et al. (2012)	43.80*
Zhang (2013)	44.70
syntax (Liu et al., 2015)	50.82
4-gram	42.26
combined	**52.38**

Table 8: Final results of all systems, where "*" means that the system uses extra POS input.

7.2 Final results

We use the WSJ, AFP and XIN for training the *N*-gram model[7]. The same WSJ, WPB and SANCL test data are used to measure performances on different domains.

The experimental results are shown in Tables 7 and 8. In both in-domain and cross-domain test data, the combined system outperforms all other systems, with a BLEU score of 52.38 been achieved in the WSJ domain. It would be overly expensive to obtain a human oracle on discusses. However, according to Papineni (2002), a BLEU

[7]For the combined model, we used the WSJ training data for training, because the syntactic model is slower to train using large data. However, we did a set of experiments to scale up the training data by sampling 900k sentences from AFP. Results show that the combined model gives BLEU scores of 42.86 and 44.44 on the WPB and SANCL tests, respectively. Cross-domain BLEU on WSJ, however falls to 49.84.

BLEU	sentences
ref	For weeks , the market had been nervous about takeovers , after Campeau Corp. 's cash crunch spurred concern about the prospects for future highly leveraged takeovers .
41.37	For weeks , Campeau Corp. 's cash had the prospects for takeovers after the market crunch spurred concern about future highly leveraged takeovers , nervous been about .
ref	Now , at 3:07 , one of the market 's post-crash " reforms " took hold as the S&P 500 futures contract had plunged 12 points , equivalent to around a 100-point drop in the Dow industrials .
51.39	Now , one of the market 's reforms plunged 12 points in the Dow industrials as " post-crash , the S&P 500 futures contract , equivalent to 3:07 took hold at around a 100-point drop had . "
ref	Canadian Utilities had 1988 revenue of C$ 1.16 billion , mainly from its natural gas and electric utility businesses in Alberta , where the company serves about 800,000 customers .
64.38	Canadian Utilities , Alberta , where the company had 1988 revenue of C$ 1.16 billion in its natural gas and electric utility businesses serves mainly from about 800,000 customers .

Table 9: Output samples.

score of over 52.38 indicate an easily understood sentence. Some sample outputs with different BLEU scores are shown in Table 9

In addition, Table 7 shows that the *N*-gram model is the fastest among the models due to its small search space. The running time of the combined system is larger than the pure syntactic system, because of *N*-gram probability computation. Table 8 compare our results with different previous methods on word ordering. Our combined model gives the best reported performance on this standard benchmarks.

8 Conclusion

We empirically compared the strengths and error distributions of syntactic and *N*-gram language models on word ordering, showing that both can benefit from large-scale raw text. The influence of parsing accuracies has relatively small impact on the syntactic language model trained on automatically-parsed data, which enables scaling up of training data for syntactic language models. However, as the size of training data increases, syntactic language models can become intolerantly slow to train, making them benefit less from the scale of training data, as compared with *N*-gram models.

Syntactic models give better performance compared with *N*-gram models, despite trained with less data. On the other hand, the two models lead to different error distributions in word ordering. As a result, we combined the advantages of both systems by integrating a syntactic model trained with relatively small data and an *N*-gram model trained with relatively large data. The resulting model gives better performance than both single models and achieves the best reported scores in a standard benchmark for word ordering.

We release our code under GPL at `https://github.com/SUTDNLP/ZGen`. Future work includes application of the system on text-to-text problem such as machine translation.

Acknowledgments

The research is funded by the Singapore ministry of education (MOE) ACRF Tier 2 project T2MOE201301. We thank the anonymous reviewers for their detailed comments.

References

Regina Barzilay and Kathleen R McKeown. 2005. Sentence fusion for multidocument news summarization. *Computational Linguistics*, 31(3):297–328.

Eugene Charniak, Kevin Knight, and Kenji Yamada. 2003. Syntax-based language models for statistical machine translation. In *Proceedings of MT Summit IX*, pages 40–46. Citeseer.

Wenliang Chen, Min Zhang, and Haizhou Li. 2012. Utilizing dependency language models for graph-based dependency parsing models. In *Proceedings of ACL*, pages 213–222.

David Chiang. 2007. Hierarchical phrase-based translation. *computational linguistics*, 33(2):201–228.

Michael Collins and Brian Roark. 2004. Incremental parsing with the perceptron algorithm. In *Proceedings of ACL*, page 111.

A De Gispert, M Tomalin, and W Byrne. 2014. Word ordering with phrase-based grammars. In *Proceedings of EACL*, pages 259–268.

Michael Denkowski and Alon Lavie. 2010. Extending the meteor machine translation evaluation metric to the phrase level. In *HLT/NAACL*, pages 250–253.

Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. 2004. What's in a translation rule. Technical report, DTIC Document.

Yuqing Guo, Haifeng Wang, and Josef Van Genabith. 2011. Dependency-based n-gram models for general purpose sentence realisation. *Natural Language Engineering*, 17(04):455–483.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of NAACL*, pages 48–54.

Yijia Liu, Yue Zhang, Wanxiang Che, and Bing Qin. 2015. Transition-based syntactic linearization. In *Proceedings of NAACL/HLT*, pages 113–122, Denver, Colorado, May–June.

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

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

Robert Parker, David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2011. English gigaword fifth edition, june. *Linguistic Data Consortium, LDC2011T07*.

Lane Schwartz, Chris Callison-Burch, William Schuler, and Stephen Wu. 2011. Incremental syntactic language models for phrase-based translation. In *Proceedings of ACL/HLT*, pages 620–631.

Libin Shen, Jinxi Xu, and Ralph M Weischedel. 2008. A new string-to-dependency machine translation algorithm with a target dependency language model. In *Proceedings of ACL*, pages 577–585.

Linfeng Song, Yue Zhang, Kai Song, and Qun Liu. 2014. Joint morphological generation and syntactic linearization. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*.

Stephen Wan, Mark Dras, Robert Dale, and Cécile Paris. 2009. Improving grammaticality in statistical sentence generation: Introducing a dependency spanning tree algorithm with an argument satisfaction model. In *Proceedings of EACL*, pages 852–860.

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

Yue Zhang and Stephen Clark. 2011b. Syntax-based grammaticality improvement using ccg and guided search. In *Proceedings of EMNLP*, pages 1147–1157.

Yue Zhang, Graeme Blackwood, and Stephen Clark. 2012. Syntax-based word ordering incorporating a large-scale language model. In *Proceedings of EACL*, pages 736–746. Association for Computational Linguistics.

Yue Zhang, Kai Song, Linfeng Song, Jingbo Zhu, and Qun Liu. 2014. Syntactic smt using a discriminative text generation model. In *Proceedings of EMNLP*, pages 177–182, Doha, Qatar, October.

Yue Zhang. 2013. Partial-tree linearization: generalized word ordering for text synthesis. In *Proceedings of IJCAI*, pages 2232–2238. AAAI Press.

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

A Neural Attention Model for Sentence Summarization

Alexander M. Rush
Facebook AI Research /
Harvard SEAS
srush@seas.harvard.edu

Sumit Chopra
Facebook AI Research
spchopra@fb.com

Jason Weston
Facebook AI Research
jase@fb.com

Abstract

Summarization based on text extraction is inherently limited, but generation-style abstractive methods have proven challenging to build. In this work, we propose a fully data-driven approach to abstractive sentence summarization. Our method utilizes a local attention-based model that generates each word of the summary conditioned on the input sentence. While the model is structurally simple, it can easily be trained end-to-end and scales to a large amount of training data. The model shows significant performance gains on the DUC-2004 shared task compared with several strong baselines.

1 Introduction

Summarization is an important challenge of natural language understanding. The aim is to produce a condensed representation of an input text that captures the core meaning of the original. Most successful summarization systems utilize *extractive* approaches that crop out and stitch together portions of the text to produce a condensed version. In contrast, *abstractive* summarization attempts to produce a bottom-up summary, aspects of which may not appear as part of the original.

We focus on the task of sentence-level summarization. While much work on this task has looked at deletion-based sentence compression techniques (Knight and Marcu (2002), among many others), studies of human summarizers show that it is common to apply various other operations while condensing, such as paraphrasing, generalization, and reordering (Jing, 2002). Past work has modeled this abstractive summarization problem either using linguistically-inspired constraints (Dorr et al., 2003; Zajic et al., 2004) or with syntactic transformations of the input text (Cohn and Lapata, 2008; Woodsend et al., 2010). These approaches are described in more detail in Section 6.

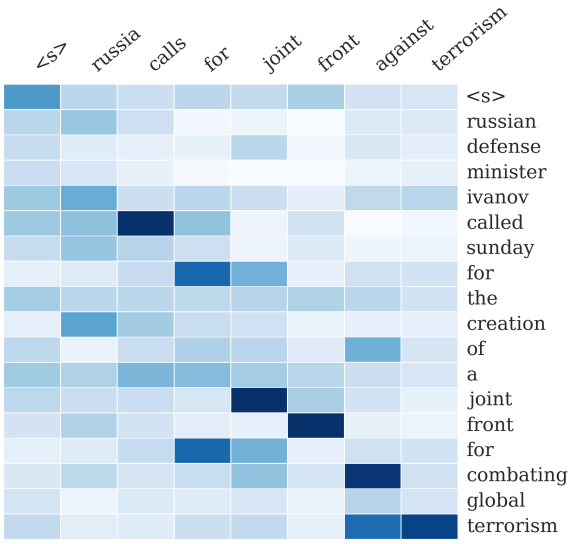


Figure 1: Example output of the attention-based summarization (ABS) system. The heatmap represents a soft alignment between the input (right) and the generated summary (top). The columns represent the distribution over the input after generating each word.

We instead explore a fully data-driven approach for generating abstractive summaries. Inspired by the recent success of neural machine translation, we combine a neural language model with a contextual input encoder. Our encoder is modeled off of the attention-based encoder of Bahdanau et al. (2014) in that it learns a latent soft alignment over the input text to help inform the summary (as shown in Figure 1). Crucially both the encoder and the generation model are trained jointly on the sentence summarization task. The model is described in detail in Section 3. Our model also incorporates a beam-search decoder as well as additional features to model extractive elements; these aspects are discussed in Sections 4 and 5.

This approach to summarization, which we call Attention-Based Summarization (ABS), incorporates less linguistic structure than comparable abstractive summarization approaches, but can easily

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 379–389,
Lisbon, Portugal, 17-21 September 2015.

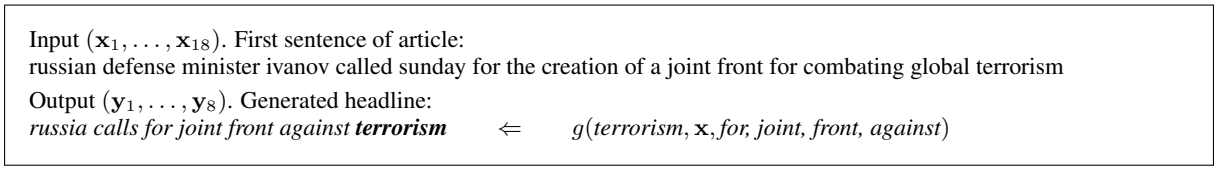


Figure 2: Example input sentence and the generated summary. The score of generating $\mathbf{y}_{i+1}$ (`terrorism`) is based on the context $\mathbf{y}_c$ (`for ... against`) as well as the input $\mathbf{x}_1 \ldots \mathbf{x}_{18}$. Note that the summary generated is abstractive which makes it possible to *generalize* (`russian defense minister` to `russia`) and *paraphrase* (`for combating` to `against`), in addition to *compressing* (dropping `the creation of`), see Jing (2002) for a survey of these editing operations.

scale to train on a large amount of data. Since our system makes no assumptions about the vocabulary of the generated summary it can be trained directly on any document-summary pair.[1] This allows us to train a summarization model for headline-generation on a corpus of article pairs from Gigaword (Graff et al., 2003) consisting of around 4 million articles. An example of generation is given in Figure 2, and we discuss the details of this task in Section 7.

To test the effectiveness of this approach we run extensive comparisons with multiple abstractive and extractive baselines, including traditional syntax-based systems, integer linear program-constrained systems, information-retrieval style approaches, as well as statistical phrase-based machine translation. Section 8 describes the results of these experiments. Our approach outperforms a machine translation system trained on the same large-scale dataset and yields a large improvement over the highest scoring system in the DUC-2004 competition.

2 Background

We begin by defining the sentence summarization task. Given an input sentence, the goal is to produce a condensed summary. Let the input consist of a sequence of M words $\mathbf{x}_1, \ldots, \mathbf{x}_M$ coming from a fixed vocabulary $\mathcal{V}$ of size $|\mathcal{V}| = V$. We will represent each word as an indicator vector $\mathbf{x}_i \in \{0,1\}^V$ for $i \in \{1, \ldots, M\}$, sentences as a sequence of indicators, and $\mathcal{X}$ as the set of possible inputs. Furthermore define the notation $\mathbf{x}_{[i,j,k]}$ to indicate the sub-sequence of elements i, j, k.

A summarizer takes $\mathbf{x}$ as input and outputs a shortened sentence $\mathbf{y}$ of length $N < M$. We will assume that the words in the summary also come from the same vocabulary $\mathcal{V}$ and that the output is a sequence $\mathbf{y}_1, \ldots, \mathbf{y}_N$. Note that in contrast to related tasks, like machine translation, we will assume that the output length N is fixed, and that the system knows the length of the summary before generation.[2]

Next consider the problem of generating summaries. Define the set $\mathcal{Y} \subset (\{0,1\}^V, \ldots, \{0,1\}^V)$ as all possible sentences of length N, i.e. for all i and $\mathbf{y} \in \mathcal{Y}$, $\mathbf{y}_i$ is an indicator. We say a system is *abstractive* if it tries to find the optimal sequence from this set $\mathcal{Y}$,

$$\arg\max_{\mathbf{y} \in \mathcal{Y}} s(\mathbf{x}, \mathbf{y}), \tag{1}$$

under a scoring function $s : \mathcal{X} \times \mathcal{Y} \mapsto \mathbb{R}$. Contrast this to a fully *extractive* sentence summary[3] which transfers words from the input:

$$\arg\max_{m \in \{1,\ldots M\}^N} s(\mathbf{x}, \mathbf{x}_{[m_1,\ldots,m_N]}), \tag{2}$$

or to the related problem of sentence *compression* that concentrates on deleting words from the input:

$$\arg\max_{m \in \{1,\ldots M\}^N, m_{i-1} < m_i} s(\mathbf{x}, \mathbf{x}_{[m_1,\ldots,m_N]}). \tag{3}$$

While abstractive summarization poses a more difficult generation challenge, the lack of hard constraints gives the system more freedom in generation and allows it to fit with a wider range of training data.

In this work we focus on factored scoring functions, s, that take into account a fixed window of previous words:

$$s(\mathbf{x}, \mathbf{y}) \approx \sum_{i=0}^{N-1} g(\mathbf{y}_{i+1}, \mathbf{x}, \mathbf{y}_c), \tag{4}$$

[1] In contrast to a large-scale sentence compression systems like Filippova and Altun (2013) which require monotonic aligned compressions.

[2] For the DUC-2004 evaluation, it is actually the number of *bytes* of the output that is capped. More detail is given in Section 7.

[3] Unfortunately the literature is inconsistent on the formal definition of this distinction. Some systems self-described as abstractive would be extractive under our definition.

where we define $\mathbf{y}_c \triangleq \mathbf{y}_{[i-C+1,\ldots,i]}$ for a window of size C.

In particular consider the conditional log-probability of a summary given the input, $s(\mathbf{x}, \mathbf{y}) = \log p(\mathbf{y}|\mathbf{x}; \theta)$. We can write this as:

$$\log p(\mathbf{y}|\mathbf{x}; \theta) \approx \sum_{i=0}^{N-1} \log p(\mathbf{y}_{i+1}|\mathbf{x}, \mathbf{y}_c; \theta),$$

where we make a Markov assumption on the length of the context as size C and assume for $i < 1$, $\mathbf{y}_i$ is a special start symbol $\langle S \rangle$.

With this scoring function in mind, our main focus will be on modelling the local conditional distribution: $p(\mathbf{y}_{i+1}|\mathbf{x}, \mathbf{y}_c; \theta)$. The next section defines a parameterization for this distribution, in Section 4, we return to the question of generation for factored models, and in Section 5 we introduce a modified factored scoring function.

3 Model

The distribution of interest, $p(\mathbf{y}_{i+1}|\mathbf{x}, \mathbf{y}_c; \theta)$, is a conditional language model based on the input sentence $\mathbf{x}$. Past work on summarization and compression has used a noisy-channel approach to split and independently estimate a language model and a conditional summarization model (Banko et al., 2000; Knight and Marcu, 2002; Daumé III and Marcu, 2002), i.e.,

$$\arg\max_{\mathbf{y}} \log p(\mathbf{y}|\mathbf{x}) = \arg\max_{\mathbf{y}} \log p(\mathbf{y}) p(\mathbf{x}|\mathbf{y})$$

where $p(y)$ and $p(x|y)$ are estimated separately. Here we instead follow work in neural machine translation and directly parameterize the original distribution as a neural network. The network contains both a neural probabilistic language model and an encoder which acts as a conditional summarization model.

3.1 Neural Language Model

The core of our parameterization is a language model for estimating the contextual probability of the next word. The language model is adapted from a standard feed-forward neural network language model (NNLM), particularly the class of NNLMs described by Bengio et al. (2003). The full model is:

$$\begin{aligned} p(\mathbf{y}_{i+1}|\mathbf{y}_c, \mathbf{x}; \theta) &\propto \exp(\mathbf{V}\mathbf{h} + \mathbf{W}\mathrm{enc}(\mathbf{x}, \mathbf{y}_c)), \\ \tilde{\mathbf{y}}_c &= [\mathbf{E}\mathbf{y}_{i-C+1}, \ldots, \mathbf{E}\mathbf{y}_i], \\ \mathbf{h} &= \tanh(\mathbf{U}\tilde{\mathbf{y}}_c). \end{aligned}$$

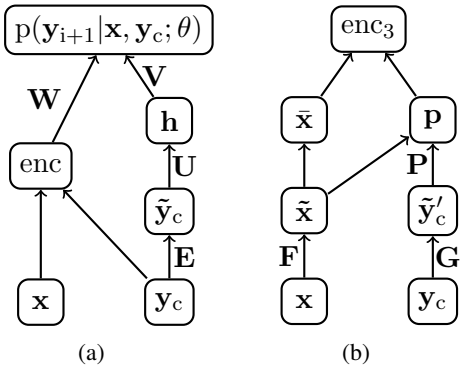


Figure 3: (a) A network diagram for the NNLM decoder with additional encoder element. (b) A network diagram for the attention-based encoder enc_3.

The parameters are $\theta = (\mathbf{E}, \mathbf{U}, \mathbf{V}, \mathbf{W})$ where $\mathbf{E} \in \mathbb{R}^{D \times V}$ is a word embedding matrix, $\mathbf{U} \in \mathbb{R}^{(CD) \times H}$, $\mathbf{V} \in \mathbb{R}^{V \times H}$, $\mathbf{W} \in \mathbb{R}^{V \times H}$ are weight matrices,[4] D is the size of the word embeddings, and $\mathbf{h}$ is a hidden layer of size H. The black-box function enc is a contextual encoder term that returns a vector of size H representing the input and current context; we consider several possible variants, described subsequently. Figure 3a gives a schematic representation of the decoder architecture.

3.2 Encoders

Note that without the encoder term this represents a standard language model. By incorporating in enc and training the two elements jointly we crucially can incorporate the input text into generation. We discuss next several possible instantiations of the encoder.

Bag-of-Words Encoder Our most basic model simply uses the bag-of-words of the input sentence embedded down to size H, while ignoring properties of the original order or relationships between neighboring words. We write this model as:

$$\begin{aligned} \mathrm{enc}_1(\mathbf{x}, \mathbf{y}_c) &= \mathbf{p}^\top \tilde{\mathbf{x}}, \\ \mathbf{p} &= [1/M, \ldots, 1/M], \\ \tilde{\mathbf{x}} &= [\mathbf{F}\mathbf{x}_1, \ldots, \mathbf{F}\mathbf{x}_M]. \end{aligned}$$

Where the input-side embedding matrix $\mathbf{F} \in \mathbb{R}^{H \times V}$ is the only new parameter of the encoder and $\mathbf{p} \in [0, 1]^M$ is a uniform distribution over the input words.

[4] Each of the weight matrices $\mathbf{U}$, $\mathbf{V}$, $\mathbf{W}$ also has a corresponding bias term. For readability, we omit these terms throughout the paper.

For summarization this model can capture the relative importance of words to distinguish content words from stop words or embellishments. Potentially the model can also learn to combine words; although it is inherently limited in representing contiguous phrases.

Convolutional Encoder To address some of the modelling issues with bag-of-words we also consider using a deep convolutional encoder for the input sentence. This architecture improves on the bag-of-words model by allowing local interactions between words while also not requiring the context $\mathbf{y}_{\mathrm{c}}$ while encoding the input.

We utilize a standard time-delay neural network (TDNN) architecture, alternating between temporal convolution layers and max pooling layers.

$$\forall j,\ \mathrm{enc}_2(\mathbf{x}, \mathbf{y}_{\mathrm{c}})_j = \max_i \tilde{\mathbf{x}}^L_{i,j}, \quad (5)$$

$$\forall i, l \in \{1, \dots L\},\ \tilde{\mathbf{x}}^l_j = \tanh(\max\{\bar{\mathbf{x}}^l_{2i-1}, \bar{\mathbf{x}}^l_{2i}\}), \quad (6)$$

$$\forall i, l \in \{1, \dots L\},\ \bar{\mathbf{x}}^l_i = \mathbf{Q}^l \tilde{\mathbf{x}}^{l-1}_{[i-Q,\dots,i+Q]}, \quad (7)$$

$$\tilde{\mathbf{x}}^0 = [\mathbf{F}\mathbf{x}_1, \dots, \mathbf{F}\mathbf{x}_M]. \quad (8)$$

Where $\mathbf{F}$ is a word embedding matrix and $\mathbf{Q}^{L \times H \times 2Q+1}$ consists of a set of filters for each layer $\{1, \dots L\}$. Eq. 7 is a temporal (1D) convolution layer, Eq. 6 consists of a 2-element temporal max pooling layer and a pointwise non-linearity, and final output Eq. 5 is a max over time. At each layer $\tilde{x}$ is one half the size of $\bar{x}$. For simplicity we assume that the convolution is padded at the boundaries, and that M is greater than 2^L so that the dimensions are well-defined.

Attention-Based Encoder While the convolutional encoder has richer capacity than bag-of-words, it still is required to produce a single representation for the entire input sentence. A similar issue in machine translation inspired Bahdanau et al. (2014) to instead utilize an attention-based contextual encoder that constructs a representation based on the generation context. Here we note that if we exploit this context, we can actually use a rather simple model similar to bag-of-words:

$$\begin{aligned}
\mathrm{enc}_3(\mathbf{x}, \mathbf{y}_{\mathrm{c}}) &= \mathbf{p}^\top \bar{\mathbf{x}}, \\
\mathbf{p} &\propto \exp(\tilde{\mathbf{x}} \mathbf{P} \tilde{\mathbf{y}}'_{\mathrm{c}}), \\
\tilde{\mathbf{x}} &= [\mathbf{F}\mathbf{x}_1, \dots, \mathbf{F}\mathbf{x}_M], \\
\tilde{\mathbf{y}}'_{\mathrm{c}} &= [\mathbf{G}\mathbf{y}_{i-C+1}, \dots, \mathbf{G}\mathbf{y}_i], \\
\forall i\ \ \bar{\mathbf{x}}_i &= \sum_{q=i-Q}^{i+Q} \tilde{\mathbf{x}}_i / Q.
\end{aligned}$$

Where $\mathbf{G} \in \mathbb{R}^{D \times V}$ is an embedding of the context, $\mathbf{P} \in \mathbb{R}^{H \times (CD)}$ is a new weight matrix parameter mapping between the context embedding and input embedding, and Q is a smoothing window. The full model is shown in Figure 3b.

Informally we can think of this model as simply replacing the uniform distribution in bag-of-words with a learned soft alignment, $\mathbf{P}$, between the input and the summary. Figure 1 shows an example of this distribution $\mathbf{p}$ as a summary is generated. The soft alignment is then used to weight the smoothed version of the input $\bar{\mathbf{x}}$ when constructing the representation. For instance if the current context aligns well with position i then the words $\mathbf{x}_{i-Q}, \dots, \mathbf{x}_{i+Q}$ are highly weighted by the encoder. Together with the NNLM, this model can be seen as a stripped-down version of the attention-based neural machine translation model.[5]

3.3 Training

The lack of generation constraints makes it possible to train the model on arbitrary input-output pairs. Once we have defined the local conditional model, $p(\mathbf{y}_{i+1} | \mathbf{x}, \mathbf{y}_{\mathrm{c}}; \theta)$, we can estimate the parameters to minimize the negative log-likelihood of a set of summaries. Define this training set as consisting of J input-summary pairs $(\mathbf{x}^{(1)}, \mathbf{y}^{(1)}), \dots, (\mathbf{x}^{(J)}, \mathbf{y}^{(J)})$. The negative log-likelihood conveniently factors[6] into a term for each token in the summary:

$$\begin{aligned}
\mathrm{NLL}(\theta) &= -\sum_{j=1}^{J} \log p(\mathbf{y}^{(j)} | \mathbf{x}^{(j)}; \theta), \\
&= -\sum_{j=1}^{J} \sum_{i=1}^{N-1} \log p(\mathbf{y}^{(j)}_{i+1} | \mathbf{x}^{(j)}, \mathbf{y}_{\mathrm{c}}; \theta).
\end{aligned}$$

We minimize NLL by using mini-batch stochastic gradient descent. The details are described further in Section 7.

[5] To be explicit, compared to Bahdanau et al. (2014) our model uses an NNLM instead of a target-side LSTM, source-side windowed averaging instead of a source-side bi-directional RNN, and a weighted dot-product for alignment instead of an alignment MLP.

[6] This is dependent on using the gold standard contexts $\mathbf{y}_{\mathrm{c}}$. An alternative is to use the predicted context within a structured or reenforcement-learning style objective.

4 Generating Summaries

We now return to the problem of generating summaries. Recall from Eq. 4 that our goal is to find,

$$\mathbf{y}^* \quad = \quad \arg\max_{\mathbf{y} \in \mathcal{Y}} \sum_{i=0}^{N-1} g(\mathbf{y}_{i+1}, \mathbf{x}, \mathbf{y}_{\mathrm{c}}).$$

Unlike phrase-based machine translation where inference is NP-hard, it actually is tractable in theory to compute $\mathbf{y}^*$. Since there is no explicit hard alignment constraint, Viterbi decoding can be applied and requires $O(NV^C)$ time to find an exact solution. In practice though V is large enough to make this difficult.

An alternative approach is to approximate the $\arg\max$ with a strictly greedy or deterministic decoder. While decoders of this form can produce very bad approximations, they have shown to be relatively effective and fast for neural MT models (Sutskever et al., 2014).

A compromise between exact and greedy decoding is to use a beam-search decoder (Algorithm 1) which maintains the full vocabulary $\mathcal{V}$ while limiting itself to K potential hypotheses at each position of the summary. The beam-search algorithm is shown here:

Algorithm 1 Beam Search

Input: Parameters θ, beam size K, input $\mathbf{x}$
Output: Approx. K-best summaries

$\pi[0] \leftarrow \{\epsilon\}$
$\mathcal{S} = \mathcal{V}$ if abstractive else $\{\mathbf{x}_i \mid \forall i\}$
for $i = 0$ to $N-1$ **do**
  ▷ Generate Hypotheses
  $\mathcal{N} \leftarrow \left\{ [\mathbf{y}, \mathbf{y}_{i+1}] \quad | \quad \mathbf{y} \in \pi[i], \mathbf{y}_{i+1} \in \mathcal{S} \right\}$
  ▷ Hypothesis Recombination
  $\mathcal{H} \leftarrow \left\{ \mathbf{y} \in \mathcal{N} \quad | \quad \begin{array}{c} s(\mathbf{y}, \mathbf{x}) > s(\mathbf{y}', \mathbf{x}) \\ \forall \mathbf{y}' \in \mathcal{N} \text{ s.t. } \mathbf{y}_{\mathrm{c}} = \mathbf{y}'_{\mathrm{c}} \end{array} \right\}$
  ▷ Filter K-Max
  $\pi[i+1] \leftarrow \underset{\mathbf{y} \in \mathcal{H}}{\text{K-arg max}}\, g(\mathbf{y}_{i+1}, \mathbf{y}_{\mathrm{c}}, \mathbf{x}) + s(\mathbf{y}, \mathbf{x})$
end for
return $\pi[N]$

As with Viterbi this beam search algorithm is much simpler than beam search for phrase-based MT. Because there is no explicit constraint that each source word be used exactly once there is no need to maintain a bit set and we can simply move from left-to-right generating words. The beam search algorithm requires $O(KNV)$ time. From a computational perspective though, each round of beam search is dominated by computing $p(\mathbf{y}_i | \mathbf{x}, \mathbf{y}_{\mathrm{c}})$ for each of the K hypotheses. These can be computed as a mini-batch, which in practice greatly reduces the factor of K.

5 Extension: Extractive Tuning

While we will see that the attention-based model is effective at generating summaries, it does miss an important aspect seen in the human-generated references. In particular the abstractive model does not have the capacity to find extractive word matches when necessary, for example transferring unseen proper noun phrases from the input. Similar issues have also been observed in neural translation models particularly in terms of translating rare words (Luong et al., 2014).

To address this issue we experiment with tuning a very small set of additional features that tradeoff the abstractive/extractive tendency of the system. We do this by modifying our scoring function to directly estimate the probability of a summary using a log-linear model, as is standard in machine translation:

$$p(\mathbf{y} | \mathbf{x}; \theta, \alpha) \quad \propto \quad \exp(\alpha^\top \sum_{i=0}^{N-1} f(\mathbf{y}_{i+1}, \mathbf{x}, \mathbf{y}_{\mathrm{c}})).$$

Where $\alpha \in \mathbb{R}^5$ is a weight vector and f is a feature function. Finding the best summary under this distribution corresponds to maximizing a factored scoring function s,

$$s(\mathbf{y}, \mathbf{x}) \quad = \quad \sum_{i=0}^{N-1} \alpha^\top f(\mathbf{y}_{i+1}, \mathbf{x}, \mathbf{y}_{\mathrm{c}}).$$

where $g(\mathbf{y}_{i+1}, \mathbf{x}, \mathbf{y}_{\mathrm{c}}) \triangleq \alpha^\top f(\mathbf{y}_{i+1}, \mathbf{x}, \mathbf{y}_{\mathrm{c}})$ to satisfy Eq. 4. The function f is defined to combine the local conditional probability with some additional indicator featrues:

$$\begin{aligned} f(\mathbf{y}_{i+1}, \mathbf{x}, \mathbf{y}_{\mathrm{c}}) = [\ & \log p(\mathbf{y}_{i+1} | \mathbf{x}, \mathbf{y}_{\mathrm{c}}; \theta), \\ & \mathbb{1}\{\exists j.\ \mathbf{y}_{i+1} = \mathbf{x}_j\ \}, \\ & \mathbb{1}\{\exists j.\ \mathbf{y}_{i+1-k} = \mathbf{x}_{j-k}\ \forall k \in \{0, 1\}\}, \\ & \mathbb{1}\{\exists j.\ \mathbf{y}_{i+1-k} = \mathbf{x}_{j-k}\ \forall k \in \{0, 1, 2\}\}, \\ & \mathbb{1}\{\exists k > j.\ \mathbf{y}_i = \mathbf{x}_k, \mathbf{y}_{i+1} = \mathbf{x}_j\}\ ]. \end{aligned}$$

These features correspond to indicators of unigram, bigram, and trigram match with the input as well as reordering of input words. Note that setting $\alpha = \langle 1, 0, \ldots, 0 \rangle$ gives a model identical to standard ABS.

After training the main neural model, we fix θ and tune the α parameters. We follow the statistical machine translation setup and use minimum-error rate training (MERT) to tune for the summarization metric on tuning data (Och, 2003). This tuning step is also identical to the one used for the phrase-based machine translation baseline.

6 Related Work

Abstractive sentence summarization has been traditionally connected to the task of headline generation. Our work is similar to early work of Banko et al. (2000) who developed a statistical machine translation-inspired approach for this task using a corpus of headline-article pairs. We extend this approach by: (1) using a neural summarization model as opposed to a count-based noisy-channel model, (2) training the model on much larger scale (25K compared to 4 million articles), (3) and allowing fully abstractive decoding.

This task was standardized around the DUC-2003 and DUC-2004 competitions (Over et al., 2007). The TOPIARY system (Zajic et al., 2004) performed the best in this task, and is described in detail in the next section. We point interested readers to the DUC web page (`http://duc.nist.gov/`) for the full list of systems entered in this shared task.

More recently, Cohn and Lapata (2008) give a compression method which allows for more arbitrary transformations. They extract tree transduction rules from aligned, parsed texts and learn weights on transfomations using a max-margin learning algorithm. Woodsend et al. (2010) propose a quasi-synchronous grammar approach utilizing both context-free parses and dependency parses to produce legible summaries. Both of these approaches differ from ours in that they directly use the syntax of the input/output sentences. The latter system is W&L in our results; we attempted to train the former system T3 on this dataset but could not train it at scale.

In addition to Banko et al. (2000) there has been some work using statistical machine translation directly for abstractive summary. Wubben et al. (2012) utilize MOSES directly as a method for text simplification.

Recently Filippova and Altun (2013) developed a strictly extractive system that is trained on a relatively large corpora (250K sentences) of article-title pairs. Because their focus is extractive compression, the sentences are transformed by a series of heuristics such that the words are in monotonic alignment. Our system does not require this alignment step but instead uses the text directly.

Neural MT This work is closely related to recent work on neural network language models (NNLM) and to work on neural machine translation. The core of our model is a NNLM based on that of Bengio et al. (2003).

Recently, there have been several papers about models for machine translation (Kalchbrenner and Blunsom, 2013; Cho et al., 2014; Sutskever et al., 2014). Of these our model is most closely related to the attention-based model of Bahdanau et al. (2014), which explicitly finds a soft alignment between the current position and the input source. Most of these models utilize recurrent neural networks (RNNs) for generation as opposed to feed-forward models. We hope to incorporate an RNN-LM in future work.

7 Experimental Setup

We experiment with our attention-based sentence summarization model on the task of headline generation. In this section we describe the corpora used for this task, the baseline methods we compare with, and implementation details of our approach.

7.1 Data Set

The standard sentence summarization evaluation set is associated with the DUC-2003 and DUC-2004 shared tasks (Over et al., 2007). The data for this task consists of 500 news articles from the New York Times and Associated Press Wire services each paired with 4 different human-generated reference summaries (not actually headlines), capped at 75 bytes. This data set is evaluation-only, although the similarly sized DUC-2003 data set was made available for the task. The expectation is for a summary of roughly 14 words, based on the text of a complete article (although we only make use of the first sentence). The full data set is available by request at `http://duc.nist.gov/data.html`.

For this shared task, systems were entered and evaluated using several variants of the recall-oriented ROUGE metric (Lin, 2004). To make recall-only evaluation unbiased to length, output of all systems is cut-off after 75-characters and no bonus is given for shorter summaries.

Unlike BLEU which interpolates various n-gram matches, there are several versions of ROUGE for different match lengths. The DUC evaluation uses ROUGE-1 (unigrams), ROUGE-2 (bigrams), and ROUGE-L (longest-common substring), all of which we report.

In addition to the standard DUC-2014 evaluation, we also report evaluation on single reference headline-generation using a randomly held-out subset of Gigaword. This evaluation is closer to the task the model is trained for, and it allows us to use a bigger evaluation set, which we will include in our code release. For this evaluation, we tune systems to generate output of the average title length.

For training data for both tasks, we utilize the annotated Gigaword data set (Graff et al., 2003; Napoles et al., 2012), which consists of standard Gigaword, preprocessed with Stanford CoreNLP tools (Manning et al., 2014). Our model only uses annotations for tokenization and sentence separation, although several of the baselines use parsing and tagging as well. Gigaword contains around 9.5 million news articles sourced from various domestic and international news services over the last two decades.

For our training set, we pair the headline of each article with its first sentence to create an input-summary pair. While the model could in theory be trained on any pair, Gigaword contains many spurious headline-article pairs. We therefore prune training based on the following heuristic filters: (1) Are there no non-stop-words in common? (2) Does the title contain a byline or other extraneous editing marks? (3) Does the title have a question mark or colon? After applying these filters, the training set consists of roughly $J = 4$ million title-article pairs. We apply a minimal preprocessing step using PTB tokenization, lower-casing, replacing all digit characters with #, and replacing of word types seen less than 5 times with UNK. We also remove all articles from the time-period of the DUC evaluation. release.

The complete input training vocabulary consists of 119 million word tokens and 110K unique word types with an average sentence size of 31.3 words. The headline vocabulary consists of 31 million tokens and 69K word types with the average title of length 8.3 words (note that this is significantly shorter than the DUC summaries). On average there are 4.6 overlapping word types between the headline and the input; although only 2.6 in the first 75-characters of the input.

7.2 Baselines

Due to the variety of approaches to the sentence summarization problem, we report a broad set of headline-generation baselines.

From the DUC-2004 task we include the PREFIX baseline that simply returns the first 75-characters of the input as the headline. We also report the winning system on this shared task, TOPIARY (Zajic et al., 2004). TOPIARY merges a compression system using linguistically-motivated transformations of the input (Dorr et al., 2003) with an unsupervised topic detection (UTD) algorithm that appends key phrases from the full article onto the compressed output. Woodsend et al. (2010) (described above) also report results on the DUC dataset.

The DUC task also includes a set of manual summaries performed by 8 human summarizers each summarizing half of the test data sentences (yielding 4 references per sentence). We report the average inter-annotater agreement score as REFERENCE. For reference, the best human evaluator scores 31.7 ROUGE-1.

We also include several baselines that have access to the same training data as our system. The first is a sentence compression baseline COMPRESS (Clarke and Lapata, 2008). This model uses the syntactic structure of the original sentence along with a language model trained on the headline data to produce a compressed output. The syntax and language model are combined with a set of linguistic constraints and decoding is performed with an ILP solver.

To control for memorizing titles from training, we implement an information retrieval baseline, IR. This baseline indexes the training set, and gives the title for the article with highest BM-25 match to the input (see Manning et al. (2008)).

Finally, we use a phrase-based statistical machine translation system trained on Gigaword to produce summaries, MOSES+ (Koehn et al., 2007). To improve the baseline for this task, we augment the phrase table with "deletion" rules mapping each article word to ϵ, include an additional deletion feature for these rules, and allow for an infinite distortion limit. We also explicitly tune the model using MERT to target the 75-byte capped ROUGE score as opposed to standard

Model	DUC-2004 ROUGE-1	DUC-2004 ROUGE-2	DUC-2004 ROUGE-L	Gigaword ROUGE-1	Gigaword ROUGE-2	Gigaword ROUGE-L	Ext. %
IR	11.06	1.67	9.67	16.91	5.55	15.58	29.2
PREFIX	22.43	6.49	19.65	23.14	8.25	21.73	100
COMPRESS	19.77	4.02	17.30	19.63	5.13	18.28	100
W&L	22	6	17	-	-	-	-
TOPIARY	25.12	6.46	20.12	-	-	-	-
MOSES+	26.50	8.13	22.85	28.77	12.10	26.44	70.5
ABS	26.55	7.06	22.05	30.88	12.22	27.77	85.4
ABS+	28.18	8.49	23.81	31.00	12.65	28.34	91.5
REFERENCE	29.21	8.38	24.46	-	-	-	45.6

Table 1: Experimental results on the main summary tasks on various ROUGE metrics . Baseline models are described in detail in Section 7.2. We report the percentage of tokens in the summary that also appear in the input for Gigaword as `Ext %`.

BLEU-based tuning. Unfortunately, one remaining issue is that it is non-trivial to modify the translation decoder to produce fixed-length outputs, so we tune the system to produce roughly the expected length.

7.3 Implementation

For training, we use mini-batch stochastic gradient descent to minimize negative log-likelihood. We use a learning rate of 0.05, and split the learning rate by half if validation log-likelihood does not improve for an epoch. Training is performed with shuffled mini-batches of size 64. The minibatches are grouped by input length. After each epoch, we renormalize the embedding tables (Hinton et al., 2012). Based on the validation set, we set hyperparameters as $D = 200$, $H = 400$, $C = 5$, $L = 3$, and $Q = 2$.

Our implementation uses the Torch numerical framework (`http://torch.ch/`) and will be openly available along with the data pipeline. Crucially, training is performed on GPUs and would be intractable or require approximations otherwise. Processing 1000 mini-batches with $D = 200$, $H = 400$ requires 160 seconds. Best validation accuracy is reached after 15 epochs through the data, which requires around 4 days of training.

Additionally, as described in Section 5 we apply a MERT tuning step after training using the DUC-2003 data. For this step we use Z-MERT (Zaidan, 2009). We refer to the main model as ABS and the tuned model as ABS+.

8 Results

Our main results are presented in Table 1. We run experiments both using the DUC-2004 evaluation data set (500 sentences, 4 references, 75 bytes) with all systems and a randomly held-out Gigaword test set (2000 sentences, 1 reference). We first note that the baselines COMPRESS and IR do relatively poorly on both datasets, indicating that neither just having article information or language model information alone is sufficient for the task. The PREFIX baseline actually performs surprisingly well on ROUGE-1 which makes sense given the earlier observed overlap between article and summary.

Both ABS and MOSES+ perform better than TOPIARY, particularly on ROUGE-2 and ROUGE-L in DUC. The full model ABS+ scores the best on these tasks, and is significantly better based on the default ROUGE confidence level than TOPIARY on all metrics, and MOSES+ on ROUGE-1 for DUC as well as ROUGE-1 and ROUGE-L for Gigaword. Note that the additional extractive features bias the system towards retaining more input words, which is useful for the underlying metric.

Next we consider ablations to the model and algorithm structure. Table 2 shows experiments for the model with various encoders. For these experiments we look at the perplexity of the system as a language model on validation data, which controls for the variable of inference and tuning. The NNLM language model with no encoder gives a gain over the standard n-gram language model. Including even the bag-of-words encoder reduces perplexity number to below 50. Both the convolutional encoder and the attention-based encoder further reduce the perplexity, with attention giving a value below 30.

We also consider model and decoding ablations on the main summary model, shown in Table 3. These experiments compare to the BoW encoding models, compare beam search and greedy decoding, as well as restricting the system to be com-

Model	Encoder	Perplexity
KN-Smoothed 5-Gram	none	183.2
Feed-Forward NNLM	none	145.9
Bag-of-Word	enc_1	43.6
Convolutional (TDNN)	enc_2	35.9
Attention-Based (ABS)	enc_3	27.1

Table 2: Perplexity results on the Gigaword validation set comparing various language models with C=5 and end-to-end summarization models. The encoders are defined in Section 3.

Decoder	Model	Cons.	R-1	R-2	R-L
Greedy	ABS+	Abs	26.67	6.72	21.70
Beam	BOW	Abs	22.15	4.60	18.23
Beam	ABS+	Ext	27.89	7.56	22.84
Beam	ABS+	Abs	28.48	8.91	23.97

Table 3: ROUGE scores on DUC-2003 development data for various versions of inference. Greedy and Beam are described in Section 4. Ext. is a purely extractive version of the system (Eq. 2)

plete extractive. Of these features, the biggest impact is from using a more powerful encoder (attention versus BoW), as well as using beam search to generate summaries. The abstractive nature of the system helps, but for ROUGE even using pure extractive generation is effective.

Finally we consider example summaries shown in Figure 4. Despite improving on the baseline scores, this model is far from human performance on this task. Generally the models are good at picking out key words from the input, such as names and places. However, both models will reorder words in syntactically incorrect ways, for instance in Sentence 7 both models have the wrong subject. ABS often uses more interesting re-wording, for instance *new nz pm after election* in Sentence 4, but this can also lead to attachment mistakes such a *russian oil giant chevron* in Sentence 11.

9 Conclusion

We have presented a neural attention-based model for abstractive summarization, based on recent developments in neural machine translation. We combine this probabilistic model with a generation algorithm which produces accurate abstractive summaries. As a next step we would like to further improve the grammaticality of the summaries in a data-driven way, as well as scale this system to generate paragraph-level summaries. Both pose additional challenges in terms of efficient alignment and consistency in generation.

I(1): a detained iranian-american academic accused of acting against national security has been released from a tehran prison after a hefty bail was posted , a to p judiciary official said tuesday .
G: iranian-american academic held in tehran released on bail
A: detained iranian-american academic released from jail after posting bail
A+: detained iranian-american academic released from prison after hefty bail

I(2): ministers from the european union and its mediterranean neighbors gathered here under heavy security on monday for an unprecedented conference on economic and political cooperation .
G: european mediterranean ministers gather for landmark conference by julie bradford
A: mediterranean neighbors gather for unprecedented conference on heavy security
A+: mediterranean neighbors gather under heavy security for unprecedented conference

I(3): the death toll from a school collapse in a haitian shanty-town rose to ## after rescue workers uncovered a classroom with ## dead students and their teacher , officials said saturday .
G: toll rises to ## in haiti school unk : official
A: death toll in haiti school accident rises to ##
A+: death toll in haiti school to ## dead students

I(4): australian foreign minister stephen smith sunday congratulated new zealand 's new prime minister-elect john key as he praised ousted leader helen clark as a " gutsy " and respected politician .
G: time caught up with nz 's gutsy clark says australian fm
A: australian foreign minister congratulates new nz pm after election
A+: australian foreign minister congratulates smith new zealand as leader

I(5): two drunken south african fans hurled racist abuse at the country 's rugby sevens coach after the team were eliminated from the weekend 's hong kong tournament , reports said tuesday .
G: rugby union : racist taunts mar hong kong sevens : report
A: south african fans hurl racist taunts at rugby sevens
A+: south african fans racist abuse at rugby sevens tournament

I(6): christian conservatives – kingmakers in the last two us presidential elections – may have less success in getting their pick elected in #### , political observers say .
G: christian conservatives power diminished ahead of #### vote
A: christian conservatives may have less success in #### election
A+: christian conservatives in the last two us presidential elections

I(7): the white house on thursday warned iran of possible new sanctions after the un nuclear watchdog reported that tehran had begun sensitive nuclear work at a key site in defiance of un resolutions .
G: us warns iran of step backward on nuclear issue
A: iran warns of possible new sanctions on nuclear work
A+: un nuclear watchdog warns iran of possible new sanctions

I(8): thousands of kashmiris chanting pro-pakistan slogans on sunday attended a rally to welcome back a hardline separatist leader who underwent cancer treatment in mumbai .
G: thousands attend rally for kashmir hardliner
A: thousands rally in support of hardline kashmiri separatist leader
A+: thousands of kashmiris rally to welcome back cancer treatment

I(9): an explosion in iraq 's restive northeastern province of diyala killed two us soldiers and wounded two more , the military reported monday .
G: two us soldiers killed in iraq blast december toll ###
A: # us two soldiers killed in restive northeast province
A+: explosion in restive northeastern province kills two us soldiers

I(10): russian world no. # nikolay davydenko became the fifth withdrawal through injury or illness at the sydney international wednesday , retiring from his second round match with a foot injury .
G: tennis : davydenko pulls out of sydney with injury
A: davydenko pulls out of sydney international with foot injury
A+: russian world no. # davydenko retires at sydney international

I(11): russia 's gas and oil giant gazprom and us oil major chevron have set up a joint venture based in resource-rich northwestern siberia , the interfax news agency reported thursday quoting gazprom officials .
G: gazprom chevron set up joint venture
A: russian oil giant chevron set up siberia joint venture
A+: russia 's gazprom set up joint venture in siberia

Figure 4: Example sentence summaries produced on Gigaword. **I** is the input, **A** is ABS, and **G** is the true headline.

References

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

Michele Banko, Vibhu O Mittal, and Michael J Witbrock. 2000. Headline generation based on statistical translation. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, pages 318–325. Association for Computational Linguistics.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *The Journal of Machine Learning Research*, 3:1137–1155.

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

James Clarke and Mirella Lapata. 2008. Global inference for sentence compression: An integer linear programming approach. *Journal of Artificial Intelligence Research*, pages 399–429.

Trevor Cohn and Mirella Lapata. 2008. Sentence compression beyond word deletion. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, pages 137–144. Association for Computational Linguistics.

Hal Daumé III and Daniel Marcu. 2002. A noisy-channel model for document compression. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 449–456. Association for Computational Linguistics.

Bonnie Dorr, David Zajic, and Richard Schwartz. 2003. Hedge trimmer: A parse-and-trim approach to headline generation. In *Proceedings of the HLT-NAACL 03 on Text summarization workshop-Volume 5*, pages 1–8. Association for Computational Linguistics.

Katja Filippova and Yasemin Altun. 2013. Overcoming the lack of parallel data in sentence compression. In *EMNLP*, pages 1481–1491.

David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2003. English gigaword. *Linguistic Data Consortium, Philadelphia*.

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

Hongyan Jing. 2002. Using hidden markov modeling to decompose human-written summaries. *Computational linguistics*, 28(4):527–543.

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

Kevin Knight and Daniel Marcu. 2002. Summarization beyond sentence extraction: A probabilistic approach to sentence compression. *Artificial Intelligence*, 139(1):91–107.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th annual meeting of the ACL on interactive poster and demonstration sessions*, pages 177–180. Association for Computational Linguistics.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, pages 74–81.

Thang Luong, Ilya Sutskever, Quoc V Le, Oriol Vinyals, and Wojciech Zaremba. 2014. Addressing the rare word problem in neural machine translation. *arXiv preprint arXiv:1410.8206*.

Christopher D Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to information retrieval*, volume 1. Cambridge university press Cambridge.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60.

Courtney Napoles, Matthew Gormley, and Benjamin Van Durme. 2012. Annotated gigaword. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*, pages 95–100. Association for Computational Linguistics.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 160–167. Association for Computational Linguistics.

Paul Over, Hoa Dang, and Donna Harman. 2007. Duc in context. *Information Processing & Management*, 43(6):1506–1520.

Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112.

Kristian Woodsend, Yansong Feng, and Mirella Lapata. 2010. Generation with quasi-synchronous grammar. In *Proceedings of the 2010 conference on empirical methods in natural language processing*, pages 513–523. Association for Computational Linguistics.

Sander Wubben, Antal Van Den Bosch, and Emiel Krahmer. 2012. Sentence simplification by monolingual machine translation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 1015–1024. Association for Computational Linguistics.

Omar Zaidan. 2009. Z-mert: A fully configurable open source tool for minimum error rate training of machine translation systems. *The Prague Bulletin of Mathematical Linguistics*, 91:79–88.

David Zajic, Bonnie Dorr, and Richard Schwartz. 2004. Bbn/umd at duc-2004: Topiary. In *Proceedings of the HLT-NAACL 2004 Document Understanding Workshop, Boston*, pages 112–119.

Scientific Article Summarization Using Citation-Context and Article's Discourse Structure

Arman Cohan
Information Retrieval Lab
Computer Science Department
Georgetown University
arman@ir.cs.georgetown.edu

Nazli Goharian
Information Retrieval Lab
Computer Science Department
Georgetown University
nazli@ir.cs.georgetown.edu

Abstract

We propose a summarization approach for scientific articles which takes advantage of citation-context and the document discourse model. While citations have been previously used in generating scientific summaries, they lack the related context from the referenced article and therefore do not accurately reflect the article's content. Our method overcomes the problem of inconsistency between the citation summary and the article's content by providing context for each citation. We also leverage the inherent scientific article's discourse for producing better summaries. We show that our proposed method effectively improves over existing summarization approaches (greater than 30% improvement over the best performing baseline) in terms of ROUGE scores on TAC2014 scientific summarization dataset. While the dataset we use for evaluation is in the biomedical domain, most of our approaches are general and therefore adaptable to other domains.

1 Introduction

Due to the expanding rate at which articles are being published in each scientific field, it has become difficult for researchers to keep up with the developments in their respective fields. Scientific summarization aims to facilitate this problem by providing readers with concise and informative representation of contributions or findings of an article. Scientific summarization is different than general summarization in three main aspects (Teufel and Moens, 2002). First, the length of scientific papers are usually much longer than general articles (e.g newswire). Second, in scientific summarization, the goal is typically to provide a technical summary of the paper which includes important findings, contributions or impacts of a paper to the community. Finally, scientific papers follow a natural discourse. A common organization for scientific paper is the one in which the problem is first introduced and is followed by the description of hypotheses, methods, experiments, findings and finally results and implications. Scientific summarization was recently further motivated by TAC2014 biomedical summarization track[1] in which they planned to investigate this problem in the domain of biomedical science.

There are currently two types of approaches towards scientific summarization. First is the articles' abstracts. While abstracts provide a general overview of the paper, they cannot be considered as an accurate scientific summary by themselves. That is due to the fact that not all the contributions and impacts of the paper are included in the abstract (Elkiss et al., 2008). In addition, the stated contributions are those that the authors deem important while they might be less important to the scientific community. Moreover, contributions are stated in a general and less focused fashion. These problems motivated the other form of scientific summaries, i.e., citation based summaries. Citation based summary is a summary which is formed by utilizing a set of citations to a referenced article (Qazvinian and Radev, 2008; Qazvinian et al., 2013). This set of citations has been previously indicated as a good representation of important findings and contributions of the article. Contributions stated in the citations are usually more focused than the abstract and contain additional information that is not in the abstract (Elkiss et al., 2008).

However, citations may not accurately represent

[1]Text Analysis Conference - *http://www.nist.gov/tac/2014*

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 390–400,
Lisbon, Portugal, 17-21 September 2015.

the content of the referenced article as they are biased towards the viewpoint of the citing authors. Moreover, citations may address a contribution or a finding regarding the referenced article without referring to the assumptions and data under which it was obtained.

The problem of inconsistency between the degree of certainty of expressing findings between the citing article and referenced article has been also reported (De Waard and Maat, 2012). Therefore, citations by themselves lack the related "context" from the original article. We call the textual spans in the reference articles that reflect the citation, the citation-context. Figure 1 shows an example of the citation-context in the reference article (green color) for a citation in the citing article (blue color).

We propose an approach to overcome the aforementioned shortcomings of existing scientific summaries. Specifically, we extract citation-context in the reference article for each citation. Then, by using the discourse facets of the citations as well as community structure of the citation-contexts, we extract candidate sentences for the summary. The final summary is formed by maximizing both novelty and informativeness of the sentences in the summary. We evaluate and compare our methods against several well-known summarization methods. Evaluation results on the TAC2014 dataset show that our proposed methods can effectively improve over the well-known existing summarization approaches. That is, we obtained greater than 30% improvement over the highest performing baseline in terms of mean ROUGE scores.

Citing article:

... The general impression that has emerged is that transformation of human cells by Ras requires the inactivation of both the pRb and p53 pathways, typically achieved by introducing DNA tumor virus oncoproteins such as SV40 large tumor antigen (T-Ag) or human papillomavirus E6 and E7 proteins (Serrano et al., 1997).
To address this question, we have been investigating the ...

Reference article (Serrano et al., 1997):

... continued to incorporate BrdU and proliferate following introduction of H-ras V12. In agreement with previous reports (66 and 60), both p53/ and p16/ MEFs expressing H-ras V12 displayed features of oncogenic transformation (e.g., refractile morphology, loss of contact inhibition), which were apparent almost immediately after H-ras V12 was transduced (data not shown). These results indicate that p53 and p16 are essential for ras-induced arrest in MEFs, and that inactivation of either p53 or p16 alone is sufficient to circumvent arrest. In REF52 and IMR90 fibroblasts, a different approach was ...

Figure 1: The blue highlighted span in the citing article (top) shows the citation text, followed by the citation marker (pink span). For this citation, the citation-context is the green highlighted span in the reference article (bottom). The text spans outside the scope of the citation text and citation-context are not highlighted.

2 Related work

Document summarization is a relatively well studied area and various types of approaches for document summarization have been proposed in the past twenty years.

Latent Semantic Analysis (LSA) has been used in text summarization first by (Gong and Liu, 2001). Other variations of LSA based summarization approaches have later been introduced (Steinberger and Jezek, 2004; Steinberger et al., 2005; Lee et al., 2009; Ozsoy et al., 2010). Summarization approaches based on topic modeling and Bayesian models have also been explored (Vanderwende et al., 2007; Haghighi and Vanderwende, 2009; Celikyilmaz and Hakkani-Tur, 2010; Ritter et al., 2010; Celikyilmaz and Hakkani-Tür, 2011; Ma and Nakagawa, 2013; Li and Li, 2014). In these approaches, the content/topic distribution in the final summary is estimated using a graphical probabilistic model. Some approaches have viewed summarization as an optimization task solved by linear programming (Clarke and Lapata, 2008; Berg-Kirkpatrick et al., 2011; Woodsend and Lapata, 2012). Many works have viewed the summarization problem as a supervised classification problem in which several features are used to predict the inclusion of document sentences in the summary. Variations of supervised models have been utilized for summary generation, such as: maximum entropy (Osborne, 2002), HMM (Conroy et al., 2011), CRF (Galley, 2006; Shen et al., 2007; Chali and Hasan, 2012), SVM (Xie and Liu, 2010), logistic regression (Louis et al., 2010) and reinforcement learning (Rioux et al., 2014). Problems with supervised models in context of summarization include the need for large amount of annotated data and domain dependency.

Graph based models have shown promising results for text summarization. In these approaches, the goal is to find the most central sentences in the document by constructing a graph in which nodes are sentences and edges are similarity between these sentences. Examples of these techniques include LexRank (Erkan and Radev, 2004), TextRank (Mihalcea and Tarau, 2004), and the work by (Paul et al., 2010). Maximizing the novelty and preventing

the redundancy in a summary is addressed by greedy selection of content summarization (Carbonell and Goldstein, 1998; Guo and Sanner, 2010; Lin et al., 2010). Rhetorical structure of the documents have also been investigated for automatic summarization. In this line of work, dependency and discourse parsing based on Rhetorical Structure Theory (RST) (Mann and Thompson, 1988) is used for analyzing the structure of the documents (Hirao et al., 2013; Kikuchi et al., 2014; Yoshida et al., 2014). Summarization based on rhetorical structure is better suited for shorter documents and is highly dependent on the quality of the discourse parser that is used. Training the discourse parser requires large amount of training data in the RST framework.

Scientific article summarization was first studied by (Teufel and Moens, 2002) in which they trained a supervised Naive Bayes classifier to select informative content for the summary. Later (Elkiss et al., 2008) argue the benefits of citations to scientific work analysis. (Cohan et al., 2015) use a search oriented approach for finding relevant parts of the reference paper to citations. (Qazvinian and Radev, 2008; Qazvinian et al., 2013) use citations to an article to construct its summary. More specifically, they perform hierarchical agglomerative clustering on citations to maximize purity and select most central sentences from each cluster for the final summary. Our work is closest to (Qazvinian and Radev, 2008) with the difference that they only make use of citations. While citations are useful for summarization, relying solely on them might not accurately capture the original context of the referenced paper. That is, the generated summary lacks the appropriate evidence to reflect the content of the original paper, such as circumstances, data and assumptions under which certain findings were obtained. We address this shortcoming by leveraging the citation-context and the inherent discourse model in the scientific articles.

3 The summarization approach

Our scientific summary generation algorithm is composed of four steps: (1) Extracting the citation-context, (2) Grouping citation-contexts, (3) Ranking the sentences within each group and (4) Selecting the sentences for final summary. We assume that the citation text (the text span in the citing article that references another article) in each citing article is already known. We describe each step in the following sub-sections. Our proposed method generates a summary of an article with the premise that the article has a number of citations to it. We call the article that is being referenced the "reference article". We shall note that we tokenized the articles' text to sentences by using the punkt unsupervised sentence boundary detection algorithm (Kiss and Strunk, 2006). We modified the original sentence boundary detection algorithm to also account for biomedical abbreviations. For the rest of the paper, "sentence" refers to units that are output of the sentence boundary detection algorithm, whereas "text span" or in short "span" can consist of multiple sentences.

3.1 Extracting the citation-context

As described in section 2, one problem with existing citation based summarization approaches is that they lack the context of the referenced paper. Therefore, our goal is to leverage citation-context in the reference article to correctly reflect the reference paper. To find citation-contexts, we consider each citation as an n-gram vector and use vector space model for locating the relevant text spans in the reference article. More specifically, given a citation c, we return the ranked list of text spans $r_1, r_2, ..., r_n$ which have the highest similarity to c. We call the retrieved text spans reference spans. These reference spans are essentially forming the context for each citation. The similarity function is the cosine similarity between the pivoted normalized vectors. We evaluated four different approaches for forming the citation vector.

1. All terms in citation except for stopwords, numeric values and citation markers i.e., name of authors or numbered citations. In figure 1 an example of citation marker is shown.

2. Terms with high inverted document frequency (idf). Idf values of terms have shown to be a good estimate of term informativeness.

3. Concepts that are represented through noun phrases in the citation, for example in the following: " ... typically achieved by introducing DNA tumor virus oncoproteins such a ... " which is part of a citation, the phrase "DNA tumor virus oncoproteins" is a noun phrase.

4. Biomedical concepts and noun phrases expanded by related biomedical concepts: This formation is specific to the biomedical domain. It selects biomedical concepts and noun phrases in the citation and uses related biomedical terminology to expand the citation vector. We used Metamap[1] for extracting biomedical concepts from the citation text (which is a tool for mapping free form text to UMLS[2] concepts). For expanding the citation vector using the related biomedical terminology, we used SNOMED CT[3] ontology by which we added synonyms of the concepts in the citation text to the citation vector.

3.2 Grouping the citation-contexts

After identifying the context for each citation, we use them to form the summary. To capture various important aspects of the reference article, we form groups of citation-contexts that are about the same topic. We use the following two approaches for forming these groups:

Community detection - We want to find diverse key aspects of the reference article. We form the graph of extracted reference spans in which nodes are sentences and edges are similarity between sentences. As for the similarity function, we use cosine similarity between tf-idf vectors of the sentences. Similar to (Qazvinian and Radev, 2008), we want to find subgraphs or communities whose intra-connectivity is high but inter-connectivity is low. Such quality is captured by the modularity measure of the graph (Newman, 2006; Newman, 2012). Graph modularity quantifies the denseness of the subgraphs in comparison with denseness of the graph of randomly distributed edges and is defined as follows:

$$Q = \frac{1}{2m} \sum_{vw} \left[A_{vw} - \frac{k_v \times k_w}{2m} \right] \delta(c_v, c_w)$$

Where A_{vw} is the weigh of the edge (v, w); k_v is the degree of the vertex v; c_v is the community of vertex v; δ is the Kronecker's delta function and $m = \sum_{vw} A_{vw}$ is the normalization factor.

While the general problem of precise partitioning of the graph into highly dense communities that optimizes the modularity is computationally prohibitive (Brandes et al., 2008), many heuristic algorithms have been proposed with reasonable results. To extract communities from the graph of reference spans, we use the algorithm proposed by (Blondel et al., 2008) which is a simple yet accurate and efficient community detection algorithm. Specifically, communities are built in a hierarchical fashion. At first, each node belongs to a separate community. Then nodes are assigned to new communities if there is a positive gain in modularity. This process is applied iteratively until no further improvement in modularity is possible.

Discourse model - A natural discourse model is followed in each scientific article. In this method, instead of finding communities to capture different important aspects of the paper, we try to select reference spans based on the discourse model of the paper. The discourse model is according to the following facets: "hypothesis", "method", "results", "implication", "discussion" and "data-set-used". The goal is to ideally include reference spans from each of these discourse facets of the article in the summary to correctly capture all aspects of the article. We use a one-vs-rest SVM supervised model with linear kernel to classify the reference spans to their respective discourse facets. Training was done on both the citation and reference spans since empirical evaluation showed marginal improvements upon including the reference spans in addition to the citation itself. We use unigram and verb features with tf-idf weighting to train the classifier.

3.3 Ranking model

To identify the most representative sentences of each group, we require a measure of importance of sentences. We consider the sentences in a group as a graph and rank nodes based on their importance. An important node is a node that has many connections with other nodes. There are various ways of measuring centrality of nodes such as nodes degree, betweenness, closeness and eigenvectors. Here, we opt for eigenvectors and we find the most central sentences in each group by using the "power method" (Erkan and Radev, 2004) which iteratively updates the eigenvector until convergence.

[1] *http://metamap.nlm.nih.gov/*

[2] Unified Medical Language System - a compendium of controlled vocabularies in the biomedical sciences, *http://www.nlm.nih.gov/research/umls*

[3] *http://www.nlm.nih.gov/research/umls/Snomed/snomed_main.html*

3.4 Selecting the sentences for final summary

After scoring and ranking the sentences in each group which were identified either by discourse model or by community detection algorithm, we employ two strategies for generating the summary within the summary length threshold.

- *Iterative*: We select top sentences iteratively from each group until we reach the summary length threshold. That is, we first pick the top sentence from all groups and if the threshold is not met, we select the second sentence and so forth. In the discourse based method, the following ordering for selecting sentences from groups is used: "hypothesis", "method","results", "implication" and "discussion". In the community detection method, no pre-determined order is specified.
- *Novelty:* We employ a greedy strategy similar to MMR (Carbonell and Goldstein, 1998) in which sentences from each group are selected based on the following scoring formula:

$$\text{score(S)} \stackrel{\text{def}}{=} \lambda Sim_1(\text{S, D}) - (1-\lambda) Sim_2(\text{S,Summary})$$

Where, for each sentence S, the score is a linear interpolation of similarity of sentence with all other sentences (Sim_1) and the similarity of sentence with the sentences already in the summary (Sim_2) and λ is a constant. We empirically set $\lambda = 0.7$ and also selected top 3 central sentences from each group as the candidates for the final summary.

4 Experimental setup

4.1 Data

We used the TAC2014 biomedical summarization dataset for evaluation of our proposed method. The TAC2014 benchmark contains 20 topics each of which consists of one reference article and several articles that have citations to each reference article (the statistics of the dataset is shown in Table 1). All articles are biomedical papers published by Elsevier. For each topic, 4 experts in biomedical domain have written a scientific summary of length not exceeding 250 words for the reference article. The data also contains annotated citation texts as well as the discourse facets. The latter were used to build the supervised discourse model. The distribution of discourse facets is shown in Table 2.

Table 1: Dataset statistics

	mean	std
# of topics (reference articles)	20	0
# of Gold summaries for each topic	4	0
# of citing articles in each topic	15.65	2.70
# of citations to the reference article in each citing article	1.57	1.17
Length of summaries (words)	235.64	31.24
Length of articles (words)	9759.86	2199.48

Table 2: Distribution of annotated discourse facets

Discourse facet	count
Hypothesis	21
Method	155
Results	490
Implication	140
Discussion	446

4.2 Baselines

We compared existing well-known and widely-used approaches discussed in section 2 with our approach and evaluated their effectiveness for scientific summarization. The first three approaches use the scientific article's text and the last approach uses the citations to the article for generating the summary.

- *LSA (Steinberger and Jezek, 2004)* - The LSA summarization method is based on singular value decomposition. In this method, a term document index $\mathbf{A}$ is created in which the values correspond to the tf-idf values of terms in the document. Then, Singular Value Decomposition, a dimension reduction approach, is applied to A. This will yield a singular value matrix $\mathbf{\Sigma}$ and a singular vector matrix $\mathbf{V}^T$. The top singular vectors are selected from $\mathbf{V}^T$ iteratively until length of the summary reaches a predefined threshold.
- *LexRank (Erkan and Radev, 2004)* - LexRank uses a measure called centrality to find the most representative sentences in given sets of sentences. It finds the most central sentences by updating the score of each sentence using an algorithm based on PageRank random walk ranking model (Page et al., 1999). More specifically, the centrality score of each sentence is represented by a centrality matrix $\mathbf{p}$ which is updated iteratively through the following equation using a method called "power method":

$$\mathbf{p} = \mathbf{A}^T \mathbf{p}$$

Where matrix $\mathbf{A}$ is based on the similarity matrix $\mathbf{B}$ of the sentences:

$$\mathbf{A} = [d\mathbf{U} + (1-d)\mathbf{B}]$$

In which $\mathbf{U}$ is a square matrix with values $1/N$ and d is a parameter called the damping factor. We set d to 0.1 which is the default suggested value.

- *MMR (Carbonell and Goldstein, 1998)* - In Maximal Marginal Relevance (MMR), sentences are greedily ranked according to a score based on their relevance to the document and the amount of redundant information they carry. It scores sentences based on the maximization of the linear interpolation of the relevance to the document and diversity:

$$\text{MMR(S,D)} \stackrel{\text{def}}{=} \lambda Sim_1(\text{S, D}) - (1-\lambda) Sim_2(\text{S,Summary})$$

Where S is the sentence being evaluated, D is the document being summarized, Sim_1 and Sim_2 are similarity function, $Summary$ is the summary formed by the previously selected sentences and λ is a parameter. We used cosine similarity as similarity functions and we set λ to 0.3, 0.5 and 0.7 for observing the effect of informativeness vs. novelty.

- *Citation summary (Qazvinian and Radev, 2008)*- In this approach, a network of citations is built and citations are clustered to maximum purity (Zhao and Karypis, 2001) and mutual information. These clusters are then used to generate the final summary by selecting the top central sentences from each cluster in a round-robin fashion. Our approach is similar to this work in that they also use centrality scores on citation network clusters. Since they only focus on citations, comparison of our approach with this work gives a better insight into how beneficial our use of citation-context and article's discourse model can be in generating scientific summaries.

5 Results and discussions

5.1 Evaluation metrics

We use the ROUGE evaluation metrics which has shown consistent correlation with manually evaluated summarization scores (Lin, 2004). More specifically, we use ROUGE-L, ROUGE-1 and ROUGE-2 to evaluate and compare the quality of the summaries generated by our system. While ROUGE-N focuses on n-gram overlaps, ROUGE-L uses the longest common subsequence to measure the quality of the summary. ROUGE-N where N is the n-gram order, is defined as follows:

$$\text{ROUGE-N} = \frac{\sum_{S \in \{\text{Gold summaries}\}} \sum_{W \in S} f_{match}(W)}{\sum_{S \in \{\text{Gold summaries}\}} \sum_{W \in S} f(W)}$$

Where W is the n-gram, $f(.)$ is the count function, $f_{match}(.)$ is the maximum number of n-grams co-occurring in the generated summary and in a set of gold summaries. For a candidate summary C with n words and a gold summary S with u sentences, ROUGE-L is defined as follows:

$$\text{ROUGE-L}_{rec} = \frac{\sum_{i=1}^{u} LCS_{\cup}(r_i, C)}{\sum_{i=1}^{u} |r_i|}$$

$$\text{ROUGE-L}_{prec} = \frac{\sum_{i=1}^{u} LCS_{\cup}(r_i, C)}{n}$$

Where $LCS_{\cup}(.,.)$ is the Longest common subsequence (LCS) score of the union of LCS between gold sentence r_i and the candidate summary C. ROUGE-L f score is the harmonic mean between precision and recall.

5.2 Comparison between summarizers

We generated two sets of summaries using the methods and baselines described in previous sections. We consider short summaries of length 100 words and longer summaries of length 250 words (which corresponds to the length threshold in gold summaries). We also considered the oracle's performance by averaging over the ROUGE scores of all human summaries calculated by considering one human summary against others in each topic. As far as 100 words summaries, since we did not have gold summaries of that length, we considered the first 100 words from each gold summary. Figure 2 shows the box-and-whisker plots with ROUGE scores. For each metric, the scores of each summarizer in comparison with the baselines for 100 word summaries and 250 words summaries are shown. The citation-context for all the methods were identified by the citation text vector method which uses the citation text except for numeric values, stop words and citation markers (first method in section 3.1). In section 5.3, we analyze the effect of various citation-context extraction methods that we discussed in section 3

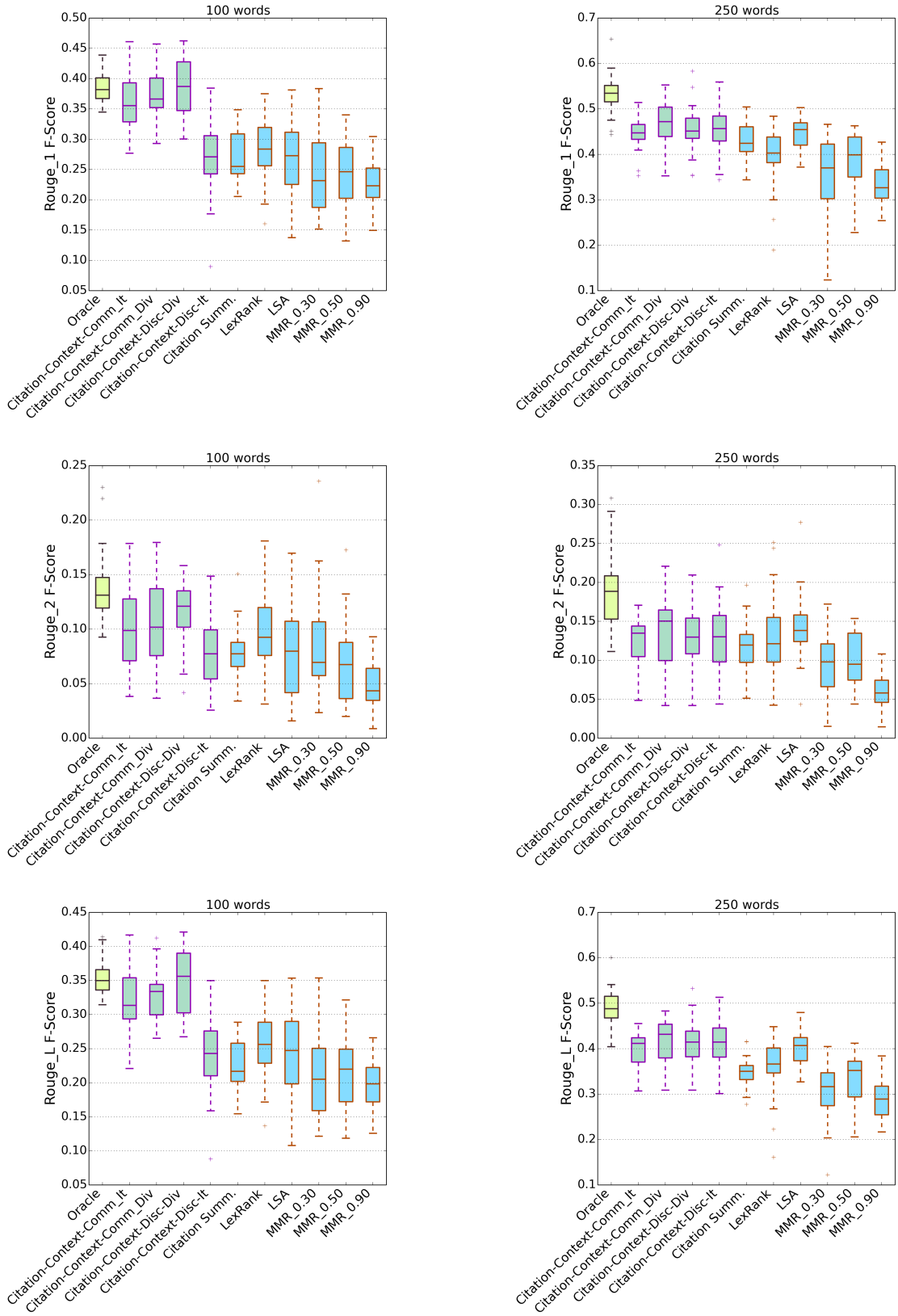


Figure 2: ROUGE-1, ROUGE-2 and ROUGE-L scores for different summarization approaches. Chartreuse (yellowish green) box shows the oracle, green boxes show the proposed summarizers and blue boxes show the baselines; From left, Oracle; Citation-Context-Comm-It: Community detection on citation-context followed by iterative selection; Citation-Context-Community-Div: Community detection on citation-context followed by relevance and diversification in sentence selection; Citation-Context-Discourse-Div: Discourse model on citation-context followed by relevance and diversification; Citation-Context-Discourse-It: Discourse model on citation-context followed by iterative selection; Citation Summ.: Citation summary; MMR_0.3: Maximal marginal relevance with $\lambda = 0.3$.

on the final summary. The name of each of our methods is shortened by the following convention: [Summarization approach]_[Sentence selection strategy]. Summarization approach is based on either community detection (Citation-Context-Comm) or discourse model of the article (Citation-Context-Disc) and sentence selection strategy can be iterative (It) or by relevance and diversification (Div).

We can clearly observe that our proposed methods achieve encouraging results in comparison with existing baselines. Specifically, for 100 words short summaries, the discourse based method (with 34.6% mean ROUGE-L improvement over the best baseline) and for 250 word summaries, the community based method (with 3.5% mean ROUGE-L improvement over the best baseline) are the best performing methods. We observe relative consistency between different rouge scores for each summarization approach. Grouping citation-context based on both the discourse structure and the communities show comparable results. The community detection approach is thus effectively able to identify diverse aspects of the article. The discourse model of the scientific article is also able to diversify selection of citation contexts for the final summary. These results confirm our hypotheses that using the citation context along with the discourse model of the scientific articles can help producing better summaries.

Comparison of performance of methods on individual topics showed that the citation-context methods consistently over perform all other methods in most of the topics (65% of all topics).

While the discourse approach shows encouraging results, we attribute its limitation in achieving higher ROUGE scores to the classification errors that we observed in intrinsic classification evaluation. In evaluating the performance of several classifiers, linear SVM achieved the highest performance with accuracy of 0.788 in comparison with human annotation performance. Many of the citations cannot exactly belong to only one of the discourse facets of the paper and thus some errors in classification are inevitable. This is also observable in disagreements between the annotators in labeling as reported by (Cohan et al., 2014). This fact influences the diversification and finally the summarization quality.

Among baseline summarization approaches, LexRank performs relatively well. Its performance is the best for short summaries among other baselines. This is expected since LexRank tries to find the most central sentences. When the length of the summary is short, the main idea in the summary is usually captured by finding the most representative sentence which LexRank can effectively achieve. However, the sentences that it chooses are usually about the same topic. Hence, the diversity in the gold summaries is not considered. This becomes more visible when we observe 250 word summaries. Our discourse based method can overcome this problem by including important contents for diverse discourse facets (34.6% mean ROUGE-L improvement for 100 words summaries and 13.9% improvement for 250 word summaries). The community based approach achieves the same diversification effect in an unsupervised fashion by forming citation-context communities (27.16% mean ROUGE-L improvement for 100 words summaries and 14.9% improvement for 250 word summaries).

The citation based summarization baseline has somewhat average performance among the baseline methods. This confirms that relying only on the citations can not be optimal for scientific summarization. While LSA approach performs relatively well, we observe lower scores for all variations of MMR approaches. We attribute the low performance of MMR to its sub optimal greedy selection of sentences from relatively long scientific articles.

By comparing the two sentence selection approaches (i.e., iterative and diversification-relevance), we observe that while for shorter length summaries the method based on diversification performs better, for the longer summaries results for the two methods are comparable. This is because when the length threshold is smaller, iterative approach may fail to select best representative sentences from all the groups. It essentially selects one sentence from each group until the length threshold is met, and consequently misses some aspects. Whereas, the diversification method selects sentences that maximize the gain in informativeness and at the same time contributes to the novelty of the summary. In longer summaries, due to larger threshold, iterative approach seems to be able

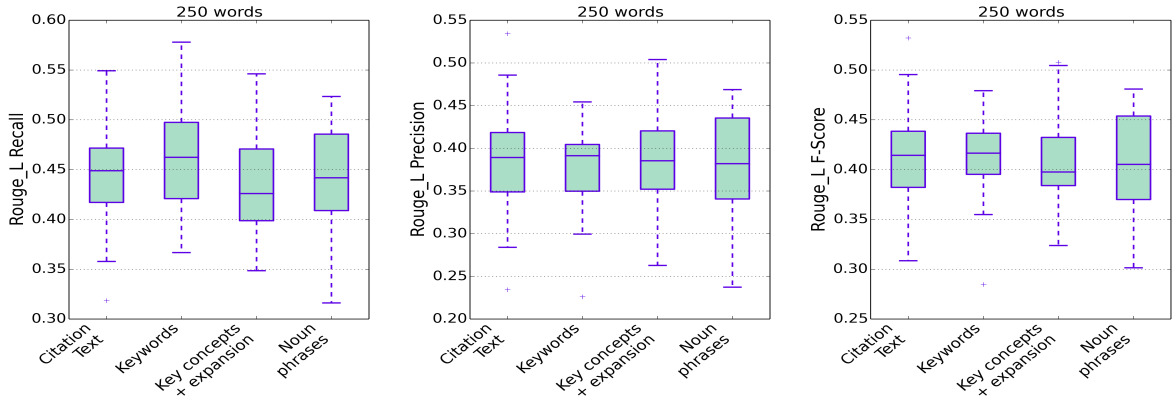


Figure 3: Comparison of the effect of different citation-context extraction methods on the quality of the final summary.

to select the top sentences from each group, enabling it to reflect different aspect of the paper. Therefore, the iterative approach performs comparably well to the diversification approach. This outcome is expected because the number of groups are small. For discourse method, there are 5 different discourse facets and for community method, on average 5.2 communities are detected. Hence, iterative selection can select sentences from most of these groups within 250 words limit summaries.

5.3 Analysis of strategies for citation-context extraction

Figure 3 shows ROUGE-L results for 250 words summaries based on using different citation-context extraction approaches, described in section 3.1. Relatively comparable performance for all the approaches is achieved. Using the citation text for extracting the context is almost as effective as other methods. Keywords approach which uses the terms with high idf values for locating the context achieves slightly higher Rouge-L precision while it has the lowest recall. This is expected since keywords approach chooses only informative terms for extracting citation-contexts. This results in missing terms that may not be keywords by themselves but help providing meaning. Noun phrases has the highest mean F-score and thus suggests the fact that noun phrases are good indicators of important concepts in scientific text. We attribute the high recall of noun phrases to the fact that most important concepts are captured by only selecting noun phrases. Interestingly, introducing biomedical concepts and expanding the citation vector by related concepts does not improve the performance. This approach achieves a relatively higher recall but a lower mean precision. While capturing domain concepts along with noun phrases helps improving the performance, adding related concepts to the citation vector causes drift from the original context as expressed in the reference article. Therefore some decline in performance is incurred.

6 Conclusion

We proposed a pipeline approach for summarization of scientific articles which takes advantage of the article's inherent discourse model and citation-contexts extracted from the reference article[1]. Our approach focuses on the problem of lack of context in existing citation based summarization approaches. We effectively achieved improvement over several well known summarization approaches on the TAC2014 biomedical summarization dataset. That is, in all cases we improved over the baselines; in some cases we obtained greater than 30% improvement for mean ROUGE scores over the best performing baseline. While the dataset we use for evaluation of scientific articles is in biomedical domain, most of our approaches are general and therefore adaptable to other scientific domains.

Acknowledgments

The authors would like to thank the three anonymous reviewers for their valuable feedback and comments. This research was partially supported by National Science Foundation (NSF) under grant CNS-1204347.

[1]Code can be found at: *https://github.com/acohan/scientific-summ*

References

Taylor Berg-Kirkpatrick, Dan Gillick, and Dan Klein. 2011. Jointly learning to extract and compress. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 481–490. Association for Computational Linguistics.

Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. 2008. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008.

Ulrik Brandes, Daniel Delling, Marco Gaertler, Robert Gorke, Martin Hoefer, Zoran Nikoloski, and Dorothea Wagner. 2008. On modularity clustering. *Knowledge and Data Engineering, IEEE Transactions on*, 20(2):172–188.

Jaime Carbonell and Jade Goldstein. 1998. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *SIGIR*, pages 335–336. ACM.

Asli Celikyilmaz and Dilek Hakkani-Tur. 2010. A hybrid hierarchical model for multi-document summarization. In *ACL*, pages 815–824. Association for Computational Linguistics.

Asli Celikyilmaz and Dilek Hakkani-Tür. 2011. Discovery of topically coherent sentences for extractive summarization. In *ACL: HLT-Volume 1*, pages 491–499. Association for Computational Linguistics.

Yllias Chali and Sadid a. Hasan. 2012. Query-focused multi-document summarization: Automatic data annotations and supervised learning approaches. *Nat. Lang. Eng.*, 18(1):109–145, January.

James Clarke and Mirella Lapata. 2008. Global inference for sentence compression an integer linear programming approach. *J. Artif. Int. Res.*, 31(1):399–429, March.

Arman Cohan, Luca Soldaini, and Nazli Goharian. 2014. Towards citation-based summarization of biomedical literature. Proceedings of the Text Analysis Conference (TAC '14).

Arman Cohan, Luca Soldaini, and Nazli Goharian. 2015. Matching citation text and cited spans in biomedical literature: a search-oriented approach. In *Proceedings of the 2015 NAACL-HLT*, pages 1042–1048. Association for Computational Linguistics.

John M Conroy, Judith D Schlesinger, Jeff Kubina, Peter A Rankel, and Dianne P OLeary. 2011. Classy 2011 at tac: Guided and multi-lingual summaries and evaluation metrics. In *Proceedings of the Text Analysis Conference*.

Anita De Waard and Henk Pander Maat. 2012. Epistemic modality and knowledge attribution in scientific discourse: A taxonomy of types and overview of features. In *Proceedings of the Workshop on Detecting Structure in Scholarly Discourse*, pages 47–55. Association for Computational Linguistics.

Aaron Elkiss, Siwei Shen, Anthony Fader, Güneş Erkan, David States, and Dragomir Radev. 2008. Blind men and elephants: What do citation summaries tell us about a research article? *Journal of the American Society for Information Science and Technology*, 59(1):51–62.

Günes Erkan and Dragomir R Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *J. Artif. Intell. Res.(JAIR)*, 22(1):457–479.

Michel Galley. 2006. A skip-chain conditional random field for ranking meeting utterances by importance. In *EMNLP*, pages 364–372. Association for Computational Linguistics.

Yihong Gong and Xin Liu. 2001. Generic text summarization using relevance measure and latent semantic analysis. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 19–25. ACM.

Shengbo Guo and Scott Sanner. 2010. Probabilistic latent maximal marginal relevance. In *SIGIR*, pages 833–834. ACM.

Aria Haghighi and Lucy Vanderwende. 2009. Exploring content models for multi-document summarization. In *NAACL-HLT*, pages 362–370. Association for Computational Linguistics.

Tsutomu Hirao, Yasuhisa Yoshida, Masaaki Nishino, Norihito Yasuda, and Masaaki Nagata. 2013. Single-document summarization as a tree knapsack problem. In *EMNLP*, pages 1515–1520.

Yuta Kikuchi, Tsutomu Hirao, Hiroya Takamura, Manabu Okumura, and Masaaki Nagata. 2014. Single document summarization based on nested tree structure. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, volume 2, pages 315–320.

Tibor Kiss and Jan Strunk. 2006. Unsupervised multilingual sentence boundary detection. *Computational Linguistics*, 32(4):485–525.

Ju-Hong Lee, Sun Park, Chan-Min Ahn, and Daeho Kim. 2009. Automatic generic document summarization based on non-negative matrix factorization. *Information Processing & Management*, 45(1):20 – 34.

Jiwei Li and Sujian Li. 2014. A novel feature-based bayesian model for query focused multi-document summarization. *Transactions of the Association for Computational Linguistics*, 1:89–98.

Jimmy Lin, Nitin Madnani, and Bonnie J Dorr. 2010. Putting the user in the loop: interactive maximal marginal relevance for query-focused summarization. In *NAACL-HLT*, pages 305–308. Association for Computational Linguistics.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, pages 74–81.

Annie Louis, Aravind Joshi, and Ani Nenkova. 2010. Discourse indicators for content selection in summarization. In *Proceedings of the 11th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 147–156. Association for Computational Linguistics.

Tengfei Ma and Hiroshi Nakagawa. 2013. Automatically determining a proper length for multi-document summarization: A bayesian nonparametric approach. In *EMNLP*, pages 736–746. Association for Computational Linguistics.

William C Mann and Sandra A Thompson. 1988. Rhetorical structure theory: Toward a functional theory of text organization. *Text*, 8(3):243–281.

Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into texts. Association for Computational Linguistics.

Mark EJ Newman. 2006. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, 103(23):8577–8582.

MEJ Newman. 2012. Communities, modules and large-scale structure in networks. *Nature Physics*, 8(1):25–31.

Miles Osborne. 2002. Using maximum entropy for sentence extraction. In *Proceedings of the ACL-02 Workshop on Automatic Summarization-Volume 4*, pages 1–8. Association for Computational Linguistics.

Makbule Gulcin Ozsoy, Ilyas Cicekli, and Ferda Nur Alpaslan. 2010. Text summarization of turkish texts using latent semantic analysis. In *COLING*, pages 869–876. Association for Computational Linguistics.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. The pagerank citation ranking: Bringing order to the web.

Michael Paul, ChengXiang Zhai, and Roxana Girju. 2010. Summarizing contrastive viewpoints in opinionated text. In *EMNLP*, pages 66–76. Association for Computational Linguistics.

Vahed Qazvinian and Dragomir R Radev. 2008. Scientific paper summarization using citation summary networks. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, pages 689–696. Association for Computational Linguistics.

Vahed Qazvinian, Dragomir R Radev, Saif Mohammad, Bonnie J Dorr, David M Zajic, Michael Whidby, and Taesun Moon. 2013. Generating extractive summaries of scientific paradigms. *J. Artif. Intell. Res.(JAIR)*, 46:165–201.

Cody Rioux, A. Sadid Hasan, and Yllias Chali. 2014. Fear the reaper: A system for automatic multi-document summarization with reinforcement learning. In *EMNLP*, pages 681–690. Association for Computational Linguistics.

Alan Ritter, Colin Cherry, and Bill Dolan. 2010. Unsupervised modeling of twitter conversations. In *NAACL-HLT*, HLT '10, pages 172–180, Stroudsburg, PA, USA. Association for Computational Linguistics.

Dou Shen, Jian-Tao Sun, Hua Li, Qiang Yang, and Zheng Chen. 2007. Document summarization using conditional random fields. In *IJCAI*, volume 7, pages 2862–2867.

Josef Steinberger and Karel Jezek. 2004. Using latent semantic analysis in text summarization and summary evaluation. In *Proc. ISIM04*, pages 93–100.

Josef Steinberger, Mijail A Kabadjov, Massimo Poesio, and Olivia Sanchez-Graillet. 2005. Improving lsa-based summarization with anaphora resolution. In *EMNLP-HLT*, pages 1–8. Association for Computational Linguistics.

Simone Teufel and Marc Moens. 2002. Summarizing scientific articles: Experiments with relevance and rhetorical status. *Comput. Linguist.*, 28(4):409–445, December.

Lucy Vanderwende, Hisami Suzuki, Chris Brockett, and Ani Nenkova. 2007. Beyond sumbasic: Task-focused summarization with sentence simplification and lexical expansion. *Information Processing & Management*, 43(6):1606–1618.

Kristian Woodsend and Mirella Lapata. 2012. Multiple aspect summarization using integer linear programming. In *EMNLP*, pages 233–243. Association for Computational Linguistics.

Shasha Xie and Yang Liu. 2010. Improving supervised learning for meeting summarization using sampling and regression. *Computer Speech & Language*, 24(3):495 – 514. Emergent Artificial Intelligence Approaches for Pattern Recognition in Speech and Language Processing.

Yasuhisa Yoshida, Jun Suzuki, Tsutomu Hirao, and Masaaki Nagata. 2014. Dependency-based discourse parser for single-document summarization. In *EMNLP*, pages 1834–1839, Doha, Qatar, October. Association for Computational Linguistics.

Ying Zhao and George Karypis. 2001. Criterion functions for document clustering: Experiments and analysis. Technical report, Citeseer.

Hashtag Recommendation Using Dirichlet Process Mixture Models Incorporating Types of Hashtags

Yeyun Gong, Qi Zhang, Xuanjing Huang
Shanghai Key Laboratory of Data Science
School of Computer Science, Fudan University
825 Zhangheng Road, Shanghai, P.R.China
{12110240006, qz, xjhuang}@fudan.edu.cn

Abstract

In recent years, the task of recommending hashtags for microblogs has been given increasing attention. Various methods have been proposed to study the problem from different aspects. However, most of the recent studies have not considered the differences in the types or uses of hashtags. In this paper, we introduce a novel nonparametric Bayesian method for this task. Based on the Dirichlet Process Mixture Models (DPMM), we incorporate the type of hashtag as a hidden variable. The results of experiments on the data collected from a real world microblogging service demonstrate that the proposed method outperforms state-of-the-art methods that do not consider these aspects. By taking these aspects into consideration, the relative improvement of the proposed method over the state-of-the-art methods is around 12.2% in F1- score.

1 Introduction

Hashtags are used to mark keywords or topics in a microblog. Over the past few years, social media services have become some of the most important communication channels for people. According to the statistic reported by the Pew Research Centers Internet & American Life Project in Aug 5, 2013, about 72% of adult internet users are also members of at least one social networking site. Hence, microblogs have also been widely used as data sources for public opinion analyses (Bermingham and Smeaton, 2010; Jiang et al., 2011), prediction (Asur and Huberman, 2010; Bollen et al., 2011), reputation management (Pang and Lee, 2008; Otsuka et al., 2012), and many other applications (Sakaki et al., 2010; Becker et al., 2010; Guy et al., 2010; Guy et al., 2013).

In addition to the limited number of characters in the content, microblogs also contain a form of metadata tag (hashtag), which is a string of characters preceded by the symbol (#). Hashtags are used to mark the keywords or topics of a microblog. They can occur anywhere in a microblog, at the beginning, middle, or end. Hashtags have been proven to be useful for many applications, including microblog retrieval (Efron, 2010), query expansion (A.Bandyopadhyay et al., 2011), and sentiment analysis (Davidov et al., 2010; Wang et al., 2011). However, only a small percentages of microblogs contain hashtags provided by their authors. Hence, the task of recommending hashtags for microblogs has become an important research topic and has received considerable attention in recent years. Existing works have studied discriminative models (Ohkura et al., 2006; Heymann et al., 2008) and generative models (Blei and Jordan, 2003; Krestel et al., 2009; Ding et al., 2013; Godin et al., 2013) based on the textual information of a single microblog.

Since microblog users are free to develop and use their own hashtags, they may select hashtags for different purposes. Based on an analysis of the hashtags crawled from a real online service, we observe that hashtags are used for events, conferences, conversation, disasters, memes, recall, quotes, and so on. To illustrate it let us take the following examples:

Example 1:*#Apple_iOS_9 includes music feature, new security and support for older iPhones.*

Example 2:*#BREAKING: Missing cyclist Natalie Donoghue has been found alive after she went missing in the Hunter Valley.*

We can see that the hashtag *#Apple_iOS_9* used in the example summarize the main topics of the corresponding microblog. While, the aim of hashtag *#BREAKING* in the example 2 is used as a label of the microblog. The different uses greatly

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 401–410,
Lisbon, Portugal, 17-21 September 2015.

impact the strategy of hashtag recommendation. However, there has been relatively few studies which take this issue into consideration.

In this paper, we propose a novel nonparametric Bayesian method to perform this problem. Inspired by the methods proposed by Liu et al. (2012), we assume that the hashtags and textual content in the corresponding microblog are parallel descriptions of the same thing in different languages. We adapt a translation model with topic distribution to achieve this task. Because of the ability of Dirichlet Process Mixture Models (DPMM) (Antoniak and others, 1974; Ferguson, 1983) to handle an unbounded number of topics, the proposed method is extended from them. Based on the different uses of hashtags, we incorporate the type of hashtag into the DPMM as a hidden variable.

The main contributions of this work can be summarized as follows:

- Through analyzing the microblogs, we propose the problem of influences of types of hashtags.
- We adopt a nonparametric Bayesian method to perform the hash tag recommendation task, which also takes the types of hashtags into consideration.
- Experimental results on the dataset we construct from a real microblogging service show that the proposed method can achieve significantly better performance than the state-of-the-arts methods.

2 The Proposed Method

In this section, we first give some brief descriptions about the Dirichlet process (DP) and Dirichlet Process Mixture Models (DPMM). Then, we detail the proposed hashtag recommendation method.

2.1 Preliminaries

2.1.1 Dirichlet Process

The Dirichlet process (DP) is a distribution over distributions. A DP, denoted by $G \sim DP(\alpha, H)$, is parameterized by a base measure H, and a concentration parameter α. After a discussion of basic definitions, we present two different perspectives on the Dirichlet process.

A perspective on the Dirichlet process is stick-breaking construction. The stick-breaking construction considers a probability mass function $\{\beta_k\}_{k=1}^{\infty}$ on a countably infinite set, where the discrete probabilities are defined as follows:

$$\begin{aligned} v_k|\alpha &\sim Beta(1,\alpha) \\ \beta_k &= v_k \prod_{l=1}^{k-1}(1-v_l). \end{aligned} \tag{1}$$

The k^{th} weight is a random proportion v_k of the remaining stick after the previous$(k-1)$ weights have been defined. This stick-breaking construction is generally denoted by $\beta \sim GEM(\alpha)$ (GEM stands for Griffiths, Engen and McCloskey). A random draw $G \sim DP(\alpha, H)$ can be expressed as:

$$G = \sum_{k=1}^{\infty} \beta_k \delta_{\theta_k} \quad \theta_k|\alpha, H \sim H, \tag{2}$$

where δ_θ is a probability measure concentrated at θ.

A second perspective on the Dirichlet process is provided by the *Pólya urn scheme* (Blackwell and MacQueen, 1973). It refers to draws from G. Let $\theta_1, \theta_2, ...$ represent a sequence of independent and identically distributed (i.i.d.) random variables distributed according to G. Blackwell and MacQueen (1973) showed that the conditional distributions of θ_i given $\theta_1, ..., \theta_{i-1}$ have the following form:

$$\begin{aligned} &\theta_i|\theta_1, ..., \theta_{i-1}, \alpha, H \\ &\sim \sum_{j=1}^{i-1} \frac{j}{i-1+\alpha}\delta_{\theta_j} + \frac{\alpha}{i-1+\alpha}H. \end{aligned} \tag{3}$$

Eq.(3) shows that θ_i has positive probability of being equal to one of the previous draws. We use $\phi_1, ..., \phi_K$ to represent the distinct values taken on by $\theta_1, ..., \theta_{i-1}$, and Eq.(3) can be re-expressed as:

$$\begin{aligned} &\theta_i|\theta_1, ..., \theta_{i-1}, \alpha, H \\ &\sim \sum_{k=1}^{K} \frac{m_k}{i-1+\alpha}\delta_{\theta_{\phi_k}} + \frac{\alpha}{i-1+\alpha}H, \end{aligned} \tag{4}$$

where m_k is the number of values $\theta_{i'} = \phi_k$ for $1 \leq i' < i$.

2.1.2 Dirichlet Process Mixture Models

In nonparametric Bayesian statistics, DPs are commonly used as prior distributions for mixture

models with an unknown number of components. Let $F(\theta_i)$ denotes the distribution of the observation x_i given θ_i. We can get the observation x_i as follows:

$$\theta_i|G \sim G$$
$$x_i|\theta_i \sim F(\theta_i).$$

Given $G \sim DP(\alpha, H)$, each observation x_i from an exchangeable data set $\mathbf{x}$ is generated by first choosing a parameter $\theta_i \sim G$, and then sampling $x_i \sim F(\theta_i)$. This model is referred to as a Dirichlet process mixture model. This process is often described by a set $\mathbf{z}$ of independently sampled variables $z_i \sim Mult(\beta)$ indicating the component of the mixture $G(\theta)$ associated with each data point $x_i \sim F(\theta_{z_i})$. Then we can get:

$$z_i|\beta \sim Mult(\beta)$$
$$x_i|\{\theta_k\}_{k=1}^{\infty}, z_i \sim F(\theta_{z_i}).$$

2.2 DPMM Based Hashtag Recommendation

2.2.1 The Generation Process

Let D represent the number of microblogs in the given corpus. A microblog contains a bag of words denoted by $w_d = \{w_{d_1}, w_{d_2}, ..., w_{d_{N_d}}\}$, where N_d is the total number of terms in the microblog. A word is defined as an item from a vocabulary with W distinct words indexed by $w = \{w_1, w_2, ..., w_W\}$. Each microblog may have a number of hashtags denoted by $h_d = \{h_{d_1}, h_{d_2}, ..., h_{d_{M_d}}\}$. M_d is the number of hashtags of microblog d. Each hashtag is from the vocabulary with V distinct hashtags indexed by $h = \{h_1, h_2, ..., h_V\}$. Given an unlabeled data set, the task of hashtag recommendation is to discover a list of hashtags for each microblog.

In standard LDA, each document is viewed as a mixture of topics, and each topic has probabilities to generate words. A LDA is a generalization of a finite mixture model. Since DP is the extension of finite mixture models to the nonparametric setting, the appropriate tool for nonparametric topic models is HDP. However, both LDA and HDP are normally suitable for long documents. For microblogs, which have limited number of words, a single microblog is most likely to talk about a single topic. Hence, in this work, we regard that each microblog associates with only one topic. The set of documents are viewed as a mixture of infinite topics. And we use DPs as prior distributions for the mixture of infinite topics.

The main assumptions of our model are as follows. When user u publishes a microblog, he will first generate the content and then generate the hashtags. When constructing the content, he will select a topic based on the topic distribution. Then he will choose a bag of words one by one from the word distribution of the topic or from the background words that captures white noise. Hashtags will be chosen according to the following two situations. In the first situation, hashtags summarize the corresponding microblogs. Hashtags of a microblog can be generated from the content through the topic-specific alignment probability between words and hashtags. In the second situation, hashtag is used as a label of the microblog. We recommend the hashtags using the words in the microblog, which is based on the frequency of words regarded as this type of hashtag.

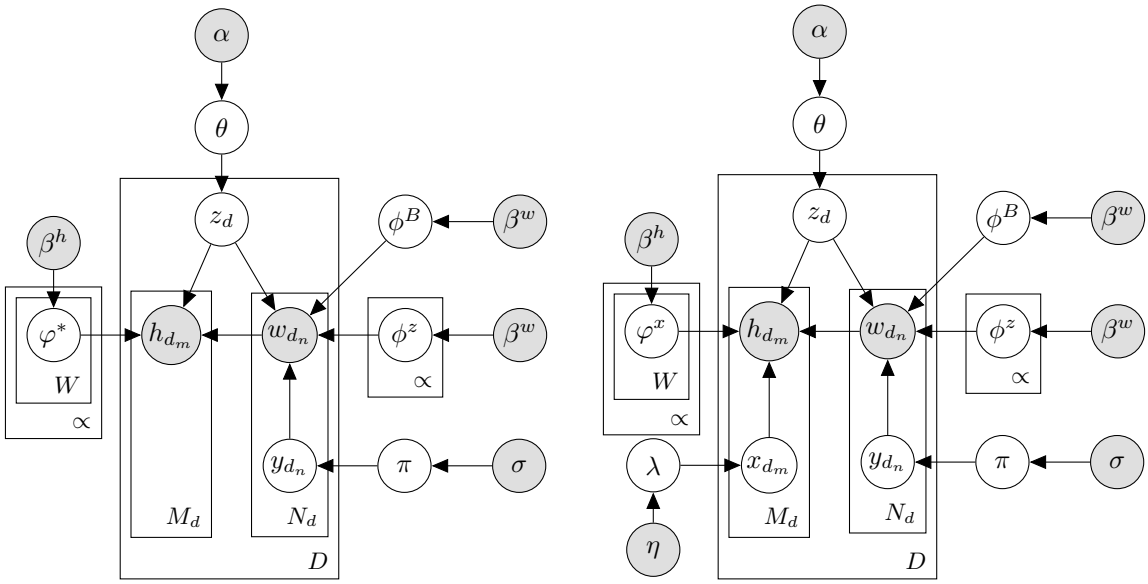
Let π be the probability of choosing a topic word or a background word, and we use $y_d = \{y_{d_n}\}_{n=1}^{N_d}$ to indicate a word to be a topic word or background word. θ denotes the topic distribution, and ϕ^k represents the word distribution for topic k. ϕ^B represents the word distribution for background words. We use x_{d_m} to represent the type of hashtag h_{d_m}, and use z_d to represent the topic of document d. Then each hashtag h_{d_m} is annotated according to the translation possibility $P(h_{d_m}|w_d, z_d, x_{d_m}, \varphi^{x_{d_m}})$, where $\varphi^{x_{d_m}}$ is the probability alignment table between words and hashtags. The generation process is as Algorithm 1.

Figure 1(a) shows the graphical representation of the generation process in Algorithm 1. Figure 1(b) is the graphical model which does not take the types of hashtags into consideration, where $\varphi^* \in \{\varphi^1, \varphi^2\}$. If $\varphi^* = \varphi^1$, the model is just considering the first situation. when $\varphi^* = \varphi^2$, only the second type of hashtag will be considered.

2.2.2 Learning

We use collapsed Gibbs sampling (Griffiths and Steyvers, 2004) to obtain samples of hidden variables assignment and to estimate the model parameters from these samples.

The sampling probability of being a topic/background word for the nth word in the microblog d can be calculated by the following



(a) CNHR (b) NHR*

Figure 1: The graphical representation of the proposed model. Shaded circles are observations or constants. Unshaded ones are hidden variables. *CNHR* represents the proposed hashtag recommendation method. *NHR** represents the model which does not take the types of hashtags into consideration.

Algorithm 1 The generation process of CNHR

Draw $\pi \sim Beta(\sigma)$, $\lambda \sim Beta(\eta)$
Draw background word distribution $\phi^B \sim Dir(\beta^w)$
Draw $\theta|\alpha \sim GEM(\alpha)$
for each microblog $d = 1, 2, ..., D$ **do**
  Draw $z_d \sim Mul(\theta)$
  Draw word distribution $\phi^{z_d} \sim Dir(\beta^w)$
  for each word $n = 1, ..., N_d$ **do**
    Draw $y_{d_n} \sim Ber(\pi)$
    if $y_{d_n} = 0$ **then**
      Draw a word w_{d_n} from the background-word distribution $w_{d_n} \sim Mul(\phi^B)$
    else
      Draw a word w_{d_n} from the topic-word distribution $w_{d_n} \sim Mul(\phi^{z_d})$
    end if
  end for
  for each hashtag $m = 1, ..., M_d$ **do**
    Draw $x_{d_m} \sim Ber(\lambda)$
    Draw $\varphi^{x_{d_m}} \sim Dir(\beta^h)$
    Draw a hashtag $h_{d_m} \sim P(h_{d_m}|w_d, z_d, x_{d_m}, \varphi^{x_{d_m}})$
  end for
end for

equation:

$$p(y_{d_n}|\mathbf{w}, \mathbf{h}, \mathbf{z}, \mathbf{y}_{\neg \mathbf{d_n}}, \sigma, \beta^w) \propto \frac{N_{\neg n,p} + \sigma}{N_{\neg n,(.)} + 2\sigma} \cdot \frac{N^{w_{d_n}}_{\neg n,l} + \beta^w}{N^{(.)}_{\neg n,l} + \beta^w W}, \quad (5)$$

where $l = B$ when $p = 0$ and $l = z_d$ when $p = 1$, $N_{\neg n,p}$ is a count of words that are assigned to background words and any topic respectively, $N^{w_{d_n}}_{\neg n,B}$ is the number of word w_{d_n} assigned to background words, $N^{w_{d_n}}_{\neg n,z_d}$ is the number of word w_{d_n} that are assigned to topic z_d. All counters are calculated with the current word w_{d_n} excluded.

We sample z_d for the microblog d using the following equation:

$$p(z_d|\mathbf{w}, \mathbf{h}, \mathbf{z}_{\neg \mathbf{d}}, \mathbf{y}, \mathbf{x}, \alpha, \beta^w, \beta^h) \propto p(z_d|\mathbf{z}_{\neg \mathbf{d}}, \alpha) \cdot p(\mathbf{w_d}|\mathbf{z}, \mathbf{w}_{\neg \mathbf{d}}, \mathbf{y}, \beta^w) \cdot p(\mathbf{h_d}|\mathbf{z}, \mathbf{w_d}, \mathbf{y}, \mathbf{x}, \beta^h). \quad (6)$$

We can also represent $p(z_d|\mathbf{z}_{\neg \mathbf{d}}, \alpha)$ with CRP as described in the previous section. Since $z_1, z_2, ...$ is a sequence of i.i.d random variables, they are exchangeable. Let us consider the dth variable z_d is the last observation, we can get the following expression:

$$p(z_d|\mathbf{z}_{\neg \mathbf{d}}, \alpha) \sim \sum_k^K \frac{N^k_{\neg d}}{N^{(.)}_{\neg d} - 1 + \alpha} \delta(z_d, k) + \frac{\alpha}{N^{(.)}_{\neg d} - 1 + \alpha} \delta(z_d, \bar{k}), \quad (7)$$

where k is an exist topic and $\bar{k}$ is a new topic, $N^k_{\neg d}$ is the number of microblogs assigned with topic k, $N^{(.)}_{\neg d}$ is the total number of microblogs, α is concentration parameter. All counters are

calculated with the current microblog d excluded.

If z_d equals an exist topic $z_d = k$, then we can calculate $p(\mathbf{w_d}|\mathbf{z}, \mathbf{w}_{\neg\mathbf{d}}, \mathbf{y}, \beta^w)$ by:

$$p(\mathbf{w_d}|\mathbf{z}, \mathbf{w}_{\neg\mathbf{d}}, \mathbf{y}, \beta^w) = \frac{\int_{\phi_k} \widetilde{f}(\mathbf{w_d}|\phi_k) \prod_{z_j=k, j\neq d} \widetilde{f}(\mathbf{w_j}|\phi_k) h(\phi_k) d\phi_k}{\int_{\phi_k} \prod_{z_j=k, j\neq d} \widetilde{f}(\mathbf{w_j}|\phi_k) h(\phi_k) d\phi_k}, \tag{8}$$

where $\widetilde{f}(\mathbf{w_d}|\phi_k) = \prod_{1\leq n\leq N_d, y_{d_n}=1} f(w_{d_n}|\phi_k)$. N_d is the number of words in microblog d. $f(w_{d_n}|\phi_k)$ is the density of word w_{d_n} given topic k. $\mathbf{w_d}$ are the words in microblog d. $h(\phi_k)$ is the density of base measure H.

If z_d is a new topic $z_d = \bar{k}$, then we can calculate $p(\mathbf{w_d}|z_d = \bar{k}, \mathbf{w}_{\neg\mathbf{d}}, \mathbf{y}, \beta^w)$ by:

$$p(\mathbf{w_d}|z_d = \bar{k}, \mathbf{w}_{\neg\mathbf{d}}, \mathbf{y}, \beta^w) = p(\mathbf{w_d}|\beta^w) = \int_{\phi_{\bar{k}}} \widetilde{p}(\mathbf{w_d}|\phi_{\bar{k}}) h(\phi_{\bar{k}}) d\phi_{\bar{k}}, \tag{9}$$

where $\widetilde{p}(\mathbf{w_d}|\phi_{\bar{k}}) = \prod_{1\leq n\leq N_d, y_{d_n}=1} p(w_{d_n}|\phi_{\bar{k}})$.

We can calculate the probabilities of generating hashtags from two situations as follows:

$$p(\mathbf{h_d}|\mathbf{z}, \mathbf{w_d}, \mathbf{y}, \mathbf{x}, \beta^h) = \begin{cases} \prod_{m=1}^{M_d} \sum_{n\in\widetilde{N}_d} \frac{M^{k,\neg d}_{w_{d_n}, h_{d_m}} + \beta^h}{M^{k,\neg d}_{w_{d_n},(.)} + \beta^h V} & x_{d_m} = 1 \\ \prod_{m=1}^{M_d} \sum_{n\in\widetilde{N}_d, w_{d_n}=h_{d_m}} \frac{M^{k,\neg d}_{w_{d_n},2} + \beta^h}{M^{k,\neg d}_{w_{d_n},(.)} + 2\beta^h} & x_{d_m} = 2, \end{cases} \tag{10}$$

where $\widetilde{N}_d$ represent the index set of topic words($y = 1$) in the microblog d, $M^{k,\neg d}_{w_{d_n}, h_{d_m}}$ is the number of occurrences that word w_{d_n} is translated to hashtag h_{d_m} given topic k, $M^{k,\neg d}_{w_{d_n},(.)}$ is the total number of occurrences that word w_{d_n} is under topic k, $M^{k,\neg d}_{w_{d_n},2}$ is the number of word w_{d_n} recommended as the second type of hashtag given topic k. All counters with $\neg d$ are calculated with the current microblog w_d excluded.

We sample the index variable x_{d_m} for mth hashtag in the microblog d as follows:

$$p(x_{d_m}|\mathbf{z}, \mathbf{w_d}, \mathbf{y}, \mathbf{x}_{\neg\mathbf{d_m}}, \mathbf{h}_d, \beta^h) \propto \sum_{n=1}^{\widetilde{N}_d} \varphi^{x_{d_m}}_{h_{d_m}, z_d, w_{d_n}} \frac{N^{\neg d_m}_{x_{d_m}} + \eta}{N^{\neg d_m}_{(.)} + 2\eta}, \tag{11}$$

where $N^{\neg d_m}_{x_{d_m}}$ is the number of hashtags that is generated by the type x_{d_m}, $N^{\neg d_m}_{(.)}$ is total number of hashtags, the counters with $\neg d_m$ are calculated with the current hashtag excluded.

After enough sampling iterations to burn in the Markov chain, φ^1 and φ^2 are estimated as follows:

$$\varphi^1_{h,k,w} = \frac{M^k_{w,h} + \beta^h}{M^k_{w,(.)} + \beta^h V}, \varphi^2_{h,k,w} = \frac{M^k_{w,2} + \beta^h}{M^k_{w,(.)} + 2\beta^h}, \tag{12}$$

The potential size of the probability alignment φ^1 between hashtag and word is $W \cdot V \cdot K$. The data sparsity may pose a more serious problem in estimating φ^1 than the topic-free word alignment case. We use interpolation smoothing technique for φ^1. In this paper, we employ smoothing as follows:

$$\varphi^{1*}_{h,k,w} = \gamma\varphi^1_{h,k,w} + (1-\gamma)P(h|w), \tag{13}$$

where $\varphi^{1*}_{h,k,w}$ is the smoothed topical alignment probabilities, $\varphi^1_{h,k,w}$ is the original topical alignment probabilities, $P(h|w)$ is topic-free word alignment probability. In this work, we obtain $P(h|w)$ by exploring IBM model-1 (Brown et al., 1993). γ is trade-off of two probabilities ranging from 0 to 1. When $\gamma = 0$, $\varphi^{1*}_{h,k,w}$ reduces to topic-free word alignment probability, and when $\gamma = 1$, there will be no smoothing in $\varphi^{1*}_{h,k,w}$.

2.2.3 Hashtag Recommendation

Suppose given an unlabeled dataset, we firstly discover the topic and determine topic/background words for each microblog. The collapsed Gibbs sampling is also applied for inference. The process is almost same as previous section described the model learning. The different is that there are no hashtags in the unlabeled dataset. Hence, when sampling z_d for the microblog d, we use the following equation:

$$p(z_d|\mathbf{w}, \mathbf{z}_{\neg\mathbf{d}}, \mathbf{y}, \alpha, \beta^w) \propto p(z_d|\mathbf{z}_{\neg\mathbf{d}}, \alpha) \cdot p(\mathbf{w_d}|\mathbf{z}, \mathbf{w}_{\neg\mathbf{d}}, \mathbf{y}, \beta^w). \tag{14}$$

Since there are no differences between the word alignments with each hashtags for a new topic in the unlabeled dataset, after the hidden variables of topic/background words and the topic of each microblog become stable, we only need to estimate the distribution of topics exist in the training dataset. Then we can estimate the distribution of topics for the microblog d in the unlabeled data by:

$$\chi_{dk} = \frac{p(k)p(w_{d_1}|k)p(w_{d_2}|k)...p(w_{d_{N_d}}|k)}{Z}, \tag{15}$$

where $p(w_{d_n}|k) = \frac{N_k^{w_{d_n}}+\beta}{N_k^{(\cdot)}+W\beta}$ and $N_k^{w_{d_n}}$ is a count of words w_{d_n} that are assigned to topic k in the corpus. And $p(k) = \frac{N_k}{N_{(\cdot)}+\alpha}$ is regarded as a prior for topic distribution, where Z is the normalized factor. With topic distribution χ and topic-specific word alignment table φ^*, we can rank hashtags for the microblog d in the unlabeled data through the following equation:

$$p(h_{d_m}|w_d, \chi_d, \varphi^*) \propto \sum_{z_d=1}^{K}\sum_{n=1}^{N_d}\sum_{x=1}^{C} p(z_d|\chi_d) \cdot p(w_{d_n}|w_d) \cdot p(x_{d_m}) \cdot p(h_{d_m}|w_{d_n}, z_d, x_{d_m}, \varphi^{x_{d_m}*}), \quad (16)$$

where C is the number of hashtag types. $p(w_{d_n}|w_d)$ is the weight of the word w_{d_n} in the microblog content w_d, which can be estimated by the IDF score of the word, $p(x_{d_m})$ is the probability of hashtag belong to the type x_{d_m}, we can estimate it with Eq.(11). Based on the ranking scores, we can suggest the top-ranked hashtags for each microblog.

3 Experiments

3.1 Data Collection

We use a dataset collected from Sina Weibo[1], which provides the Twitter-like service and is one of the most popular one in China, to evaluate the proposed approach and alternative methods. The original data set contains 282.2 million microblogs posted by around 1.1 million users. These microblogs were obtained by starting from a set of seed users and their follower/followee relations. We extract the microblogs posted with hashtags between Jan. 2012 and July 2013. Finally, 1,118,792 microblogs posted are selected for this work. The unique number of hashtags in the corpus is 305,227. We randomly select 100K as training data, 10K as development data, and 10K as test set. The hashtags marked in the original microblogs are considered as the golden standards.

3.2 Experiment Configurations

We use precision (P), recall (R), and F1-score (F_1) to evaluate the performance. Precision is calculated based on the percentage of "hashtags truly assigned" among "hashtags assigned by system". Recall is calculated based on the "hashtags truly assigned" among "hashtags manually assigned". F_1 is the harmonic mean of precision and recall. We do 500 iterations of Gibbs sampling to train the model. For optimizing the hyperparmeters of the proposed method and alternative methods, we use development data set to do it. In this work, the scale parameter α is set to $Gamma(5, 0.5)$. The other settings of hyperparameters are as follows: $\beta^w = 0.1$, $\beta^h = 0.1$, $\eta = 0.01$, and $\sigma = 0.01$. The smoothing factor γ in Eq.(13) is set to 0.8. For estimating the translation probability without topical information, we use GIZA++ 1.07 (Och and Ney, 2003) to do it.

Since hashtag recommendation task can also be modeled as a classification problem, we compare the proposed model with the following alternative methods:

- **Naive Bayes (NB)**: We formulate hashtag recommendation as a binary classification task and apply NB to model the posterior probability of each hashtag given a microblog.
- **Support Vector Machine (SVM)**: Similar to Naive Bayes, each hashtag can be regarded as one label and we use SVM to classify these microblogs.
- **Translation model (IBM-1)**: IBM model 1 is directly applied to obtain the alignment probability between the word and the hashtag (Liu et al., 2011).
- **Topical translation model (TTM)**: Ding et al. (2013) proposed the TTM for hashtag extraction. We implemented and extended their method for evaluating on the corpus constructed in this work. The number of topics in TTM is set to 20, and α is set to $50/K$. The hyperparameters used in TTM are also selected based on the development data set.

3.3 Experimental Results

Table 1 shows the comparisons of the proposed method with the state-of-the-art methods on the constructed evaluation dataset. "*CNHR*" denotes the method proposed in this paper. "*NHR1*" is a degenerate variation of CNHR, in which we consider all the hashtags are generated from distribution φ^1. "*NHR2*" is a model in which we consider all the hashtags are generated from

[1] http://www.weibo.com

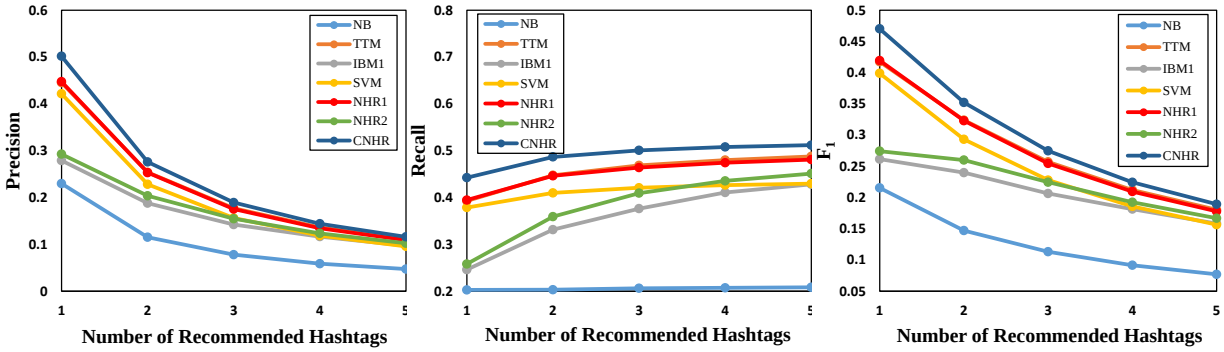


Figure 2: Precision, Recall and F_1 with recommended Hashtags range from 1 to 5

Table 1: Evaluation results of different methods on the evaluation collection.

Methods	Precision	Recall	F_1
NB	0.230	0.203	0.215
SVM	0.426	0.376	0.399
IBM1	0.279	0.246	0.261
TTM	0.445	0.393	0.417
NHR1	0.448	0.395	0.419
NHR2	0.293	0.258	0.274
CNHR	**0.502**	**0.442**	**0.470**

distribution φ^2. From the results, we can observe that discriminative methods achieve worse results than generative methods. We think that the large number of hashtags is one of the main reasons of the low performances.

From the results shown in Table 1, we also observe that the proposed method can achieve significantly better performance than existing methods. The relative improvement of proposed CNHR over TTM is around 12.7% in F_1. And we can see that the performances of TTM are similar as the results of NHR1. Since TTM and NHR1 are similar with each other except that TTM is based on LDA and NHR1 is adapted from DPMM. The results demonstrate the advantage of using DPMM over LDA. It does not need prior knowledge about number of topics. Comparing the results of the method CNHR with the methods NHR1 and NHR2 which do not take the types of hashtags into consideration, we can see that the proposed method benefits a lot from incorporating the types of hashtags.

Figure 2 shows the Precision, Recall, and F_1 curves of NB, IBM1, SVM, TTM, NHR1, NHR2 and CNHR on the test data. Each point of a curve represents the extraction of a different number of hashtags ranging from 1 to 5 respectively. In curves, the curve that is the highest of the graph indicates the best performance. Based on the results, we can observe that the performance of CNHR is the highest in all the curves. This indicates that the proposed method was significantly better than the other methods.

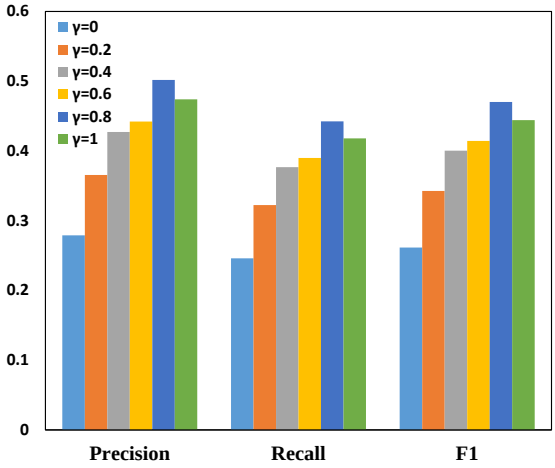
Table 2: The influence of the number of topics K of TTM.

K	Precision	Recall	F_1
10	0.441	0.389	0.413
20	**0.445**	**0.393**	**0.417**
30	0.432	0.381	0.405
40	0.413	0.364	0.387
50	0.391	0.345	0.367

In TTM, the number of topics K is also crucial factor. Table 2 shows the impact of the number of topics. From the table, we can observe that TTM obtains the best performance when K is set to 20. And performance decreases with more number of topics. We think that data sparsity may be one of the main reasons. With much more topic number, the data sparsity problem will be more serious when estimating topic-specific translation probability. We compare our method with the best performance of TTM.

From the description of the proposed model, we can know that there is a smooth parameter γ in the proposed method CNHR. To evaluate the impact of it, Figure 3 shows the influence of the translation probability smoothing parameter γ. When γ is set to 0.0, it means that the topical information is omitted. Comparing the results of $\gamma = 0.0$ and other values, we can observe that the topical information can benefit this task.

Figure 3: The influence of the smoothing parameter γ of CNHR.



When γ is set to 1.0, it represents the method without smoothing. The results indicate that it is necessary to address the sparsity problem through smoothing.

4 Related Works

Due to the usefulness of tag recommendation, many methods have been proposed from different perspectives (Heymann et al., 2008; Krestel et al., 2009; Rendle et al., 2009; Liu et al., 2012; Ding et al., 2013). Heymann et al. (Heymann et al., 2008) investigated the tag recommendation problem using the data collected from social bookmarking system. They introduced an entropy-based metric to capture the generality of a particular tag. In (Song et al., 2008), a Poisson Mixture Model based method is introduced to achieve the tag recommendation task. Krestel et al. (Krestel et al., 2009) introduced a Latent Dirichlet Allocation to elicit a shared topical structure from the collaborative tagging effort of multiple users for recommending tags. Ding et al. (2013) proposed to use translation process to model this task.

Based on the the observation that similar web pages tend to have the same tags, Lu et al. (2009) proposed a method taking both tag information and page content into account to achieve the task. They extended the translation based method and introduced a topic-specific translation model to process the various meanings of words in different topics. In (Tariq et al., 2013), discriminative-term-weights were used to establish topic-term relationships, of which users' perception were learned to suggest suitable hashtags for users. To handle the vocabulary problem in keyphrase extraction task, Liu et al. proposed a topical word trigger model, which treated the keyphrase extraction problem as a translation process with latent topics (Liu et al., 2012).

Most of the works mentioned above are based on textual information. Besides these methods, personalized methods for different recommendation tasks have also been paid lots of attentions (Liang et al., 2007; Shepitsen et al., 2008; Garg and Weber, 2008; Li et al., 2010; Liang et al., 2010; Rendle and Schmidt-Thieme, 2010; Huang et al., 2012). Shepitsen et al. (2008) proposed to use hierarchical agglomerative clustering to take into account personalized navigation context in cluster selection. In (Garg and Weber, 2008), the problem of personalized, interactive tag recommendation was also studied based on the statistics of the tags co-occurrence. Liang et al. (2010) proposed to the multiple relationships among users, items and tags to find the semantic meaning of each tag for each user individually and used this information for personalized item recommendation.

From the brief descriptions given above, we can observe that most of the previous works on hashtag suggestion did not take the types of hashtags into consideration. In this work, we propose to incorporate it into the generative methods.

5 Conclusions

In this paper, we study the problem of hashtag recommendation for microblogs. Since existing translation model based methods for this task regard all the hashtags generated from the same distribution, we propose a novel method which incorporates different type of hashtags have different distribution into the topical translation model for hashtag recommendation task. To evaluate the proposed method, we also construct a dataset from real world microblogging services. The results of experiments on the constructed dataset demonstrate that the proposed method outperforms state-of-the-art methods that do not consider these aspects.

6 Acknowledgement

The authors wish to thank the anonymous reviewers for their helpful comments. This work was par-

tially funded by National Natural Science Foundation of China (No. 61473092 and 61472088), the National High Technology Research and Development Program of China (No. 2015AA011802), and Shanghai Science and Technology Development Funds (13dz226020013511504300).

References

A.Bandyopadhyay, M. Mitra, and P. Majumder. 2011. Query expansion for microblog retrieval. In *Proceedings of The Twentieth Text REtrieval Conference*, TREC 2011.

Charles E Antoniak et al. 1974. Mixtures of dirichlet processes with applications to bayesian nonparametric problems. *The annals of statistics*, 2(6):1152–1174.

S. Asur and B.A. Huberman. 2010. Predicting the future with social media. In *WI-IAT'10*, volume 1, pages 492–499.

Hila Becker, Mor Naaman, and Luis Gravano. 2010. Learning similarity metrics for event identification in social media. In *Proceedings of WSDM '10*.

Adam Bermingham and Alan F. Smeaton. 2010. Classifying sentiment in microblogs: is brevity an advantage? In *Proceedings of CIKM'10*.

David Blackwell and James B MacQueen. 1973. Ferguson distributions via pólya urn schemes. *The annals of statistics*, pages 353–355.

D.M. Blei and M.I. Jordan. 2003. Modeling annotated data. In *Proceedings of SIGIR*, pages 127–134.

Johan Bollen, Huina Mao, and Xiaojun Zeng. 2011. Twitter mood predicts the stock market. *Journal of Computational Science*, 2(1):1 – 8.

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

Dmitry Davidov, Oren Tsur, and Ari Rappoport. 2010. Enhanced sentiment learning using twitter hashtags and smileys. In *Proceedings of COLING '10*.

Zhuoye Ding, Xipeng Qiu, Qi Zhang, and Xuanjing Huang. 2013. Learning topical translation model for microblog hashtag suggestion. In *Proceedings of IJCAI 2013*.

Miles Efron. 2010. Hashtag retrieval in a microblogging environment. In *Proceedings of SIGIR '10*.

Thomas S Ferguson. 1983. Bayesian density estimation by mixtures of normal distributions. *Recent advances in statistics*, 24:287–302.

Nikhil Garg and Ingmar Weber. 2008. Personalized, interactive tag recommendation for flickr. In *Proceedings of RecSys '08*.

Fréderic Godin, Viktor Slavkovikj, Wesley De Neve, Benjamin Schrauwen, and Rik Van de Walle. 2013. Using topic models for twitter hashtag recommendation. In *Proceedings of the 22Nd International Conference on World Wide Web Companion*, WWW '13 Companion, pages 593–596, Republic and Canton of Geneva, Switzerland. International World Wide Web Conferences Steering Committee.

T. L. Griffiths and M. Steyvers. 2004. Finding scientific topics. *Proceedings of the National Academy of Sciences*.

Ido Guy, Naama Zwerdling, Inbal Ronen, David Carmel, and Erel Uziel. 2010. Social media recommendation based on people and tags. In *Proceedings of SIGIR '10*.

Ido Guy, Uri Avraham, David Carmel, Sigalit Ur, Michal Jacovi, and Inbal Ronen. 2013. Mining expertise and interests from social media. In *Proceedings of WWW '13*.

Paul Heymann, Daniel Ramage, and Hector Garcia-Molina. 2008. Social tag prediction. In *Proceedings of SIGIR '08*.

Wenyi Huang, Saurabh Kataria, Cornelia Caragea, Prasenjit Mitra, C Lee Giles, and Lior Rokach. 2012. Recommending citations: translating papers into references. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 1910–1914. ACM.

Long Jiang, Mo Yu, Ming Zhou, Xiaohua Liu, and Tiejun Zhao. 2011. Target-dependent twitter sentiment classification. In *Proceedings of ACL 2011*, Portland, Oregon, USA.

Ralf Krestel, Peter Fankhauser, and Wolfgang Nejdl. 2009. Latent dirichlet allocation for tag recommendation. In *Proceedings of RecSys '09*.

Lihong Li, Wei Chu, John Langford, and Robert E Schapire. 2010. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*, pages 661–670. ACM.

Ting-Peng Liang, Hung-Jen Lai, and Yi-Cheng Ku. 2007. Personalized content recommendation and user satisfaction: Theoretical synthesis and empirical findings. *Journal of Management Information Systems*, 23(3):45–70.

Huizhi Liang, Yue Xu, Yuefeng Li, Richi Nayak, and Xiaohui Tao. 2010. Connecting users and items with weighted tags for personalized item recommendations. In *Proceedings of the 21st ACM conference on Hypertext and hypermedia*, pages 51–60. ACM.

Zhiyuan Liu, Xinxiong Chen, and Maosong Sun. 2011. A simple word trigger method for social tag suggestion. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1577–1588. Association for Computational Linguistics.

Zhiyuan Liu, Chen Liang, and Maosong Sun. 2012. Topical word trigger model for keyphrase extraction. In *Proceedings of COLING*.

Yu-Ta Lu, Shoou-I Yu, Tsung-Chieh Chang, and Jane Yung-jen Hsu. 2009. A content-based method to enhance tag recommendation. In *Proceedings of IJCAI'09*.

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

Tsutomu Ohkura, Yoji Kiyota, and Hiroshi Nakagawa. 2006. Browsing system for weblog articles based on automated folksonomy. *Workshop on the Weblogging Ecosystem Aggregation Analysis and Dynamics at WWW*.

Takanobu Otsuka, Takuya Yoshimura, and Takayuki Ito. 2012. Evaluation of the reputation network using realistic distance between facebook data. In *Proceedings of WI-IAT '12*.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Found. Trends Inf. Retr.*, 2(1-2):1–135, January.

Steffen Rendle and Lars Schmidt-Thieme. 2010. Pairwise interaction tensor factorization for personalized tag recommendation. In *Proceedings of the third ACM international conference on Web search and data mining*, pages 81–90. ACM.

Steffen Rendle, Leandro Balby Marinho, Alexandros Nanopoulos, and Lars Schmidt-Thieme. 2009. Learning optimal ranking with tensor factorization for tag recommendation. In *Proceedings of KDD '09*.

Takeshi Sakaki, Makoto Okazaki, and Yutaka Matsuo. 2010. Earthquake shakes twitter users: real-time event detection by social sensors. In *Proceedings of WWW '10*.

Andriy Shepitsen, Jonathan Gemmell, Bamshad Mobasher, and Robin Burke. 2008. Personalized recommendation in social tagging systems using hierarchical clustering. In *Proceedings of the 2008 ACM Conference on Recommender Systems*, RecSys '08, pages 259–266, New York, NY, USA. ACM.

Yang Song, Ziming Zhuang, Huajing Li, Qiankun Zhao, Jia Li, Wang-Chien Lee, and C. Lee Giles. 2008. Real-time automatic tag recommendation. In *Proceedings of SIGIR '08*.

Amara Tariq, Asim Karim, Fernando Gomez, and Hassan Foroosh. 2013. Exploiting topical perceptions over multi-lingual text for hashtag suggestion on twitter. In *FLAIRS Conference*.

Xiaolong Wang, Furu Wei, Xiaohua Liu, Ming Zhou, and Ming Zhang. 2011. Topic sentiment analysis in twitter: a graph-based hashtag sentiment classification approach. In *Proceedings of CIKM '11*.

A Graph-based Readability Assessment Method using Word Coupling

Zhiwei Jiang, Gang Sun, Qing Gu*, Tao Bai, Daoxu Chen
State Key Laboratory for Novel Software Technology
Nanjing University, Nanjing 210023, China
jiangzhiwei@outlook.com, sungangnju@163.com,
guq@nju.edu.cn, bt@xjau.edu.cn, cdx@nju.edu.cn

Abstract

This paper proposes a graph-based readability assessment method using word coupling. Compared to the state-of-the-art methods such as the readability formulae, the word-based and feature-based methods, our method develops a coupled bag-of-words model which combines the merits of word frequencies and text features. Unlike the general bag-of-words model which assumes words are independent, our model correlates the words based on their similarities on readability. By applying TF-IDF (Term Frequency and Inverse Document Frequency), the coupled TF-IDF matrix is built, and used in the graph-based classification framework, which involves graph building, merging and label propagation. Experiments are conducted on both English and Chinese datasets. The results demonstrate both effectiveness and potential of the method.

1 Introduction

Readability assessment is a task that aims to evaluate the reading difficulty or comprehending easiness of text documents. It is helpful for educationists to select texts appropriate to the reading/grade levels of the students, and for web designers to organize texts on web pages for the users doing personalized searches for information retrieval.

Research on readability assessment starts from the early 20th century (Dale and Chall, 1948). Many useful readability formulae have been developed since then (Dale and Chall, 1948; McLaughlin, 1969; Kincaid et al., 1975). Currently, due to the development of natural language processing, the methods on readability assessment have made a great progress (Zakaluk and Samuels, 1988; Benjamin, 2012; Gonzalez-Dios et al., 2014). The word-based methods compute word frequencies in documents to estimate their readability (Collins-Thompson and Callan, 2004; Kidwell et al., 2009). The feature-based methods extract text features from documents and train classification models to classify the readability (Schwarm and Ostendorf, 2005; Feng et al., 2010; François and Fairon, 2012; Hancke et al., 2012).

In this paper, we propose a graph-based method using word coupling, which combines the merits of both word frequencies and text features for readability assessment. We design a coupled bag-of-words model, which correlates words based on their similarities on sentence-level readability computed using text features. The model is used in a graph-based classification framework, which involves graph building, graph merging/combination, and label propagation. We perform experiments on datasets of both English and Chinese. The results demonstrate both effectiveness and potential of our method.

The rest of this paper is organized as follows: Section 2 introduces backgrounds of our work. Section 3 presents the details of the method. Section 4 designs the experiments and explains the results. Finally, Section 5 concludes the paper with planned future work.

2 Background

In this section, we introduce briefly three research topics relevant to our work: readability assessment, the bag-of-words model and the graph-based label propagation method.

2.1 Readability Assessment

Research on readability assessment has developed three types of methods: the readability formula, the word-based methods and the feature-based methods (Kincaid et al., 1975; Collins-Thompson and Callan, 2004; Schwarm and Os-

*Corresponding author.

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 411–420,
Lisbon, Portugal, 17-21 September 2015.

tendorf, 2005). During the early time, many well-known readability formulae have been developed to assess the readability of text documents (Dale and Chall, 1948; McLaughlin, 1969; Kincaid et al., 1975). Surface text features are defined in these formulae to measure both lexical and grammatical complexities of a document. The word-based methods focus on words and their frequencies in a document to assess its readability, which mainly include the unigram/bigram/n-gram models (Collins-Thompson and Callan, 2004; Schwarm and Ostendorf, 2005) and the word acquisition model (Kidwell et al., 2009). The feature-based methods focus on extracting text features from a document and training a classification model to classify its readability (Feng et al., 2010; François and Fairon, 2012; Hancke et al., 2012). Suitable text features are usually essential to the success of these methods. The Support vector machine and logistic regression model are two classification models commonly used in these methods.

2.2 The Bag-of-Words Model

The bag-of-words model is mostly used for document classification. It constructs a feature space that contains all the distinct words in a language (or the document set). A document is represented by a vector, whose components reflect the weight of every distinct word contained in the document. Normally, it assumes the words are independent. Now the capturing of the relationship among words has attracted considerable attention (Wong et al., 1985; Cheng et al., 2013). Inspired by these works, this paper adopts the bag-of-words model in readability assessment, and refines the model by computing similarity among words on reading difficulty.

2.3 The Graph-based Label Propagation Method

Graph-based label propagation is applied on a graph to propagate class labels from labeled nodes to unlabeled ones (Kim et al., 2013). It has been successfully applied in various applications, such as dictionary construction (Kim et al., 2013), word segmentation and tagging (Zeng et al., 2013), and sentiment classification (Ponomareva and Thelwall, 2012). Typically, a graph-based label propagation method consists of two main steps: graph construction and label propagation (Zeng et al., 2013). During the first step, a similarity function is required to build edges and compute weights between pairs of the nodes (Daitch et al., 2009). Some form of edge pruning is required to refine the graph (Jebara et al., 2009). After that, effective algorithms have been developed to propagate the label distributions to all the nodes (Subramanya et al., 2010; Kim et al., 2013).

3 The Proposed Method

In this section, we present GRAW (Graph-based Readability Assessment method using Word coupling), which constructs a coupled bag-of-words model by exploiting the correlation of readability among the words. Unlike the general bag-of-words model which models document relationship on topic, the coupled bag-of-words model extends it to model the relationship among documents on readability. In the following sections, we describe in detail how to build the coupled bag-of-words model. The model is then used in the graph-based classification framework for readability assessment.

3.1 The General Bag-of-Words Model

TF-IDF (Term Frequency and Inverse Document Frequency) is the most popular scheme of the bag-of-words model. Given the set of documents $\mathcal{D}$, the TF-IDF matrix M can be calculated based on the logarithmically scaled term (i.e. word) frequency (Salton and Buckley, 1988) as follows.

$$\begin{aligned} M_{t,d} &= tf_{t,d} \cdot idf_{t,d} \\ &= (1 + \log f(t,d)) \cdot \log \frac{|\mathcal{D}|}{|\{d|t \in d\}|} \end{aligned} \quad (1)$$

where $f(t,d)$ is the number of times that a term (word) t occurs in a document $d \in \mathcal{D}$.

3.2 The Coupled Bag-of-Words Model

As shown in Figure 1, three main stages are required to construct the coupled bag-of-words model: per-sentence readability estimation, word coupling matrix construction and coupled TF-IDF matrix calculation. The following sections describe the details of these stages.

3.2.1 Per-Sentence Readability Estimation

Two steps are required for the per-sentence readability estimation. The first is to compute a reading score of a sentence by heuristic functions. The second is to determine the difficulty level of the sentence by discretizing the score.

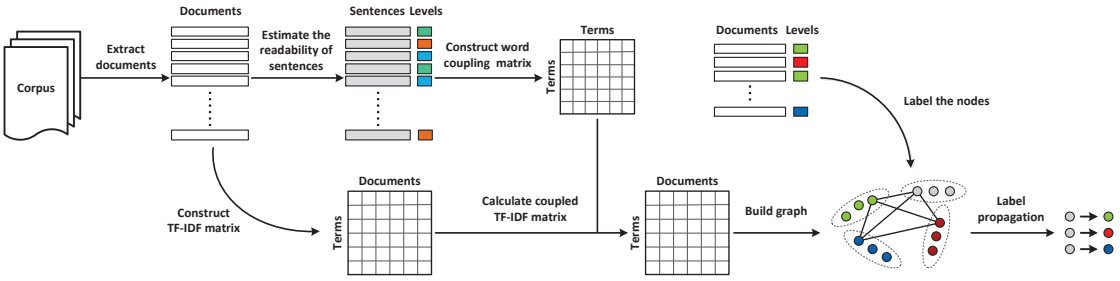


Figure 1: The Framework of GRAW

Step 1. Given a sentence s, its reading difficulty can be quantified as a reading score which is a continuous variable denoted by $r(s)$. The more difficult s is, the greater $r(s)$ will be. Based on text features of s, $r(s)$ can be computed by one of the eight heuristic functions listed in Table 1 which are grouped into three aspects.

Aspect	Function	Description
Surface	$len(s)$	the length of the sentence s.
	$ans(s)$	the average number of syllables (or strokes for Chinese) per word (or character for Chinese) in s.
	$anc(s)$	the average number of characters per word in s.
Lexical	$lv(s)$	the number of distinct types of POS, i.e. part of speech, in s.
	$atr(s)$	the ratio of adjectives in s.
	$ntr(s)$	the ratio of nouns in s.
Syntatic	$pth(s)$	the height of the syntax parser tree of s.
	$anp(s)$	the average number of (noun, verb, and preposition) phrases in s.

Table 1: Three aspects of estimating reading difficulty of sentences using heuristic functions

Step 2. Let η denote the pre-determined number of difficulty levels, r_{max} and r_{min} denote the maximum and minimum reading score respectively of all the sentences in $\mathcal{D}$. To determine the difficulty level $l^*(s)$ ($l^*(s) \in [1, \eta]$) of a sentence s, the range $[r_{min}, r_{max}]$ is divided into η intervals, so that each interval contains the reading scores of $\frac{1}{\eta}$ of all the sentences. The assumption is that all the sentences are equally distributed among the difficulty levels. $l^*(s)$ will be i, if the reading score $r(s)$ resides in the i-th interval.

For each of the three aspects, we compute one $l^*(s)$ for a sentence s by combining the heuristic functions using the following equations. The assumption is that the reading difficulty of a sentence may be determined by the maximum measure on the text features.

$$\begin{aligned} l^{sur}(s) &= \max\left[l^{len}(s), l^{ans}(s), l^{anc}(s)\right] \\ l^{lex}(s) &= \max\left[l^{lv}(s), l^{atr}(s), l^{ntr}(s)\right] \\ l^{syn}(s) &= \max\left[l^{pth}(s), l^{anp}(s)\right] \end{aligned} \tag{2}$$

3.2.2 Word Coupling Matrix Construction

Let $\mathcal{V}$ denote the set of all the words, a word coupling matrix is defined as $C^* \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$, the element of which reflects the correlation between two words (i.e. terms). Two steps are required to construct this matrix. The first is to count the difficulty distributions of words, and the second is to compute the correlation between each pair of words according to the similarity of their difficulty distributions.

Step 1. Let $\mathcal{S}$ denote the set of all the sentences, p_t denote the difficulty distribution of a word (term) t. p_t is a vector containing η (i.e. the number of difficulty levels) values, the i-th part of which can be calculated by the following formula.

$$p_t(i) = \frac{1}{n_t} \cdot \sum_{s \in \mathcal{S}} \delta(t \in s) \cdot \delta(l^*(s) = i) \tag{3}$$

where n_t refers to the number of sentences in which t appears. The indicator function $\delta(x)$ returns 1 if x is true and 0 otherwise. $l^*(s)$ refers to one of the functions $l^{sur}(s)$, $l^{lex}(s)$ or $l^{syn}(s)$.

Step 2. Given two words (terms) t_1 and t_2, whose level distributions are p_{t_1} and p_{t_2} respectively, we measure the distribution difference $c_{KL}(t_1, t_2)$ using the Kullback-Leibler divergence (Kullback and Leibler, 1951), computed by the following formula.

$$c_{KL}(t_1, t_2) = \frac{1}{2}\left(KL(p_{t_1}||p_{t_2}) + KL(p_{t_2}||p_{t_1})\right) \tag{4}$$

where $KL(p||q) = \sum_i p(i) \log \frac{p(i)}{q(i)}$. After that, the logistic function is applied on the computed difference to get the normalized distribution similarity, i.e.

$$sim(t_1, t_2) = \frac{2}{1 + e^{c_{KL}(t_1, t_2)}} \tag{5}$$

Given a word t_i, only λ other words with highest correlation (similarity) are selected to build the

neighbor set of t_i, denoted as $\mathcal{N}(t_i)$. If a word t_j is not selected (i.e. $t_j \notin \mathcal{N}(t_i)$), the corresponding $sim(t_i, t_j)$ will be assigned 0. After that, the word coupling matrix (i.e. C^*) with $sim(t_i, t_j)$ as elements is normalized along the rows so that the sum of each row is 1. Based on three different $l^*(s)$, we construct three word coupling matrices C^{sur}, C^{lex} and C^{syn}.

3.2.3 Coupled TF-IDF Matrix Calculation

In the general bag-of-words model, the words are treated as independent of each other. However, for readability assessment, words may be correlated according to the similarity of their difficulty distributions. To improve the TF-IDF matrix M described in Section 3.1, we multiply it by the word coupling matrix C^*, so that the term frequencies are shared among the highly correlated (coupled) words. We denote the coupled TF-IDF matrix as M^*, obtained by the following formula.

$$M^* = C^* \cdot M \tag{6}$$

Specifically, three homogenous coupled TF-IDF matrices M^{sur}, M^{lex} and M^{syn} can be built according to the three word coupling matrices C^*.

3.3 Graph-based Readability Assessment

We employ the coupled bag-of-words model for readability assessment under the graph-based classification framework as described in the previous work (Zhu and Ghahramani, 2002). Firstly, we construct a graph representing the readability relationship among documents by using the coupled bag-of-words model to compute the relations among these documents. Secondly, we estimate reading levels of documents by applying label propagation on the graph.

3.3.1 Graph Construction

We build a directed graph G^* to represent the readability relation among documents, where nodes represent documents, and edges are weighted by the similarities between pairs of documents. Given a similarity function, we link documents d_i to d_j with an edge of weight G^*_{ij}, defined as:

$$G^*_{i,j} = \begin{cases} sim(d_i, d_j) & \text{if } d_j \in \mathcal{N}(d_i) \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

where $\mathcal{N}(d_i)$ is the set of k-nearest neighbors of d_i determined by the similarity function.

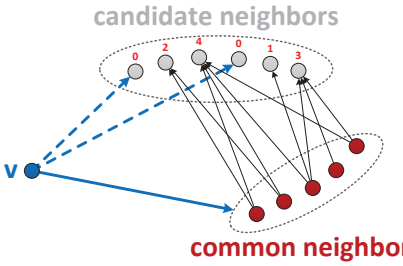


Figure 2: Illustration of the graph merging strategy

Given the coupled matrix $M^* \in \mathbb{R}^{m \times |\mathcal{D}|}$ which maps each document into a m-dimension feature space, the similarity function $sim(d_i, d_j)$ can be defined by the Euclidean distance as follows.

$$sim(d_i, d_j) = \frac{1}{\sqrt{\sum_{k=1}^{m} (M_{k,i} - M_{k,j})^2 + \epsilon}} \tag{8}$$

where ϵ is a small constant to avoid zero denominators.

Merge the three graphs Refer to Section 3.2, the three coupled TF-IDF matrices will lead to three different document graphs, denoted as G^{sur}, G^{lex} and G^{syn} respectively. To take advantage of the three aspects at one time, we need to merge the three graphs into one, denoted as G^c.

In G^c, each node also keeps k neighbors, and some edges shall be filtered out from the three graphs. The basic idea is to remove edges containing redundant information, as shown in Figure 2. For each node v, we firstly select the neighbors which are common in all the three graphs (i.e. $\mathcal{N}^{sur}(v) \cap \mathcal{N}^{lex}(v) \cap \mathcal{N}^{syn}(v)$). Secondly, for the rest candidate nodes, which are the neighbors of v in at least one graph, we select one by one the node which possesses the least number of common neighbors (from all the three graphs) with the nodes that are already selected in $\mathcal{N}^c(v)$. The objective is to keep the number of triangles in G^c to a minimum. The edge weights of G^c are averaged on the corresponding edges appeared in the three graphs.

Combine with the feature-based graph Previous studies usually extract text features from documents to assess the readability using classification models. Here, we also take into consideration the feature-based graph, where similarities among documents are computed on text features. We use the features defined in (Jiang et al., 2014), where the model based features are eliminated since the

computation depends on pre-assigned class labels, and represent a document as a vector of the feature values. We compute the similarity between any pair of documents using the Euclidean distance, and built the feature-based graph (denoted as G^f) in the same way as above.

Additionally, to take advantage of both graphs, we combine them into one (denoted as G^{cf}) using the following formula.

$$G^{cf}_{i,j} = \max\left[G^{c}_{i,j}, G^{f}_{i,j}\right] \quad (9)$$

3.3.2 Graph Propagation

Given a graph G^* constructed in previous sections, its nodes are divided into two sets: the labeled set V_l and the unlabeled set V_u. The goal of label propagation is to propagate class labels from the labeled nodes (i.e. documents) to the entire graph. Here, we use a simplified version of the label propagation method presented in (Subramanya et al., 2010), which has been proved effective (Kim et al., 2013). The method iteratively updates the label distribution on a document node using the following equation.

$$p_d^{(i)}(l) = \frac{1}{\kappa_d}\left(p_d^0(l)\delta(d \in V_l) + \sum_{v \in \mathcal{N}(d)} G_{d,v} p_v^{(i-1)}(l)\right) \quad (10)$$

At the left side of Eq.10, $p_d^{(i)}(l)$ is the afterward probability of l (i.e. the class label) on a node d at the i-th iteration. At the right side, κ_d is the normalizing constant to make sure the sum of all the probabilities is 1, and $p_d^0(l)$ is the initial probability of l on d if d is initially labeled (i.e. belonging to the labeled set V_l). $\delta(x)$ is the indicator function. $\mathcal{N}(d)$ denotes the set of neighbors of d. The iteration stops when the changes in $p_d^{(i)}(l)$ for all the nodes and label values are small enough (e.g. less than e^{-3}), or i exceeds a predefined number (e.g. greater than 30).

4 Empirical Studies

In this section, we conduct experiments on datasets of both English and Chinese, to investigate the following three research questions:

RQ1: Whether the proposed method (i.e. GRAW) outperforms the state-of-the-art methods for readability assessment?

RQ2: What are the effects of adding the word coupling matrix to the general bag-of-words model?

RQ3: Whether the graph merging strategy is effective, and whether the performance can be further improved by combining the feature-based graph.

4.1 Corpus and Metrics

To evaluate our proposed method, we collected two datasets. The first is CPT (*Chinese primary textbook*) (Jiang et al., 2014), which contains Chinese documents of six reading levels. The second is ENCT (*English New Concept textbook*) which contains English documents of four reading levels. Both datasets are built from well-known textbooks where documents are labeled as grade levels by credible educationists. The details of the datasets are listed in Table 2.

Dataset	Language	#Grade	#Doc	#Sent	#Word
CPT	Chinese	6	637	16145	234372
ENCT	English	4	279	4671	62921

Table 2: Statistics of the datasets on both English and Chinese

We conduct experiments on both datasets using the cross-validation which randomly divides a dataset into labeled (training) and unlabeled (test) sets. The labeling proportion is varied to investigate the performance of GRAW under different circumstances. To reduce variability, given certain labeling proportion, 100 rounds of cross-validation are performed, and the validation results are averaged over all the rounds. We choose the precision (P), recall (R) and F1-measure (F1) as the performance metrics.

4.2 Comparison to the State-of-the-Art Methods

To address RQ1, we implement the following readability assessment methods and compare GRAW to them: (1) SMOG (McLaughlin, 1969) and FK (Kincaid et al., 1975) are two widely used readability formulae. We reserve their core measures (i.e. text features, and number of strokes for Chinese instead of number of syllables), and refine the coefficients on both datasets to befit the reading (grade) levels. (2) SUM (Collins-Thompson and Callan, 2004) is a word-based method, which trains one unigram model for each grade level, and applies model smoothing both inter and intra the grade levels. (3) LR and SVM refer to two feature-based methods which incorporate text features defined in (Jiang et al., 2014) to represent documents as instances. The logistic regression model and

Dataset	Level	Metric	Methods						
			SMOG	FK	SUM	LR	SVM	GRAW$_c$	GRAW$_{cf}$
CPT (Chinese)	Gr.1	P	57.48	74.07	71.76	71.87	73.18	73.26	**75.29**
		R	17.31	9.69	36.31	71.17	67.28	73.28	**83.17**
		F1	26.14	15.25	47.67	71.23	69.70	72.98	**78.89**
	Gr.2	P	34.73	31.44	37.94	51.62	50.78	52.05	**55.83**
		R	31.06	28.00	29.73	56.48	59.45	57.36	**66.06**
		F1	32.66	29.42	33.13	53.67	54.53	54.40	**60.37**
	Gr.3	P	20.05	20.79	28.12	44.15	48.89	46.33	**51.72**
		R	58.84	75.53	25.06	43.94	49.94	58.59	**68.41**
		F1	29.89	32.40	26.35	43.72	49.04	51.57	**58.74**
	Gr.4	P	25.06	28.94	25.60	33.35	33.92	39.90	**44.15**
		R	**41.06**	31.82	28.76	31.82	33.64	35.42	28.88
		F1	31.03	29.69	26.91	32.24	33.58	**37.32**	34.57
	Gr.5	P	33.57	45.00	28.71	37.70	37.30	**45.02**	37.33
		R	4.00	2.71	34.41	**36.12**	34.29	27.12	19.00
		F1	7.02	4.95	31.10	**36.61**	35.47	33.35	24.45
	Gr.6	P	0.00	6.67	32.21	40.47	**46.53**	45.91	44.24
		R	0.00	0.35	45.81	39.03	43.48	51.81	**54.06**
		F1	0.00	0.67	37.55	39.48	44.65	**48.38**	48.15
	Avg.	P	28.48	34.48	37.39	46.53	48.43	50.41	**51.43**
		R	25.38	24.68	33.35	46.43	48.01	50.60	**53.26**
		F1	21.12	18.73	33.78	46.16	47.83	49.67	**50.86**
ENCT (English)	Gr.1	P	54.65	60.79	**96.59**	88.60	90.74	95.42	95.53
		R	67.50	73.50	84.77	**89.32**	85.45	83.77	83.95
		F1	60.18	66.36	**90.09**	88.64	87.76	89.01	89.18
	Gr.2	P	50.11	56.23	78.30	85.51	**90.80**	88.60	89.03
		R	59.28	63.93	35.07	86.07	92.86	96.76	**96.86**
		F1	54.17	59.69	48.11	85.54	91.68	92.42	**92.70**
	Gr.3	P	29.49	32.09	40.53	88.31	89.08	85.36	**89.73**
		R	24.22	26.94	68.33	86.17	84.78	94.17	**96.56**
		F1	26.40	29.15	50.77	86.94	86.16	89.40	**92.92**
	Gr.4	P	85.73	94.00	69.30	89.79	81.20	91.70	**95.26**
		R	14.64	18.21	**97.64**	87.07	85.21	77.93	85.36
		F1	24.06	29.46	80.81	88.02	81.79	83.84	**89.81**
	Avg.	P	55.00	60.78	71.18	88.05	87.95	90.27	**92.39**
		R	41.41	45.65	71.45	87.16	87.08	88.16	**90.68**
		F1	41.20	46.16	67.44	87.28	86.85	88.67	**91.15**

Table 3: The average Precision, Recall and F1-measure (%) per reading level of the seven methods for readability assessment on both datasets when the labeling proportion is 0.7

support vector machine are used as the classifiers respectively.

For GRAW, we implement label propagation on both the merged graph G^c and the final graph G^{cf} (Section 3.3), denoted as GRAW$_c$ and GRAW$_{cf}$ respectively. Table 3 gives the average performance measure per reading level resulted by the implemented methods on both datasets. Unless otherwise specified, we fixed η to 3, and λ to 2800 for CPT and 2000 for ENCT. The proportion of the labeled (training) set is set to 0.7.

In Table 3, the precision, recall and F1-measure of all the seven methods are calculated per reading (grade) level on both English and Chinese datasets. The values marked in bold in each row refer to the maximum (best) measure gained by the methods.

From Table 3, the readability formulae (SMOG and FK) perform poorly on either the precision or recall measure, and their F1-measure values are generally the poorest. Both SMOG and FK are designed for English, and have acceptable performance on the English dataset ENCT. The unigram model (SUM) performs a little better than the readability formulae. On ENCT, It has relatively good performance on grade levels 1 and 4, while on the Chinese dataset CPT, the performance is not satisfactory. The feature-based methods (LR and SVM) perform well on both ENCT and CPT, which means both the text features developed and the classifiers trained are useful. In general, GRAW$_c$ performs better than both LR and SVM, which demonstrates the effectiveness of our method. In addition, by combining the feature-based graph (GRAW$_{cf}$), GRAW can be improved, and performs the best on all the three metrics over the majority of reading levels on both datasets. The only exception is on level 5 in CPT, which suggests the requirement of further improvements.

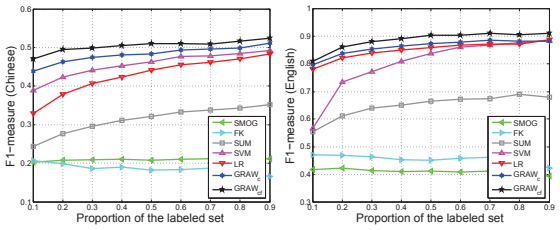
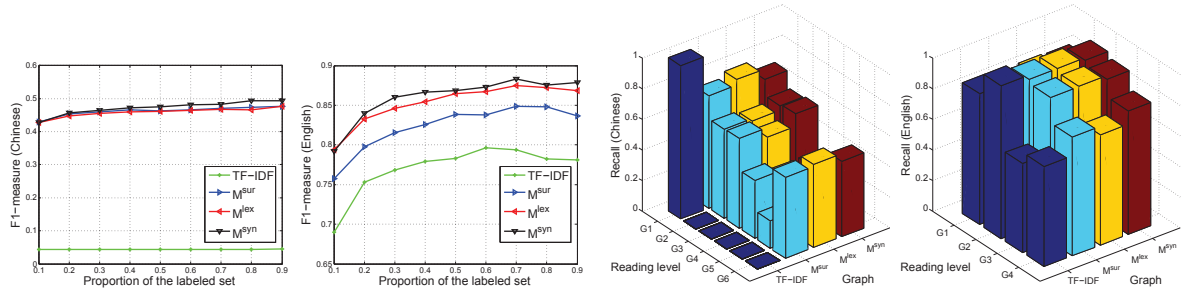


Figure 3: The average F1-measure of the seven methods on both datasets with the labeling proportion varied from 0.1 to 0.9

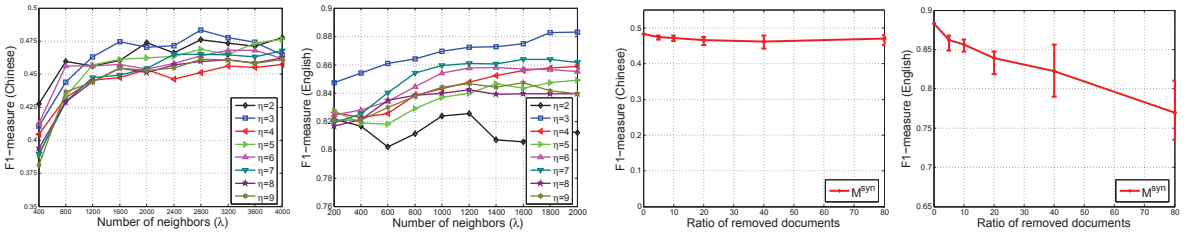
We study the effect of labeling proportion on the performance of these methods on both datasets. The F1-measure averaged over the reading levels is used, since it is a good representative of the three metrics according to Table 3. Figure 3 depicts the performance trends of all the methods.

From Figure 3, neither SMOG nor FK benefits



(a) Comparison of the average F1-measure between the coupling and general TF-IDF matrices

(b) Comparison of the Recall rate per reading level between the coupling and general TF-IDF matrices



(c) The effects of η and λ on the performance of the word coupling matrix

(d) The effect of corpus size on the performance of the word coupling matrix

Figure 4: Four perspectives on the effectiveness of the word coupling matrices

from the increasing size of the labeled set. This suggests that the performance of the readability formulae can hardly be improved by accumulating training data. The other 5 methods achieve better performance on larger labeled set, and outperform the two formulae even if the labeling proportion is small. Both LR and SVM perform better than SUM, but the performance is not good when the labeling proportion is less than 0.3, especially on the Chinese dataset. On the Chinese dataset, SVM performs better than LR, while on the English dataset, the situation is reversed. Both versions of GRAW outperform the other methods over the labeling ranges on both datasets. In addition, GRAW performs well when the labeling proportion is still small. Again, by combining the feature-based graph, the performance of GRAW is consistently improved.

In summary, GRAW can outperform the state-of-the-art methods for readability assessment on both English and Chinese datasets. By combining the feature-based graph, the performance of GRAW can be further improved.

4.3 Effects of the Word Coupling Matrix

For RQ2, we firstly compare the coupled bag-of-words model to the general model in the process of graph construction. Four graphs are built by using each of the three word coupling matrices (i.e. M^{sur}, M^{lex} and M^{syn}) and the TF-IDF matrix respectively. Label propagation is applied on each graph to predict reading levels of unlabeled documents. The labeling proportion is varied from 0.1 to 0.9 on both the English and Chinese datasets. Figure 4(a) depicts the average F1-measure resulted from the four graphs.

From Figure 4(a), the three word coupling matrices greatly outperform the TF-IDF matrix, especially on the Chinese dataset. This demonstrates that the word coupling matrices are very effective in improving the performance of the general bag-of-words model for readability assessment.

Secondly, we investigate the performance of the four matrices per reading level. Figure 4(b) depicts the recall rate per reading level of the four corresponding graphs in bar charts. The labeling proportion is set to 0.7. The recall rate is used because it makes the reason evident that the TF-IDF matrix performs poorly. From Figure 4(b), on the Chinese dataset, nearly all the unlabeled documents are classified as level 1 by the TF-IDF matrix, in which the word frequencies are too few to make meaningful discrimination among the reading levels. On the English dataset, the TF-IDF matrix performs better, but still prefers to classify documents into lower levels.

As described in Section 3.2.2, η (the number of difficulty levels of sentences) and λ (the number of neighbors pertained for each document node) are two parameters in building the word coupling

matrices. To investigate their effects on the performance of the built matrices, we vary the values of both η and λ, and compute the average F1-measure on the two datasets. Figure 4(c) depicts the results in line charts, where η varies from 2 to 9 step by 1, while λ varies from 400 to 4000 step by 400 on Chinese and from 200 to 2000 step by 200 on English (the difference is due to the dissimilar number of documents between the two datasets). The three word coupling matrices exhibit similar behavior during experiments, hence, only M^{syn} is depicted.

From Figure 4(c), a small η (e.g. 2 or 3) is good on the Chinese dataset. However, on the English dataset, $\eta = 2$ leads to the poorest performance. It seems the increasing of η causes vibrated performance, and the trend is further complicated when involving λ. Above all, $\eta = 3$ gives a preferable option on both datasets. For λ, most of the lines exhibit a similar trend that rises first and then keeps stable on both datasets, although some may drop when λ is too large. This suggests that making a relatively large number of the other words as the neighbors of one (i.e. $\lambda = 2800$ on the Chinese dataset and $\lambda = 2000$ on the English dataset) will make an effective word coupling matrix.

The word coupling matrix constructed in GRAW uses the whole corpus on either English or Chinese. To investigate if the corpus size takes effects on the performance of GRAW, we vary the proportion of the corpus used by randomly removing documents from each reading level. Figure 4(d) depicts the average F1-measure resulted by M^{syn}. The removing ratio is selected from $\{0, 0.05, 0.1, 0.2, 0.4, 0.8\}$. Both the mean values and deviations are shown on the line chart.

From Figure 4(d), on the Chinese dataset, the performance of GRAW suffers little from removing documents, even if only 20% documents are left for building the word coupling matrix. However, on the English dataset, the mean performance drops sharply and the deviation increases evidently. This suggests that cumulating sufficient corpus is required for building a suitable word coupling matrix in GRAW, and factors other than number of documents may influence the corpus quality, which deserves further study.

In summary, the word coupling matrix plays an essential role in GRAW. For building a suitable word coupling matrix, the number of difficulty levels of sentences (η) can be set to 3, and a relatively large number of the other words should be selected as the neighbors of a word. A sufficient corpus is required for refining the matrix.

4.4 Effectiveness of Graph Combination

For RQ3, we compare graphs built on each singular word coupling matrix (i.e. M^{sur}, M^{lex} and M^{syn}) to the merged graph (i.e. GRAW$_c$) and the combined graph (i.e. GRAW$_{cf}$). Figure 5 depicts the average F1-measure resulted after applying label propagation on these graphs with labeling proportion varied from 0.1 to 0.9. The feature-based graph (i.e. G^f) is also depicted for comparison.

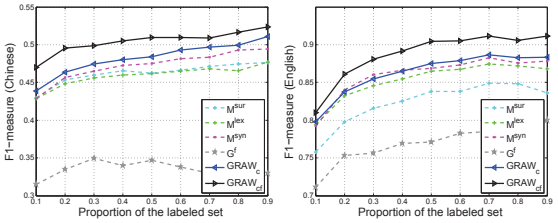


Figure 5: The average F1-measure of different types of graphs on the English and Chinese datasets

From Figure 5, the merged graph GRAW$_c$ outperforms the three basic graphs on both datasets in most cases. Within the three, M^{syn} performs best, especially on the English dataset, where it can outperform GRAW$_c$ slightly when the labeling proportion is small $(0.2 - 0.4)$. By combining the feature-based graph, GRAW$_{cf}$ performs even better on both datasets, although G^f performs poorest among all the graphs. In summary, the graph merging strategy is effective, and by combining the feature-based graph, the performance of GRAW can be improved. This demonstrates the potential of GRAW.

5 Conclusion

In this paper, we propose a graph-based readability assessment method using word coupling. The coupled bag-of-words model is designed, which exploits the correlation of readability among the words, and by applying TF-IDF, models the relationship among documents on reading levels. The model is employed in the graph-based classification framework for readability assessment, which involves graph building, merging, and label propagation. Experiments are conducted on both Chinese and English datasets. The results show that our method can outperform the commonly used

methods for readability assessment. In addition, the evaluation demonstrates the potential of the coupled bag-of-words model and the graph combination/merging strategies.

In our future work, we plan to verify the soundness of the results by applying our method on large volume corpus of both English and Chinese. In addition, we will investigate other ways of computing the word coupling matrices, such as incorporating word coherency or semantics, and develop efficient merging strategies which can be used for training classification models, as well as for building graphs.

Acknowledgments

This work was supported by the National NSFC projects under Grant Nos. 61373012, 61321491, and 91218302.

References

Rebekah George Benjamin. 2012. Reconstructing readability: Recent developments and recommendations in the analysis of text difficulty. *Educational Psychology Review*, 24(1):63–88.

Xin Cheng, Duoqian Miao, Can Wang, and Longbing Cao. 2013. Coupled term-term relation analysis for document clustering. In *Proceedings of the 2013 International Joint Conference on Neural Networks*, pages 1–8. IEEE.

Kevyn Collins-Thompson and James P Callan. 2004. A language modeling approach to predicting reading difficulty. In *Proceedings of the 2004 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 193–200. Association for Computational Linguistics.

Samuel I Daitch, Jonathan A Kelner, and Daniel A Spielman. 2009. Fitting a graph to vector data. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 201–208. ACM.

Edgar Dale and Jeanne S. Chall. 1948. A formula for predicting readability. *Educational Research Bulletin*, 27(1):11–28.

Lijun Feng, Martin Jansche, Matt Huenerfauth, and Noémie Elhadad. 2010. A comparison of features for automatic readability assessment. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 276–284. Association for Computational Linguistics.

Thomas François and Cédrick Fairon. 2012. An ai readability formula for french as a foreign language. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 466–477. Association for Computational Linguistics.

Itziar Gonzalez-Dios, Marıa Jesús Aranzabe, Arantza Dıaz de Ilarraza, and Haritz Salaberri. 2014. Simple or complex? assessing the readability of basque texts. In *Proceedings of the 25th International Conference on Computational Linguistics*, pages 334–344. Association for Computational Linguistics.

Julia Hancke, Sowmya Vajjala, and Detmar Meurers. 2012. Readability classification for german using lexical, syntactic, and morphological features. In *Proceedings of the 24th International Conference on Computational Linguistics*, pages 1063–1080. Association for Computational Linguistics.

Tony Jebara, Jun Wang, and Shih-Fu Chang. 2009. Graph construction and b-matching for semi-supervised learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 441–448. ACM.

Zhiwei Jiang, Gang Sun, Qing Gu, and Daoxu Chen. 2014. An ordinal multi-class classification method for readability assessment of chinese documents. In *Knowledge Science, Engineering and Management*, pages 61–72. Springer.

Paul Kidwell, Guy Lebanon, and Kevyn Collins-Thompson. 2009. Statistical estimation of word acquisition with application to readability prediction. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 900–909. Association for Computational Linguistics.

Doo Soon Kim, Kunal Verma, and Peter Z Yeh. 2013. Joint extraction and labeling via graph propagation for dictionary construction. In *Proceedings of the 27th AAAI Conference on Artificial Intelligence*, pages 510–517.

J Peter Kincaid, Robert P Fishburne Jr, Richard L Rogers, and Brad S Chissom. 1975. Derivation of new readability formulas (automated readability index, fog count and flesch reading ease formula) for navy enlisted personnel. Technical report, Naval Air Station, Memphis, TN.

Solomon Kullback and Richard A Leibler. 1951. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86.

G Harry McLaughlin. 1969. Smog grading: A new readability formula. *Journal of reading*, 12(8):639–646.

Natalia Ponomareva and Mike Thelwall. 2012. Do neighbours help?: an exploration of graph-based algorithms for cross-domain sentiment classification. In *Proceedings of the 2012 Joint Conference on*

Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pages 655–665. Association for Computational Linguistics.

Gerard Salton and Christopher Buckley. 1988. Term-weighting approaches in automatic text retrieval. *Information processing and management*, 24(5):513–523.

Sarah E Schwarm and Mari Ostendorf. 2005. Reading level assessment using support vector machines and statistical language models. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 523–530. Association for Computational Linguistics.

Amarnag Subramanya, Slav Petrov, and Fernando Pereira. 2010. Efficient graph-based semi-supervised learning of structured tagging models. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 167–176. Association for Computational Linguistics.

S. K. Michael Wong, Wojciech Ziarko, and P. C. N. Wong. 1985. Generalized vector space model in information retrieval. In *Proceedings of the 8th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 18–25. ACM.

Beverley L. Zakaluk and S. Jay Samuels. 1988. *Readability: Its Past, Present, and Future*. ERIC.

Xiaodong Zeng, Derek F Wong, Lidia S Chao, and Isabel Trancoso. 2013. Graph-based semi-supervised model for joint chinese word segmentation and part-of-speech tagging. In *Proceeding of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 770–779. Association for Computational Linguistics.

Xiaojin Zhu and Zoubin Ghahramani. 2002. Learning from labeled and unlabeled data with label propagation. Technical Report CMU-CALD-02-107, Carnegie Mellon University.

More Features Are Not Always Better: Evaluating Generalizing Models in Incident Type Classification of Tweets

Axel Schulz
Business Intelligence Marketing
DB Fernverkehr AG
Germany
schulz.axel@gmx.net

Christian Guckelsberger
Computational Creativity Group
Goldsmiths College, University of London
United Kingdom
c.guckelsberger@gold.ac.uk

Benedikt Schmidt
Telecooperation Lab, Technische Universität Darmstadt, Germany
benedikt.schmidt@tk.informatik.tu-darmstadt.de

Abstract

Social media represents a rich source of up-to-date information about events such as incidents. The sheer amount of available information makes machine learning approaches a necessity for further processing. This learning problem is often concerned with regionally restricted datasets such as data from only one city. Because social media data such as tweets varies considerably across different cities, the training of efficient models requires labeling data from each city of interest, which is costly and time consuming.

In this study, we investigate which features are most suitable for training generalizable models, i.e., models that show good performance across different datasets. We reimplemented the most popular features from the state of the art in addition to other novel approaches, and evaluated them on data from ten different cities. We show that many sophisticated features are not necessarily valuable for training a generalized model and are outperformed by classic features such as plain word-n-grams and character-n-grams.

1 Introduction

Incident information contained in social media has proven to frequently include information not captured by standard emergency channels (e.g. 911 calls, bystander reports). Therefore, stakeholders like emergency management and city administration can highly benefit from social media. Due to its unstructured and unfocused nature, automatic filtering of social media content is a necessity for further analysis. A standard approach for this filtering is automatic classification using a trained machine learning model (Agarwal et al., 2012; Schulz et al., 2013; Schulz et al., 2015b).

A problem for the classification approach is that language, style and named entities used in social media highly vary across different regions. Consider the following two tweets as examples: "RT: @People 0noe friday afternoon in heavy traffic, car crash on I-90, right lane closed" and "Road blocked due to traffic collision on I-495". Both tweets comprise entities that might refer to the same thing with different wording, either on a semantically low ("accident" and "car collision") or more abstract level ("I90" and "I-495"). With simple syntactical text similarity approaches using standard bag of words features, it is not easily possible to make use of this semantic similarity, even though it is highly valuable for classification.

These limitations impose constraints on the dataset, because tokens are likely to be related to the location where the text was created or contain location- or incident-sensitive topics. Models trained using spatially and temporally restricted data from one region are bound by the specific aspects of language and style expressed in the training data, thus, model reuse is not easily possible.

In this paper, we focus on the creation of generalized models. Such models avoid the use of features that — overfitting like — are only useful for a specific region. Generalized models are intended to work in different regions, even if training data originates only from one ore few regions. This can ensure high classification rates even in areas where only few training samples are available. Finally, in

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 421–430,
Lisbon, Portugal, 17-21 September 2015.

times of increasing growth of cities and the merging with surrounding towns to large metropolitan areas, they allow to cope with the latent transitions in token use.

To create generalized models for incident type classification (and social media classification in general) the most important step is an appropriate feature generation. Therefore, in this paper we investigate the suitability of standard and novel features and different machine learning algorithms for the creation of generalized classification models for incident type classification. We conduct intensive feature engineering and evaluation. For this purpose, we have collected and labeled 10 datasets with high regional variation. To the best of our knowledge, this is the first investigation of the challenges of heterogeneous datasets in this domain, and of the suitability of state of the art classification and feature extraction techniques.

In summary, our contributions are: 1) Investigation of features and feature groups for generalized social media/incident type classification models. 2) Identification of the best feature combinations and classifiers for a generalized model. For an evaluation (qualitative and inferential statistics) of ten tweet datasets with high regional variation we get an overall F-measure of $> 83\%$. 3) The evaluation shows that features extending a plain n-gram-based approach are not necessarily valuable for training a generalized model as these provide little improvement.

Following this introduction, we give an overview of related work in Section 2. In Section 3, we provide a description of our datasets followed by a comprehensive evaluation in Section 4. We close with our conclusion and future work in Section 5.

2 Related Work

A review of existing work on the classification of social media content shows which features, feature groups and algorithms are generally used (see table 1). Furthermore, the number of classes and the dominating approaches unfold. We report the ratios of labeled tweets for the individual approaches; however, we omit performance measures as these are directly related to the respective datasets used for evaluation.

Classifiers based on Support Vector Machines (SVM) or Naive Bayes (NB) clearly dominate in terms of performance for incident type classification. (Sakaki and Okazaki, 2010; Carvalho et al., 2010; Agarwal et al., 2012; Robert Power, 2013; Schulz and Janssen, 2014) trained an SVM, whereas (Agarwal et al., 2012; Imran et al., 2013; Schulz and Janssen, 2014) also evaluated an NB classifier. In contrast to these works, (Wanichayapong et al., 2011) followed a dictionary-based approach using traffic-related keywords. (Li et al., 2012) do not provide any information about the classifier used.

Feature groups are mostly based on word-n-grams, such as unigrams (Carvalho et al., 2010), bigrams (Imran et al., 2013), or the combination of unigrams and bigrams (Robert Power, 2013; Karimi et al., 2013; Agarwal et al., 2012). (Schulz and Janssen, 2014) combined unigrams, bigrams, and trigrams. Also, based on the words present in the text named entities such as locations, organizations, or persons were used by (Agarwal et al., 2012; Li et al., 2012; Schulz and Janssen, 2014).

Twitter-specific features were also used, including the number of hashtags, @-mentions or web-text features such as the presence of numbers or URLs (Li et al., 2012; Agarwal et al., 2012; Robert Power, 2013; Karimi et al., 2013; Imran et al., 2013; Schulz and Janssen, 2014).

Keywords also play a crucial role in feature design. (Sakaki and Okazaki, 2010) used earthquake-specific keywords, statistical features (the number of words in a tweet and the position of keywords), and word context features (the words before and after the earthquake-related keyword). (Wanichayapong et al., 2011) used traffic-related keywords in combination with location-related keywords. Furthermore, Li et al. (2012) iteratively refined a keyword-based search for retrieving a higher number of incident-related tweets.

Two approaches rely on more specific feature groups. The approach of (Schulz and Janssen, 2014) is the only one that uses TF-IDF scores. (Imran et al., 2013) use Kipper et al.'s (2006) extension of the Verbnet ontology for verbs.

The related approaches mostly use word-n-grams and a variety of Twitter-specific features. Datasets are spatially and temporally restricted and limited to a small number, complicating generalizability.

3 Data Collection and Processing

We are interested in generalizable models for different regions, user-generated content has been

Table 1: Overview of related approaches for incident type classification. (NEs = Named Entities)

Approach	Classifier	#Classes	#Tweets	N-Grams	#NEs	#URLs	TF-IDF	Twitter	Other
(Sakaki and Okazaki, 2010)	SVM	2	597	x					Context
(Carvalho et al., 2010)	SVM	2	3,300	x					
(Wanichayapong et al., 2011)	Keyw.	2	1,249						
(Agarwal et al., 2012)	NB, SVM	2	1,400	x	x	x			
(Li et al., 2012)	Undefined	2	Undef.		x			x	
(Robert Power, 2013)	Keyw., SVM	2	794	x				x	
(Karimi et al., 2013)	SVM	6	5,747	x				x	
(Imran et al., 2013)	NB	3	1,233	x		x		x	Verbnet
(Schulz and Janssen, 2014)	SVM, NB	4	2,000	x	x	x	x	x	

created in. For this purpose, we created 10 datasets with more than 20k labeled tweets to train and test models with respect to their generalization. In the following, we describe how this data was collected, preprocessed, and which features were generated.

3.1 Data Collection

We focus on tweets as suitable example for unstructured textual information shared in social media. The classification of incident-related tweets represents a challenge that is relevant for many cities. We use a complex four-class classification problem, where new tweets can be assigned to the classes "crash", "fire", "shooting", and a neutral class "not incident related". This goes beyond related work with a focus on two-class classification. Our classes were identified as the most common incident types in Seattle using the Fire Calls data set (http://seattle.data.gov), an official incident information source.

As ground truth data, we collected several city-specific datasets using the Twitter Search API. These datasets were collected in a 15 km radius around the city centres of Boston (USA), Brisbane (AUS), Chicago (USA), Dublin (IRE), London (UK), Memphis (USA), New York City (USA), San Francisco (USA), Seattle (USA), Sydney (AUS).

We selected these cities because of their huge regional distance, which allows us to evaluate our approaches with respect not only to geographical, but also to cultural variations. Also, for all cities, sufficiently many English tweets can be retrieved. We chose 15 km as radius to collect a representative data sample even from cities with large metropolitan areas. Despite the limitations of the Twitter Search API with respect to the number of geotagged tweets, we assume that our sample is, although by definition incomplete, highly relevant to our experiments.

We collected all available Tweets during limited time periods, resulting in three initial sets of tweets: 7.5M tweets collected from November, 2012 to February, 2013 for Memphis and Seattle (**SET_CITY_1**); 2.5M tweets collected from January, 2014 to March, 2014 for New York City, Chicago, and San Francisco (**SET_CITY_2**); 5M tweets collected from July, 2014 to August, 2014 for Boston, Brisbane, Dublin, London, and Sydney (**SET_CITY_3**).

For the manual labeling process, we had to select a subset of our original tweet set which included our classes of interest for model training and testing. Generating subsets is required because manual labeling of social media data is very expensive, especially if multiple annotators are involved. To generate subsets we used the approach of (Schulz et al., 2013) of extracting microposts using incident-related keywords. As a result, more than 200 keywords were identified for each class. Based on these incident-related keywords, we were able to accurately and efficiently filter the datasets. After applying keyword-filtering, we randomly selected 5.000 microposts for each city. Though one might assume that this pre-filtering leads to a biased dataset, (Schulz and Janssen, 2014) showed that keyword sampling does not influence the classification process as the performance of a keyword-based classifier is notably worse compared to supervised classifiers.

In the next step, we removed all redundant tweets as well as those with no textual content from the resulting sets as a couple of tweets contain keywords that are part of hashtags or @-mentions, but have no useful textual content. The tweets were then labeled manually by five annotators using the CrowdFlower (http://www.crowdflower.com/) platform. We retrieved the manual labels and selected those for

Table 2: Class distributions for all datasets.

Dataset	Classes			
	Crash	Fire	Shooting	No
Boston	347	188	28	2257
Sydney	587	189	39	2208
Brisbane	497	164	12	1915
Chicago	129	81	4	1270
Dublin	131	33	21	2630
London	283	95	29	2475
Memphis	23	30	27	721
NYC	129	239	45	1446
SF	161	82	61	1176
Seattle	204	153	139	390

which all coders agreed to at least 75%. In the case of disagreement, the tweets were removed. This resulted in ten datasets with regional diversity to be used for evaluation.

Table 2 lists the class distributions for each dataset. The distributions vary considerably, allowing us to evaluate with typical city-specific samples. Also, the "crash" class seems to be the most prominent incident type, whereas "shootings" are less frequent. One reason for this is that "shootings" do not occur as frequent as other incidents, whereas another less obvious reason might be that people tend to report more about specific incident types and that there is not necessarily a correlation between the real-world incidents and those mentioned in tweets. Although the data sets have been filtered based on keywords, the "no incident" class remains the largest class.

One of the key questions that motivates our work is to which extent the used words vary in each dataset as an effect of the spatial and cultural context. We thus analysed how similar all datasets are by calculating the intersection of tokens. We found that after preprocessing, between 14% and 23% tokens are shared between the datasets. We do not assume that every unique token is a city-specific token, but the large number of tweets in our evaluations gives a first indication that there is a diversity in the samples that either requires the training of several individual- or one generalizing model which is the focus of this paper.

3.2 Preprocessing and Feature Generation

To use our datasets for feature generation, i.e., for deriving different *feature groups* that are used for training a classification model, we had to convert the texts into a structured representation by means of preprocessing. Following this, we extracted several features for training classification models. To evaluate the best feature groups for incident type classification, we re-implemented commonly used feature extraction approaches from the state of the art. We further extended these feature groups by additional ones that seemed promising in this problem domain:

Preprocessing As a first step, the text was converted to Unicode to preserve non-Unicode characters. Specific URLs would not be useful for the classification process, therefore we replaced them with a common token "URL". We then removed stopwords and conducted tokenization. Every token was then analysed and non-alphanumeric characters were removed or replaced. Finally, we applied lemmatization to normalize all tokens. All preprocessing steps were performed by DKPro TC (de Castilho and Gurevych, 2014), a popular framework for text classification. After preprocessing, we generated several features (see Table 3). In the following, we give a description of the different feature groups.

Baseline Feature Sets As the most simple approach and as used in all related works, we represented tweets as a set of words and also as a set of characters with varying lengths. As features, we used a vector with the frequency of each n-gram. Most importantly, we evaluated the powerset of all different combinations of n-grams. For instance, if a length of $n = 2$ was chosen, we evaluated the three combinations $(n = 1), (n = 1, 2), (n = 2)$. Furthermore, as not all tokens are necessarily important for the classification process, we evaluated several top-k selection strategies, i.e., taking only the k most frequent n-grams into account. For this, we tested $k = 100, 1000, 5000$ as well as the approach using all n-grams. We treat these features as the baseline approach, and extend it by additional features, e.g. similarity, sentiment scores, Twitter-specific features.

Sentiment Features Emoticons are widely used to express emotions in textual content. Various text classification approaches make use of these, e.g. for sentiment analysis (Agarwal et al., 2011; Go et al., 2009). For incident type classification, they could also be useful as people link emotions with ongoing incidents, thus, we re-implemented three approaches for extracting sentiment features.

Table 3: Overview of all feature groups implemented for comparison

Feature Group	Description
Word-n-grams	Each tweet is represented as a powerset of word-n-grams of length $n = 1$ to $n = 3$.
Char-n-grams	Each tweet is represented as a powerset of char-n-grams of length $n = 1$ to $n = 5$.
POS_EMO	The Tweet NLP part-of-speech tagger (Owoputi et al., 2013) was used to identify emoticons. The ratio of emoticons to all tokens is calculated.
DICT_EMO	An emoticon library that is based on the suggestions from Agarwal et al. (Agarwal et al., 2011) was used comprising a set of 63 emoticons from Wikipedia. The number of positive and the number of negative emoticons in a tweet is calculated.
AGG_EMO	One single sentiment score based on the second approach by aggregating the number of positive and negative emoticons.
NER	We used the Stanford Named Entity Recognizer (Finkel et al., 2005) and applied the three class model to count the number of location, organization, and person mentions.
NR_CHAR	The number of characters in a tweet.
NR_SENT	The number of sentences in a tweet.
NR_TOKEN	The number of tokens in a tweet.
QUEST_RT	The proportion of question marks and sentences in a tweet.
EXCLA_RT	The proportion of exclamation marks and sentences in a tweet.
NR_AT_MN	The number of @-mentions in a tweet.
NR_HASHTAG	The number of hashtags in a tweet.
NR_URL	The number of URLs present in a tweet.
NR_SLANG	The number of colloquial words (i.e., *lol* or *ugh*). Feature extraction is based on the Tweet NLP POS-tags (Owoputi et al., 2013).
IS_RT	A boolean to indicate whether a tweet is a retweet.
NR_CARD	In conjunction with the named entities present in tweets, people tend to refer to street names (e.g.,*I-95*) or the number of injured people (e.g.,*2-people*). Thus, we create a feature for the number of cardinal numbers present in a tweet.
GREEDY_ST	Similarity scores following Greedy String Tiling (Wise, 1996) as a method to deal with shared sub-strings that do not appear in the same order.
LEVENST	The Levenshtein distance (Levenshtein, 1966) as an edit-distance metrics, i.e., the minimum number of edit operations that transform one tweet into another.
TF_IDF	As the baseline relies on plain frequency-based weighting, we calculate the traditional TF-IDF scores (Manning et al., 2009) for every tweet.

Named Entities: As shown in the state of the art, named entities, i.e. entities that have been assigned a name such as *Seattle*, are commonly used in tweets. Named entities might be valuable, as these are used frequently in incident-related tweets. Thus, we also incorporated Named Entity Recognition (NER) for feature extraction.

Stylistic Features: The style of a tweet could be an additional indicator for incident relatedness. For instance, a repetition of punctuations could point at a person that is expressing emotions resulting from an ongoing incident. Structured representation might indicate high quality.

Twitter-specific features As shown in related work, several Twitter-specific features seem to be valuable for incident type classification such as the number of @-mentions and hashtags.

Similarity Features The similarity of individual tweets might be helpful to identify common topics. We therefore implemented several similarity scores[1]. The rationale behind this is to embrace additional features that do not only take the raw frequencies of words into account, but also which words appear in which document.

To sum up, we re-implemented two approaches that will serve as a baseline, and 18 additional feature groups to extend them. In the following section, we will evaluate the usefulness of these approaches for training a generalizing model.

4 Evaluation

The goal of our evaluation is to determine which features were most useful for creating a generalizing model. We first describe our method, including the feature sets, the classification algorithms used, and our sampling procedure. This is followed by a results section in which we report differences in performance by means of qualitative and inferential statistics.

4.1 Method

The indicators for well-performing features in related work allows us to perform a condensed evaluation, compared to similar studies such as (Hasanain et al., 2014).

Our approach comprises three steps: First, we evaluated the baseline approaches, i.e., word- and char-n-grams. Second, we combined each of the remaining features with the best performing baseline feature. Third, we again selected the best performing combinations and evaluated their power set. To evaluate the suitability of different features for training generalizing models, we picked one dataset from the 10 presented in Section 3.1 for training, and tested on the remaining 9 datasets. We did not evaluate different models on datasets from only one city, as we were interested in generalizing models.

Selecting each city as training set resulted in 90 performance samples per model. The models were formed by combining the feature sets described in the previous section 3.2 or respectively, their combinations, with an SVM and NB classifier. We decided for these classification algorithms since they were the most successful in related work. Another reason for the choice of NB is its good performance in text classification tasks, as demonstrated by Rennie et al. (2003). We relied on the LibLinear implementation of an SVM because it has been shown that for a large number of features and a small number of instances, a linear kernel is comparable to a non-linear one (Hsu et al., 2003). As for SVMs parameter tuning is inevitable, we evaluated the best settings for the slack variable c whenever an SVM was used. For training and testing, we used the reference implementations in WEKA (Hall et al., 2009).

We calculated the F1-Measure for assessing performance, because it is well established in text classification, cannot be manipulated by the classification threshold parameter and allows to measure the overall performance of the approaches with an emphasis on the individual classes (Jakowicz and Shah, 2011). In Section 3.1, we demonstrated that the proportion of data representing individual classes varies strongly. We therefore weighted the F1-measure by this ratio and report the micro-averaged results over all datasets $\overline{F1}$. Given our focus on training a generalizable model, we deliberately did not focus on the performance variation in the individual datasets.

4.2 Results

In order to check whether our findings persist at least across the two learning algorithms, we did not aggregate the model performance samples but analyzed them for each algorithm separately. We therefore only have one independent variable, our feature groups, that affects the model performance. In order to keep p-value inflation low, we only compared the ten best performing models for each algorithm with respect to the F1-Measure. Note that even if the difference in performance between these models appears small, there are thus many worse models that were not explicitly listed.

Our samples generally do not fulfill the assumptions of normality and sphericity required by parametric tests for comparing more than two groups. Under the violation of these assumptions, non-parametric tests have more power and are less prone to outliers (Demsar, 2006). We therefore relied exclusively on the non-parametric tests suggested in literature: Friedman's test was used as non-parametric alternative to a repeated-measures one-way ANOVA, and Nemenyi's test[2] was used post-hoc as a replacement for Tukey's test.

In contrast to its parametric counterpart, Friedman's test is based on a ranking of the models induced by the performance measure, and therefore only relies implicitly on the latter. Each model is ranked from best to worst, with mean ranks being

[1]The respective similarity scores have been calculated on the whole document corpus after preprocessing.

[2]We chose Nemenyi's test because it is widely accepted in the machine learning community. A discussion of alternatives can be found in Herrera et al. (Herrera, 2008).

Feature Group	words(1000,1,2)	words(1000,1,3)	words(ALL,1,1)	words(5000,1,1)	words(100,1,1)	words(100,1,2)	words(100,1,3)	words(5000,1,3)	words(1000,1,1)	words(5000,1,1)
$\overline{F1}$	82.10	82.00	82.86	82.87	80.62	80.66	80.76	81.15	82.71	81.28

(a) LibLinear

Feature Group	words(1000,1,2)	words(1000,1,3)	words(1000,1,1)	words(5000,1,2)	chars(5000,2,3)	chars(5000,2,4)	chars(1000,2,4)	chars(1000,2,5)	chars(1000,2,3)	chars(5000,2,5)
$\overline{F1}$	80.10	79.56	80.10	78.09	78.01	80.27	80.22	79.73	79.86	80.48

(b) NaiveBayes

Table 4: Average F1-Measure $\overline{F1}$ for the ten best performing baseline feature groups

used in case of ties. The *Friedman statistic* is calculated by dividing the sum of squares of the mean ranks by the sum of squares error. For sufficiently many samples, the statistic follows a χ^2 distribution with $k-1$ degrees of freedom. The *q statistic* used in Nemenyi's test is similar to the one used by Tukey, but uses rank differences. It utilises the previous ranking from the Frieman test to calculate and relate the average ranks of two models, for each available pair. Two models are considered significantly different, if their difference in mean ranks exceeds a critical value, which varies for different significance levels. For a detailed description and examples of these tests, see (Jakowicz and Shah, 2011).

We illustrated the ranks and significant differences between the feature groups by means of the critical difference (CD) diagram. Introduced by Demsar (2006), this diagram lists the feature groups ordered by their rank, where lower rank numbers indicate higher performance. Feature groups are connected with bars if they are not significantly different, given $\alpha = 0.05$.

In the following, we will use shortcuts like *words(1000,1,2)* to denote the 1000 most frequent uni- and bigrams. The same applies for char-n-grams. Abbreviations can be found in Table 3.

4.2.1 Evaluation using LibLinear Classifier

We first evaluated which of our 20 baseline feature sets, as described in Section 3, lead to the best classification performance over different datasets. Notably, the ten best-performing approaches were all combinations of word-n-grams. Table 4 contains the average F-Measures for these approaches. The Friedman test indicated strong significant differences between the performances of these groups ($\chi^2_r(9) = 112.467, p < 0.001, \alpha = 0.01$). The subsequent Nemenyi test indicated strong significant pairwise differences between the performances of the models ($\alpha = 0.01$), with p-values listed in Table 2 in the supplementary.

Figure 1 illustrates the differences by means of a CD diagram: the approaches of using simple unigrams of the most frequent 5000 and all words provide the best results, i.e. they have the lowest rank. They are not significantly different from the 1000 most frequent word-uni and bigrams. Nevertheless, they are significantly better than all other baseline approaches.

This also applies to the char-n-gram approaches, that were not considered in this statistical comparison due to their inferior performance. It is important to note that the differences between the worst word-n-gram approaches and the best char-n-gram approaches could still be statistically non-significant.

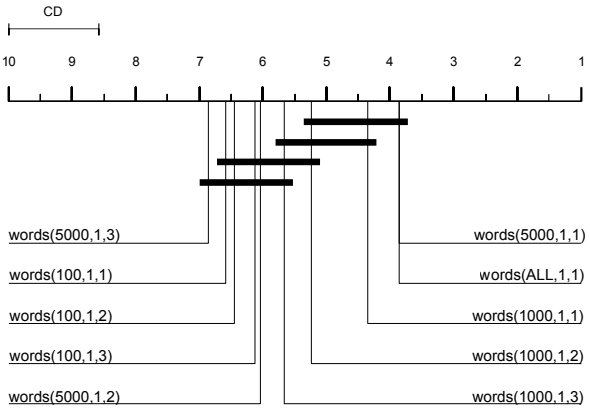


Figure 1: CD diagram with the ranks of the ten best performing baseline feature groups for LibLinear. Feature groups are connected if they are not significantly different ($\alpha = 0.05$).

The best performing baseline approach for LibLinear is using unigrams of the top 5000 words, i.e. *words(5000,1,1)*, with $\overline{F1} = 82.87$. We therefore picked this baseline feature group for the second part of our evaluation. We added each non-baseline feature individually to the selected baseline approach and compared the performances of these combinations and the non-extended baseline group. Table 6 lists the averaged F-Measure. When comparing the ten best-performing groups, the Friedman test showed strong significant differences between the performances of the models ($\chi^2_r(9) = 87.274, p < 0.001, \alpha = 0.01$). The Nemenyi test partly showed strong significant differences between the performances of the models (for the corresponding p-values see Table 3 in the supplementary). They are illustrated in the CD diagram in Figure 2. The tests indicate that adding *NER* and *NR_AT_MT* to the baseline approach provides the best performances with $\overline{F1} = 83.32$ and $\overline{F1} = 83.03$ respectively.

Finally, we evaluated the power set of these feature groups, i.e. we compared the individual groups and their combination. Table 5 contains the corresponding averaged F-Measures. For the resulting performance samples, the Friedman test showed strong significant differences between the models ($\chi^2_r(3) = 72.014, p < 0.001, \alpha = 0.01$). The Nemenyi test partly showed strong significant

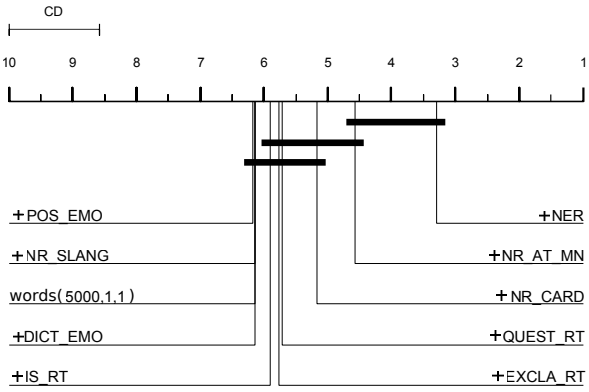


Figure 2: CD diagram with the ranks of the ten best performing feature groups for LibLinear, comprising the baseline and the baseline with an additional feature. Feature groups are connected if they are not significantly different ($\alpha = 0.05$).

Feature Group	words(5000,1,1)	+NER	+NER+NR_AT_MN	+NR_AT_MN
$\overline{F1}$	82.87	83.32	83.48	83.03

Table 5: Average F1-Measure $\overline{F1}$ for the power set of best performing feature groups and LibLinear.

differences ($\alpha = 0.001$), with p-values listed in Table 4 in the supplementary and illustrated in Figure 3. The diagram shows that the combination of *NER* and *NR_AT_MN* with the *words(5000,1,1)* baseline outperforms all other models with respect to F1 ($\overline{F1} = 83.48$), but does not differ significantly from the plain *NER* approach ($\overline{F1} = 83.32$). This combination gives us the final and best feature set for training a generalizing model over our datasets. As can be seen, the plain n-gram approach ($F1 = 82.87$) can be improved further by 0.5%.

4.2.2 Evaluation using Naive Bayes Classifier

In this section, we repeat the previous steps for the NB classifier. As baseline feature sets, we first evaluated the word-n-gram and char-n-gram approaches. The averaged F-Measures can be found in Table 4. The Friedman test showed strong significant differences between the performances of the models ($\chi^2_r(9) = 110.293, p < 0.001, \alpha = 0.01$). The Nemenyi test partly showed strong significant differences between the performances of the models (for the corresponding p-values see Table 1 in the supplementary). In contrast to the LibLinear classifier, using the 5000 most frequent combinations of two to five subsequent characters, i.e. *chars(5000,2,5)* provide the best F1 score ($\overline{F1} = 80.48$). Thus, char-n-grams outperform the word-n-gram approaches with respect to F1.

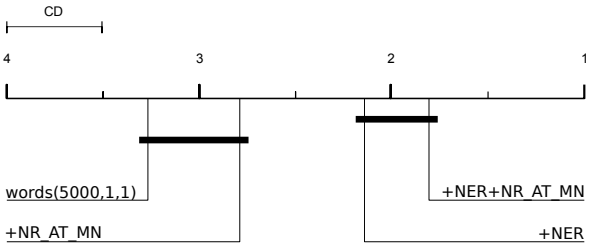


Figure 3: CD diagram with the ranks of the best baseline feature groups, complemented with a combination of the best performing feature sets, for LibLinear. Feature groups are connected if they are not significantly different ($\alpha = 0.05$).

The CD diagram in Figure 5 shows that using either the 5000 most frequent char-n-grams with a length of two to five and two to four respectively, the 1000 most frequent word-n-grams with a length of one and one to two respectively, and the 1000 most frequent char-n-grams with a length of two to four do not differ significantly. However, using either the 5000 most frequent char-n-grams with a length of two to five and two to four respectively significantly outperform all other baseline approaches. As a subsequent step, we added each single feature to *chars(5000,2,5)* as the best baseline approach to find if these provide better performance for the NB classifier. Table 6 contains the corresponding averaged F-Measures. Though the Friedman test indicated strong significant differences between the performances of the models ($\chi^2_r(9) = 22.209, p = 0.008, \alpha = 0.01$), the subsequent Nemenyi test did not indicate significant pairwise differences. We can therefore not repeat the third step of our evaluation, but infer that for a NB classifier, the plain char-n-gram-based approach is sufficient for training a generalizing model for our dataset.

The results indicate that LibLinear provides a better avg. performance ($\overline{F1} = 83.32$) when training a generalizing model, compared to the NB classifier ($\overline{F1} = 80.48$).

5 Conclusion and Future Work

In this paper, we compared the performance of different popular feature groups and classification algorithms for the task of training a generalizing model for incident type classification. We carefully selected the most popular feature groups from related work, and separately evaluated them

Feature Group	words(5000,1,1)	+DICT_EMO	+NER	+NR_CARD	+NR_AT_MN	+POS_EMO	+NR_SLANG	+EXCLA_RT	+QUEST_RT	+IS_RT
$\overline{F1}$	82.87	82.87	83.32	83.06	83.03	82.87	82.87	82.88	82.88	82.88

(a) LibLinear

Feature Group	chars(5000,2,5)	+DICT_EMO	+QUEST_RT	+NER	+NR_AT_MN	+NR_HASHTAG	+POS_EMO	+NR_SLANG	+NR_SENT	+EXCLA_RT
$\overline{F1}$	80.48	80.48	80.49	80.55	80.51	80.48	80.48	80.48	80.48	80.50

(b) NaiveBayes

Table 6: Average F1-Measure $\overline{F1}$ for the ten best performing combinations of the best baseline and an additional feature

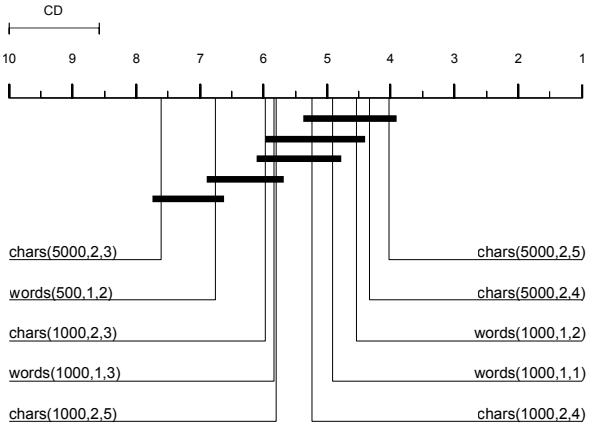


Figure 4: CD diagram with the ranks of the ten best performing baseline feature groups for Naive Bayes. Feature groups are connected if they are not significantly different ($\alpha = 0.05$).

Figure 5: Ranks of NB baseline feature groups.

for the LibLinear and NB classification algorithms on ten spatially and temporally diverse datasets. The resulting F1-measure samples indicate that training a generalizing model, i.e., a model that is applicable on previously unseen incident-related data, is still a challenging task. We found that LibLinear provides a better averaged performance compared to the NB classifier. More surprisingly, additional feature groups that are commonly used in related work do not necessarily outperform a plain n-gram-based approach. This highlights the need for other novel approaches for training generalizing classification models. Especially in the domain of incident detection and emergency management, our findings are important because less time consuming techniques showed nearly the same performance as sophisticated ones.

There are two main topics for our future work. First, we will investigate the performance of models generated with biased datasets on unfiltered datasets. This is relevant, if a technique like filtering is used to include more relevant class examples in a dataset than provided with an original sample – a necessary step to realize a labeled dataset for model learning of a rare-class task. Second, we will work on using novel features for the creation of generalized models. One example is the utilization of the Semantic Web to generate abstract features, utilizing a technique called Semantic Abstraction (Schulz et al., 2015a). Semantic Abstraction has shown to improve the generalization of tweet classification by deriving features from Linked Open Data and using location and temporal mentions.

References

Apoorv Agarwal, Boyi Xie, Ilia Vovsha, Owen Rambow, and Rebecca Passonneau. 2011. Sentiment analysis of twitter data. In *Proceedings of the Workshop on Languages in Social Media, LSM '11*, pages 30–38. ACL.

Puneet Agarwal, Rajgopal Vaithiyanathan, Saurabh Sharma, and Gautam Shroff. 2012. Catching the long-tail: Extracting local news events from twitter. In *Proceedings of the Sixth International Conference on Weblogs and Social Media, ICWSM 2012*. AAAI Press.

S. Carvalho, L. Sarmento, and R. J. F. Rossetti. 2010. Real-time sensing of traffic information in twitter messages. In *Proceedings of the 4th Workshop on Artificial Transportation Systems and Simulation ATSS, ITSC'10*, pages 19–22. IEEE Computer Society.

Richard Eckart de Castilho and Iryna Gurevych. 2014. A broad-coverage collection of portable nlp components for building shareable analysis pipelines. In *Proceedings OIAF4HLT at COLING 2014*, pages 1–11.

Janez Demsar. 2006. Statistical Comparisons of Classifiers over Multiple Data Sets. *Journal of Machine Learning Research*, 7:1–30.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of ACL*, pages 363–370.

Alec Go, Richa Bhayani, and Lei Huang. 2009. Twitter sentiment classification using distant supervision. In *Proceedings of the Workshop on Languages in Social Media, LSM '11*, pages 30–38. Association for Computational Linguistics.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. 2009. The weka data mining software: an update. *SIGKDD Explor. Newsl.*, 11(1):10–18.

Maram Hasanain, Tamer Elsayed, and Walid Magdy. 2014. Identification of answer-seeking questions in arabic microblogs. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 1839–1842. ACM.

Francisco Herrera. 2008. An Extension on Statistical Comparisons of Classifiers over Multiple Data Sets for all Pairwise Comparisons. *Journal of Machine Learning Research*, 9:2677–2694.

Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin. 2003. A practical guide to support vector classification. Technical report, Department of Computer Science, National Taiwan University.

Muhammad Imran, Shady Elbassuoni, Carlos Castillo, Fernando Diaz, and Patrick Meier, 2013. *Extracting information nuggets from disaster- Related messages in social media*, pages 791–801. Karlsruher Institut fur Technologie (KIT).

Nathalie Jakowicz and Mohak Shah. 2011. *Evaluating Learning Algorithms. A Classification Perspective*. Cambridge University Press, Cambridge.

Sarvnaz Karimi, Jie Yin, and Cecile Paris. 2013. Classifying microblogs for disasters. In *Proceedings of the 18th Australasian Document Computing Symposium, ADCS '13*, pages 26–33. ACM.

Karen Kipper, Anna Korhonen, Neville Ryant, and Martha Palmer. 2006. Extending verbnet with novel verb classes. In *Proceedings LREC'06*.

VI Levenshtein. 1966. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady*, 10:707.

Rui Li, Kin Hou Lei, Ravi Khadiwala, and Kevin Chen-Chuan Chang. 2012. Tedas: A twitter-based event detection and analysis system. In *Proceedings of the 28th International Conference on Data Engineering, ICDE'12*, pages 1273–1276. IEEE Computer Society.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schtze., 2009. *An Introduction to Information Retrieval*, pages 117–120. Cambridge University Press.

Olutobi Owoputi, Chris Dyer, Kevin Gimpel, Nathan Schneider, and Noah A. Smith. 2013. Improved part-of-speech tagging for online conversational text with word clusters. In *In Proceedings of NAACL*.

Jason D. Rennie, Lawrence Shih, Jaime Teevan, and David R. Karger. 2003. Tackling the poor assumptions of naive bayes text classifiers. In Tom Fawcett and Nina Mishra, editors, *International Conference on Machine Learning (ICML-03)*, pages 616–623. AAAI Press.

David Ratcliffe Robert Power, Bella Robinson. 2013. Finding fires with twitter. In *Australasian Language Technology Association Workshop*, pages 80–89. Association for Computational Linguistics.

Takeshi Sakaki and M Okazaki. 2010. Earthquake shakes twitter users: real-time event detection by social sensors. In *Proceedings of the 19th international conference on World Wide Web, WWW '10*, pages 851–860. ACM.

Axel Schulz and Frederik Janssen. 2014. What is good for one city may not be good for another one: Evaluating generalization for tweet classification based on semantic abstraction. In CEUR, editor, *Proceedings of the Fifth Workshop on Semantics for Smarter Cities a Workshop at the 13th International Semantic Web Conference*, volume 1280, pages 53–67.

Axel Schulz, Petar Ristoski, and Heiko Paulheim. 2013. I see a car crash: Real-time detection of small scale incidents in microblogs. In *ESWC'13*, pages 22–33.

Axel Schulz, Christian Guckelsberger, and Frederik Janssen. 2015a. Semantic abstraction for generalization of tweet classification: An evaluation on incident-related tweets. In *Semantic Web Journal: Special Issue on Smart Cities*.

Axel Schulz, Benedikt Schmidt, and Thorsten Strufe. 2015b. Small-scale incident detection based on microposts. In *Proceedings of the 26th ACM Conference on Hypertext and Social Media*. ACM.

N. Wanichayapong, W. Pruthipunyaskul, W. Pattara-Atikom, and P. Chaovalit. 2011. Social-based traffic information extraction and classification. In *Proceedings of the 11th International Conference on ITS Telecommunications, ITST'11*, pages 107–112. IEEE Computer Society.

Michael J. Wise. 1996. Yap3: Improved detection of similarities in computer program and other texts. In *SIGCSEB: SIGCSE Bulletin (ACM Special Interest Group on Computer Science Education*, pages 130–134. ACM Press.

Flexible Domain Adaptation for Automated Essay Scoring Using Correlated Linear Regression

Peter Phandi[1] **Kian Ming A. Chai**[2] **Hwee Tou Ng**[1]

[1] Department of Computer Science, National University of Singapore
{peter-p,nght}@comp.nus.edu.sg

[2] DSO National Laboratories
ckianmin@dso.org.sg

Abstract

Most of the current automated essay scoring (AES) systems are trained using manually graded essays from a specific prompt. These systems experience a drop in accuracy when used to grade an essay from a different prompt. Obtaining a large number of manually graded essays each time a new prompt is introduced is costly and not viable. We propose domain adaptation as a solution to adapt an AES system from an initial prompt to a new prompt. We also propose a novel domain adaptation technique that uses Bayesian linear ridge regression. We evaluate our domain adaptation technique on the publicly available Automated Student Assessment Prize (ASAP) dataset and show that our proposed technique is a competitive default domain adaptation algorithm for the AES task.

1 Introduction

Essay writing is a common task evaluated in schools and universities. In this task, students are typically given a prompt or essay topic to write about. Essay writing is included in high-stakes assessments, such as Test of English as a Foreign Language (TOEFL) and Graduate Record Examination (GRE). Manually grading all essays takes a lot of time and effort for the graders. This is what Automated Essay Scoring (AES) systems are trying to alleviate.

Automated Essay Scoring uses computer software to automatically evaluate an essay written in an educational setting by giving it a score. Work related to essay scoring can be traced back to 1966 when Ellis Page created a computer grading software called Project Essay Grade (PEG). Research on AES has continued through the years. The recent Automated Student Assessment Prize (ASAP) Competition[1] sponsored by the Hewlett Foundation in 2012 has renewed interest on this topic. The agreement between the scores assigned by state-of-the-art AES systems and the scores assigned by human raters has been shown to be relatively high. See Shermis and Burstein (2013) for a recent overview of AES.

AES is usually treated as a supervised machine learning problem, either as a classification, regression, or rank preference task. Using this approach, a training set in the form of human graded essays is needed. However, human graded essays are not readily available. This is perhaps why research in this area was mostly done by commercial organizations. After the ASAP competition, research interest in this area has been rekindled because of the released dataset.

Most of the recent AES related work is prompt-specific. That is, an AES system is trained using essays from a specific prompt and tested against essays from the same prompt. These AES systems will not work as well when tested against a different prompt. Furthermore, generating the training data each time a new prompt is introduced will be costly and time consuming.

In this paper, we propose domain adaptation as a solution to this problem. Instead of hiring people to grade new essays each time a new prompt is introduced, domain adaptation can be used to adapt the old prompt-specific system to suit the new prompt. This way, a smaller number of training essays from the new prompt is needed. In this paper, we propose a *novel* domain adaptation technique based on Bayesian linear ridge regression.

The rest of this paper is organized as follows. In Section 2, we give an overview of related work on AES and domain adaptation. Section 3 describes the AES task and the features used. Section 4 presents our novel domain adaptation algorithm.

[1] http://www.kaggle.com/c/asap-aes

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 431–439,
Lisbon, Portugal, 17-21 September 2015.

Section 5 describes our data, experimental setup, and evaluation metric. Section 6 presents and discusses the results. We conclude in Section 7.

2 Related Work

We first introduce related work on automated essay scoring, followed by domain adaptation in the context of natural language processing.

2.1 Automated Essay Scoring

Since the first AES system, Project Essay Grade, was created in 1966, a number of commercial systems have been deployed. One such system, e-rater (Attali and Burstein, 2004), is even used as a replacement for the second human grader in the Test of English as a Foreign Language (TOEFL) and Graduate Record Examination (GRE). Other AES commercial systems also exist, such as IntelliMetric[2] and Intelligent Essay Assessor (Foltz et al., 1999).

AES is generally considered as a machine learning problem. Some work, such as PEG (Page, 1994) and e-rater, considers it as a regression problem. PEG uses a large number of features with regression to predict the human score. e-rater uses natural language processing (NLP) techniques to extract a smaller number of complex features, such as grammatical error and lexical complexity, and uses them with stepwise linear regression (Attali and Burstein, 2004). Others like (Larkey, 1998) take the classification approach. (Rudner and Liang, 2002) uses Bayesian models for classification and treats AES as a text classification problem. Intelligent Essay Assessor uses Latent Semantic Analysis (LSA) (Landauer et al., 1998) as a measure of semantic similarity between essays. Other recent work uses the preference ranking based approach (Yannakoudakis et al., 2011; Chen and He, 2013).

In this paper, we also treat AES as a regression problem, following PEG and e-rater. We use regression because the range of scores of the essays could be very large and a classification approach does not work well in this case. It also allows us to model essay scores as continuous values and scale them easily in the case of different score ranges between the source essay prompt and the target essay prompt.

The features used differ among the systems, ranging from simple features (e.g., word length, essay length, etc) to more complex features (e.g., grammatical errors). Some of these features are generic in the sense that they could apply to all kinds of prompts. Such features include the number of spelling errors, grammatical errors, lexical complexity, etc. Others are prompt-specific features such as bag of words features.

[2]http://www.vantagelearning.com/products/intellimetric/

2.2 Domain Adaptation

The knowledge learned from a single domain might not be directly applicable to another domain. For example, a named entity recognition system trained on labeled news data might not perform as well on biomedical texts (Jiang and Zhai, 2007). We can solve this problem either by getting labeled data from the other domain, which might not be available, or by performing domain adaptation.

Domain adaptation is the task of adapting knowledge learned in a source domain to a target domain. Various approaches to this task have been proposed and used in the context of NLP. Some commonly used approaches include EasyAdapt (Daumé III, 2007), instance weighting (IW) (Jiang and Zhai, 2007), and structural correspondence learning (SCL) (Blitzer et al., 2006).

We can divide the approaches of domain adaptation into two categories based on the availability of labeled target data. The case where a small number of labeled target data is available is usually referred to as *supervised* domain adaptation (such as EasyAdapt and IW). The case where no labeled target domain data is available is usually referred to as *unsupervised* domain adaptation (such as SCL). In our work, we focus on *supervised* domain adaptation.

Daumé III (2007) described a domain adaptation scheme called EasyAdapt which makes use of feature augmentation. Suppose we have a feature vector $\boldsymbol{x}$ in the original feature space. This scheme will map this instance using the mapping functions $\Phi^{\mathrm{s}}(\boldsymbol{x})$ and $\Phi^{\mathrm{t}}(\boldsymbol{x})$ for the source and target domain respectively, where

$$\Phi^{\mathrm{s}}(x) = \langle \boldsymbol{x}, \boldsymbol{x}, \mathbf{0} \rangle$$
$$\Phi^{\mathrm{t}}(x) = \langle \boldsymbol{x}, \mathbf{0}, \boldsymbol{x} \rangle,$$

and $\mathbf{0}$ is a zero vector of length $|\boldsymbol{x}|$. This adaptation scheme is attractive because of its simplicity and ease of use as a pre-processing step, and also because it performs quite well despite its simplicity. It has been used in various NLP tasks such

as word segmentation (Monroe et al., 2014), machine translation (Green et al., 2014), word sense disambiguation (Zhong et al., 2008), and short answer scoring (Heilman and Madnani, 2013). Our work is an extension of this scheme in the sense that our work is a generalization of EasyAdapt.

3 Automated Essay Scoring

This section describes the Automated Essay Scoring (AES) task and the features we use for the task.

3.1 Task Description

In AES, the input to the system is a student essay, and the output is the score assigned to the essay. The score assigned by the AES system will be compared against the human assigned score to measure their agreement. Common agreement measures used include Pearson's correlation, Spearman's correlation, and quadratic weighted Kappa (QWK). We use QWK in this paper, which is also the evaluation metric in the ASAP competition.

3.2 Features and Learning Algorithm

We model the AES task as a regression problem and use Bayesian linear ridge regression (BLRR) as our learning algorithm. We choose BLRR as our learning algorithm so as to use the correlated BLRR approach which will be explained in Section 4. We use an open source essay scoring system, EASE (Enhanced AI Scoring Engine)[3], to extract the features. EASE is created by one of the winners of the ASAP competition so the features they use have been proven to be robust. Table 1 gives the features used by EASE.

Useful n-grams are defined as n-grams that separate good scoring essays and bad scoring essays, determined using the Fisher test (Fisher, 1922). Good scoring essays are essays with a score greater than or equal to the average score, and the remainder are considered as bad scoring essays. The top 201 n-grams with the highest Fisher values are then chosen as the bag features. We perform the calculation of useful n-grams separately for source and target domain essays, and join them together using set union during the domain adaptation experiment. This is done to prevent the system from choosing only n-grams from the source domain as the useful n-grams, since the number of source domain essays is much larger than the target domain essays.

EASE uses NLTK (Bird et al., 2009) for POS tagging and stemming, aspell for spellchecking, and WordNet (Fellbaum, 1998) to get the synonyms. Correct POS tags are generated using a grammatically correct text (provided by EASE). The POS tag sequences not included in the correct POS tags are considered as bad POS. EASE uses scikit-learn (Pedregosa et al., 2011) for extracting unigram and bigram features. For linear regression, a constant feature of value one is appended for the bias.

4 Correlated Bayesian Linear Ridge Regression

First, consider the single-task setting. Let $\boldsymbol{x} \in \mathbb{R}^p$ be the feature vector of an essay. p represents the number of features in $\boldsymbol{x}$. The generative model for an observed real-valued score y is

$$\begin{aligned} &\alpha \sim \Gamma(\alpha_1, \alpha_2), && \lambda \sim \Gamma(\lambda_1, \lambda_2), \\ &\boldsymbol{w} \sim \mathcal{N}(\mathbf{0}, \lambda^{-1} I), && f(\boldsymbol{x}) \stackrel{\text{def}}{=} \boldsymbol{x}^{\mathrm{T}} \boldsymbol{w}, \\ &y \sim \mathcal{N}(f(\boldsymbol{x}_i), \alpha^{-1}). \end{aligned}$$

Here, α and λ are Gamma distributed hyper-parameters of the model; $\boldsymbol{w} \in \mathbb{R}^p$ is the Normal distributed weight vector of the model; f is the latent function that returns the "true" score of an essay represented by $\boldsymbol{x}$ by linear combination; and y is the noisy observed score of $\boldsymbol{x}$.

Now, consider the two-task setting, where we indicate the source task and the target task by superscripts s and t. Given an essay with feature vector $\boldsymbol{x}$, we consider its observed scores y^{s} and y^{t} when evaluated in task s and task t separately. We have scale hyper-parameters α and λ sampled as before. In addition, we have the correlation ρ between the two tasks. The generative model relating the two tasks is

$$\begin{aligned} \rho &\sim p_\rho, \\ \boldsymbol{w}^{\mathrm{t}}, \boldsymbol{w}^{\mathrm{s}} &\sim \mathcal{N}(\mathbf{0}, \lambda^{-1} I), \\ f^{\mathrm{t}}(\boldsymbol{x}) &\stackrel{\text{def}}{=} \boldsymbol{x}^{\mathrm{T}} \boldsymbol{w}^{\mathrm{t}}, \\ f^{\mathrm{s}}(\boldsymbol{x}) &\stackrel{\text{def}}{=} \rho \boldsymbol{x}^{\mathrm{T}} \boldsymbol{w}^{\mathrm{t}} + (1-\rho^2)^{1/2} \boldsymbol{x}^{\mathrm{T}} \boldsymbol{w}^{\mathrm{s}}, \\ y^{\mathrm{t}} &\sim \mathcal{N}(f^{\mathrm{t}}(\boldsymbol{x}), \alpha^{-1}), \\ y^{\mathrm{s}} &\sim \mathcal{N}(f^{\mathrm{s}}(\boldsymbol{x}), \alpha^{-1}), \end{aligned}$$

where p_ρ is a chosen distribution over the correlation; and $\boldsymbol{w}^{\mathrm{t}}$ and $\boldsymbol{w}^{\mathrm{s}}$ are the weight vectors of the

[3] https://github.com/edx/ease

Feature Type	Feature Description
Length	Number of characters Number of words Number of commas Number of apostrophes Number of sentence ending punctuation symbols (".", "?", or "!") Average word length
Part of speech (POS)	Number of bad POS n-grams Number of bad POS n-grams divided by the total number of words in the essay
Prompt	Number of words in the essay that appears in the prompt Number of words in the essay that appears in the prompt divided by the total number of words in the essay Number of words in the essay which is a word or a synonym of a word that appears in the prompt Number of words in the essay which is a word or a synonym of a word that appears in the prompt divided by the total number of words in the essay
Bag of words	Count of useful unigrams and bigrams (unstemmed) Count of stemmed and spell corrected useful unigrams and bigrams

Table 1: Description of the features used by EASE.

target and the source tasks respectively, and they are identically distributed but independent. In this setting, it can be shown that the correlation between latent scoring functions for the target and the source tasks is ρ. That is,

$$\mathbb{E}(f^{\mathrm{t}}(\boldsymbol{x})f^{\mathrm{s}}(\boldsymbol{x}')) = \lambda^{-1}\rho\boldsymbol{x}^{\mathrm{T}}\boldsymbol{x}'. \qquad (1)$$

This, in fact, is a generalization of the EasyAdapt scheme, for which the correlation ρ is fixed at 0.5 [(Daumé III, 2007), see eq. 3]. Two other common values for ρ are 1 and 0; the former corresponds to a straightforward concatenation of the source and target data, while the latter is the shared-hyperparameter setting which shares α and λ between the source and target domain. Through adjusting ρ, the model traverses smoothly between these three regimes of domain adaptation.

EasyAdapt is attractive because of its (frustratingly) ease of use via encoding the correlation within an expanded feature representation scheme. In the same way, the current setup can be achieved readily by the expanded feature representation

$$\begin{aligned}\Phi^{\mathrm{t}}(\boldsymbol{x}) &= \langle \boldsymbol{x}, \mathbf{0}_p \rangle\,, \\ \Phi^{\mathrm{s}}(\boldsymbol{x}) &= \left\langle \rho\boldsymbol{x}, (1-\rho^2)^{1/2}\boldsymbol{x} \right\rangle\end{aligned} \qquad (2)$$

in $\mathbb{R}^{2p}$ for the target and the source tasks. Associated with this expanded feature representation is the weight vector $\boldsymbol{w} \stackrel{\text{def}}{=} (\boldsymbol{w}^{\mathrm{t}}, \boldsymbol{w}^{\mathrm{s}})$ also in $\mathbb{R}^{2p}$. As we shall see in Section 4.1, such a representation eases the estimation of the parameters.

The above model is related to the multi-task Gaussian Process model that has been used for joint emotion analysis (Beck et al., 2014). There, the *intrinsic coregionalisation model* (ICM) has been used with *squared-exponential covariance function*. Here, we use the simpler *linear covariance function* (Rasmussen and Williams, 2006), and this leads to Bayesian linear ridge regression. There are two reasons for this choice. The first is that linear combination of carefully chosen features, especially lexical ones, usually gives good performance in NLP tasks. The second is in the preceding paragraph: an intuitive feature expansion representation of the domain adaptation process that allows ease of parameter estimation.

The above model is derived from the Cholesky decomposition

$$\begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ \rho & (1-\rho^2)^{1/2} \end{pmatrix} \begin{pmatrix} 1 & \rho \\ 0 & (1-\rho^2)^{1/2} \end{pmatrix}$$

of the desired correlation matrix that will eventually lead to equation (1). Other choices are possible, as long as equation (1) is satisfied. However, the current choice has the desired property that the $\boldsymbol{w}^{\mathrm{t}}$ portion of the combined weight vector is di-

rectly interpretable as the weights for the features in the target domain.

4.1 Maximum Likelihood Estimation

We estimate the parameters (α, λ, ρ) of the model using penalized maximum likelihood. For α and λ, the gamma distributions are used. For ρ, we impose a distribution with density $p_\rho(\rho) = 1 + a - 2a\rho$, $a \in [-1, 1]$. This distribution is supported only in $[0, 1]$; negative ρs are not supported because we think that negative transfer of information from source to target domain prompts in this essay scoring task is improbable. In our application, we slightly bias the correlations towards zero with $a = 1/10$ in order to ameliorate spurious correlations.

For the training data, let there be n^{t} examples in the target domain and n^{s} in the source domain. Let X^{t} (resp. X^{s}) be the n^{t}-by-p (resp. n^{s}-by-p) design matrix for the training data in the target (resp. source) domain. Let $\boldsymbol{y}^{\mathrm{t}}$ and $\boldsymbol{y}^{\mathrm{s}}$ be the corresponding observed essay scores. The expanded feature matrix due to equation (2) is

$$X \stackrel{\text{def}}{=} \begin{pmatrix} X^{\mathrm{t}} & 0 \\ \rho X^{\mathrm{s}} & (1-\rho^2)^{1/2} X^{\mathrm{s}} \end{pmatrix}.$$

Similarly, let $\boldsymbol{y}$ be the stacking of $\boldsymbol{y}^{\mathrm{t}}$ and $\boldsymbol{y}^{\mathrm{s}}$. Let $K \stackrel{\text{def}}{=} \lambda^{-1} X X^{\mathrm{T}} + \alpha^{-1} I$, which is also known as the Gramian for the observations. The log marginal likelihood of the training data is (Rasmussen and Williams, 2006)

$$L = -\frac{1}{2}\boldsymbol{y}^{\mathrm{T}} K^{-1} \boldsymbol{y} - \frac{1}{2}\log|K| - \frac{n^{\mathrm{t}} + n^{\mathrm{s}}}{2}\log 2\pi.$$

This is penalized to give L_p by adding

$$\begin{gathered}(\alpha_1 - 1)\log(\alpha) - \alpha_2\alpha + \alpha_1 \log \alpha_2 - \log\Gamma(\alpha_1) \\ +(\lambda_1 - 1)\log(\lambda) - \lambda_2\lambda + \lambda_1 \log \lambda_2 - \log\Gamma(\lambda_1) \\ + \log(1 + a - 2a\rho).\end{gathered}$$

The estimation of these parameters is then done by optimising L_p. In our implementation, we use scikit-learn for estimating α and λ in an inner loop, and we use gradient descent for estimating ρ in the outer loop using

$$\frac{\partial L_p}{\partial \rho} = \frac{1}{2}\operatorname{tr}\left(\left(\gamma\gamma^T - K^{-1}\right)\frac{\partial K}{\partial \rho}\right) - \frac{2a}{1 + a - 2a\rho},$$

where $\gamma \stackrel{\text{def}}{=} K^{-1} y$ and

$$\frac{\partial K}{\partial \rho} = \lambda^{-1}\begin{pmatrix} 0 & X^{\mathrm{t}}(X^{\mathrm{s}})^{\mathrm{T}} \\ X^{\mathrm{s}}(X^{\mathrm{t}})^{\mathrm{T}} & 0 \end{pmatrix}.$$

4.2 Prediction

We report the mean prediction as the score of an essay. This uses the mean weight vector $\bar{\boldsymbol{w}} = \lambda^{-1} X^{\mathrm{T}} K^{-1} \boldsymbol{y} \in \mathbb{R}^{2p}$, which may be partitioned into two vectors $\bar{\boldsymbol{w}}^{\mathrm{t}}$ and $\bar{\boldsymbol{w}}^{\mathrm{s}}$, each in $\mathbb{R}^p$. The prediction of a new essay represented by $\boldsymbol{x}_*$ in the target domain is then given by $\boldsymbol{x}_*^{\mathrm{T}} \bar{\boldsymbol{w}}^{\mathrm{t}}$.

5 Experiments

In this section, we will give a brief description of the dataset we use, describe our experimental setup, and explain the evaluation metric we use.

5.1 Data

We use the ASAP dataset[4] for our domain adaptation experiments. This dataset contains 8 prompts of different genres. The average length of the essays differs for each prompt, ranging from 150 to 650 words. The essays were written by students ranging in grade 7 to grade 10. All the essays were graded by at least 2 human graders. The genres include narrative, argumentative, or response. The prompts also have different score ranges, as shown in Table 2.

We pick four pairs of essay prompts to perform our experiments. In each experiment, one of the essay prompts from the pair will be the source domain and the other essay prompt will be the target domain. The essay set pairs we choose are $1 \rightarrow 2$, $3 \rightarrow 4$, $5 \rightarrow 6$, and $7 \rightarrow 8$, where the pair $1 \rightarrow 2$ denotes using prompt 1 as the source domain and prompt 2 as the target domain, for example. These pairs are chosen based on the similarities in their genres, score ranges, and median scores. The aim is to have similar source and target domains for effective domain adaptation.

5.2 Experimental Setup

We use 5-fold cross validation on the ASAP training data for evaluation. This is because the official test data of the competition is not released to the public. We divide the target domain data randomly into 5 folds. One fold is used as the test data, while the remaining four folds are collected together and then sub-sampled to obtain the target-domain training data. The sizes of the sub-sampled target-domain training data are 10, 25, 50 and 100, with the larger sets containing the smaller sets. All essays from the source domain are used.

[4] https://www.kaggle.com/c/asap-aes/data

Set	# Essays	Genre	Avg len	Score	
				Range	Median
1	1,783	ARG	350	2–12	8
2	1,800	ARG	350	1–6	3
3	1,726	RES	150	0–3	1
4	1,772	RES	150	0–3	1
5	1,805	RES	150	0–4	2
6	1,800	RES	150	0–4	2
7	1,569	NAR	250	0–30	16
8	723	NAR	650	0–60	36

Table 2: Selected details of the ASAP data. For the genre column, ARG denotes *argumentative* essays, RES denotes *response* essays, and NAR denotes *narrative* essays.

Our evaluation considers the following four ways in which we train the AES model:

SourceOnly Using essays from the source domain only;

TargetOnly Using 10, 25, 50, and 100 sampled essays from the target domain only;

SharedHyper Using correlated Bayesian linear ridge regression (BLRR) with ρ fixed to 0 on source domain essays and sampled essays from the target domain.

EasyAdapt As SharedHyper, but with $\rho = 0.5$;

Concat As SharedHyper, but with ρ fixed to 1.0;

ML-ρ Using correlated BLRR with ρ maximizing the likelihood of the data.

Since the source and target domain may have different score ranges, we scale the scores linearly to range from -1 to 1. When predicting on the test essays, the predicted scores of our system will be linearly scaled back to the target domain score range and rounded to the nearest integer.

We build upon scikit-learn's implementation of BLRR for our learning algorithm. To ameliorate the effects of different scales of features, we normalize the features: length, POS, and prompt features are linearly scaled to range from 0 to 1 according to the training data; and the feature values for bag-of-words features are $\log(1 + \text{count})$ instead of the actual counts.

We use scikit-learn version 0.15.2, NLTK version 2.0b7, and aspell version 0.60.6.1 in this experiment. The BLRR code (`bayes.py`) in scikit-learn is modified to obtain valid likelihoods for use in the outer loop for estimating ρ. We use scikit-learn's default value for the parameters α_1, α_2, λ_1, and λ_2 which is 10^{-6}.

Set #	QWK scores		
	BLRR	SVM	Human
1	0.761	0.781	0.721
2	0.606	0.621	0.814
3	0.621	0.630	0.769
4	0.742	0.749	0.851
5	0.784	0.782	0.753
6	0.775	0.771	0.776
7	0.730	0.727	0.721
8	0.617	0.534	0.629

Table 3: In-domain experimental results.

5.3 Evaluation Metric

Quadratic weighted Kappa (QWK) is used to measure the agreement between the human rater and the system. We choose to use this evaluation metric since it is the official evaluation metric of the ASAP competition. Other work such as (Chen and He, 2013) that uses the ASAP dataset also uses this evaluation metric. QWK is calculated using

$$\kappa = 1 - \frac{\sum_{i,j} w_{i,j} O_{i,j}}{\sum_{i,j} w_{i,j} E_{i,j}},$$

where matrices O, $(w_{i,j})$, and E are the matrices of observed scores, weights, and expected scores respectively. Matrix $O_{i,j}$ corresponds to the number of essays that receive a score i by the first rater and a score j by the second rater. The weight entries are $w_{i,j} = (i-j)^2/(N-1)^2$, where N is the number of possible ratings. Matrix E is calculated by taking the outer product between the score vectors of the two raters, which are then normalized to have the same sum as O.

6 Results and Discussion

In-domain results for comparison First, we determine indicative upper bounds on the QWK scores using Bayesian linear ridge regression (BLRR). To this end, we perform 5-fold cross validation by training and testing within each domain. This is also done with linear support vector machine (SVM) regression to confirm that BLRR is a competitive method for this task. In addition, since the ASAP data has at least 2 human annotators for each essay, we also calculate the human agreement score. The results are shown in Table 3. We see that the BLRR scores are close to the the human agreement scores for prompt 1 and

Method	QWK Scores n^{t} =10	25	50	100
1 → 2				
SourceOnly	0.434			
TargetOnly	0.069	0.169	0.279	0.395
SharedHyper	0.158	0.218	0.332	0.390
EasyAdapt	0.425	0.422	0.442	0.467
Concat	**0.484**	**0.507**	**0.529**	**0.545**
ML-ρ	0.463	0.457	0.492	0.510
3 → 4				
SourceOnly	0.522			
TargetOnly	0.117	0.398	0.545	0.626
SharedHyper	0.113	0.350	0.487	0.575
EasyAdapt	0.461	0.541	0.589	0.628
Concat	**0.594**	**0.611**	0.617	0.638
ML-ρ	0.593	0.609	**0.618**	**0.646**
5 → 6				
SourceOnly	0.187			
TargetOnly	0.416	0.506	0.554	0.608
SharedHyper	0.380	0.500	0.544	0.600
EasyAdapt	0.553	0.621	0.652	0.698
Concat	**0.649**	**0.689**	**0.708**	**0.722**
ML-ρ	0.539	0.662	0.680	0.713
7 → 8				
SourceOnly	0.171			
TargetOnly	0.290	0.381	0.426	0.477
SharedHyper	0.302	0.383	0.444	0.484
EasyAdapt	**0.594**	**0.616**	0.605	0.610
Concat	0.332	0.362	0.396	0.463
ML-ρ	0.586	0.607	**0.613**	**0.621**

Table 4: QWK scores of the six methods on four domain adaptation experiments, ranging from using 10 target-domain essays (second column) to 100 target-domain essays (fifth column). The scores are the averages over 5 folds. Setting $a \rightarrow b$ means the AES system is trained on essay set a and tested on essay set b. For each set of six results comparing the methods, the best score is boldfaced and the second-best score is underlined.

prompts 5 to 8, but fall short by 10% to 20% for prompts 2 to 4. We also see that BLRR is comparable to linear SVM regression, giving almost the same performance for prompts 4 to 7; slightly poorer performance for prompts 1 to 3; and much better performance for prompt 8. The subsequent discussion in this section will refer to the BLRR scores in Table 3 for in-domain scores.

Importance of domain adaptation The results of the domain adaptation experiments are tabulated in Table 4, where the best scores are boldfaced and the second-best scores are underlined. As expected, for pairs $1 \rightarrow 2$, $3 \rightarrow 4$, and $5 \rightarrow 6$, all the scores are below their corresponding upper bounds from the in-domain setting in Table 3. However, for pair $7 \rightarrow 8$, the QWK score for domain adaptation with 100 target essays outperforms that of the in-domain, albeit only by 0.4%. This can be explained by the small number of essays in prompt 8 that can be used in both the in-domain and domain adaptation settings, and that domain adaptation additionally involves prompt 7 which has more than twice the number of essays; see column two in Table 2. Hence, domain adaptation is effective in the context of small number of target essays with large number of source essays. This can also be seen in Table 4 where we have simulated small number of target essays with sizes 10, 25, 50, and 100. When we compare the scores of TargetOnly against the best scores and second-best scores, we find that domain adaptation is effective and important in improving the QWK scores.

By the above argument alone, one might have thought that an overwhelming large number of source domain essays was sufficient for the target domain. However, this is not true. When we compare the scores of SourceOnly against the best scores and second-best scores, we find that domain adaptation again improves the QWK scores. In fact, with just 10 additional target domain essays, effective domain adaptation can improve over SourceOnly for all target domains 2, 4, 6, and 8 respectively.

This is the first time where the effects of domain adaptation are shown in the AES task. In addition, the large improvement with a small number of additional target domain essays in $5 \rightarrow 6$ and $7 \rightarrow 8$ suggests the high domain-dependence nature of the task: *learning on one essay prompt and testing on another should be strongly discouraged.*

Contributions by target-domain essays It is instructive to understand why domain adaptation is important for AES. To this end, we estimate the contribution of bag-of-words features to the overall prediction by computing the ratio

$$\frac{\sum_{i \text{ over bag-of-words features}} w_i^2}{\sum_{i \text{ over all features}} w_i^2}$$

using weights learned in the in-domain setting; see Table 1 for the complete list of features. For domains 2, 4, 6, and 8, which are the target domains in the domain adaptation experiments, these ratios are 0.37, 0.73, 0.69, and 0.93. The ratios for the other four domains are similarly high. This shows that bag-of-words features play a significant role in the prediction of the essay scores. We examine the number of bag-of-words features that 100 additional target domain essays would add to SourceOnly; that is, we compare the bag-of-words features for SourceOnly with those of SharedHyper, EasyAdapt, Concat, and ML-ρ for $n^{\mathrm{t}} = 100$. The numbers of these additional features, averaged over the five folds, are 269, 351, 377, and 291 for target domains 2, 4, 6, and 8 respectively. In terms of percentages, these are 67%, 87%, 94%, and 72% more features over SourceOnly. Such a large number of additional bag-of-words features contributed by target-domain essays, together with the fact that these features are given high weights, means that target-domain essays are important.

Comparing domain adaptation methods We now compare the four domain adaptation methods: SharedHyper, EasyAdapt, Concat, and ML-ρ. We recall that the first three are constrained cases of the last by fixing ρ to 0, 0.5, and 1 respectively. First, we see that SharedHyper is a rather poor domain adaptation method for AES, because it gives the lowest QWK score, except for the case of using 25, 50, and 100 target essays in adapting from prompt 7 to prompt 8, where it is better than Concat. In fact, its scores are generally close to the TargetOnly scores. This is unsurprising, since in SharedHyper the weights are effectively not shared between the target and source training examples: only the hyper-parameters α and λ are shared. This is a weak form of information sharing between the target and source domains. Hence, we expect this to perform suboptimally when the target and source domains bear more than spurious relationship, which is indeed the case here because we have chosen the source and target domain pairs based on their similarities, as described in Section 5.1.

We now focus on EasyAdapt, Concat, and ML-ρ, which are the better domain adaptation methods from our results. We see that ML-ρ either gives the best or second-best scores, except for the one case of $5 \rightarrow 6$ with 10 target essays. In comparison, although Concat performs consistently well for $1 \rightarrow 2$, $3 \rightarrow 4$, and $5 \rightarrow 6$, its QWK scores for $7 \rightarrow 8$ are quite poor and even lower than those of TargetOnly for 25 or more target essays. In contrast to Concat, EasyAdapt performs well for $7 \rightarrow 8$ but not so well for the other three domain pairs.

Let us examine the reason for contrasting results between EasyAdapt and Concat to appreciate the flexibility afforded by ML-ρ. The ρ estimated by ML-ρ for the pairs $1 \rightarrow 2$, $3 \rightarrow 4$, $5 \rightarrow 6$, and $7 \rightarrow 8$ with 100 target essays are 0.81, 0.97, 0.76, and 0.63 averaged over five folds. The lower estimated correlation ρ for $7 \rightarrow 8$ means that prompt 7 and prompt 8 are not as similar as the other pairs are. In such a case as this, Concat, which in effect considers the target domain to be exactly the same as the source domain, can perform very poorly. For the other three pairs which are more similar, the correlation of 0.5 assumed by EasyAdapt is not strong enough to fully exploit the similarities between the domains. Unlike Concat and EasyAdapt, ML-ρ has the flexibility to allow it to traverse effectively between the different degrees of domain similarity or relatedness based on the source domain and target domain training data. In view of this, we consider *ML-ρ to be a competitive default domain adaptation algorithm for the AES task*.

In retrospect of our present results, it can be obvious why prompts 7 and 8 are not as similar as we would have hoped for more effective domain adaptation. Both prompts ask for narrative essays, and these by nature are very prompt-specific and require words and phrases relating directly to the prompts. In fact, referring to a previous discussion on the *contributions by target-domain essays*, we see that weights for the bag-of-words features for prompt 8 contribute a high of 93% of the total. When we examine the bag-of-words features, we see that prompt 7 (which is to write about patience) contributes only 19% to the bag-of-words features of prompt 8 (which is to write about laughter) in the in-domain experiment. This means that 81% of the bag-of-words features, which are important to narrative essays, must be contributed by the target-domain essays relating to prompt 8. Future work on domain adaptation for AES can explore chosing the prior p_ρ on ρ to better reflect the nature of the essays involved.

7 Conclusion

In this work, we investigate the effectiveness of using domain adaptation when we only have a small number of target domain essays. We have shown that domain adaptation can achieve better results compared to using just the small number of target domain data or just using a large amount of data from a different domain. As such, our research will help reduce the amount of annotation work needed to be done by human graders to introduce a new prompt.

Acknowledgments

This research is supported by Singapore Ministry of Education Academic Research Fund Tier 2 grant MOE2013-T2-1-150.

References

Yigal Attali and Jill Burstein. 2004. Automated essay scoring with e-rater® v. 2.0. Technical report, Educational Testing Service.

Daniel Beck, Trevor Cohn, and Lucia Specia. 2014. Joint emotion analysis via multi-task Gaussian processes. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python*. O'Reilly Media.

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*.

Hongbo Chen and Ben He. 2013. Automated essay scoring by maximizing human-machine agreement. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*.

Hal Daumé III. 2007. Frustratingly easy domain adaptation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*.

Christiane Fellbaum, editor. 1998. *WordNet: An Electronic Lexical Database*. Bradford.

Ronald A Fisher. 1922. On the interpretation of χ^2 from contingency tables, and the calculation of p. *Journal of the Royal Statistical Society*.

Peter W Foltz, Darrell Laham, and Thomas K Landauer. 1999. The intelligent essay assessor: Applications to educational technology. *Interactive Multimedia Electronic Journal of Computer-Enhanced Learning*.

Spence Green, Daniel Cer, and Christopher D. Manning. 2014. An empirical comparison of features and tuning for phrase-based machine translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*.

Michael Heilman and Nitin Madnani. 2013. Ets: domain adaptation and stacking for short answer scoring. In *Proceedings of the Seventh International Workshop on Semantic Evaluation*.

Jing Jiang and ChengXiang Zhai. 2007. Instance weighting for domain adaptation in NLP. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*.

Thomas K Landauer, Peter W Foltz, and Darrell Laham. 1998. An introduction to latent semantic analysis. *Discourse Processes*.

Leah S Larkey. 1998. Automatic essay grading using text categorization techniques. In *Proceedings of the 21st International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Will Monroe, Spence Green, and Christopher D Manning. 2014. Word segmentation of informal Arabic with domain adaptation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*.

Ellis Batten Page. 1994. Computer grading of student prose, using modern concepts and software. *The Journal of Experimental Education*.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*.

Carl Edward Rasmussen and Christopher K. I. Williams. 2006. *Gaussian Processes for Machine Learning*. MIT Press.

Lawrence M Rudner and Tahung Liang. 2002. Automated essay scoring using Bayes' theorem. *The Journal of Technology, Learning and Assessment*.

Mark D. Shermis and Jill Burstein, editors. 2013. *Handbook of Automated Essay Evaluation: Current Applications and New Directions*. Routledge.

Helen Yannakoudakis, Ted Briscoe, and Ben Medlock. 2011. A new dataset and method for automatically grading ESOL texts. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*.

Zhi Zhong, Hwee Tou Ng, and Yee Seng Chan. 2008. Word sense disambiguation using OntoNotes: An empirical study. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*.

Show Me Your Evidence – an Automatic Method for Context Dependent Evidence Detection

Ruty Rinott[1], Lena Dankin[1], Carlos Alzate[2], Mitesh M. Khapra[3],
Ehud Aharoni[1], Noam Slonim[1]

[1]IBM Research - Haifa, Mount Carmel, Haifa, 31905, Israel,
[2]IBM Research - Ireland, Damastown Industrial Estate, Dublin 15, Ireland,
[3]IBM Research - Bangalore, India,
{rutyr,lenad,aehud,noams}@il.ibm.com
carlos.alzate@ie.ibm.com mikhapra@in.ibm.com

Abstract

Engaging in a debate with oneself or others to take decisions is an integral part of our day-to-day life. A debate on a topic (say, *use of performance enhancing drugs*) typically proceeds by one party making an assertion/claim (say, *PEDs are bad for health*) and then providing an evidence to support the claim (say, *a 2006 study shows that PEDs have psychiatric side effects*). In this work, we propose the task of automatically detecting such evidences from unstructured text that support a given claim. This task has many practical applications in decision support and persuasion enhancement in a wide range of domains. We first introduce an extensive benchmark data set tailored for this task, which allows training statistical models and assessing their performance. Then, we suggest a system architecture based on supervised learning to address the evidence detection task. Finally, promising experimental results are reported.

1 Introduction

In recent years there has been a growing interest in the area of argumentation mining (Green et al., 2014; Cardie et al., 2015; Wells, 2014). Part of this awakening is the The Debater™ project[1] whose goal is to develop technologies that will assist humans to debate and reason, e.g., by automatically suggesting arguments relevant to an examined topic. The minimal definition of such an argument (Walton, 2009) is a set of statements, made up of three parts – a claim (aka conclusion, proposition), a set of evidence (aka premises), and an inference from the evidence to the claim. Needless to say, evidence plays a critical role in a persuasive argument.

In most debate related skills, such as natural language understanding and generation, humans currently have an inherent advantage over a machine. However, in the ability to provide high quality and diverse evidence, machines have a very promising potential, being able to swiftly process large quantities of information. Nonetheless, since most of the relevant information is represented by unstructured text, successfully exploiting these resources requires the ability to identify evidence in free text. This is exactly the focus of our work. Specifically, we formally define the task of evidence detection, introduce an architecture for attacking this problem, and demonstrate its performance over dedicated manually labeled data.

Before defining the task formally, we introduce three concepts which will be used throughout this paper. These concepts were earlier defined in (Aharoni et al., 2014) and we use the same definitions here. **Topic**: a short phrase that frames the discussion. **Claim**: a general, concise statement that directly supports or contests the topic. **Context Dependent Evidence (CDE)**: a text segment that directly supports a claim in the context of the topic. The first three rows of Table 1 show examples of a topic, a claim and CDE.

For the purpose of this work, we assume that we are given a concrete topic, a relevant claim, and potentially relevant documents, provided either manually or by automatic methods (Cartright et al., 2011; Levy et al., 2014). Our task, which we term Context Dependent Evidence Detection (CDED), is to automatically pinpoint CDE within these documents. We further require that a detected CDE is reasonably well phrased, and easily understandable in the given context, so that it can be instantly and naturally used to support the claim in a discussion. Table 1 gives examples of valid CDE (V) and non-valid CDE (X) according to the definition mentioned above.

It is well recognized that one can support a claim using different types of evidence (Rieke and Sillars, 2001; Seech, 2008). Furthermore, for different use cases, different evidence types could be more suitable. Correspondingly, we develop a classification approach that is able to identify and distinguish between three common evidence types (Rieke and Sillars, 1984; Seech, 2008):

- **Study** Results of a quantitative analysis of data, given as numbers, or as conclusions. (Table 1 S1);

[1]http://researcher.ibm.com/researcher/view_group.php?id=5443

[2]Note ibuprofen is considered a PED

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 440–450,
Lisbon, Portugal, 17-21 September 2015.

Topic: Use of performance enhancing drugs (PEDs) in professional sports	
Claim A: PEDs can be harmful to athletes health	
S1: A 2006 study examined 320 athletes for psychiatric side effects induced by anabolic steroid use. The study found a higher incidence of mood disorders in these athletes compared to a control group.	V
S2: The International Agency for Research on Cancer classifies androgenic steroids as "Probably carcinogenic to humans."	V
S3: Rica Reinisch, a triple Olympic champion and world record-setter at the Moscow Games in 1980, has suffered numerous miscarriages and recurring ovarian cysts following drug abuse.	V
S4: The UN estimates that there are more than 50 million regular users of heroin, cocaine and synthetic drugs.	X
S5: FDA does not approve ibuprofen[2] for babies younger than six months due to risk of liver damage.	X
S6: Doping can ultimately damage your health.	X
Claim B: Use of PED is inline with the spirit of sport	
S7: Professor Savulescu, a philosopher and bioethicist, believes that biological manipulation embodies the sports spirit: the capacity to improve ourselves on the basis of reason and judgment.	V

Table 1: Examples for defined concepts. The V/X indicates if the candidate is a CDE to the claim above it, according to our definition.

- **Expert** Testimony by a person / group / committee / organization with some known expertise / authority on the topic. (Table 1 S2, S7);
- **Anecdotal** A description of an episode(s), centered on individual(s) or clearly located in place and/or in time. (Table 1 S3);

Examining the valid and non-valid CDEs in Table 1 it should be clear that the distinction between them is often quite subtle. For example, it is possible that a piece of text has the characteristics of a certain evidence type, but does not support the claim (see S4 in Table 1). It is also possible that a piece of text supports the claim, but is irrelevant in the context of the topic (see S5 in Table 1). It could also be the case that a piece of text entails the claim, but adds no new information to support it (see S6 in Table 1).

We present here a pipeline architecture, relying on supervised learning, to handle the different aspects of CDED which shows promising results over a variety of topics. We demonstrate that the proposed solution and features can generalize well, namely that models learned over different topics can perform reasonably well on an entirely new topic. On average, for a significant fraction of claims the proposed system succeeds to propose relevant CDE amongst its top 4 predictions, and properly determines the evidence type. Furthermore we show that we are able to automatically pinpoint claims for which the performance of the system are of even greater quality, enabling the user to obtain higher precision for these claims.

We believe that the ability to automatically provide evidence for given claims will have many practical uses, helping layman and professionals in different domains, to reach decisions and prepare for discussions, from a lawyer presenting a case in court, to a politician considering a new policy.

2 Related work

CDED is related to several other information retrieval and NLP tasks. Probably the closest of which is the relatively unexplored task of Evidence Retrieval (ER) (Cartright et al., 2011; Bellot et al., 2013). However, while ER focus is on identifying whole documents, in CDED the goal is to pinpoint a typically much shorter text segment which can be used directly to support a claim. Furthermore, ER is typically performed for factual assertions, while in CDED one may want to consider a wider range of claim types (Rieke and Sillars, 2001), cf. claim B in Table 1.

Another important line of related work is the Textual Entailment (TE) framework (Dagan et al., 2009; Glickman et al., 2005). A text fragment, T, is said to entail a textual hypothesis H if the truth of H can be most likely inferred from T. While TE can be an important component in a CDED approach, and perhaps vice versa, the tasks are quite different. Namely, the goal of TE is detecting semantic inference while the goal of CDED is to provide evidence which can enhance the persuasion of a claim. For example, common instances of TE are rephrases or summarizations of a sentence, however they cannot serve to support a claim within a discussion, as they merely repeat it (Table 1, S6). On the other hand, an anecdotal story may have strong emotional impact that will effectively support a claim during a discussion, although the truth of the claim cannot be inferred from such evidence. Furthermore, similar to ER, TE focuses only on factual assertions, while we focus on a wider range of claims (Rieke and Sillars, 2001), cf. claim B in Table 1.

Question answering (QA) (Dang et al., 2007) also has some similar aspects to the proposed task, although aiming at a very different goal, which is to provide an explicit – typically unique and concise – answer, to a question.

The proposed CDED task should be seen as another contribution in the emerging field of argumentation mining, with several important distinct characteristics. Previous works suggested extracting full ar-

	Topics	Claims	Articles with CDE	CDE	avg. % of claims with CDE	avg. # CDE per claim
Study	30	1587	136	1018	31 (22)	2.2 (0.9)
Expert	37	1702	214	1896	46 (22)	1.9 (0.8)
Anecdotal	22	1137	70	382	17 (11)	2.0 (1.6)
Total	39	1734	274	3057	60 (17)	2.9 (3.7)

Table 2: 'Topics' indicate the number of topics included for each CDE type. This determines the number of claims considered for each type. The next columns indicate the number of articles in which at least one CDE was found; the total number of CDE detected for each type; the average percent of claims for which at least one CDE was found; and for these claims, the average number of CDE found. Note that the total number of CDE is not a simple sum of the CDE per type, as CDE can be assigned with more than one type. Standard deviations of distribution across topics are given in parenthesis where relevant.

guments (Mochales Palau and Moens, 2009), analyzing argument structure (Peldszus, 2014), and identifying relations between arguments (Cabrio and Villata, 2012; Ghosh et al., 2014). Other works focused on specific domains such as evidence-based legal documents (Mochales Palau and Moens, 2011; Ashley and Walker, 2013), online debates (Cabrio and Villata, 2012; Boltužić and Šnajder, 2014), and product reviews (Villalba and Saint-Dizier, 2012; Yessenalina et al., 2010). In addition, some works based on machine-learning techniques, used the same topic in training and testing (Rosenfeld and Kraus, 2015; Boltužić and Šnajder, 2014), relying on features from the topic itself in identifying arguments. In contrast, here, we focus on detecting an essential constituent of an argument – the evidence – rather then detecting whole arguments, or detecting other argument parts like claims (Levy et al., 2014; Lippi and Torroni, 2015). In addition, we do not limit ourselves to a particular domain, nor assume that the topic of the discussion is known in advance. Finally, we aim to pinpoint evidence in a clearly defined context, given by the pre–specified claim. Thus, the developed system should not only find pieces of text that have general evidence characteristics but further identify which of these candidates can be used to support a specific claim. Hence, as we demonstrate in our results, an essential part of a CDED system should be dedicated to model and assess the semantic relation of a candidate evidence to the given claim and topic.

3 Data

Since CDED is a new and rather complicated task, it is beneficial to examine and understand the nature of the data before moving on to developing a working solution. We therefore start by explaining the manual data annotation process, and several important observations over the resulting data.

To train and assess the classifiers in our system we rely on data collected by the procedure described in (Aharoni et al., 2014). Briefly, given a topic and a corresponding relevant claim, extracted from a Wikipedia article by human annotators, the annotators were asked to mark corresponding evidence – text segments supporting the claim. To limit the amount of time annotators spend on these tasks, labeling was restricted to the article in which the claim was found. The task was split into two stages. First, in the detection stage, five annotators read the article, and mark all CDE candidates they locate. Next, in the confirmation stage all the candidates suggested by the annotators are presented to another set of five annotators, which confirm or reject each candidate, and determine the type(s) of accepted candidates. Candidates which were confirmed by the majority of the annotators are considered CDE, and are assigned the type(s) suggested by at least three annotators.

A total of 547 Wikipedia articles associated with 58 different topics were annotated through this procedure. The topics were selected at random from *Debatabase*[3] covering a wide variety of domains, from atheism to the role of wind power in future energy supply. Out of these topics, 39 were selected at random for training and testing the classifiers included in the system. We refer to these data as the *train and test data*. The remaining 19 topics were used for tuning various feature parameters, and developing auxiliary classifiers, as described in Section 5. We refer to these data as the *held-out* data.

In the 39 topics comprising the train and test data, a total of $3,057$ distinct CDE were found in 274 articles (See Table 2). The data is highly unbalanced towards non CDE sentences. For example, for type Study, only 31% of the claims had at least one CDE. Of these 31% claims, on average, a claim was associated with 2.17 CDEs. Further, on average these 2.17 CDEs together span 1.5 sentences, whereas an average article in our data consists of 150 sentences. In other words, even for claims with at least one CDE of type Study, on average only 2% of the sentences in the claim's article are part of such Study CDE.

In general, CDE in the examined data varied in length from less than a sentence to more than a paragraph. However, 90% of these CDE were composed of segments of up to three sentences within the same paragraph. Furthermore, in 95% of the cases, CDEs were

[3] http://idebate.org/debatabase

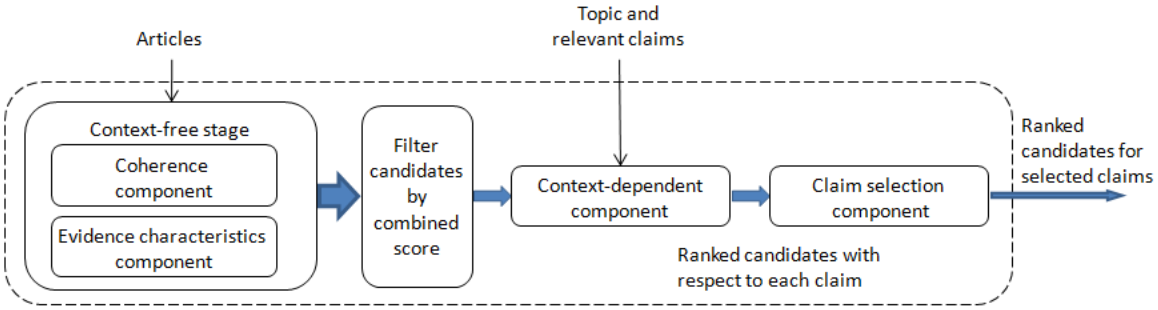


Figure 1: Schematic description of the CDED system proposed in this work.

comprised of full sentences. Examining CDEs that start or end mid-sentence, reveals that in most cases the CDE is more concise in these boundaries, but is still a valid CDE when extending the boundaries to include the full sentence. We therefore decided not to address this issue here, and we extend all CDE boundaries to full sentences.

Apparently CDE of type Study and type Expert are far more common in Wikipedia compared to Anecdotal CDE. We expect this distribution to change in other less scientifically inclined corpora.

Finally, the variance between different topics was substantial, as depicted in Table 2 (refer to the standard deviations mentioned in parenthesis). For example, the percentage of claims with Expert CDE varies from 10% in the topic *banning gambling* to 95% in the topic *US responsibility for the Mexican drug wars*. This observed variability obviously adds to the difficulty and complexity of the task.

In the experiments reported in this paper, out of the 39 topics in the train and test data, we exclude from the evaluation of each type, topics that had less than three CDE of that type. This leaves a total of 30, 37, and 22 topics for types Expert, Study, and Anecdotal, respectively.

The current work is the first to report results over these CDE data, which are more than 4 times larger compared to the data released in (Aharoni et al., 2014). These data are now freely available for research purposes [4].

4 System Architecture

The input to our system is a topic, a set of related articles and a set of relevant claims detected within these articles. Given this input, our system provides the user with a ranked list of candidate CDEs, originating from the text in the claim's article, for an automatically selected subset of the input claims.

In general, we observe that a text segment should satisfy three criteria to be considered CDE of a specific type. It must be coherent; it must have characteristics of the relevant Evidence type; and finally, of course, it should support the claim.

In addition to these observations, we note that a priori, we do not expect all claims to be supported by all CDE types (Park and Cardie, 2014). For example, opinion claims like claim B in Table 1 are expected to be less supported by Study evidence compared to factual claims, like claim A in Table 1. Moreover, as evident from Table 2, many claims do not have any associated CDE in the same article. Thus, the system performance may naturally improve if it will propose candidate CDE of a particular type, only to an automatically identified subset of the input claims.

Based on these observations, we are led to suggest an architecture which approaches CDED via a pipeline of modular components. Each of these components relies upon the results of its precedents, and is specifically designed to address a single aspect of those mentioned above. The resulting architecture is depicted in Figure 1. Briefly, in the proposed architecture, the first two components are *context-free*, i.e., focused on the general characteristics of a candidate, still not taking into account the context of the claim, nor the topic. The third component is *context-dependent*, considering the relation of the candidate to the claim and topic. Finally, the fourth component aims to identify a subset of the claims for which CDE will be proposed.

We consider all text segments composed of one, two or three consecutive sentences, included within the same paragraph as candidates (see Section 5 for more details). Given a set of such *candidate CDEs* – or simply, *candidates* – the first component, termed the *coherence component*, estimates the coherence of each candidate. For example, consider CDE S1 in Table 1. A candidate which includes only the second sentence is incoherent, as it includes critical unresolved anaphora, that cannot be understood without the previous sentence. In parallel, the second component, termed the *evidence characteristics component*, estimates the extent to which the candidate's statistical signature matches that of the examined evidence type. For example, if no quantitative analysis of data is reported, the candidate typically cannot be considered Study evidence, regardless of the claim and topic. Next, we only

[4] https://www.research.ibm.com/haifa/dept/vst/mlta_data.shtml

retain candidates for which the average score of the first two components was relatively high, aiming to further focus our attention on the most promising candidates.

The retained candidates are then considered by the *context-dependent* component which aims to determine if the examined candidate indeed supports the provided claim in the context of the topic. Thus, this component ranks all retained candidates with respect to each claim. Finally, the *claim selection* component aims to rank all input claims, according to the probability that CDEs are indeed found amongst top-ranking candidate for the claim.

Dividing the overall task into sub-tasks has several benefits. First, it allows training each component over its most suitable data, in which the signal of the relevant features is easier to capture. For example, many of the features for the *context-dependent* component aim to determine the semantic relatedness between the claim and a candidate. If one would have tried to tackle the entire CDED task simultaneously, the training data for this component would have been masked by many candidates that are highly related to the claim, although are not CDE – e.g., definitions of some aspects of the claim. These candidates would have blurred the signal that should be captured by the semantic relatedness features, as they represent candidates with negative labels that are nonetheless semantically related to the claim. By separating the tasks, we allow the *context-dependent* component to avoid this inherent difficulty, and train over much cleaner data.

Second, our pipeline allows efficient handling of the CDED task in terms of run time. Semantic relatedness features are often relatively complex and demanding in terms of run time. The significant filtering done after the *context-free* stage, reduces the number of candidates for which we have to calculate these features.

Finally, we note that some of the modular components we develop as part of the pipeline might be of interest by themselves. For example, context-free evidence detection might be useful in cases in which the claim and topic are not defined (Lippi and Torroni, 2015).

Naturally, we expect that different evidence types will have different characteristics. For example, numbers are expected to be more common in CDE of type Study compared to CDE of type Expert. Anecdotal CDE is perhaps expected to be less semantically related to the corresponding claim, as it may have a more associative relation to the claim, compared to CDE of types Study or type Expert. Correspondingly, all components are developed, trained, and assessed, independently for each CDE type.

In summary, the full flow of our system upon receiving a new topic with associated articles and claims, is as follows:

1. All articles are split into sentences, and all consecutive segments up to three sentences within a paragraph are generated as candidates.
2. Each candidate is assigned a score by the two components in the *context-free* stage and their scores are averaged. [5]
3. A dynamic programming algorithm selects a complete coverage of the article by non-overlapping candidates with the maximal average *context-free* score. The rest of the candidates are discarded.
4. The remaining candidates across all articles are sorted and only the top 15% of these candidates are retained. [6]
5. For each claim, the *context-dependent* component ranks all retained candidates within the claim's article with respect to the claim.
6. The claim selection component considers all claims and the candidates ranked with respect to each claim and assigns a score per claim. If the claim–score is below a pre–computed threshold, no candidate CDE will be presented for that claim.

All components are based on a Logistic Regression (LR) classifier, and the class probability is used as the candidate score.

5 Technical approach

In this section, we provide more technical details for each of the components in our architecture.

5.1 Coherence component

This component aims to score a candidate according to its coherence. For example, a candidate with an unresolved anaphora, or one that breaks a quotation in the middle, is expected to receive a relatively low score. As mentioned, this component considers all text segments composed of 1–3 consecutive sentences included within the same paragraph. This decision is based on the observation that such segments cover 90% of CDE in the labeled data. Reaching a full coverage requires examining segments up to 25 sentences, which would vastly increase run time, for a relatively small gain. Thus, for example, for a single paragraph with five sentences, our system will examine a total of $5 + 4 + 3 = 12$ candidates. For an article including 30 such paragraphs, a total of 360 candidates will be considered. During training, segments that conform to a labeled CDE were considered positive examples, while segments that overlap a labeled CDE, but either include additional sentence(s), or exclude part of the CDE sentences were considered negative examples.

Dominant features for this classifier included: presence of incomplete quotes; presence of contrast related conjunctive adverbs – e.g., `however`,

[5] With additional training data, we might be able to learn a more sophisticated function to combine both scores.

[6] This percentage was determined according to performance on the held-out data set. We have also experimented with methods where the threshold is score–based rather than percentage–based, which gave similar results.

nevertheless; segment length; and presence of unresolved co-references.

5.2 Evidence characteristics component

This component aims to estimate to what extent a candidate represents evidence of a certain type. The train and test data for this component consisted of all text segments composed of 1-3 consecutive sentences, included within the same paragraph. Positive examples are all labeled CDE of the corresponding evidence type. Negative examples are all candidates that do not overlap labeled CDE of the relevant type, including CDE of other types.

The dominant features for the classifier used in this component relied on the following mechanisms:

- **Lexicons** – including external lexicons (the Harvard IV-4 dictionary) and manually and automatically compiled in-house lexicons. Specifically, for each evidence type, we manually compiled a lexicon of words characterizing this type by looking at examples from the held-out data. This resulted with high–precision / low–recall lexicons. For example, for type Expert we used a lexicon of words describing persons and organizations that may have some relevant expertise, such as: economist, philosopher, court. In addition, we used the held-out data to automatically learn wider lexicons of words that are significantly associated with each type. All the in-house lexicons are described in detail in the supplementary material.

- **Named Entity Recognition (NER)**. We used the Stanford NER (Finkel et al., 2005) to extract named entities such as person and organization, and an in-house NER (Lally et al., 2012) to extract more fine grained categories such as "educational organization" and "leader".

- **Patterns**. We used regular expressions to represent features like: does that candidate contain a quote; does it contain a citation; does it contain numeric quantitative results. In addition, we generated complex regular expressions which combine the above lexicons with NER results to capture patterns indicative of different types. For example, the pattern [Person/organization, 0 to 10 wildcard words, an opinion verb - such as believe, conclude, etc.] was highly indicative of Expert evidence (cf. Table 1 S7).

- **Subjectivity classifier**. We manually labeled $1,750$ sentences, selected at random from articles in the held-out data, as either subjective or objective. Next, each sentence was represented by a concatenation of two feature vectors – (i) a bag-of-words representation, limited to a handcrafted subjectivity lexicon containing 100 words; (ii) a bag-of-patterns representation based on patterns observed as frequent in the subjective sentences, detected by a modification of the SPM algorithm (Srikant and Agrawal, 1996). An LR classifier was then trained over the labeled sentences.

5.3 Context-dependent component

The goal of this component is to estimate whether a candidate can be used to support a claim while discussing the given topic. The training data for this component are [topic / claim / CDE] triplets. Triplets in which the CDE and claim were linked in the labeled data – namely, the CDE was identified as evidence for the claim – were considered as positive examples. Negative examples were generated by combining claims and CDEs detected in the same topic and article, but that were not linked in our labeled data.

The features for the classifier used in this component can be conceptually divided into four types: (i) Semantic relatedness between the candidate and the claim (ii) Semantic relatedness between text related to the candidate and the claim (iii) Relative location of the candidate with respect to the claim and (iv) sentiment–agreement between the candidate and the claim.

In general, we rely on two methods to assess the semantic relatedness between two texts. The first is based on the cosine similarity between TF-IDF vectors representing each text. Before constructing the TF-IDF vectors each text is augmented with acronym expansions, and lexical relations (including antonym, derivationally related and pertainym) from WordNet (Miller, 1995). The second, relies on the average cosine similarity between the Word2Vec (Mikolov et al., 2013) representation of all pairs of words in the two texts, where in each pair one word is taken from the first text and the other word from the second.

For each of these two methods, we consider the semantic relatedness between the claim and: Specified slots in the candidate as detected by an in-house slot grammar parser (McCord, 1990; McCord et al., 2012); The entire candidate text; The header/sub-header of the section/subsection containing the candidate; Titles of citations referred to from the candidate.

5.4 Claim selection component

The goal of this component is to rank all claims according to the probability that the claim's article includes CDE of the relevant type, associated with the claim. The training data consisted of all claims, where positive examples included claims for which at least one CDE of the relevant type existed in the labeled data and negative examples included all remaining claims.

A thresholding mechanism on the component score is used to determine the claims for which candidates will be presented. This threshold was selected by optimizing the F1 score over the set of held-out topics.

The features used by this component exploited three types of information:

- **Claim properties**: We used the held-out data to

generate two types of lexicons. The first lexicon is generated separately per evidence type. It includes claim words that were found to be significantly associated with positive examples, namely with claims for which CDE were found. For example, for type Study, this lexicon included words such as `lead`, `result`, `development` and `significant`. The second lexicon aimed to characterize words that are significantly associated with factual claims vs. non–factual claims, with the expectation that certain evidence types might be more/less common for each of these two claim categories. For this, 550 randomly selected claims were annotated as factual/non–factual. Words identified as characterizing factual claims included `increase`, `important`, and `relate`, while words like `natural`, `freedom`, and `right` were found dominant for non-factual claims.

- **Claim's relevance to topic and article**: We expect that when an article's main topic is highly related to the claim, it will more likely include CDE for that claim. Similarly we expect that for claims at the heart of the topic, CDE is more likely to be provided. These properties are assessed by measuring the semantic relatedness between (i) the claim and the content of the claim's article and (ii) the claim and topic.

- **Properties of claim's article** : Specifically, we mainly consider the scores provided by the context-dependent component to all candidates examined in the claim's article. If the observed scores are relatively high/low, we expect the article to be more/less likely to include evidence of the considered type. Various statistics of these scores, such as the maximum score and the standard deviation are used as features aiming to capture this intuition.

6 Experimental Results

6.1 Evaluation

We evaluated our approach using the Leave-One-Out schema: for every topic, we trained the classifiers using the claims and associated CDE in all other topics and then applied the resulting models to the left out topic.

In general, we consider a candidate as true-positive if it includes all sentences included in the CDE and no additional sentences. However, for our analysis it is also interesting to separate between (i) errors in selecting the segment boundaries and (ii) errors of down the line components that are affected by these errors. Thus, we also include the *overlap* measure where we consider a candidate as true-positive if at least one sentence within it overlaps a sentence in a labeled CDE.

Our final assessment measure is the mean reciprocal rank (MRR), that is the inverse of the rank of the first CDE detected for a particular claim, averaged over all claims selected by the *claim selection component.* This is motivated by the observation that in most practical use cases, it is usually more important to be able to support many claims, than to provide all the CDE available for a single claim. We define the MRR of a claim with no CDE (errors of the claim selection component) to be 0.

Finally, we report the macro-averaged results over the different topics, that is all topics have the same weight regardless the amount of labeled claims and labeled CDE detected for them. The rational behind this is that we wish to ensure that our system does reasonably well across all topics examined. We note that micro-averaging gave overall similar results.

6.2 Comparison to Baselines

To assess the necessity and contribution of the different components we compare our full pipeline to partial pipelines, where some of the component are disabled or replaced by simple baselines. These baselines are described below.

First, we consider the **No Context-Free Stage (NCFS)** baseline which aims to assess the contribution of the *context-free* stage by skipping this stage, and passing all candidates directly to the *context-dependent* component.

Next, we consider the **Basic Claim Selection (BCS)** baseline which replaces the claim selection component. It ranks claims according to the top score of the candidate CDE for the claim. A threshold was selected on top of the training data, such that the average percentage of claims passing the threshold is equal to the average percentage of claims with CDEs in the labeled data.

Since, to the best of our knowledge, this is the first work to address CDED, there is no prior-art to compare our results to. However, to ensure that this task is indeed empirically different from related tasks, and demands a specialized pipeline to handle, we compare with two baselines that are often used in related tasks.

The **BM25** basline handles CDED as an IR task, where the claim represents the query, and all CDE candidates represent the documents in a standard IR setting. After pre-processing, which includes tokenization, stop word removal, and stemming (Porter, 1997) we use BM25 (Robertson et al., 1996) to rank all relevant candidates according to their similarity to the query, namely to the input claim.

The **W2V** baseline handles CDED as a purely semantic relatedness task using state of the art semantic relatedness measure of Word2Vec (Mikolov et al., 2013). Thus, we use the average cosine similarity between the Word2Vec representations of all words in a given candidate to all words in the claim, to rank all relevant candidates with respect to each claim.

Type	MRR				MRR overlap			
	Pipeline	NCFS	W2V	BM25	Pipeline	NCFS	W2V	BM25
Study	0.37	0.19	0.09	0.14	0.51	0.39	0.24	0.23
Expert	0.41	0.29	0.28	0.15	0.58	0.52	0.50	0.24
Anecdotal	0.18	0.04	0.04	0.04	0.31	0.11	0.11	0.11

Table 3: Macro-averaged MRR for each CDE type. Only claims with CDE in the labeled data were considered in these results.

6.3 Results

We start by assessing the proposed pipeline prior to the claim selection component. Table 3 reports the MRR following the context-dependent component when filtering out claims for which no CDE were found in the labeled data.

Impact of context free stage: Comparing the pipeline performance to the baseline using only the context dependent component (NCFS baseline), the results indicate the necessity of the context-free stage in our pipeline. That is, assessing the coherence of candidates, as well as their evidence characteristics, seems to be essential to properly address CDED. In particular, the fact that the gain is observed both in the MRR measure and in the MRR-overlap measure suggests that both the context-free components are valuable.

Impact of context dependent stage: Comparing the NCFS basline to W2V and BM25 baselines shows that for type Study, the context-dependent component alone still has an advantage over a single semantic relatedness feature. Observing feature weights learned by the LR classifier, we estimate that much of this advantage is due to also taking into consideration semantic relatedness of the claim to texts related to the candidate, namely the header of the section containing the candidate and titles of citations referred to from the candidate.

For types Expert and Anecdotal the performance of the context-dependent component are similar to those of the W2V baseline. For type Expert, this suggests that most of the signal in the context-dependent component comes from semantic relatedness between the claim and candidate CDE. Results for type Anecdotal are significantly lower. This was somewhat expected, given the smaller size of Anecdotal data available to train our classifiers (Table 2). The declined performance of the W2V and BM25 baselines for this type, further suggests that the semantic relatedness of CDE and claims for this type are less direct.

Impact of detecting segment boundaries: Comparing the *overlap* MRR measure to the exact MRR highlights that identifying the correct segment boundaries is still a challenge, and once we improve this aspect, we can expect a significant improvement in the results.

Impact of claim selection component: We next turn to assess the contribution of the claim selection component. Table 4 compares the final MRR results – at the end of the pipeline – for claims selected by the claim selection component, vs. claims selected by the BCS baseline. Additionally, to demonstrate the value of claim selection in general, we add results when considering all claims. For types Expert and Study the claim selection component shows a clear advantage over the baselines. Furthermore, the improved performance is achieved when passing a higher percentage of claims than the BCS baseline (34% vs 31% for Study and 52% vs 46% for Expert, Figure 2). Admittedly, for Anecdotal CDE the performance of claim selection are poor. For this component the small sample size for Anecdotal CDE was even more acute – there were only 151 claims with CDE of type Anecdotal – thus few positive examples to train this component.

Type	Pipeline	BCS	All claims
Study	0.25	0.16	0.12
Expert	0.34	0.23	0.20
Anecdotal	0.04	0.05	0.03

Table 4: Macro-averaged MRR over: 1) claims selected by the claim selection component, 2) claims selected by basic claim selection, and 3) all claims.

Recall that the claim selection component's threshold was tuned over the held-out data to optimize the F1 measure with respect to claims with/without CDE. However, for some applications one may favor higher precision at the expense of providing candidate CDEs for less claims. Figure 2 shows that indeed, for type Study, considering more strict thresholds of the claim selection component monotonically improves the system's overall precision, as reflected by the improved MRR. Similar results were obtained for type Expert.

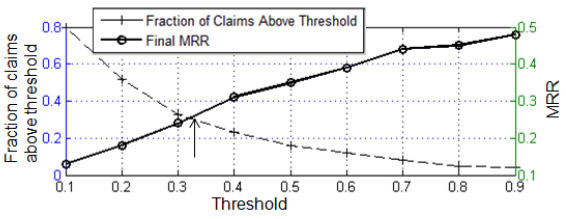


Figure 2: MRR and average fraction of passed claims as function of the claim selection threshold for type Study. Arrow indicates threshold used to obtain the results in Table 4.

6.4 Examples of System Performance

To provide some intuition for the results of our system, Table 5 shows the 4 top ranking candidate CDE

According to econometric studies, negative side effects of aid can include an unbalanced appreciation of the recipient's currency, increasing corruption, and adverse political effects such as postponements of necessary economic and democratic reforms.	X
Many econometric studies in recent years have supported the view that development aid has no effect on the speed with which countries develop.	V
An inquiry into aid effectiveness by the UK All Party Parliamentary Group (APPG) for Debt, Aid and Trade featured evidence from Rosalind Eyben, a Fellow at the Institute of Development Studies.	X
A very large part of the spend money on development aid is simply wasted uselessly. According to Gerbert van der Aa, for the Netherlands, only 33% of the development aid is successful, another 33% fails and of the remaining 33% the effect is unclear. This means that for example for the Netherlands, 1.33 to 2.66 billion is lost as it spends 4 billion in total of development aid.	V

Table 5: Top ranking candidates for the claim *aid is ineffective* in the context of the topic *trade vs. aid*

of type Study for the claim *aid is ineffective* in the context of the topic *trade vs. aid*. Among these, 2 were indeed labeled as CDE. The other two exemplify common errors of our system. Candidate 1 can be used to support a highly related claim such as *aid has negative side effects*, but does not directly support the claim under consideration. Candidate 3 mentions a relevant study, but does not present its results, hence cannot be used to support the claim.

7 Conclusions and Future Work

We have provided the definitions for the CDED task, and described a system architecture that addresses the issues at the heart of the task. We assessed the performance of the proposed approach over a novel benchmark dataset, demonstrating the validity of our architecture, and the necessity of all its components.

There are still many open issues to address and directions in which to expand the task and labeled data which we hope to address in future work.

In this paper we define CDE only in the context of *supporting* a claim. However, in many scenarios providing *counter* evidence can also be very useful. As evidence supporting and contesting a claim share many semantic and syntactic features, we believe that detecting both cases simultaneously might be easier to accomplish, although to enhance the practical use of such a solution, one may need to develop an additional component, determining the polarity of the detected CDE.

Another natural direction to pursue is expanding the documents which are considered for CDED beyond the article containing the claim. These can include additional Wikipedia articles and other resources such as newspaper archives, scientific literature, blogs, etc. This poses additional challenges in gathering labeled data, as it will require a mechanism to decide which documents to label per claim and will probably increase the number of documents to be labeled. Expanding to additional corpora will probably require development of additional features, to capture signals unique to each corpus. For example, in newspaper archives, the identity of the author might prove an important feature.

Finally, in this work we used manually identified claims and articles. Combining a CDED solution with recent works in the field of argumentation mining (Cartright et al., 2011; Levy et al., 2014; Lippi and Torroni, 2015), may give rise to a new generation of methods, that will be able to automatically construct relevant arguments on demand, for a variety of topics.

Acknowledgments

The authors would like to thank Oren Tsur, Vikas C. Raykar, Matan Orbach and Ido Dagan for many helpful discussions.

References

Ehud Aharoni, Anatoly Polnarov, Tamar Lavee, Daniel Hershcovich, Ran Levy, Ruty Rinott, Dan Gutfreund, and Noam Slonim. 2014. A benchmark dataset for automatic detection of claims and evidence in the context of controversial topics. In *Proceedings of the First Workshop on Argumentation Mining*, pages 64–68, Baltimore, Maryland, June. Association for Computational Linguistics.

Kevin D. Ashley and Vern R. Walker. 2013. Toward constructing evidence-based legal arguments using legal decision documents and machine learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Law*, ICAIL '13, pages 176–180, New York, NY, USA. ACM.

Patrice Bellot, Antoine Doucet, Shlomo Geva, Sairam Gurajada, Jaap Kamps, Gabriella Kazai, Marijn Koolen, Arunav Mishra, Veronique Moriceau, Josiane Mothe, Michael Preminger, Eric SanJuan, Ralf Schenkel, Xavier Tannier, Martin Theobald, Matthew Trappett, and Qiuyue Wang. 2013. Overview of inex 2013. In *CLEF Lab Reports*, Valencia, Spain, September.

Filip Boltužić and Jan Šnajder. 2014. Back up your stance: Recognizing arguments in online discussions. In *Proceedings of the First Workshop on Argumentation Mining*, pages 49–58, Baltimore, Maryland, June. Association for Computational Linguistics.

Elena Cabrio and Serena Villata. 2012. Combining textual entailment and argumentation theory for supporting online debates interactions. In *ACL (2)*, pages 208–212.

Claire Cardie, Nancy Green, Iryna Gurevych, Graeme Hirst, Diane Litman, Smaranda Muresan, Georgios Petasis, Manfred Stede, Marilyn Walker, and Janyce Wiebe, editors. 2015. *Proceedings of the Second*

Workshop on Argumentation Mining. Association for Computational Linguistics, Denver, Colorado, June.

Marc-Allen Cartright, Henry A. Feild, and James Allan. 2011. Evidence finding using a collection of books. In *Proceedings of the 4th ACM Workshop on Online Books, Complementary Social Media and Crowdsourcing*, BooksOnline '11, pages 11–18, New York, NY, USA. ACM.

I. Dagan, B. Dolan, B. Magnini, and D. Roth. 2009. Recognizing textual entailment: Rational, evaluation and approaches. *Natural Language Engineering*, 15(04).

Hoa Trang Dang, Diane Kelly, and Jimmy J. Lin. 2007. Overview of the trec 2007 question answering track. In *TREC*.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 363–370, Stroudsburg, PA, USA. Association for Computational Linguistics.

Debanjan Ghosh, Smaranda Muresan, Nina Wacholder, Mark Aakhus, and Matthew Mitsui. 2014. Analyzing argumentative discourse units in online interactions. In *Proceedings of the First Workshop on Argumentation Mining*, pages 39–48, Baltimore, Maryland, June. Association for Computational Linguistics.

Oren Glickman, Ido Dagan, and Moshe Koppel. 2005. A probabilistic classification approach for lexical textual entailment. In Manuela M. Veloso and Subbarao Kambhampati, editors, *AAAI*, pages 1050–1055. AAAI Press / The MIT Press.

Nancy Green, Kevin Ashley, Diane Litman, Chris Reed, and Vern Walker, editors. 2014. *Proceedings of the First Workshop on Argumentation Mining*. Association for Computational Linguistics, Baltimore, Maryland, June.

Adam Lally, John M. Prager, Michael C. McCord, Branimir Boguraev, Siddharth Patwardhan, James Fan, Paul Fodor, and Jennifer Chu-Carroll. 2012. Question analysis: How watson reads a clue. *IBM Journal of Research and Development*, 56(3):2.

Ran Levy, Yonatan Bilu, Daniel Hershcovich, Ehud Aharoni, and Noam Slonim. 2014. Context dependent claim detection. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 1489–1500, Dublin, Ireland, August. Dublin City University and Association for Computational Linguistics.

Marco Lippi and Paolo Torroni. 2015. Context-independent claim detection for argumentation mining. In *Proceedings of the Twenty Fourth International Joint Conference on Artificial Intelligence*. AAAI Press.

Michael C. McCord, William J. Murdock, and Bill K. Boguraev. 2012. Deep parsing in watson. *IBM J. Res. Dev.*, 56(3):264–278, May.

Michael C. McCord. 1990. Slot grammar: A system for simpler construction of practical natural language grammars. In R. Studer, editor, *Natural Language and Logic: Proc. of the International Scientific Symposium, Hamburg, FRG*, pages 118–145. Springer, Berlin, Heidelberg.

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

George A. Miller. 1995. Wordnet: A lexical database for english. *COMMUNICATIONS OF THE ACM*, 38:39–41.

Raquel Mochales Palau and Marie-Francine Moens. 2009. Argumentation mining: the detection, classification and structure of arguments in text. In *Proceedings of the Twelfth International Conference on Artificial Intelligence and Law (ICAIL 2009),*, pages 98–109. ACM.

Raquel Mochales Palau and Marie-Francine Moens. 2011. Argumentation mining. *Artificial Intelligence and Law*, 19(1):1–22.

Joonsuk Park and Claire Cardie. 2014. Identifying appropriate support for propositions in online user comments. In *Proceedings of the First Workshop on Argumentation Mining*, pages 29–38, Baltimore, Maryland, June. Association for Computational Linguistics.

Andreas Peldszus. 2014. Towards segment-based recognition of argumentation structure in short texts. In *Proceedings of the First Workshop on Argumentation Mining*, pages 88–97, Baltimore, Maryland, June. Association for Computational Linguistics.

Martin F. Porter. 1997. Readings in information retrieval. chapter An Algorithm for Suffix Stripping, pages 313–316. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Richard D. Rieke and Malcolm O. Sillars. 1984. *Argumentation and the decision making process*. Harper Collins, New York, NY, USA.

Richard D. Rieke and Malcolm O. Sillars. 2001. *Argumentation and Critical Decision Making*. Longman.

Stephen E. Robertson, Steve Walker, Susan Jones, Micheline Hancock-Beaulieu, and Mike Gatford. 1996. Okapi at trec-3. pages 109–126.

Ariel Rosenfeld and Sarit Kraus. 2015. Providing arguments in discussions based on the prediction of human argumentative behavior. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI-15)*.

Zachary Seech. 2008. *Writing Philosophy Papers*. Cengage Learning.

Ramakrishnan Srikant and Rakesh Agrawal. 1996. Mining sequential patterns: Generalizations and performance improvements. In *Proceedings of the 5th International Conference on Extending Database Technology: Advances in Database Technology*, EDBT '96, pages 3–17, London, UK, UK. Springer-Verlag.

Maria Paz Garcia Villalba and Patrick Saint-Dizier. 2012. Some facets of argument mining for opinion analysis. In *Computational Models of Argument - Proceedings of COMMA 2012, Vienna, Austria, September 10-12, 2012*, pages 23–34.

Douglas Walton. 2009. Argumentation theory: A very short introduction. In Guillermo Simari and Iyad Rahwan, editors, *Argumentation in Artificial Intelligence*, pages 1–22. Springer US.

Simon Wells. 2014. Argument mining: Was ist das? In *Proceedings of the 14th International Workshop on Computational Models of Natural Argument*, CMNA14.

Ainur Yessenalina, Yejin Choi, and Claire Cardie. 2010. Automatically generating annotator rationales to improve sentiment classification. In *Proceedings of the ACL 2010 Conference Short Papers*, ACLShort '10, pages 336–341, Stroudsburg, PA, USA. Association for Computational Linguistics.

Spelling Correction of User Search Queries through Statistical Machine Translation

Saša Hasan[*] sasa.hasan@gmail.com

Carmen Heger[†] heger.carmen@gmail.com

Saab Mansour saamansour@ebay.com

eBay Inc.
2065 Hamilton Ave
San Jose, CA 95125, USA

Abstract

We use character-based statistical machine translation in order to correct user search queries in the e-commerce domain. The training data is automatically extracted from event logs where users re-issue their search queries with potentially corrected spelling within the same session. We show results on a test set which was annotated by humans and compare against online autocorrection capabilities of three additional web sites. Overall, the methods presented in this paper outperform fully productized spellchecking and autocorrection services in terms of accuracy and F1 score. We also propose novel evaluation steps based on retrieved search results of the corrected queries in terms of quantity and relevance.

1 Introduction

Spelling correction is an important feature for any interactive service that takes input generated by users, e.g. an e-commerce web site that allows searching for goods and products. Misspellings are very common with user-generated input and the reason why many web sites offer spelling correction in the form of "Did you mean?" suggestions or automatic corrections. Autocorrection increases user satisfaction by correcting obvious errors, whereas suggestions make it convenient for users to accept a proposed correction without retyping or correcting the query manually. Spelling correction is not a trivial task, as search queries are often short and lack context. Misspelled queries might be considered correct by a statistical spelling correction system as there is evidence in the data through frequent occurrences.

While common successful methods (cf. Section 1.1) rely on either human-annotated data or the entire web, we wanted to use easily accessible in-domain data and on top of that technology that is already available. In this work, we use user event logs from an e-commerce web site to fetch similar search query pairs within an active session. The main idea is that users issue a search query but alter it into something similar within a given time window which might be the correction of a potential typo. To the best of our knowledge, the idea of collecting query corrections using user session and time information is novel. Previous work suggested collecting queries using information that a user clicked on a proposed correction. Our proposed method for collecting training data has several advantages. First, we do not rely on a previous spelling correction system, but on user formulations. Second, for many search queries, especially from the tail where search recall is generally low, these misspellings yield few results, and thus, users looking for certain products are inclined to correct the query themselves in order to find what they are looking for. We use Damerau-Levenshtein distance (Damerau, 1964) on character level as similarity criterion, i.e. queries within a specific edit distance are considered to be related.

The steps proposed in this work are:

1. Extraction of similar user queries from search logs for bootstrapping training data (Section 2),
2. classification and filtering of data to remove noisy entries (Section 3), and

[*] The author is now affiliated with Lilt Inc., Stanford, CA, USA.

[†] The author is now affiliated with Stylight GmbH, Munich, Germany.

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 451–460, Lisbon, Portugal, 17-21 September 2015.

3. spelling correction cast into a statistical machine translation framework based on character bigram sequences (Section 4).

We also evaluate the work thoroughly in Section 5 where we compare our method to three other online sites, two of them from the e-commerce domain, and present a novel approach that determines quality based on retrieved search results. We show examples indicating that our method can handle both corrections of misspelled queries and queries with segmentation issues (i.e. missing whitespace delimiters). A summary can be found in Section 6.

To our knowledge, this is the first work that uses character-based machine translation technology on user-generated data for spelling correction. Moreover, it is the first to evaluate the performance in an e-commerce setting with there relevant measures.

1.1 Related Work

One of the more prominent papers on autocorrection of misspelled input is (Whitelaw et al., 2009). The three-step approach incorporates a classification step that determines whether a word is misspelled, computes the most likely correction candidate and then, again, classifies whether this candidate is likely to be correct. An error model based after (Brill and Moore, 2000) is trained on similar word pairs extracted from large amounts of web sites, and a language model is used to disambiguate correction candidates based on left and right context around the current position. Our method differs in several ways. First, we consider full query pairs as training data, and do not use single words as primary mode of operation. Second, we do not train explicit error models $P(w|s)$ for words w and observed corrections s, but use standard phrase-based machine translation modeling to derive phrase and lexical translation models. Although our level of context is shorter, especially for long words, the system automatically uses cross-word level context.

The idea of using consecutive user queries from a stream of events to improve ranking of web search results was described in (Radlinski and Joachims, 2005). The authors introduce the notion of *query chains* that take advantage of users reformulating their queries as a means to learn better ranking functions. The classification of query chains is performed by support vector machines, and its training data is generated in a supervised fashion by manual inspection and annotation. In contrast, we do not manually annotate any of our training data. Since our initial sets are quite noisy, we apply a couple of heuristics that try to produce a cleaner subset of the data that contains mostly misspelled queries and their potential correction candidates.

Another focus of researchers was specifically to tackle misspelled web search queries and use search engine logs for training and evaluation data (Gao et al., 2010), which differs from our work by collecting data using "click-through" enforcement. The user is presented with a spelling correction, and if she clicks on it, they learn that the correction is valid. Our method does not need a previous spelling correction to work. In addition, our proposed method has the potential to learn corrections of new and rare terms that will not be produced by an automatic spelling correction.

In (Zhang et al., 2006), the authors use a conventional spellchecker to correct web queries through additional reranking of its output by a ranking SVM. The training data is in part automatically extracted, but also contains manually annotated pairs and, thus, is a semi-supervised approach. In this paper, we use an unsupervised approach to generate training data. Query spelling correction that is based on click-through data and uses a phrase-based error model is reported in (Sun et al., 2010). Our models operate on character sequences instead of words, and we do not observe issues with identity transformations (i.e. non-corrections for correctly spelled input).

In (Cucerzan and Brill, 2004), the authors investigate a transformation method that corrects unlikely queries into more likely variants based on web query logs. The iterative approach transforms a search query based on word uni- and bigram decompositions, and the authors evaluate on both a large set that contains around 17% misspelled queries and a smaller set that is based on successive user-reformulated similar queries, a similar setup that we use to extract our training data. They stress the importance of a good language model, as performance drops drastically going from a bigram to a unigram LM.

The use of character-based models in combination with statistical machine translation is not novel and was proposed for spelling correction, e.g., in (Formiga and Fonollosa, 2012), (Liu et al.,

2013) or (Chiu et al., 2013). The authors compare a distance-based approach including a language model, a confusion network-based approach, a translation approach through a heuristically defined phrase table coding all character transformations, and a character-based machine translation approach using standard procedures (automatic word alignment and phrase extraction). The training data is manually created in contrast to our work where we automatically bootstrap training data from query logs. Research has also been done for translation of closely related languages (e.g. (Vilar et al., 2007) and (Nakov and Tiedemann, 2012)) and transliteration (e.g. (Deselaers et al., 2009)). An early summary paper with various spelling-related problems (non-word error detection, isolated-word error correction, and context-dependent word correction) can be found in (Kukich, 1992).

2 Extraction of training data

We use event logs that track user interactions on an e-commerce web site. Each action of a user visiting the site is stored in a data warehouse and HDFS (Hadoop Distributed File System). We store several billion of these user records on a daily basis. The setup allows us to efficiently process large amounts of data points using a Map-Reduce framework via Hadoop[1]. As part of user event tracking, all interactions on the site are stored in the database, e.g. which search terms were entered, which links were clicked, and what actions resulted in this (i.e. buying an item, adding it to a like or watch list, or simply advancing to the next page of search results, and so on). We focus on search terms that users enter within a given period of time.

Our hypothesis is that users that enter consecutive search terms are not satisfied with the results and try to modify the query until the results are acceptable. We then analyze the sequence of search queries as part of each user session. We only extract consecutive search queries that are similar in terms of character-level Damerau-Levenshtein edit distance which is the minimum number of character insertions, deletions, substitutions and transpositions (i.e. the swapping of two characters). Note that spelling correction in this kind of environment also needs to address segmentation of queries in case of missing whitespace. It is quite common to find query terms being concatenated without the use of a space character, e.g. *calvinklein*, *ipadair* or *xboxone*. Also, search queries are short in nature and often lack context, and particularly for the e-commerce domain largely consist of brand and product names and associated attributes (e.g. size of clothing).

Our method extracts similar search query pairs that will be used in our statistical machine translation setup as training data. Table 1 shows examples that we extract from the event logs. We use edit distance thresholds of 3 and 5 characters, where the latter is generally noisier. Noise in this context is everything that is not related to misspelled queries. After a closer look, we observe that many queries are simply rewrites, i.e., users either make refinements to their original query by adding more words to narrow down results, e.g., *leather wallet* → *leather wallet men*, deleting words to decrease specificity, e.g., *gucci belt* → *gucci*, or simply going through various related products or attributes, e.g., *iphone 5* → *iphone 5s* → *iphone 6* where they progress through different products or *nike air 9* → *nike air 9.5* → *nike air 10* where they iterate through different sizes.

The logs on HDFS are organized as sessions where each session contains a stream of user events up to a specific time of inactivity. We use event timestamps to determine how long the users need between consecutive queries, and discard similar query pairs if they are above a threshold of 20 seconds. We use Hadoop-based mapper and reducer steps for the data extraction procedure. In the mappers, pairs of similar user search queries get emitted as per above edit distance criterion on character level, whereas the reducer simply accumulates all counts for identical search query pairs. Due to the size of the data, we run the Hadoop extraction jobs on 24-hour log portions, thus obtaining separate data sets for each day. Overall, we can extract several hundred thousand to several million similar query pairs on a daily basis for edit distance thresholds of 3 and 5, respectively.

We pull several months of data from the Hadoop logs and accumulate each daily pull with unique entries for training our spelling correction system. As mentioned above, search queries that are similar but where the original query is not misspelled make up a big portion of the extracted data. The following section focuses on how to filter the data to result in containing mostly query pairs that fit

[1] http://hadoop.apache.org

Search query	Similar consecutive query	Edit distance	User correction?
nike air hurache	nike air huarache	1	Yes
jordan size 9	jordan size 9.5	2	No
galaxy s4	galaxy s5	1	No
pawer cord forplaystation 3	power cord for playstation 3	2	Yes
iphine 6	iphone 6	1	Yes
iphone 6 plus	iphone 6 plus case	5	No
micheal korrs wstches	michael kors watches	3	Yes
calvin klien men boit	calvin klein men boot	2	Yes
boots	boots men	4	No
sueter	sweater	2	Yes

Table 1: Extracted query pairs found in user event logs. Labels in column 4 indicate whether user-initiated spelling correction (Yes) has taken place vs. a search reformulation that entails the source query not being misspelled (No).

our use case, i.e. mappings from misspelled to corrected ones.

3 Filtering non-misspelled input

We tested a heuristic approach to filtering search query pairs: a combination of regular expressions and thresholded feature scores that detect search progressions and refinements which should be removed from the training data as they do not represent valid user corrections.

The manual filtering heuristic calculates a sequence of features for each search query pair, and as soon as a feature fires, the entry is removed. In the following, we will use the notation (x, y) for a search query pair that is extracted from the logs explained by our method in the previous section. Query x is a user search query, and query y is a similar query issued by the user in the same session within a specific time window. Example query pairs are (*babydoll*, *baby dolls*), (*bike wherl*, *bike wheel*) or (*size 12 yellow dress*, *size 14 yellow dress*). We use the following features as part of this process:

Regular expressions. We remove search query pairs (x, y) if y is a rewrite of x using search-related operators, e.g. $y =$"x" which adds quotation marks around the query (and, thus, has an edit distance of 2). Example: (*bluetooth speakers*, *"bluetooth speakers"*).

LM ratio. We use an English language model trained on publicly available corpora (e.g. Europarl), frequent queries and web content from the e-commerce domain to calculate log-likelihoods for each query and filter entries if the ratio is above zero, i.e. $\log p(x/y) = \log p(x) - \log p(y) > 0$. This step essentially removes query pairs (x, y) if the log-likelihood of query y is smaller than x which usually indicates that the correction is more perplexing than the original query. Example: (*bluetooth ear phones*, *bluetooth ear hpones*) with a log-likelihood ratio of -7.31 + 16.43 > 0 is removed as a typo actually appears on the "corrected" side.

Edit operations. We use a simple heuristic that detects search refinements in terms of word insertions and deletions: a word-level edit distance criterion is used to remove entries where edit operations indicate insertions or deletions of full words. We also detect substitutions on number tokens which are also excluded from training data. Examples: (*polo shirt*, *polo shirt xl*), (*nikon d700*, *nikon d7100*).

Frequent terms. We look at queries (x, y) and use a vocabulary with relative frequencies based on the query data y to determine whether substitutions on word level change a frequent token into another frequent token. Examples: (*snake bat wooden*, *snake bat wood*), (*hd dvds*, *hd dvd*) where *wood*/*wooden* and *dvds*/*dvd* are both frequent tokens and, thus, most likely rewrites and not corrections.

Language ID. The primary search language on US sites is English, but there is also a non-negligible mix of other languages, Spanish in particular. We do not remove all queries that are not identified as English because language identification on (often short) search queries is a non-trivial

Method	#queries	#tokens	MT Acc
all data	80.5M	235.6M	62.0%
heuristic filter	12.6M	40.1M	65.5%

Table 2: Filtering training data with a heuristic set of features. Accuracies are given on DEV for MT baselines with differences only in the data setup.

task with a high error rate and filtering for only English would remove a lot of valid entries. We simply remove query pairs where x is identified as either Spanish or Unknown based on Google's Compact Language Detector[2]. Example: (*accesorios cuarto*, *accesorios de cuarto*).

These heuristics help us to reduce the training data size from 80 million noisy search query pairs with around 235 million tokens to 12.5 million query pairs with roughly 40 million tokens that are of higher quality and most likely spelling corrections.

Table 2 shows results of this filtering step in terms of data sizes and the accuracy on the dev set (cf. Section 4). We observe that the filtering scheme reduces overall training size by almost 85% and increases accuracy on the development set by 3.5% absolute. The removal of query pairs is very aggressive at this point, and overall quality of the spelling correction framework might benefit from a more careful selection. We will look into improved variants of filtering through a maximum entropy classifier trained on actual search results in the future.

4 Autocorrection framework

We cast the autocorrection task into character-based statistical machine translation. For this, we prepare the data by splitting words into sequences of lowercased characters and use a special character to mark whitespace that indicates word boundaries. Once these sequences of characters are corrected, i.e. translated, they are merged back to the full word forms. Table 3 shows an example search query being preprocessed, translated and postprocessed. We use bigram characters instead of single characters, as suggested in (Tiedemann, 2012), in order to improve the statistical alignment models and make them more expressive.

For training the autocorrection system we use basic methods and open-source tools for statistical machine translation. The character alignment is obtained by using GIZA++ (Och and Ney, 2003) for 4, 3 and 2 iterations of IBM Model 1, HMM, and IBM Model 3, respectively. As opposed to the standard machine translation task, we did not observe improvements from IBM Model 4 and do not use it as part of the alignment process.

We use Moses (Koehn et al., 2007) for standard phrase extraction, building KenLM language models (Heafield, 2011) and tuning. The standard set of features is used, including a phrase model, word lexicon model, length penalty, jump penalty and a language model. The model weights are optimized using MERT (Och, 2003). The Moses framework allows us to easily conduct experiments with several settings and find the optimal one for our task at hand. We experiment with varying context sizes for phrase table and language model, additional features and different scoring methods, e.g. BLEU (Papineni et al., 2002) in comparison to directly maximizing accuracy on the dev set. A more detailed description of those experiments including results will follow in Section 5.

Evaluation data. In order to evaluate our proposed framework, we extracted 10,000 query pairs from a one week period not part of the training data. The initial size of 3.5M query pairs was reduced to 10k by exponential reservoir sampling (Osborne et al., 2014) after sorting by frequency. The result is a set of representative query pairs that focuses more on frequent misspellings, but also contains rare queries from the tail of the distribution.

We asked humans to create a gold reference annotation that we can use for tuning and evaluation purposes. The guidelines were to check whether for a search query pair (x, y), the left query x is misspelled, and if so, whether the similar candidate y is a correct correction or else provide the most likely correction. If x was not misspelled, the guidelines instructed to propagate query x to the gold reference y, i.e. for those cases, we have identity or a true negative. If query x is not English, the annotators had to mark those entries and we removed them from the set. This step affected 798 queries (i.e. around 8%, mostly Spanish), and the final set contains 9202 query pairs. We split those into half to produce 4,600 queries for dev and 4,602 for test. The true negatives in those sets (i.e. entries that do not need to be corrected) are ~15%. We are aware that this approach focuses on

[2]https://code.google.com/p/cld2

	characters	character bigrams
Original:	hollowin custome	
Preprocessing:	h o l l o w i n S c u s t o m e	ho ol ll lo ow wi in nS Sc cu st to om me
Translation:	h a l l o w e e n S c o s t u m e	ha al ll lo ow we ee en nS Sc co st tu um me
Postprocessing:	halloween costume	

Table 3: Preprocessing of input data as single character or character bigram sequences. "S" denotes whitespace. After translation, postprocessing transforms back to word-level surface forms.

Eval data	DEV	TEST
#queries	4,600	4,602
#tokens	15,593	15,557
CER[%]	6.1	6.0
SER[%]	84.2	84.3

Table 4: Statistics on dev and test portions of the post-edited evaluation data. CER is the character error rate of the misspelled queries against the gold reference, SER is the sentence (i.e. here query-level) error rate of the sets.

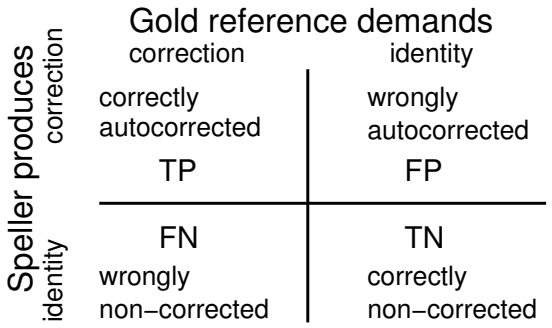


Figure 1: Scoring schema.

recall over precision, as the majority of queries is usually not misspelled. Nevertheless, exponential reservoir sampling helps us to focus on the head of that distribution, and our main goal is to evaluate the capabilities of the spelling correction framework, not the overall system integration. A final investigation in combination with query expansion that will be evaluated in the context of the live site is left for future work. Detailed statistics on the two sets can be found in Table 4.

Additional training data. In addition to the parallel data pull from the query logs as described in Section 2, we also take the top queries from those logs where we are sure they are most likely correct and can be used as gold reference (e.g. search queries like *handbags* or *wedding dress* appear thousands of times), and generate artificially noisified variants based on an English keyboard layout and statistics derived from our development set. On the dev set, we calculated a character error rate of roughly 6%, and this rate is used as a mutation rate for the artificially introduced noise. We also determine the following edit operation rates based on the dev set: 6% character transpositions, 18% for deletions, 33% for insertions, and 43% for substitutions. The target character for substitutions and insertions is based on a random Gaussian distribution with a distance mean 1 and standard deviation of 1 around the selected character, i.e. we target neighboring keys on the keyboard. For the query *wedding dress*, e.g., this method introduces misspelled variants such as *weddig dress*, *weedimg dress*, or *weddinb dreess*. We run this method 10 times on a set of 688k queries and remove all duplicates, resulting in additional 4.6M search query pairs with around 13M tokens for the training data.

As a final step, we add frequent queries from user event logs but also product titles of the live inventory to the language model. In total, the monolingual portion of the data contains roughly 31M entries with 319M tokens which are added to the target side of the bilingual training portion.

5 Experiments

In this section, we report on a series of hillclimbing experiments used to optimize performance. In the following, we use standard information retrieval criteria for evaluation, namely accuracy, precision, recall, and F1 score. Figure 1 depicts a natural way of how the scoring is performed. TP denotes true positives, FN false negatives, FP false positives, and TN true negatives, respectively. Note that for the case where the gold reference demands a correction, and the speller produces a wrong correction different from the source query, we have to increase both false positives FP and false negatives FN. With this, we can do stan-

TEST [%]	Acc	Prec	Rec	F1
online A	68.8	**84.0**	64.2	72.8
online B	63.0	77.4	60.2	67.7
online C	56.3	58.6	60.7	59.7
MT (this work)	**70.6**	74.0	**72.3**	**73.1**
MT (alt. setup)	70.2	77.1	69.3	73.0

Table 5: Comparison of accuracy, precision, recall and F1 score against other online sites on the test set.

dard calculation of accuracy as $(TP + TN)/(P + N)$ with $P = TP + FN$ and $N = TN + FP$, precision as $TP/(TP + FP)$, recall as TP/P, and F1 score as $2TP/(2TP + FP + FN)$.

5.1 Hillclimbing and performance

We compare the performance of our autocorrection system with that of three other online search engines, both from the e-commerce as well as web search domain. For this, we automated entering our search terms on the corresponding sites and extracted candidates from "Did you mean..." suggestions or "Showing results for..." autocorrections. Table 5 shows that our method outperforms all other systems in accuracy, recall and F1 score. We argue that recall in our setup is more important than precision because the goal is to correct as much as possible as autocorrection is usually invoked by the search backend as a result of low or *null* recall size.

Gradual improvements were made to the system setup and we track those on the development set. Table 6 gives a couple of major waypoints in the hillclimbing process. The baseline incorporates all extracted training data and uses phrases up to length 3 (i.e. up to three character bigrams for both source and target side). The baseline language model uses 6-grams. This setup is trained on very noisy data that contains a lot of search refinements that are not actual misspellings. The filtered data improves results. In general, we observe a 3-4% relative improvement across all scores when using bigrams instead of single characters. We experiment with additional phrase features as part of the phrase table and add 5 features based on phrase pair-specific edit operations, i.e. the number of insertions, deletions, substitutions, transpositions and final overall edit distance, which helps to increase precision. The artificially noisified data gives additional small gains, as well as direct optimization of accuracy instead of BLEU or WER. We did not observe significant differences when tuning on BLEU versus WER. Most of the improvement though comes from increasing context size, i.e., 10-gram language models and lengths up to 5 bigram character spans for both source and target phrases. We also observe that iterative correction, i.e. running the speller a second time over already corrected data, further improves performance slightly which is in-line with findings in (Cucerzan and Brill, 2004) and (Gubanov et al., 2014).

Increasing precision. We also investigated a system setup that focuses on precision over recall in order to be more in sync with the online systems that have been most likely optimized to a more cautious correction mode. Our previous experiments prefer recall which is due to the 85:15 split of misspelled vs. correct queries in the dev set. For a more conservative mode that focuses on precision, we updated the dev set by automatically adding "mostly-correct" queries with identity as correction candidate. For this, we extracted the most frequent queries with a high number of search results which can be deemed to be "almost" correct. The dev size increased to roughly 22k queries with an approximate split of 85:15 for "correct" vs. misspelled. This is a more realistic setting if the correction is applied to *all* incoming queries, irrespective of the number of corresponding search results (note also that this mode is different from our initial one where we apply corrections only to queries with low or *null* results). We also found that tuning on Matthews Correlation Coefficient (Matthews, 1975) balances better precision versus recall, especially for unbalanced classes which is the case here (i.e. 85:15 split).

In the last line of Table 6 we added more data extracted over several additional months. The additional data amounts to 60M queries, therefore increasing the total training size to 72M queries. This final setup, as described, improves precision but hurts recall slightly. Overall, the F1 and accuracy measures are still improved which is most likely due to the additional training data.

Finally, we ran a large-scale experiment and extracted the 100k most frequent unique queries (which account for 7.8M queries based on their actual frequencies). Since they represent the most common queries, some of them typed thousands of times by the users, they are deemed to be

DEV [%]	Acc	Prec	Rec	F1
baseline	62.0	66.2	61.0	63.5
+filtered data	65.5	68.8	67.6	68.2
+phrase length 5	67.2	70.0	69.6	69.8
+phrase features	67.3	71.2	68.3	69.8
+artificial noise	67.8	71.4	68.3	69.8
+10-gram LM	68.7	71.9	70.5	71.2
+tune on Acc	68.9	72.1	70.3	71.2
+iterative 2nd run	69.0	72.2	70.5	71.4
+alt. setup (Prec)	69.5	75.4	68.5	71.8

Table 6: Hillclimbing on the dev set.

Query	Category distribution
moto x	50% cell phones, 30% automotive, 5% clothing, ...
tory burch fitbit	75% jewelry, 20% sports, ...
madden 15	65% games, 18% toys, 7% collectibles, ...

Table 7: Item category distributions for queries.

"mostly" correct. We autocorrect the set through our best system which results in changed queries in <0.5% of the cases after manual filtering of non-errors (e.g. segmentation differences that are handled separately by our query expansion system, e.g. *rayban* → *ray ban* which are equivalent in the search backend) and when compared against the original input set. This means that most of the queries are left unchanged and that the decoding process of the SMT system does a good job in classifying whether a correction is needed or not. Manual inspection of the most frequent differences shows that changes actually happen on frequently misspelled words which are part of those top 100k queries, e.g. *micheal* → *michael*, *infared* → *infrared*, or *accesories* → *accessories*.

5.2 Search evaluation

An interesting way to evaluate the performance of the autocorrection system is to look at it from a search perspective. If the goal is to improve user experience in search and retrieve more relevant results, we need to compare the search result sets in terms of quantity and quality. For quantity we are mainly interested in how many queries lead to 0 search results before and after autocorrection. Table 8 shows these numbers for our test set. We improve the rate of *null* results by 51.8% absolute. In a deeper analysis we see that out of the 81.6% in the source set we improve the query behavior for 63.8% queries after autocorrection while we only decrease from non-*null* to *null* for 1.6%. A manual inspection of the 1.6% shows that those queries already led to very low search results (<10) even for the source.

TEST [%]	*null* results	KL div. < 1
misspelled	81.6	40.9
gold corrections	27.1	100.0
online A	36.5	84.4
online B	36.0	82.1
online C	33.0	76.4
MT (this work)	**29.8**	**86.0**

Table 8: Search results evaluation on the test set. For *null* results, lower rates are better. For KL div. evaluation, higher rates are better, as they indicate a closer match with the category distribution from the gold corrections.

For quality, we compare the search result sets for the autocorrected queries with those of the reference and get an estimate of how similar each autocorrected query behaves to the expected behavior given by its gold correction. In particular, we extract the categories from all items which are returned as part of the search API calls (see Table 7). We use Kullback-Leibler divergence (Kullback and Leibler, 1951) to compute the difference between the category distributions. The KL divergence defines the information lost when using one distribution to approximate another:

$$D(P||Q) = \sum_i P(i) \cdot \ln\left(\frac{P(i)}{Q(i)}\right)$$

In our case, we use the category distribution of the results of the reference query as the observed data P which we want to model with the category distribution Q from the results of the autocorrected query. This gives us a more realistic evaluation of the method, as it is more relevant to the retrieval problem that autocorrection is trying to address. The queries *perfumes for women* and *perfume for women*, e.g., retrieve similar amounts and types of items although their strings are different (note the plural *s* in the first one) which is penalized when computing accuracy on string level.

In order to deal with zero-probability events, we smooth both distributions such that they contain exactly the same categories. For this we add categories that are present in one distribution but not

Misspelled query	Correction
adidda whatch blooto	adidas watch bluetooth
awrppstale buttin shirt	aeropostale button shirt
bangen olafsn headset	bang olufsen headset
camra exesers	camera accessory
crystal and righston cuf ring	crystal and rhinestone cuff ring
fauxfurmidcalfwesternboots	faux fur mid calf western boots
fotbool chus	football shoes
otherbooxiphon5	otterbox iphone 5
womens realhairsaltandpeper	womens real hair salt and pepper

Table 9: Examples of misspelled user search queries that the presented system is able to correct.

the other with a very small probability which we subtract from the highest probability in the distribution so that all probabilities still sum up to 1. In the case of getting 0 search results for either the autocorrected or reference query we cannot compute the KL divergence and we simply mark those cases with a high number. If both queries return 0 search results we return a distance of 0. Note that with this we do not penalize queries that are misspelled and should have been corrected even if they still retrieve 0 items in search after correction. However, in this test we are only interested in the search results we get and not in the correction itself.

Table 8 shows that the presented method is closest in terms of KL divergence to category distributions of search results based on the gold references. Even though this is a nice and easy way to indicate quality of search results, we do not claim this method to be an in-depth analysis of relevance of search results and leave this for future work.

5.3 Examples

The examples in Table 9 demonstrate the abilities of the presented spelling correction framework. We are able to handle extremely misspelled queries, e.g. due to phonetic spelling by possibly non-native speakers (*camra* → *camera*, *chus* → *shoes*), words being glued together due to missing whitespace (*fauxfurmid...* → *faux fur mid...*), or brand name confusions (*righston* → *rhinestone*).

6 Conclusion

In this paper, we presented a powerful spelling correction system for the e-commerce domain using statistical phrase-based machine translation based on character bigram sequences. We extracted training data from event logs where users issue similar search queries within a certain period of time. Filtering through various heuristics is used to clean the initially noisy data set and remove entries not related to spelling errors. We evaluated our system against established online search sites, both in the general and e-commerce domain, and showed favorable results in terms of recall and retrieval rates.

We plan to further invest in improving precision. For this, we feel that adding full word-level models will help to overcome the somewhat limited context present in our character sequence-based models. We also plan to investigate the prototype directly in query expansion as part of the search backend.

Acknowledgments

The authors would like to thank the Localization team at eBay for creating gold corrections for the evaluation sets, and members of the HLT and Search teams for fruitful discussions as well as reading early drafts of this work and giving valuable feedback.

References

Eric Brill and Robert C. Moore. 2000. An improved error model for noisy channel spelling correction. In *Proc. of the 38th Annual Meeting of the Association for Computational Linguistics*, pages 286–293, Hong Kong.

Hsun-wen Chiu, Jian-cheng Wu, and Jason S. Chang. 2013. Chinese spelling checker based on statistical machine translation. In *Proc. of the Seventh SIGHAN Workshop on Chinese Language Processing*, pages 49–53, Nagoya, Japan.

Silviu Cucerzan and Eric Brill. 2004. Spelling correction as an iterative process that exploits the collective

knowledge of web users. In *Proc. of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 293–300, Barcelona, Spain.

Frederick J. Damerau. 1964. A technique for computer detection and correction of spelling errors. *Communication of ACM*, 7(3):171–176.

Thomas Deselaers, Saša Hasan, Oliver Bender, and Hermann Ney. 2009. A deep learning approach to machine transliteration. In *Fourth EACL Workshop on Statistical Machine Translation*, pages 233–241, Athens, Greece.

Lluís Formiga and José A. R. Fonollosa. 2012. Dealing with input noise in statistical machine translation. In *Proc. of the 24th International Conference on Computational Linguistics (Coling 2012): Posters*, pages 319–328, Mumbai, India.

Jianfeng Gao, Xiaolong Li, Daniel Micol, Chris Quirk, and Xu Sun. 2010. A large scale ranker-based system for search query spelling correction. In *Proc. of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 358–366, Beijing, China.

Sergey Gubanov, Irina Galinskaya, and Alexey Baytin. 2014. Improved iterative correction for distant spelling errors. In *Proc. of the 52nd Annual Meeting of the Association for Computational Linguistics: Short Papers*, pages 168–173, Baltimore, MD.

Kenneth Heafield. 2011. KenLM: faster and smaller language model queries. In *Proc. of the EMNLP 2011 Sixth Workshop on Statistical Machine Translation*, pages 187–197, Edinburgh, UK.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proc. of the 45th Annual Meeting of the Association for Computational Linguistics: Demos and Posters*, pages 177–180, Prague, Czech Republic.

Karen Kukich. 1992. Techniques for automatically correcting words in text. *ACM Computing Surveys*, 24(4):377–439.

Solomon Kullback and Richard Leibler. 1951. On information and sufficiency. *Annals of Mathematical Statistics*, 22(1):79–86.

Xiaodong Liu, Kevin Cheng, Yanyan Luo, Kevin Duh, and Yuji Matsumoto. 2013. A hybrid Chinese spelling correction using language model and statistical machine translation with reranking. In *Proc. of the Seventh SIGHAN Workshop on Chinese Language Processing*, pages 54–58, Nagoya, Japan.

Brian W. Matthews. 1975. Comparison of the predicted and observed secondary structure of T4 phage lysozyme. *Biochimica et Biophysica Acta (BBA) - Protein Structure*, 405(2):442–451.

Preslav Nakov and Jörg Tiedemann. 2012. Combining word-level and character-level models for machine translation between closely-related languages. In *Proc. of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers*, pages 301–305, Jeju Island, Korea.

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

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proc. of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 160–167, Sapporo, Japan.

Miles Osborne, Ashwin Lall, and Benjamin Van Durme. 2014. Exponential reservoir sampling for streaming language models. In *Proc. of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 687–692, Baltimore, Maryland.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proc. of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA.

Filip Radlinski and Thorsten Joachims. 2005. Query chains: Learning to rank from implicit feedback. In *Proc. of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, pages 239–248, Chicago, IL.

Xu Sun, Jianfeng Gao, Daniel Micol, and Chris Quirk. 2010. Learning phrase-based spelling error models from clickthrough data. In *Proc. of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 266–274, Uppsala, Sweden.

Jörg Tiedemann. 2012. Character-based pivot translation for under-resourced languages and domains. In *Proc. of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 141–151, Avignon, France.

David Vilar, Jan-Thorsten Peter, and Hermann Ney. 2007. Can we translate letters? In *Second Workshop on Statistical Machine Translation*, pages 33–39, Prague, Czech Republic.

Casey Whitelaw, Ben Hutchinson, Grace Y Chung, and Ged Ellis. 2009. Using the Web for language independent spellchecking and autocorrection. In *Proc. of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 890–899, Singapore.

Yang Zhang, Pilian He, Wei Xiang, and Mu Li. 2006. Discriminative reranking for spelling correction. In *Proc. of the 20th Pacific Asia Conference on Language, Information and Computation*, pages 64–71, Wuhan, China.

Human Evaluation of Grammatical Error Correction Systems

Roman Grundkiewicz[1] and **Marcin Junczys-Dowmunt**[1] and **Edward Gillian**[2]

[1]Information Systems Laboratory
Faculty of Mathematics and Computer Science, Adam Mickiewicz University
{romang,junczys}@amu.edu.pl

[2]Faculty of English, Adam Mickiewicz University
egillian@pwsz.pl

Abstract

The paper presents the results of the first large-scale human evaluation of automatic grammatical error correction (GEC) systems. Twelve participating systems and the unchanged input of the CoNLL-2014 shared task have been reassessed in a WMT-inspired human evaluation procedure. Methods introduced for the Workshop of Machine Translation evaluation campaigns have been adapted to GEC and extended where necessary. The produced rankings are used to evaluate standard metrics for grammatical error correction in terms of correlation with human judgment.

1 Introduction

The field of automatic grammatical error correction (GEC) has seen a number of shared tasks of different scope and for different languages. The most impactful were the CoNLL-2013 and CoNLL-2014 (Ng et al., 2013; Ng et al., 2014) shared tasks on Grammatical Error Correction for ESL (English as a second language) learners. They were preceded by the HOO shared tasks (Dale and Kilgarriff, 2011; Dale et al., 2012). Shared tasks for other languages took place as well, including the QALB workshops for Arabic (Mohit et al., 2014) and NLP-TEA competitions for Chinese. These tasks use automatic metrics to determine the quality of the participating systems.

However, these efforts pale in comparison to competitions organized in other fields, e.g. during the annual Workshops for Machine Translation (WMT). It is a central idea of the WMTs that automatic measures of machine translation quality are an imperfect substitute for human assessments. Therefore, manual evaluation of the system outputs are conducted and their results are reported as the final rankings of the workshops. These human evaluation campaigns are an important driving factor for the advancement of MT and produce insightful "by-products", such as a huge number of human assessments of machine translation outputs that have been used to evaluate automatic metrics.

We believe that the unavailability of this kind of quality assessment may stall the development of GEC, as all the shared tasks and the entire field have to cope with an inherent uncertainty of their methods and metrics. We hope to make a step towards alleviating this lack of confidence by presenting the results of the first[1] large-scale human evaluation of automatic grammatical error correction systems submitted to the CoNLL-2014 shared task. Most of our inspiration is drawn from the recent WMT edition (Bojar et al., 2014) and its metrics task (Macháček and Bojar, 2014).

We also provide an analysis of correlation between the standard metrics in GEC and human judgment and show that the commonly used parameters for standard metrics in the shared task may not be optimal. The uncertainty about metrics quality leads to proposals of new metrics, with Felice and Briscoe (2015) being a recent example. Based on human judgments we can show that this proposed metric maybe less useful than hoped.

2 Evaluation of GEC systems

Madnani et al. (2011) addresses two problems of GEC evaluation: 1) a lack of informative metrics and 2) an inability to directly compare the performance of systems developed by different researchers. Two evaluation methodologies are presented, both based on crowdsourcing which are used to grade types of errors rather than system performance as presented in this work. Chodorow et al. (2012) draw attention to the many evalua-

[1]During the camera-ready preparation phase, we learned about similar research by Napoles et al. (2015). After contacting the authors, it was agreed to treat both works as fully concurrent. Future work will compare the results.

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 461–470,
Lisbon, Portugal, 17-21 September 2015.

tion issues in error detection which make it hard to compare different approaches. The lack of consensus is due to the nature of the error detection task. The authors argue that the choice of the metric should take into account factors such as the skew of the data and the application that the system is used for.

The most recent addition is Felice and Briscoe (2015) who present a novel evaluation method for grammatical error correction that scores systems in terms of improvement on the original text.

3 The CoNLL-2014 shared task

The goal of the CoNLL-2014 shared task (Ng et al., 2014) was to evaluate algorithms and systems for automatically correcting grammatical errors in English essays written by second language learners of English. Training and test data was annotated with 28 error types. Participating teams were given training data with manually annotated corrections of grammatical errors and were allowed to use publicly available resources for training.

Twenty-five student non-native speakers of English were recruited to write essays to be used as test data. Each student wrote two essays. The 50 test essays were error-annotated by two English native speakers. The essays and error annotations were made available after the task. The MaxMatch (M^2) scorer (Dahlmeier and Ng, 2012) has been used as the official shared task evaluation metric.

4 Data collection

4.1 Sampling sentences for evaluation

The system outputs of the CoNLL-2014 shared task serve as evaluation data. The test set consists of 1312 sentences, there are twelve system outputs available. The thirteenth participant NARA is missing from this set. However, in GEC evaluation there is also the input to consider. Often system outputs are equal to the unmodified input, as it is most desirable if there are in fact no errors. We include INPUT as the thirteenth system.

Due to the small number of modifications that GEC systems apply to the input, there is not only a large overlap with the input, but also among all systems (Figure 1). If we sample systems uniformly, we lose easily obtainable pairwise judgments for systems with the same output, and if we collapse before sampling we introduce a strong bias towards ties. To counter that bias, we abandon uniform sampling of test set sentences and use

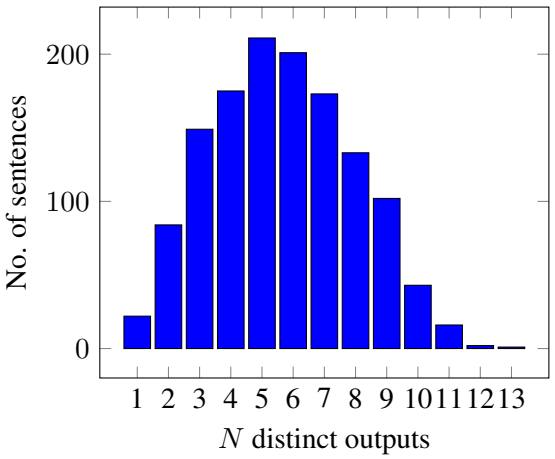


Figure 1: Frequencies of distinct corrected sentences produced by 13 systems per input sentence.

instead a parametrized distribution that favors diverse sets of outputs.

The probability p_i for a set of outputs O_i is calculated as follows: N is the number of systems to be evaluated, M is the maximum number of sentences presented to the evaluator in a single ranking (we use $M = 5$). The set of system outputs to be evaluated $E = \{O_1, \ldots, O_n\} \quad \forall_{1 \leq i \leq n} |O_i| = N$, consists of n $(= 1312)$ sets O_i of N output sentences each. Every sentence in O_i can overlap with other sentences multiple times, so for each set O_i we define the corresponding multiset of multiplicities U_i, such that $\sum_{u \in U_i} u = N$.

We define $c_i(j)$ as the number of possible ways to choose at most M different sentences that cover j systems for the i-th set of outputs:

$$c_i(j) = \left| \left\{ S \subseteq U_i : |S| \leq M \wedge \sum_{u \in S} u = j \right\} \right|.$$

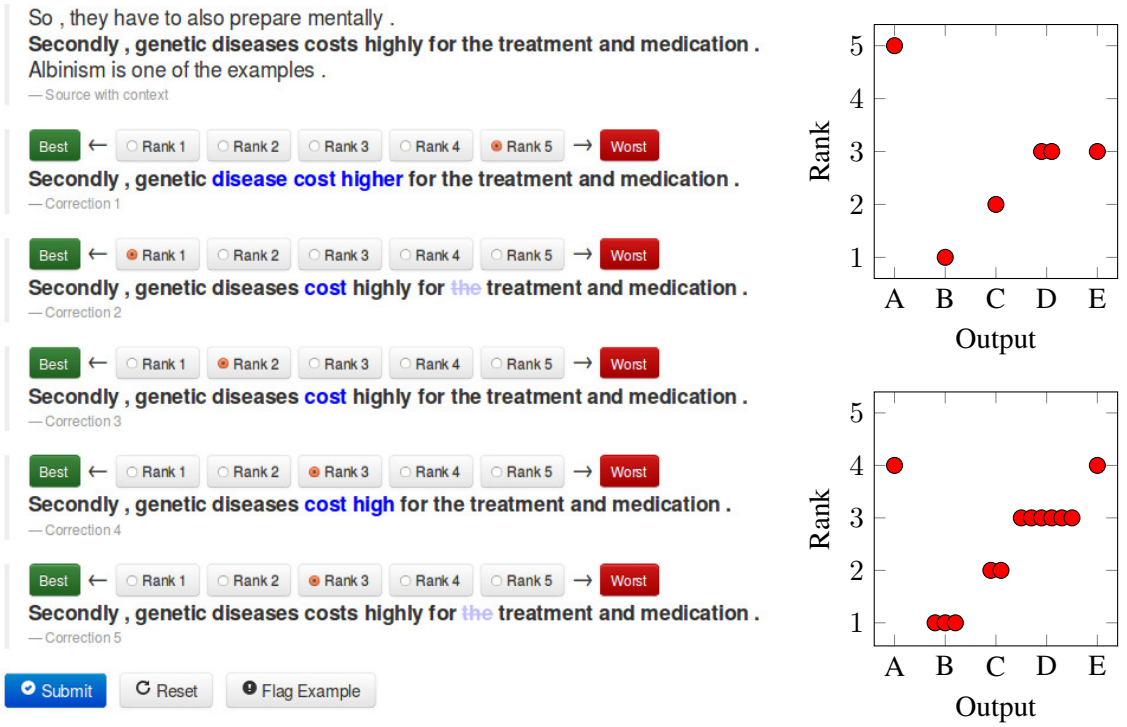
Then the expected number C_i of systems covered by choosing at most M sentences is

$$C_i = \frac{\sum_{j=M}^{N} c_i(j) \cdot j}{\sum_{j=M}^{N} c_i(j)}.$$

The pseudo-probability p'_i of sampling the i-th sentence is defined as

$$p'_i = \frac{\binom{M}{2}}{\binom{C_i}{2}} \quad \text{where} \quad \binom{C_i}{2} = \frac{C_i(C_i - 1)}{2}$$

which is the ratio of pairwise comparisons of M versus C_i different systems. By normalizing over



(a) Screenshot of Appraise modified for GEC judgment.

(b) Overlapping rankings.

Figure 2: Displayed ranking and corresponding overlapping rankings.

the entire set of output sets we obtain the probability p_i of sampling the i-th set of outputs as

$$p_i = \frac{p'_i}{\sum_{j=1}^{|E|} p'_j}.$$

4.2 Collecting system rankings

The sets of outputs sampled with the described method have been prepared for Appraise (Federmann, 2010) and presented to the judges. Judges were asked to rank sentences from best to worse. Ties are allowed. Judges were aware that the absolute ranks bear no relevance as ranks are later turned into relative pairwise judgments. No notion of "better" or "worse" was imposed by the authors, we relied on the judges to develop their own intuition. All eight judges are English native speakers and have extensive backgrounds in linguistics.

Figure 2a displays a screen shot of Appraise with a judged sentence. Several modifications to the Appraise framework[2] were implemented to account for the specific nature of GEC:

Only the input sentence is displayed (top, bold), no reference correction is given. The input sentence is surrounded by one preceding and one following sentence. Identical corrections are collapsed into one output, system names with the same output are recorded internally. Edited fragments are highlighted, blue for insertions and substitutions, pale blue and crossed-out for deletions.

4.3 Pairwise judgments

As conducted during the WMT campaigns, we turn rankings into sets of relative judgments of the form `A>B`, `A=B`, `A<B` where the lower ranked system scores a win. Absolute ranks and differences are lost. As mentioned above, due to the collapsing of identical outputs we obtain significantly more data than the usual 10 pairs from one ranking with five sentences. Figure 2a contains a ranking with overlapping outputs as displayed in the top graph of Figure 2b. Pairs from within overlaps result in ties, pairs between overlaps are expanded as products, $\binom{6}{2} = 15$ pairwise judgments can be extracted. Greater overlap leads to more pairwise judgments (bottom, $\binom{13}{2} = 78$).

Table 1 lists the full statistics for collected rankings by individual annotators. Unexpanded pairs are WMT-style pairwise judgments before an output `A` gets split into overlapping systems $\texttt{A}_1$, $\texttt{A}_2$, $\texttt{A}_3$, etc. The large number of ties for expanded pairs is to be expected due to the high overlap

[2] A fork of the original source code with can be found at `https://github.com/snukky/Appraise`

Judge	Ranks	Unexpanded	Expanded
1	400	3525 (1022)	18400 (10166)
2	299	2684 (1099)	13657 (8429)
3	400	3523 (914)	18912 (9684)
4	201	1750 (550)	9478 (5539)
5	349	3099 (766)	17107 (8972)
6	400	3474 (517)	19313 (9209)
7	70	646 (145)	3383 (1593)
8	200	1815 (681)	8848 (5525)
Total	2319	20516 (5694)	109098 (59117)

Table 1: Statistics for collected rankings (Ranks), unexpanded and expanded pairwise judgments, numbers for ties are given in parentheses.

between systems (on average there are only 5.7 unique outputs among 13 systems).

4.4 Inter- and intra-annotator agreement

Again inspired by the WMT evaluation campaigns, we compute annotator agreement as a measure of reliability of the pairwise judgments with Cohen's kappa coefficient (Cohen, 1960):

$$\kappa = \frac{P(A) - P(E)}{1 - P(E)}.$$

where $P(A)$ is the proportion of times that annotators agree, and $P(E)$ is the proportion of times that they would agree by chance. κ assumes values from 0 (no agreement) to 1 (perfect agreement).

All probabilities are computed as ratios of empirically counted pairwise judgments. As the judges worked on collapsed outputs, we calculate agreement scores for unexpanded pairs; otherwise, the high overlap would unfairly increase agreement.

$P(A)$ is calculated by examining all pairs of outputs which have been judged by two or more judges, and counting the proportion of times that they agreed that A<B, A=B, or A>B.

$P(E) = P(\texttt{A<B})^2 + P(\texttt{A=B})^2 + P(\texttt{A>B})^2$ is the probability that two judges agree randomly. Intra-annotator agreement as a measure of consistency is calculated for output sets that have been judged more than one time by the same annotator.

The agreement numbers in Table 2 are in the lower range of values reported during WMT. However, it should be noted that judges never saw the repeated outputs within one ranking which probably decreases agreement compared to the MT-specific task.

Agreement	Value	Degree
Inter-annotator	0.29	Weak
Intra-annotator	0.46	Moderate

(a) Inter-annotator and intra-annotator agreement for all judges

	1	2	3	4	5	6	7	8
1	.42	.26	.30	.37	.34	.26	.31	.24
2	–	.30	.25	.28	.23	.20	.10	.20
3	–	–	.50	.35	.44	.34	.46	.26
4	–	–	–	.34	.34	.30	.20	.26
5	–	–	–	–	.60	.36	.34	.32
6	–	–	–	–	–	.44	.35	.25
7	–	–	–	–	–	–	*	*
8	–	–	–	–	–	–	–	.48

(b) Pairwise inter-annotator and intra-annotator agreement per judge. Stars indicate too few overlapping judgements.

Table 2: Inter-annotator and intra-annotator agreement (Cohen's κ) on unexpanded pairwise judgments.

5 Computing ranks

In this section, it is our aim to produce a system ranking from best to worse by computing the average number of times each system was judged better than other systems based on the collected pairwise rankings. While previously introduced methods for producing rankings, total orderings, as well as partial orderings at chosen confidence-levels, can be directly applied to our data, determining which ranking is more accurate turns out to be methodologically and computationally more involved due to the specific nature of GEC outputs.

5.1 Ranking methods

We adapt two ranking methods applied during WMT13 and WMT14 to GEC evaluation: the Expected Wins method and a version of TrueSkill.

Expected Wins. Expected Wins (EW) has been introduced for WMT13 (Bojar et al., 2013) and is based on an underlying model of "relative ability" proposed in Koehn (2012). One advantage of this method is its intuitiveness; the scores reflect the probability that a system S_i will be ranked better than another system that has been randomly chosen from a pool of opponents $\{S_j : j \neq i\}$. Defining the function $\text{win}(A, B)$ as the number of times system A is ranked better than system B, Bojar et

#	System	P	R	$M^2_{0.5}$
1	CAMB	0.397	0.301	0.373
2	CUUI	0.417	0.248	0.367
3	AMU	0.416	0.214	0.350
4	POST	0.345	0.217	0.308
5	NTHU	0.350	0.188	0.299
6	RAC	0.331	0.149	0.266
7	UMC	0.312	0.144	0.253
8	PKU	0.322	0.136	0.253
9	SJTU	0.301	0.051	0.151
10	UFC	0.700	0.017	0.078
11	IPN	0.112	0.028	0.071
12	IITB	0.307	0.013	0.059
13	INPUT	0.000	0.000	0.000

(a) Official CoNLL-2014 ranking without unpublished NARA system.

#	Score	Range	System
1	0.628	1	AMU
2	0.566	2-3	RAC
	0.561	2-4	CAMB
	0.550	3-5	CUUI
	0.539	4-5	POST
3	0.513	6-8	UFC
	0.506	6-8	PKU
	0.495	7-9	UMC
	0.485	7-10	IITB
	0.463	10-11	SJTU
	0.456	9-12	INPUT
	0.437	11-12	NTHU
4	0.300	13	IPN

(b) Human ExpectedWins ranking (final manual ranking).

#	Score	Range	System
1	0.273	1	AMU
2	0.182	2	CAMB
3	0.114	3-4	RAC
	0.105	3-5	CUUI
	0.080	4-5	POST
4	-0.001	6-7	PKU
	-0.022	6-8	UMC
	-0.041	7-10	UFC
	-0.055	8-11	IITB
	-0.062	8-11	INPUT
	-0.074	9-11	SJTU
5	-0.142	12	NTHU
6	-0.358	13	IPN

(c) Human TrueSkill ranking.

	AMU	RAC	CAMB	CUUI	POST	UFC	PKU	UMC	IITB	SJTU	INPUT	NTHU	IPN
AMU	–	.44 ‡	.47 ⋆	.46 †	.44 ‡	.34 ‡	.40 ‡	.37 ‡	.32 ‡	.34 ‡	.32 ‡	.31 ‡	.24 ‡
RAC	**.56** ‡	–	**.53**	.48	.48	.40 ‡	.45 †	.44 ‡	.39 ‡	.38 ‡	.38 ‡	.43 ‡	.28 ‡
CAMB	**.53** ⋆	.47	–	.49	.45 ‡	.43 ‡	.43 ‡	.42 ‡	.42 ‡	.43 ‡	.42 ‡	.43 ‡	.34 ‡
CUUI	**.54** †	**.52**	**.51**	–	.49	.42 ‡	.47	.46 †	.42 ‡	.41 ‡	.41 ‡	.42 ‡	.32 ‡
POST	**.56** ‡	**.52**	**.55** ‡	**.51**	–	.45 ‡	.47	.46 ⋆	.44 ‡	.44 ‡	.43 ‡	.42 ‡	.29 ‡
UFC	**.66** ‡	**.60** ‡	**.57** ‡	**.58** ‡	**.55** ‡	–	**.54** ⋆	**.50**	.49	.44 ⋆	.27 †	.42 ‡	.21 ‡
PKU	**.60** ‡	**.55** †	**.57** ‡	**.53**	**.53**	.46 ⋆	–	.50	.47	.46 ⋆	.46 ⋆	.46 †	.35 ‡
UMC	**.63** ‡	**.56** ‡	**.58** ‡	**.54** †	**.54** ⋆	.50	**.50**	–	.48	.47	.48	.45 ‡	.35 ‡
IITB	**.68** ‡	**.61** ‡	**.58** ‡	**.58** ‡	**.56** ‡	**.51**	**.53**	**.52**	–	.48	.43	.43 ‡	.27 ‡
SJTU	**.66** ‡	**.62** ‡	**.57** ‡	**.59** ‡	**.56** ‡	**.56** ⋆	**.54** ⋆	**.53**	**.52**	–	**.53**	.46 ⋆	.30 ‡
INPUT	**.68** ‡	**.62** ‡	**.58** ‡	**.59** ‡	**.57** ‡	**.73** †	**.54** ⋆	**.52**	**.57**	.47	–	.43 ‡	.22 ‡
NTHU	**.69** ‡	**.57** ‡	**.57** ‡	**.58** ‡	**.58** ‡	**.58** ‡	**.54** †	**.55** ‡	**.57** ‡	**.54** ⋆	**.57** ‡	–	.41 ‡
IPN	**.76** ‡	**.72** ‡	**.66** ‡	**.68** ‡	**.71** ‡	**.79** ‡	**.65** ‡	**.65** ‡	**.73** ‡	**.70** ‡	**.78** ‡	**.59** ‡	–

(d) Head-to-head comparison for ExpectedWins at $p \leq 0.10$ (⋆), $p \leq 0.05$ (†), and $p \leq 0.01$ (‡).

Table 3: Comparison of official CoNLL-2014 ranking and human rankings. Ranges and clusters have been calculated with bootstrap resampling at $p \leq 0.05$.

al. (2013) calculate EW scores as follows:

$$\text{score}_{\text{EW}}(S_i) = \frac{1}{|\{S_j\}|} \sum_{j,j \neq i} \frac{\text{win}(S_i, S_j)}{\text{win}(S_i, S_j) + \text{win}(S_j, S_i)}.$$

TrueSkill. The TrueSkill ranking system (Herbrich et al., 2007) is a skill based ranking system for Xbox Live developed at Microsoft Research. It is used to identify and model player (GEC systems in our case) ability in a game to assign players to competitive matches. The TrueSkill ranking system models each player S_i by two parameters: the average relative ability μ_{S_i} and the degree of uncertainty in the player's ability $\sigma^2_{S_i}$. Maintaining uncertainty allows TS to make greater changes to the ability estimates at the beginning and smaller changes after a number of consistent matches has been played. Due to that TS can identify the ability of individual players from a smaller number of pairwise comparisons.

A modification of this approach to the WMT manual evaluation procedure by Sakaguchi et al. (2014) has been adopted as the official ranking method during WMT14 replacing EW. The TrueSkill scores are calculated as inferred means:

$$\text{score}_{TS}(S_i) = \mu_{S_i}.$$

5.2 Rank clusters

Both ranking methods produce total orderings without information on the statistical significance of the obtained ranks. Bojar et al. (2014) notice that the similarity of the participants in terms of methods and training data causes some of them to be very similar and group systems into equivalence classes as proposed by Koehn (2012).

Although the methods and training data among the systems examined in this paper are quite diverse, a great similarity of produced outputs is an inherent property of GEC. Therefore, in this section, for each system S_j placed on rank r_j we also try to determine the true systems rank ranges $[r'_j, \ldots, r''_j]$ at a confidence-level of 95% and clusters of equivalent systems by following the procedure outlined by Koehn (2012).

This is accomplished by applying bootstrap resampling. Pairwise rankings are drawn from the set of judgments with multiple drawings. Based on this sample a new ranking is produced. After repeating this process a 1000 times the obtained 1000 ranks for S_j are sorted, with the top 25 and bottom 25 ranks being discarded. The interval of the remaining ranks serves as the final rank range. Next, these rank ranges are used to produce clusters of overlapping rank ranges. This is the last step required to produce the rankings in Tables 3b and 3c for both methods, EW and TS, respectively.

5.3 Choosing the final ranking

Now, we face the question which ranking should be presented as the final result of the human evaluation task. Again, we turn to Bojar et al. (2014) who choose their rankings based on the ranking model's ability to predict pairwise rankings. Accuracy is computed by 100-fold cross-validation. For each fold a new ranking is trained from 99 parts with the left-over part serving as test data.

In a first step, we calculate the accuracy of the unclustered total orderings discarding ties. A ranking based on model scores alone cannot predict ties, this requires equivalence classes. Bojar et al. (2014) define a draw radius r such that systems whose scores differ by less than r are assigned to one cluster, r is tuned to maximize accuracy.

In our case, due to the large number of ties, their method of tuning r is trapped in local maxima and assigns all systems to a single cluster. Alternatively, we propose to calculate clusters according to the method described in the previous section.

Method	EW	TS
Total ordering (non-ties)	58.18	58.15
Bootstrapped clusters	40.12	39.48

Table 4: Accuracy for ranking-based prediction of pairwise judgments.

By fixing $p \leq 0.05$ we directly evaluate rankings of the form given in Table 3. The absolute values of scores and their different interpretations between methods become irrelevant which makes it unnecessary to tune a parameter like r. The main drawback of this approach is its computational cost. For each of the 100 folds we bootstrap another 100 rankings with EW and TS, fix $p \leq 0.05$ and calculate rank clusters. The single clustered ranking for each fold is then used to calculate accuracy for the held-out test data.

For our data, contrary to the MT-specific results from Bojar et al. (2014), EW beats TS in both cases (Table 4). We therefore present the ExpectedWins-based ranking (Table 3b) as the final result of the human evaluation effort described in this work and refer to it in the remainder of the paper when the human ranking is mentioned.

5.4 Analysis

The final human-created ranking (Table 3b) consists of four non-overlapping rank clusters. Rank ranges have been calculated at a confidence level of 95%. Comparing the official CoNLL-2014 ranking (Table 3a) with the manually created ExpectedWins ranking shows interesting differences.

The AMU system is judged to be a clear leader by human judges in its own rank cluster. For six out of eight judges, AMU has the highest score (Table 7). The officially winning system CAMB occupies third place in terms of EW scores and is placed in the second cluster with four systems. Only one judge put CAMB in first place. RAC, a middling system, is elevated to second place occupying a rank cluster with three other systems. NTHU, another middling system that based on M^2 should be similar to RAC, is put in the second to last position. Two systems are judged to be worse than INPUT. The rank cluster that includes INPUT is the largest among the four clusters.

We also include pairwise comparisons between all systems according to EW in Table 3d. Each cell contains the percentage of times the system in that column was judged to be better than the system in

that row. Bold values mark the winner. We applied the Sign Test to measure statistically significant differences, $\star$ indicates statistical significance at $p \leq 0.10$, † at $p \leq 0.05$, and ‡ at $p \leq 0.01$.

6 Correlation with GEC metrics

Since WMT08 (Callison-Burch et al., 2008) the "metrics task" has been part of the WMT. The aim of the metrics task is to assess the quality of automatic evaluation metrics for MT in terms of correlation with the collected human judgments. We attempt the same in the context of GEC.

6.1 Measures of correlation

Based on Macháček and Bojar (2013), we use Spearman's rank correlation ρ and Pearson's r to compare the similarity of rankings produced by various metrics to the manual ranking from the previous section.

Spearman's rank correlation ρ. Spearman's ρ for rankings with no ties is defined as

$$\rho = 1 - \frac{6 \sum d_i^2}{n(n^2-1)}$$

where d_i is the distance between human and metric rank for system i, n is the number of systems.

Pearson's r. Macháček and Bojar (2013) find that Spearman's ρ is too harsh and propose to also use Pearson's r, calculated as

$$r = \frac{\sum_{i=1}^{n}(H_i - \bar{H})(M_i - \bar{M})}{\sqrt{\sum_{i=1}^{n}(H_i - \bar{H})^2}\sqrt{\sum_{i=1}^{n}(M_i - \bar{M})^2}}$$

where H and M are the vectors of human and metric scores, $\bar{H}$ and $\bar{M}$ are corresponding means.

6.2 Metrics

The inventory of evaluation metrics for GEC is significantly smaller than for MT. We hope that making our data available will fuel the interest in this area. The following metrics are assessed:

MaxMatch (M^2). Due to its adoption as the main evaluation metric of the CoNLL shared tasks and the QALB shared tasks (Mohit et al., 2014), the M^2 metric (Dahlmeier and Ng, 2012) can be seen as a *de facto* standard. Being an F_β-score, M^2 results are most influenced by the choice of β. Between the CoNLL-2013 and CoNLL-2014 shared tasks, the organizers changed β from 1.0 to 0.5, and motivate this with intuition alone. The QALB shared tasks for Arabic continue to use $\beta = 1.0$.

Metric	Spearman's ρ	Pearson's r
M^2 $F_{1.0}$	0.648	0.610
M^2 $F_{0.5}$*	0.692	0.627
M^2 $F_{0.25}$	0.720	0.680
M^2 $F_{0.18}$	**0.758**	**0.701**
M^2 $F_{0.1}$	0.670	0.652
I-WAcc	-0.154	-0.098
BLEU	-0.346	-0.240
METEOR	-0.374	-0.241

Table 5: Correlation results for various metrics and human ranking.

	1	2	3	4	5	6	7	8	ρ	$\bar{\rho}$
1	–	.70	.31	.76	.74	.19	.62	.48	.70	
2	.72	–	.77	.84	.90	.57	.59	.64	.93	
3	.53	.89	–	.66	.70	.58	.42	.64	.63	
4	.82	.79	.69	–	.91	.42	.67	.54	.91	.72
5	.65	.85	.82	.87	–	.63	.63	.51	.93	
6	.32	.71	.67	.56	.86	–	.63	.39	.42	
7	.72	.74	.57	.76	.72	.63	–	.63	.76	
8	.64	.85	.86	.69	.72	.57	.75	–	.60	
r	.67	.93	.82	.87	.92	.66	.80	.82	–	
$\bar{r}$				.80						

Table 6: Inter-annotator correlation (Spearman's ρ above the diagonal, Pearson's r below).

I-measure/Weighted Accuracy (I-WAcc). The recently proposed I-WAcc metric (Felice and Briscoe, 2015) tries to address the shortcomings of M^2. The inclusion of true negatives into the formula makes this a very conservative metric; quite similar to the MT metrics described below. The metric assigns negative weights to systems that are harmful with regard to the input text, values from the range $[1, -1]$ are possible. The reported correlation values have been calculated for the ranking presented in Felice and Briscoe (2015).

Machine translation evaluation metrics. Basing most of our results on findings from MT, we also take a look at two machine translation evaluation metrics, BLEU (Papineni et al., 2002) and METEOR (Denkowski and Lavie, 2011). In order to use the CoNLL-2014 gold standard with these metrics, the edit-based annotation has been converted into two plain text files, one per annotator.

6.3 Analysis

The correlation results are collected in table 5. The M^2 metric is generally moderately correlated with

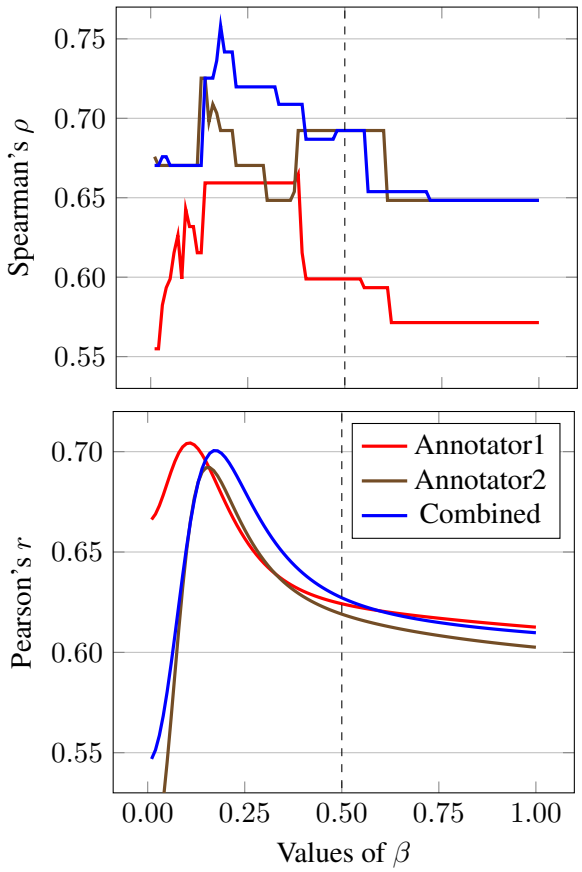


Figure 3: Spearman's ρ and Pearson's r correlation of M^2 with human judgment w.r.t. β. Dashed line marks official CoNLL-2014 choice $\beta = 0.5$.

human judgment and is on the brink of high correlation for values of β closer to 0.2. Compared to M^2, the other metrics are weakly or moderately inversely correlated to human judgment. Inverse correlation with human judgments for metrics that all assign higher scores to better systems seems problematic. In the case of I-WAcc, we would go as far as to state an absence of correlation. It seems the conservative approach adopted for I-WAcc does not correspond to the notion of quality that our judges worked out for themselves. The switch to $\beta = 0.5$ from $\beta = 1.0$ for the CoNLL-2014 shared task was a good choice, but a higher correlation can be achieved for $\beta = 0.25$, the maximum is reached for $\beta = 0.18$. Correlation drops sharply for $\beta = 0.1$. The lack of positive correlation for the MT-metrics is interesting in the light of improvement that results from a shift towards precision for M^2 as BLEU is based on precision.

Figure 3 contains detailed plots of ρ and r with regard to β within the $[0, 1]$ range. As the CoNLL-2014 test data included edits from two annotators, we plot curves for both annotators separately and for the combined gold standard. In the case of Spearman's ρ having alternative error annotations, this leads to higher correlation values. Based on the plots we would recommend setting $0.2 \leq \beta \leq 0.3$ instead of 0.5 or even 1.0.

Inter-annotator correlations of rankings computed for individual judges (Table 6) can be treated as human-level upper bounds for metric correlation. The penultimate column and row contain correlations of rankings for individual judges with rankings computed from all judges minus the respective judge. The last column and row contain the respective weighted (w.r.t. judgments per judge) average of these correlations.

7 Conclusions and future work

We have successfully adapted methods from the WMT human evaluation campaigns to automatic grammatical error correction. The collected and produced data has been made available and should be useful for other researchers. Although we set out to provide answers, we probably ended up with more questions. The following (and more) might be investigated in the future: What makes the winning system special and why do the standard metrics fail at identifying this system? Can we come up with better system-level metrics? Can meaningful sentence-level metrics be developed?

Outside the scope of the particular data, we need to wonder if our results generalize to other shared tasks and other languages. The CoNLL-2014 data concerns ESL learners only and may not be transferable to systems for native speakers. This would be in line with the ideas developed by Chodorow et al. (2012). We would hope to see similar endeavors for the other shared tasks as this would enable the field to draw more general conclusions.

Obtaining the data

The presented data and tools are available from: `https://github.com/grammatical/evaluation`

Acknowledgements

Partially funded by the Polish National Science Centre (Grant No. 2014/15/N/ST6/02330).

The authors would like to thank the following judges for their hard work on the ranking task: Sam Bennett, Peter Dunne, Stacia Levy, Kenneth Turner, and John Winward.

#	Score	Range	System
1	0.674	1-2	CAMB
	0.658	1-2	AMU
2	0.573	3-6	CUUI
	0.573	3-6	PKU
	0.566	3-7	POST
	0.553	3-8	RAC
	0.544	4-8	NTHU
	0.537	5-8	UMC
3	0.436	9-10	SJTU
	0.419	9-11	UFC
	0.387	10-12	IITB
	0.332	11-12	INPUT
4	0.276	13	IPN

(a) Judge 1

#	Score	Range	System
1	0.640	1	AMU
2	0.559	2-6	CUUI
	0.558	2-6	RAC
	0.557	2-7	UFC
	0.543	2-7	CAMB
	0.526	3-9	PKU
	0.511	5-10	POST
	0.511	4-11	IITB
	0.508	5-10	UMC
	0.473	7-11	INPUT
	0.472	8-11	NTHU
3	0.398	12	SJTU
4	0.286	13	IPN

(b) Judge 2

#	Score	Range	System
1	0.612	1-2	AMU
	0.578	2-3	CUUI
	0.564	2-5	UFC
	0.534	3-7	RAC
	0.526	4-7	POST
	0.517	4-8	UMC
	0.508	4-9	IITB
	0.474	5-11	INPUT
	0.467	8-12	SJTU
	0.464	8-12	PKU
	0.463	8-12	CAMB
	0.458	9-12	NTHU
2	0.333	13	IPN

(c) Judge 3

#	Score	Range	System
1	0.641	1-2	AMU
	0.631	1-2	CAMB
2	0.581	3-4	RAC
	0.578	3-4	CUUI
3	0.507	5-10	UFC
	0.494	5-10	POST
	0.490	5-10	UMC
	0.488	5-11	SJTU
	0.486	5-10	PKU
	0.473	5-11	INPUT
	0.441	9-12	NTHU
	0.378	11-12	IITB
4	0.313	13	IPN

(d) Judge 4

#	Score	Range	System
1	0.613	1-2	AMU
	0.608	1-2	RAC
2	0.568	3-5	CUUI
	0.554	3-6	CAMB
	0.535	4-7	POST
	0.526	4-9	UFC
	0.515	5-9	PKU
	0.496	6-10	SJTU
	0.487	6-11	INPUT
	0.472	8-11	IITB
	0.461	9-11	UMC
3	0.375	12	NTHU
4	0.290	13	IPN

(e) Judge 5

#	Score	Range	System
1	0.601	1-2	RAC
	0.579	1-4	AMU
	0.565	1-6	IITB
	0.562	2-6	POST
	0.548	2-8	INPUT
	0.535	3-8	UFC
	0.519	6-9	PKU
	0.516	6-9	CAMB
	0.491	7-10	SJTU
	0.474	9-10	CUUI
2	0.428	11	UMC
3	0.368	12	NTHU
4	0.313	13	IPN

(f) Judge 6

#	Score	Range	System
1	0.788	1	AMU
2	0.697	2	CAMB
3	0.553	3-7	RAC
	0.544	3-7	UMC
	0.537	3-7	POST
	0.533	3-10	IITB
	0.487	5-10	PKU
	0.474	5-12	INPUT
	0.462	6-11	CUUI
	0.436	6-12	SJTU
	0.408	8-12	UFC
	0.386	9-12	NTHU
4	0.303	13	IPN

(g) Judge 7

#	Score	Range	System
1	0.660	1-2	AMU
	0.625	1-3	CUUI
	0.568	2-7	IITB
	0.556	3-6	CAMB
	0.543	3-7	UMC
	0.537	3-7	POST
	0.483	5-10	INPUT
	0.481	5-10	UFC
	0.478	7-11	RAC
	0.466	7-11	NTHU
	0.420	10-12	PKU
	0.419	10-12	SJTU
2	0.260	13	IPN

(h) Judge 8

Table 7: Rankings by individual annotators. Cluster ranks and rank ranges have been computed with bootstrap resampling at $p \leq 0.1$ to accomodate for the reduced number of judgments per judge.

References

Ondřej Bojar, Christian Buck, Chris Callison-Burch, Christian Federmann, Barry Haddow, Philipp Koehn, Christof Monz, Matt Post, Radu Soricut, and Lucia Specia. 2013. Findings of the 2013 Workshop on Statistical Machine Translation. In *Proc. of the Eighth Workshop on Statistical Machine Translation*, pages 1–44. ACL.

Ondrej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Johannes Leveling, Christof Monz, Pavel Pecina, Matt Post, Herve Saint-Amand, et al. 2014. Findings of the 2014 Workshop on Statistical Machine Translation. In *Proc. of the Ninth Workshop on Statistical Machine Translation*, pages 12–58. ACL.

Chris Callison-Burch, Cameron Fordyce, Philipp Koehn, Christof Monz, and Josh Schroeder. 2008. Further meta-evaluation of machine translation. In *Proc. of the Third Workshop on Statistical Machine Translation*, pages 70–106. ACL.

Martin Chodorow, Markus Dickinson, Ross Israel, and Joel R Tetreault. 2012. Problems in evaluating grammatical error detection systems. In *Proc. of COLING 2012*, pages 611–628. ACL.

Jacob Cohen. 1960. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1):37.

Daniel Dahlmeier and Hwee Tou Ng. 2012. Better evaluation for grammatical error correction. In *Proc. of the 2012 Conference of the North American Chapter of the ACL: Human Language Technologies*, pages 568–572. ACL.

Robert Dale and Adam Kilgarriff. 2011. Helping our own: The HOO 2011 pilot shared task. In *Proc. of the 13th European Workshop on Natural Language Generation*, pages 242–249. ACL.

Robert Dale, Ilya Anisimoff, and George Narroway. 2012. Hoo 2012: A report on the preposition and determiner error correction shared task. In *Proc. of the Seventh Workshop on Building Educational Applications Using NLP*, pages 54–62. ACL.

Michael Denkowski and Alon Lavie. 2011. Meteor 1.3: Automatic Metric for Reliable Optimization and Evaluation of Machine Translation Systems. In *Proc. of the EMNLP 2011 Workshop on Statistical Machine Translation*.

Christian Federmann. 2010. Appraise: An open-source toolkit for manual phrase-based evaluation of translations. In *Proc. of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*. ELRA.

Mariano Felice and Ted Briscoe. 2015. Towards a standard evaluation method for grammatical error detection and correction. In *Proc. of the 2015 Conference of the North American Chapter of the ACL: Human Language Technologies (NAACL-HLT 2015)*. ACL.

Ralf Herbrich, Tom Minka, and Thore Graepel. 2007. Trueskill(tm): A bayesian skill rating system. In *Advances in Neural Information Processing Systems 20*, pages 569–576. MIT Press.

Philipp Koehn. 2012. Simulating human judgment in machine translation evaluation campaigns. In *International Workshop on Spoken Language Translation*, pages 179–184.

Matouš Macháček and Ondřej Bojar. 2013. Results of the WMT13 metrics shared task. In *Proc. of the Eighth Workshop on Statistical Machine Translation*, pages 45–51. ACL.

Matouš Macháček and Ondřej Bojar. 2014. Results of the WMT14 metrics shared task. In *Proc. of the Ninth Workshop on Statistical Machine Translation*, pages 293–301. ACL.

Nitin Madnani, Joel Tetreault, Martin Chodorow, and Alla Rozovskaya. 2011. They can help: using crowdsourcing to improve the evaluation of grammatical error detection systems. In *Proc. of the 49th Annual Meeting of the ACL: Human Language Technologies*, pages 508–513. ACL.

Behrang Mohit, Alla Rozovskaya, Nizar Habash, Wajdi Zaghouani, and Ossama Obeid. 2014. The first QALB shared task on automatic text correction for arabic. *Proc. of the EMNLP 2014 Workshop on Arabic Natural Language Processing*, pages 39–47.

Courtney Napoles, Keisuke Sakaguchi, Matt Post, and Joel Tetreault. 2015. Ground truth for grammaticality correction metrics. In *Proc. of the 53rd Annual Meeting of the Association for Computational Linguistics*, pages 588–593. ACL.

Hwee Tou Ng, Siew Mei Wu, Yuanbin Wu, Christian Hadiwinoto, and Joel Tetreault. 2013. The CoNLL-2013 shared task on grammatical error correction. In *Proc. of the 17th Conference on Computational Natural Language Learning*, pages 1–12. ACL.

Hwee Tou Ng, Siew Mei Wu, Ted Briscoe, Christian Hadiwinoto, Raymond Hendy Susanto, , and Christopher Bryant. 2014. The CoNLL-2014 shared task on grammatical error correction. In *Proc. of the Eighteenth Conference on Computational Natural Language Learning*, pages 1–14. ACL.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *Proc. of the 40th Annual Meeting on ACL*, pages 311–318. ACL.

Keisuke Sakaguchi, Matt Post, and Benjamin Van Durme. 2014. Efficient elicitation of annotations for human evaluation of machine translation. In *Proc. of the Ninth Workshop on Statistical Machine Translation*, pages 1–11. ACL.

Learning a Deep Hybrid Model for Semi-Supervised Text Classification

Alexander G. Ororbia II, C. Lee Giles, David Reitter
College of Information Sciences and Technology
The Pennsylvania State University, University Park, PA
{ago109, giles, reitter}@psu.edu

Abstract

We present a novel fine-tuning algorithm in a deep hybrid architecture for semi-supervised text classification. During each increment of the online learning process, the fine-tuning algorithm serves as a top-down mechanism for pseudo-jointly modifying model parameters following a bottom-up generative learning pass. The resulting model, trained under what we call the *Bottom-Up-Top-Down* learning algorithm, is shown to outperform a variety of competitive models and baselines trained across a wide range of splits between supervised and unsupervised training data.

1 Introduction

Recent breakthroughs in learning expressive neural architectures have addressed challenging problems in domains such as computer vision, speech recognition, and natural language processing. This success is owed to the representational power afforded by deeper architectures supported by long-standing theoretical arguments (Hastad, 1987). These architectures efficiently model complex, highly varying functions via multiple layers of non-linearities, which would otherwise require very "wide" shallow models that need large quantities of samples (Bengio, 2012). However, many of these deeper models have relied on mini-batch training on large-scale, labeled data-sets, either using unsupervised pre-training (Bengio et al., 2007) or improved architectural components (such as activation functions) (Schmidhuber, 2015).

In an online learning problem, samples are presented to the learning architecture at a given rate (usually with one-time access to these data points), and, as in the case of a web crawling agent, most of these are unlabeled. Given this, batch training and supervised learning frameworks are no longer applicable. While incremental approaches such as co-training have been employed to help these models learn in a more update-able fashion (Blum and Mitchell, 1998; Gollapalli et al., 2013), neural architectures can naturally be trained in an online manner through the use of stochastic gradient descent (SGD).

Semi-supervised online learning does not only address practical applications, but it also reflects some challenges of human category acquisition (Tomasello, 2001). Consider the case of a child learning to discriminate between object categories and mapping them to words, given only a small amount of explicitly labeled data (the mother pointing to the object), and a large portion of unsupervised learning, where the child comprehends an adult's speech or experiences positive feedback for his or her own utterances regardless of their correctness. The original argument in this respect applied to grammar (e.g., Chomsky, 1980; Pullum & Scholz, 2002). While neural networks are not necessarily models of actual cognitive processes, semi-supervised models can show learnability and illustrate possible constraints inherent to the learning process.

The contribution of this paper is the development of the *Bottom-Up-Top-Down* learning algorithm for training a Stacked Boltzmann Experts Network (SBEN) (Ororbia II et al., 2015) hybrid architecture. This procedure combines our proposed top-down fine-tuning procedure for jointly modifying the parameters of a SBEN with a modified form of the model's original layer-wise bottom-up learning pass (Ororbia II et al., 2015). We investigate the performance of the constructed deep model when applied to semi-supervised text classification problems and find that our hybrid architecture outperforms all baselines.

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 471–481,
Lisbon, Portugal, 17-21 September 2015.

2 Related Work

Recent successes in the domain of connectionist learning stem from the expressive power afforded by models, such as the Deep Belief Network (DBN) (Hinton et al., 2006; Bengio et al., 2007) or Stacked Denoising Autoencoder (Vincent et al., 2010), that greedily learn layers of stacked non-linear feature detectors, equivalent to levels of abstraction of the original representation. In a variety of language-based problems, deep architectures have outperformed popular shallow models and classifiers (Salakhutdinov and Hinton, 2009; Liu, 2010; Socher et al., 2011; Glorot et al., 2011b; Lu and Li, 2013; Lu et al., 2014). However, these architectures often operate in a multi-stage learning process, where a generative architecture is pre-trained and then used to initialize parameters of a second architecture that can be discriminatively fine-tuned (using back-propagation of errors or *drop-out*: Hinton et al., 2012). Several ideas have been proposed to help deep models deal with potentially uncooperative input distributions or encourage learning of discriminative information earlier in the process, many leveraging auxiliary models in various ways (Bengio et al., 2007; Zhang et al., 2014; Lee et al., 2014). A few methods for adapting deep architecture construction to an incremental learning setting have also been proposed (Calandra et al., 2012; Zhou et al., 2012). Recently, it was shown in (Ororbia II et al., 2015) that deep hybrid architectures, or multi-level models that integrate discriminative and generative learning objectives, offer a strong viable alternative to multi-stage learners and are readily usable for categorization tasks.

For text-based classification, a dominating model is the support vector machine (SVM) (Cortes and Vapnik, 1995) with many useful innovations to yet further improve its discriminative performance (Subramanya and Bilmes, 2008). When used in tandem with prior human knowledge to hand-craft good features, this simple architecture has proven effective in solving practical text-based tasks, such as academic document classification (Caragea et al., 2014). However, while model construction may be fast (especially when using a linear kernel), this process is costly in that it requires a great deal of human labor to annotate the training corpus. Our approach, which builds on that of (Ororbia II et al., 2015), provides a means for improving classification performance when labeled data is in scarce supply, learning structure and regularity within the text to reduce classification error incrementally.

3 A Deep Hybrid Model for Semi-Supervised Learning

To directly handle the problem of discriminative learning when labeled data is scarce, (Ororbia II et al., 2015) proposed deep hybrid architectures that could effectively leverage small amounts of labeled and large amounts of unlabeled data. In particular, the best-performing architecture was the Stacked Boltzmann Experts Network (SBEN), which is a variant of the DBN. In its construction and training, the SBEN design borrows many recent insights from efficiently learning good DBN models (Hinton et al., 2006) and is essentially a stack of building block models where each layer of model parameters is greedily modified while freezing the parameters of all others. In contrast to the DBN, which stacks restricted Boltzmann machines (RBM's) and is often used to initialize a deep multi-layer perceptron (MLP), the SBEN model is constructed by composing hybrid restricted Boltzmann machines and can be directly applied to the discriminative task in a single learning phase.

The hybrid restricted Boltzmann machine (HRBM) (Schmah et al., 2008; Larochelle and Bengio, 2008; Larochelle et al., 2012) building block of the SBEN is itself an extension of the RBM meant to ultimately perform classification. The HRBM graphical model is defined via parameters $\Theta = (\mathbf{W}, \mathbf{U}, \mathbf{b}, \mathbf{c}, \mathbf{d})$ (where $\mathbf{W}$ is the input-to-hidden weight matrix, $\mathbf{U}$ the hidden-to-class weight matrix, $\mathbf{b}$ is the visible bias vector, $\mathbf{c}$ is the hidden unit bias vector, and $\mathbf{d}$ is the class unit bias vector), and is a model of the joint distribution of a binary feature vector $\mathbf{x} = (x_1, \cdots, x_D)$ and its label $y \in \{1, \cdots, C\}$ that makes use of a latent variable set $\mathbf{h} = (h_1, \cdots, h_H)$. The model assigns a probability to the triplet (y,$\mathbf{x}$,$\mathbf{h}$) using:

$$p(y, \mathbf{x}, \mathbf{h}) = \frac{e^{-E(y,\mathbf{x},\mathbf{h})}}{Z}, \tag{1}$$

$$p(y, \mathbf{x}) = \frac{1}{Z} \sum_{\mathbf{h}} e^{-E(y,\mathbf{x},\mathbf{h})} \tag{2}$$

where Z is known as the partition function. The model's energy function is defined as

$$E(y, \mathbf{x}, \mathbf{h}) = -\mathbf{h}^T\mathbf{W}\mathbf{x} - \mathbf{b}^T\mathbf{x} - \mathbf{c}^T\mathbf{h} - \mathbf{d}^T\mathbf{e}_y - \mathbf{h}^T\mathbf{U}\mathbf{e}_y. \tag{3}$$

where $\mathbf{e}_y = (\mathbf{1}_{i=y})_{i=1}^{C}$ is the one-hot vector encoding of y. It is often not possible to compute $p(y, \mathbf{x}, \mathbf{h})$ or the marginal $p(y, \mathbf{x})$ due to the intractable normalization constant. However, exploiting the model's lack of intra-layer connections, block Gibbs sampling may be used to draw samples of the HRBM's latent variable layer given the current state of the visible layer and vice versa. This yields the following equations:

$$\begin{aligned} p(\mathbf{h}|y, \mathbf{x}) &= \prod_j p(h_j|y, \mathbf{x}), \\ p(h_j = 1|y, \mathbf{x}) &= \sigma(c_j + U_{jy} + \sum_i W_{ji}x_i) \end{aligned} \tag{4}$$

$$\begin{aligned} p(\mathbf{x}|\mathbf{h}) &= \prod_i p(x_i|\mathbf{h}), \\ p(x_i = 1|\mathbf{h}) &= \sigma(b_i + \sum_j W_{ji}h_j) \end{aligned} \tag{5}$$

$$p(y|\mathbf{h}) = \frac{e^{d_y + \sum_j U_{jy}h_j}}{\sum_{y^\star} e^{d_{y^\star} + \sum_j U_{jy^\star}h_j}} \tag{6}$$

where $\sigma(v) = 1/(1 + e^{-v})$. Classification may be performed directly with the HRBM by using its free energy function $F(y, \mathbf{x})$ to compute the conditional distribution

$$p(y|\mathbf{x}) = \frac{e^{-F(y,\mathbf{x})}}{\sum_{y^\star \in \{1,\cdots,C\}} e^{-F(y^\star,\mathbf{x})}} \tag{7}$$

where the free energy is formally defined as

$$-F(y, \mathbf{x}) = (d_y + \sum_j \psi(c_j + U_{jy} + \sum W_{ji}x_i)) \tag{8}$$

and ψ is the softplus activation function $\psi(v) = \log(1 + e^v)$.

To construct an *N*-layer SBEN (or *N*-SBEN), as was shown in (Ororbia II et al., 2015), one may learn a stack of HRBMs in one of two ways: (1) in a strict greedy, layer-wise manner, where layers are each trained in isolation on all of the data samples one at a time from the bottom-up; or (2) in a more relaxed disjoint fashion, where all layers are trained together on all of the data but still in a layer-wise bottom-up pass. To properly compute intermediate data representations during training and prediction in the SBEN, one must combine Equations 4 and 7. (The specific procedure for doing this can be found in the *computeLayerwiseStatistics* sub-routine in Algorithm 1.) This gives rise to the full SBEN architecture, which is depicted in Figure 1.

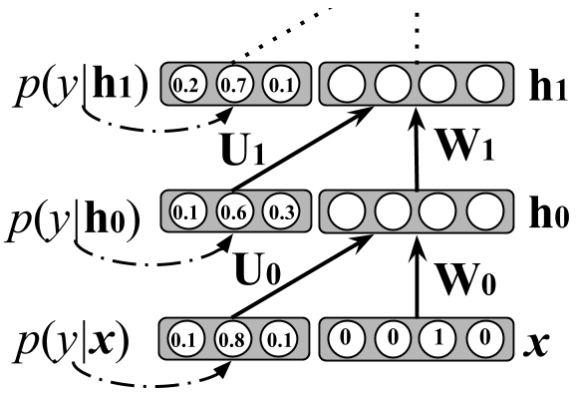


Figure 1: Architecture of the SBEN model. The model in feedforward mode can be viewed as a directed model, however, during training, connections are bi-directional.

3.1 Ensembling of Layer-Wise Experts

The SBEN may be viewed as a natural vertical ensemble of layer-wise "experts", where each layer maps latent representations to predictions, which differs from standard methods such as boosting (Schapire, 1990). Traditional feedforward neural models propagate data through the final network to obtain an output prediction y_t from a penultimate layer for a given $\mathbf{x}_t$. In contrast, this hybrid model is capable of a producing a label y_t^n at each level *n* for $\mathbf{x}_t$.

To vertically aggregate layer-wise expert outputs, we compute a simple mean predictor, $p(y|\mathbf{x})_{ensemble}$, as follows:

$$p(y|\mathbf{x})_{ensemble} = \frac{1}{N}\sum_{n=1}^{N} p(y|\mathbf{x})_n \tag{9}$$

This ensembling scheme provides a simple way to incorporate acquired discriminative knowledge of different levels of abstraction into the model's final prediction. We note that the SBEN's inherent layer-wise discriminative ability stands as an alternative to coupling helper classifiers (Bengio et al., 2007) or the "companion objectives" (Lee et al., 2014).

3.2 The Bottom-Up-Top-Down Learning Algorithm

With the SBEN architecture defined, we next present its simple two-step training algorithm, or the *Bottom-Up-Top-Down* procedure (*BUTD*), which combines a greedy, bottom-up pass with a subsequent top-down fine-tuning step. At every iteration of training, the model makes use of a single labeled sample (taken from an available, small labeled data subset) and an example from either a large unlabeled pool or a data-stream. We describe each of the two phases in Sections 3.2.1 and 3.2.2.

3.2.1 Bottom-Up Layer-wise Learning (BU)

The first phase of *N*-SBEN learning consists of a bottom-up pass where each layerwise HRBM can be trained using a compound objective function. Data samples are propagated up the model to the layer targeted for layer-wise training using the feedforward schema described above. Each HRBM layer of the SBEN is greedily trained using the frozen latent representations of the one below, which are generated by using the lower level expert's input and prediction. The loss function for each layer balances a discriminative objective $\mathcal{L}_{disc}$, a supervised generative objective $\mathcal{L}_{gen}$, and an unsupervised generative objective $\mathcal{L}_{unsup}$, fully defined as follows:

$$\begin{aligned}\mathcal{L}_{semi}(\mathcal{D}_{train}, \mathcal{D}_{unlab}) = \gamma\mathcal{L}_{disc}(\mathcal{D}_{train}) \\ +\alpha\mathcal{L}_{gen}(\mathcal{D}_{train}) \\ +\beta\mathcal{L}_{unsup}(\mathcal{D}_{unlab})\end{aligned} \quad (10)$$

Unlike generative pre-training of neural architectures (Bengio et al., 2007), the additional free parameters γ, α, and β offer explicit control over the extent to which the final parameters discovered are influenced by generative learning (Larochelle et al., 2012; Ororbia II et al., 2015). More importantly, the generative objectives may be viewed as providing data-dependent regularization on the discriminative learning gradient of each layer.

The objectives themselves are defined as:

$$\mathcal{L}_{disc}(\mathcal{D}_{train}) = -\sum_{t=1}^{|\mathcal{D}_{train}|} \log p(y|\mathbf{x}_t), \quad (11)$$

$$\mathcal{L}_{gen}(\mathcal{D}_{train}) = -\sum_{t=1}^{|\mathcal{D}_{train}|} \log p(y_t, \mathbf{x}_t), \text{ and} \quad (12)$$

$$\mathcal{L}_{unsup}(\mathcal{D}_{unlab}) = -\sum_{t=1}^{|\mathcal{D}_{unlab}|} \log p(\mathbf{x}_t) \quad (13)$$

where $\mathcal{D}_{train} = \{(\mathbf{x}_t, y)\}$ is the labeled training data-set and $\mathcal{D}_{unlab} = \{(\mathbf{u}_t)\}$ is the unlabeled training data-set. The gradient for $\mathcal{L}_{disc}$ may be computed directly, which follows the general form

$$\begin{aligned}\frac{\partial \log p(y_t|\mathbf{x})}{\partial\theta} = -\mathbb{E}_{\mathbf{h}|y_t,\mathbf{x}_t}\left[\frac{\partial}{\partial\theta}\left(\mathbb{E}(y_t, \mathbf{x}_t, \mathbf{h})\right)\right] \\ +\mathbb{E}_{y,\mathbf{h}|,\mathbf{x}}\left[\frac{\partial}{\partial\theta}\left(\mathbb{E}(y, \mathbf{x}, \mathbf{h})\right)\right]\end{aligned} \quad (14)$$

and can be calculated directly (see Larochelle et al., 2012 , for details) or through a form of *Dropping*, such as *Drop-Out* or *Drop-Connect* (Tomczak, 2013). The generative gradients themselves follow the form

$$\begin{aligned}\frac{\partial \log p(y_t, \mathbf{x})}{\partial\theta} = -\mathbb{E}_{\mathbf{h}|y_t,\mathbf{x}_t}\left[\frac{\partial}{\partial\theta}\left(\mathbb{E}(y_t, \mathbf{x}_t, \mathbf{h})\right)\right] \\ +\mathbb{E}_{y,\mathbf{x},\mathbf{h}}\left[\frac{\partial}{\partial\theta}\left(\mathbb{E}(y, \mathbf{x}, \mathbf{h})\right)\right]\end{aligned} \quad (15)$$

and, despite being intractable for any sample $(\mathbf{x}_t, y_t)$, may be approximated via the contrastive divergence procedure (Hinton, 2002). The intractable second expectation is replaced with a point estimate using a single Gibbs sampling step. To calculate the generative gradient for an unlabeled sample $\mathbf{u}$, a pseudo-label must be obtained by using a layer-wise HRBM's current estimate of $p(y|\mathbf{u})$, which can be viewed as a form of self-training or *Entropy Regularization* (Lee, 2013). The online procedure for computing the generative gradient (either labeled or unlabeled example) for a single HRBM can be found in Ororbia et al., (2015).

Setting the coefficients that control learning objective influences can lead to different model configurations (especially with respect to γ) as well as impact the gradient-based training of each model layer (i.e., α and β). In this paper, we shall explore two particular configurations, namely 1) by setting $\gamma = 0$ and $\alpha = 1$, which leads to constructing a purely generative model of $\mathcal{D}_{train}$ and

Algorithm 1 Top-down fine-tuning of an *N*-SBEN (*ensemble back-propagation*). Note that "$\cdot$" indicates a Hadamard product, ξ is an error signal vector, the prime superscript indicates a derivative (i.e., σ' means derivative function of the sigmoid), and $\widehat{\mathbf{z}}$ is the symbol for linear pre-activation values.

Input: $(\mathbf{x}_t, y_t) \in \mathcal{D}$, learning rate λ and model parameters $\Theta = \{\Theta_1, \Theta_2, ..., \Theta_N\}$
function FINETUNEMODEL($(\mathbf{x}_t, y_t)$, λ, Θ)
  $\Omega \leftarrow \emptyset$, $\mathbf{x}_n \leftarrow \mathbf{x}_t$, $y_n \leftarrow \emptyset$ ▷ Initialize list of layer-wise model statistics & variables
  // Conduct feed-forward pass to collect layer-wise statistics
  for $\Theta_n \in \Theta$ **do**
    $(\mathbf{h}_n, \widehat{\mathbf{z}}_n, y_n^h, \mathbf{x}_n) \leftarrow$ COMPUTELAYERWISESTATISTICS$(\mathbf{x}_n, \Theta_n)$
    $\Omega_n \leftarrow (\mathbf{h}_n, \widehat{\mathbf{z}}_n, y_n^h, \mathbf{x}_n)$, $\mathbf{x}_n \leftarrow \mathbf{h}_n$, $y_n \leftarrow y_n^h$
  // Conduct error back-propagation pass to adjust layer-wise parameters
  $\xi_l \leftarrow \emptyset$
  for $l \leftarrow N, l--,\ while\ l \geq 1$ **do**
    $(\mathbf{h}_l, \widehat{\mathbf{z}}_l, y_l^h, \mathbf{x}_l) \leftarrow \Omega[l]$ ▷ Grab relevant statistics for layer *l* of model
    if $i = N$ **then**
      $(\bigtriangledown_{disc}, \xi_l) \leftarrow$ COMPUTEDISCIMINATIVEGRADIENT$(y_t, \mathbf{x}_l, \emptyset, \mathbf{h}_n, \widehat{\mathbf{z}}, \Theta_l)$
    else
      $\xi_l \leftarrow \xi_l \cdot \sigma'(\widehat{\mathbf{z}}_l)$
      $(\bigtriangledown_{disc}, \xi_l) \leftarrow$ COMPUTEDISCIMINATIVEGRADIENT$(y_t, \mathbf{x}_l, \xi_l, \mathbf{h}_n, \widehat{\mathbf{z}}, \Theta_l)$
    $\Theta_n \leftarrow \Theta_n - \lambda(\bigtriangledown_{disc})$
function COMPUTELAYERWISESTATISTICS($\mathbf{x}_t, \Theta_n$)
  $y_t^h \leftarrow p(y_t|\mathbf{x}_t, \Theta_n)$ ▷ Equation 7 under the layerwise model
  $\widehat{\mathbf{z}} \leftarrow \mathbf{c} + W\mathbf{x}_t + U\mathbf{e}_{y_t}$ ▷ Can re-use $\widehat{\mathbf{z}}$ to perform next step
  $\mathbf{h}_t \sim p(\mathbf{h}|y_t^h, \mathbf{x}_t, \Theta_n)$ ▷ Equation 4 under the layerwise model
  return $(\mathbf{h}_t, \widehat{\mathbf{z}}, y_t^h, \mathbf{x}_t)$
function COMPUTEDISCIMINATIVEGRADIENT($y_t, \mathbf{x}_l, \xi_l, \mathbf{h}_n, \widehat{\mathbf{z}}, \Theta_l$)
  $\mathbf{o} \leftarrow p(y|\mathbf{h}_n, \Theta_l)$, $\xi \leftarrow softmax'(\mathbf{o}) \cdot -(y_t/\mathbf{o})$
  $\bigtriangledown_U \leftarrow \xi\mathbf{h}_n^T$, $\bigtriangledown_d \leftarrow \xi$, $\xi \leftarrow U\xi$, $\xi \leftarrow \xi \cdot \sigma'(\widehat{\mathbf{z}})$
  if $\xi_l \neq \emptyset$ **then**
    $\xi \leftarrow \xi \cdot \xi_l$
  $\bigtriangledown_W \leftarrow \xi\mathbf{x}_l^T$, $\bigtriangledown_c \leftarrow \xi$, $\bigtriangledown_b \leftarrow 0$, $\bigtriangledown_U \leftarrow \bigtriangledown_U + (\xi\mathbf{e}_{y_t}^T)$, $\xi \leftarrow W^T\xi$
  return $(\bigtriangledown \leftarrow (\bigtriangledown_W, \bigtriangledown_U, \bigtriangledown_b, \bigtriangledown_c, \bigtriangledown_d), \xi)$

$\mathcal{D}_{unsup}$, and 2) by setting $\gamma = 1$ with α freely varying (which recovers the model of Ororbia et al., 2015). In both scenarios, β is allowed to vary as a user-defined hyper-parameter. The second setting of γ allows for training the SBEN directly with only the bottom-up phase defined in this section. However, if the first setting is used, a second phase may be used to incorporate a top-down fine-tuning phase. A bottom-up pass simply entails computing this compound gradient for each layer of the model for 1 or 2 samples per training iteration. Notice that the first scenario reduces the number of hyper-parameters to explore in model selection, requiring only an appropriate value for β to be found.

3.2.2 Top-Down Fine-tuning (TD)

Although efficient, the bottom-up procedure described above is greedy, which means that the gradients are computed for each layer-wise HRBM independent of gradient information from other layers of the model. One way we propose to introduce a degree of joint training of parameters is to incorporate a second phase that adjusts the SBEN parameters via a modified form of back-propagation. Such a routine can further exploit the SBEN's multiple predictors (or entry points) where additional error signals may be computed and aggregated while signals are reverse-propagated down the network. We hypothesize that holistic fine-tuning ensures that discriminative information is incorporated into the generative

Algorithm 2 The *Bottom-Up-Top-Down* training procedure for learning an *N*-SBEN.

Input: $(\mathbf{x}_t, y_t) \in \mathcal{D}_{train}$, $(\mathbf{u}_t) \in \mathcal{D}_{unlab}$, rates λ & β, $\bar{p}$, & parameters $\Theta = \{\Theta_1, \Theta_2, ..., \Theta_N\}$
function BOTTOMUPTOPDOWN($(y_t, \mathbf{x}_t, \mathbf{u}_t, \lambda, \beta, \Theta)$
  APPLYBOTTOMUPPASS($y_t, \mathbf{x}_t, \mathbf{u}_t, \lambda, \gamma = 0, \alpha = 1, \beta, \Theta$) ▷ See (Ororbia II et al., 2015)
  // Up to two calls can be made to the top-down tuning routine
  FINETUNEMODEL($\mathbf{x}_t, y_t, \lambda, \Theta$) ▷ See Algorithm 1 for details
  $\mathbf{v}_t \leftarrow p_{ensemble}(y|\mathbf{x}, \Theta_n)$ ▷ Calculate pseudo-label probability using Equation 9
  if $max[\mathbf{v}_t] > \bar{p}$ **then**
    $\mathbf{v}_t \leftarrow$ TOONEHOT($\mathbf{v}_t$) ▷ Convert to 1-hot vector using argmax of model conditionals
    FINETUNEMODEL($\mathbf{u}_t, \mathbf{v}_t, \lambda, \Theta$)

features being constructed in the bottom-up learning step. Furthermore, errors from experts above are propagated down to lower layers, which were initially frozen during the greedy, bottom-up training phase.

Fine-tuning in the context of training an SBEN is different from using a pre-trained MLP that is subsequently fine-tuned with back-propagation. First, since the SBEN is a more complex architecture than an MLP, pre-initializing an MLP would be insufficient given that one would be tossing potentially useful information stored in the SBEN's class filters (and corresponding class bias vectors) of each layer-wise expert (i.e., U and $\mathbf{d}$). Second, merely using the SBEN as an intermediate model ignores the fact the SBEN can already perform classification directly. To avoid losing such information and to fully exploit the model's predictive ability, we adapt the back-propagation algorithm for training MLP's to operate on the SBEN, which we shall call *ensemble back-propagation* since the fine-tuning method propagates error derivatives down the network from many points of entry. Ensemble back-propagation is described in Algorithm 1.

With this second online training step, the *Bottom-Up-Top-Down* (*BUTD*) training algorithm for fully training an SBEN proceeds with a single bottom-up modification step followed by a single top-down joint fine-tuning step using the *ensemble back-propagation* procedure defined in Algorithm 1 for each training time step. A full top-down phase can consist of up to two calls to the *ensemble back-propagation* procedure. One is used to jointly modify the SBEN's parameters with respect to the sample taken from $\mathcal{D}_{train}$. A second one is potentially needed to tune parameters with respect to the sample drawn from $\mathcal{D}_{unlab}$. For the unlabeled sample, if the highest class probability assigned by the SBEN (using Equation 9) is greater than a pre-set threshold (i.e., $max[p_{ensemble}(y|\mathbf{u})] > \bar{p}$), a pseudo-label is created for that sample by converting the model's mean vector to a 1-hot encoding. The probability threshold $\bar{p}$ for the potential second call to the *ensemble back-propagation* routine allows us to incorporate a tunable form of pseudo-labeling (Lee, 2013) into the *Bottom-Up-Top-Down* learning algorithm.

The high-level view of the *BUTD* learning algorithm is depicted in Algorithm 2.

4 Experimental Results

We investigate the viability of our deep hybrid architecture for semi-supervised text categorization. Model performance was evaluated on the WebKB data-set [1] and a small-scale version of the 20NewsGroup data-set [2].

The original WebKB collection contains pages from a variety of universities (Cornell, Texas, Washington, and Wisconsin as well as miscellaneous pages from others). The 4-class classification problem we defined using this data-set was to determine if a web-page could be identified as one belonging to a Student, Faculty, Course, or a Project, yielding a subset of usable 4,199 samples. We applied simple pre-processing to the text, namely stop-word removal and stemming, chose to leverage only the k most frequently occurring terms (this varied across the two experiments), and binarized the document low-level representation (only 1 page vector was discarded due to presence of 0 terms). The 20NewsGroup data-set, on the other hand, contained 16242 total samples and was already pre-processed, containing 100 terms, binary-occurrence low-level representation, with

[1] The exact data-set we used can be found and downloaded at http://www.cs.cmu.edu/afs/cs/project/theo-20/www/data/

[2] The exact data-set we used can be found and downloaded at http://www.cs.nyu.edu/roweis/data.html.

tags for the four top-most highest level domains or meta-topics in the newsgroups array.

For both data-sets, we evaluated model generalization performance using a stratified 5-fold cross-validation (CV) scheme. For each possible train/test split, we automatically partitioned the training fold into separate labeled, unlabeled, and validation subsets using stratified random sampling without replacement. Generalization performance was evaluated by estimating classification error, average precision, average recall, and average F-Measure, where F-Measure was chosen to be the harmonic mean of precision and recall, $F1 = 2(\text{precision} \cdot \text{recall})/(\text{precision} + \text{recall})$.

4.1 Model Designs

We evaluated the BUTD version of our model, the *3-SBEN,BUTD*, as described in Algorithm 2. For simplicity, the number of latent variables at each level of the SBEN was held equal to the dimensionality of the data (i.e., a complete representation). We compared this model trained with *BUTD* against a version utilizing only the bottom-up phase (*3-SBEN,BU*) as in Ororbia et al. (2015). Both SBEN models contained 3 layers of latent variables.

We compared against an array of baseline classifiers. We used our implementation of an incremental version of Maximum Entropy, or *MaxEnt-ST* (which, as explained in Sarikaya et al., 2014, is equivalent to a softmax classifier). Furthermore, we used our implementation of the Pegasos algorithm (*SVM-ST*) (Shalev-Shwartz et al., 2011) which was extended to follow a proper multi-class scheme (Crammer and Singer, 2002). This is the online formulation of the SVM, trained via sub-gradient descent on the primal objective followed by a projection step (for simplicity, we opted to using a linear-kernel). Additionally, we implemented a semi-supervised Bernoulli Naive Bayes classifier (*NB-EM*) trained via Expectation-Maximization as in (Nigam et al., 1999). We also compared our model against the HRBM (Larochelle and Bengio, 2008) (effectively a single layer SBEN), which serves as a powerful, non-linear shallow classifier in of itself, as well as a 3-layer sparse deep Rectifier Network (Glorot et al., 2011a), or *Rect*, composed of leaky rectifier units.

All shallow classifiers (except *NB-EM* and the HRBM) were extended to the semi-supervised setting by leveraging a simple self-training scheme in order to learn from unlabeled data samples. The self-training scheme entailed using a classifier's estimate of $p(y|\mathbf{u})$ for an unlabeled sample and, if $max[p(y|\mathbf{u})] > \bar{p}$, we created a 1-hot proxy encoding using the *argmax* of model's predictor, where $\bar{p}$ is a threshold meta-parameter. Since we found this simple pseudo-labeling approach, similar in spirit to (Lee, 2013), to improve the results for all classifiers, and thus we report all results utilizing this scheme. [3] All classes of models (*SBEN*, *HRBM*, *Rect*, *SVM-ST*, *MaxEnt-ST*, *NB-ST*) were subject to the same model selection procedure described in the next section.

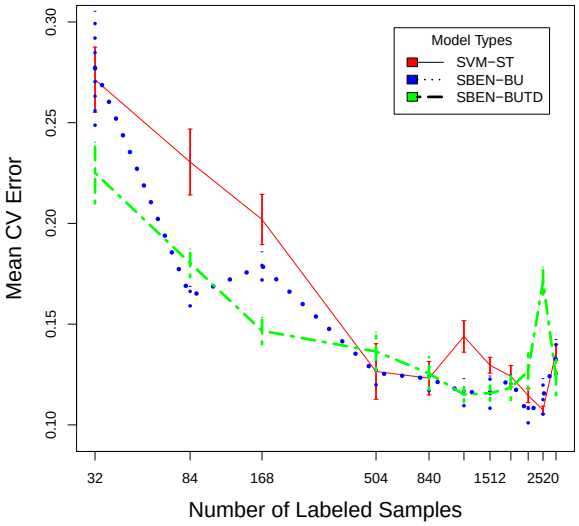


Figure 2: Mean CV generalization performance as a function of labeled sample subset size (using 200 features).

4.2 Model Selection

Model selection was conducted using a parallelized multi-setting scheme, where a configuration file for each model was specified, describing a set of hyper-parameter combinations to explore (this is akin to a course-grained grid search, where the points of model evaluation are set manually a priori). For the SBEN's, we varied the learning rate ($[0.01, 0.25]$) and β coefficient ($[0.1, 1.0]$) and

[3] All model implementations were computationally verified for correctness when applicable. Since most discriminative objectives followed a gradient descent optimization scheme and could be realized in an automatic differentiation framework, we checked gradient validity via finite difference approximation.

Table 1: WEBKB categorization results on 1% of the training data labeled (8 examples per class), rest unlabeled (i.e., 5-fold means with standard error of the mean, 250 features).

	Error	**Precision**	**Recall**	**F1-Score**
NB-EM	0.369 ± 0.039	0.684 ± 0.022	0.680 ± 0.028	0.625 ± 0.043
MaxEnt-ST	0.402 ± 0.026	0.623 ± 0.025	0.593 ± 0.015	0.583 ± 0.020
SVM-ST	0.342 ± 0.020	0.663 ± 0.010	0.665 ± 0.014	0.644 ± 0.015
HRBM	0.252 ± 0.023	0.740 ± 0.019	0.765 ± 0.016	0.741 ± 0.021
3-Rect	0.328 ± 0.020	0.673 ± 0.017	0.680 ± 0.021	0.654 ± 0.023
3-SBEN,BU	0.239 ± 0.015	0.754 ± 0.014	0.780 ± 0.016	0.754 ± 0.015
3-SBEN,BUTD	$\mathbf{0.210 \pm 0.011}$	$\mathbf{0.786 \pm 0.009}$	$\mathbf{0.784 \pm 0.014}$	$\mathbf{0.777 \pm 0.012}$

Table 2: 20NewsGroup data-set categorization results on 1% of the training data labeled (8 examples per class), rest unlabeled (i.e., 5-fold means with standard error of the mean).

	Error	**Precision**	**Recall**	**F1-Score**
NB-EM	0.275 ± 0.006	0.7176 ± 0.010	0.6685 ± 0.010	0.6697 ± 0.010
MaxEnt-ST	0.335 ± 0.005	0.643 ± 0.007	0.643 ± 0.007	0.639 ± 0.007
SVM-ST	0.346 ± 0.008	0.669 ± 0.016	0.644 ± 0.012	0.634 ± 0.011
HRBM	0.284 ± 0.006	0.706 ± 0.012	0.699 ± 0.009	0.696 ± 0.008
3-Rect	0.318 ± 0.009	0.661 ± 0.011	0.661 ± 0.012	0.657 ± 0.011
3-SBEN,BU	0.270 ± 0.006	0.715 ± 0.009	0.714 ± 0.009	0.710 ± 0.007
3-SBEN,BUTD	$\mathbf{0.256 \pm 0.007}$	$\mathbf{0.732 \pm 0.005}$	$\mathbf{0.727 \pm 0.006}$	$\mathbf{0.725 \pm 0.006}$

experimented with stochastic and mean-field versions of the models [4] (we found that mean-field did slightly better for this experiment and thus report the performance of this model in this paper). The HRBM's meta-parameters were tuned using a similar set-up to (Larochelle et al., 2012) with learning rate varied in ($[0.01, 0.25]$), α in ($[0.1, 0.5]$), and β in ($\{0.01, 0.1\}$). For the *SVM-ST* algorithm, we tuned its slack variable λ, searching in the interval $[0.0001, 0.5]$, for *MaxEnt-ST* its learning rate in $[0.0001, 0.1]$, and for $\bar{p}$ of all models (shallow and deep) that used pseudo-labeling we searched the interval $[0.1, 1.0]$. All models of all configurations were trained for a 10,000 iteration sweep incrementally on the data and the model state with lowest validation error for that particular run was used. The *SBEN*, *HRBM*, and *Rect* models were also set to use a momentum term of 0.9 (linearly increased from 0.1 in the first 1000 training iterations) and the *Rect* model made use of a small *L1* regularization penalty to encourage additional hidden sparsity. For a data-set like the 20NewsGroup, which contained a number of unlabeled samples greater than training iterations, we view our schema as simulating access to a data-stream, since all models had access to any given unlabeled example only once during a training run.

[4] Mean-field simply means no sampling steps were taken after computing probability vectors, or "means" in any stage of the computation.

4.3 Model Performance

We first conducted an experiment, using the WebKB data-set, exploring classification error as a function of labeled data subset cardinality (Figure 2). In this setup, we repeated the stratified cross-fold scheme for each possible labeled data subset size, comparing the performance of the SVM model against 3-SBEN,BU (blue dotted curve) and 3-SBEN,*BUTD* (green dash-dotted curve). We see that as the number of labeled examples increases (which entails greater human annotation effort) all models improve, nearly reaching 90% accuracy. However, while the performance difference between models becomes negligible as the training set becomes more supervised, as expected, it is in the less scarce regions of the plot we are interested in. We see that for small proportions, both variants of the SBEN outperform the SVM, and furthermore, the SBEN trained via full *BUTD* can reach lower error, especially for the most extreme scenario where only 8 labeled examples per class are available. We notice a bump in the performance of *BUTD* as nearly the whole training set becomes labeled and posit that since the *BUTD* involves additional pseudo-

Table 3: Top-most words that the SBEN (*BUTD*) model associates with the 4 NewsGroup meta-topics.

Meta-Topic	Associated Terms
*comp.**	$windows, graphics, card, driver, scsi, dos, files, display$
*rec.**	$players, hockey, season, nhl, team, league, baseball, games$
*sci.**	$orbit, shuttle, space, earth, mission, nasa, moon, doctor$
*talk.**	$jews, christian, religion, jesus, bible, war, israel, president$

labeling steps (as in the top-down phase), there is greater risk of reinforcing incorrect predictions in the pseudo-joint [5] tuning of layerwise expert parameters. For text collections where most of the data is labeled and unlabeled data is minimal, only a simple bottom-up pass is needed to learn a good hybrid model of the data.

The next set of experiments was conducted with only 1% of the training sets labeled. We observe (Tables 1 and 2) that our deep hybrid architecture trained via *BUTD* outperforms all other models with respect to all performance metrics. While the SBEN trained with simply an online bottom-up performs significantly better than the SVM model, we note a further reduction of error using our proposed *BUTD* training procedure. The additional top-down phase serves as a mechanism for unifying the layer-wise experts, where error signals for both labeled and pseudo-labeled examples increase agreement among all model layer experts.

For the 20NewsGroup data-set, we conducted a simple experiment to uncover some of the knowledge acquired by our model with respect to the target categorization task. We applied the mechanism from (Larochelle et al., 2012) to extract the variables that are most strongly associated with each of the clamped target variables in the lowest layer of a *BUTD*-trained SBEN. The top-scored terms associated with each class variable are shown in Table 3, using the 10 hidden nodes most highly triggered by the clamped class node, in a model trained on all of the 20NewsGroup data using a model configuration determined from CV results for the 20NewsGroup data-set reported in the paper. Since the SBEN is a composition of layer-wise experts each capable of classification, we note that this procedure could be applied to each level to uncover which unobserved variables are most strongly associated with each class target. We speculate that this could serve the basis for uncovering the model's underlying learnt hierarchy of the data and be potentially used for knowledge extraction, a subject for future work in analyzing black box neural models such as our own.

[5] We use the phrase "pseudo-joint" to differentiate a model that has all its parameters trained jointly from our own, where only the top-down phase of *BUTD* introduces any form of joint parameter modification.

5 Conclusions

We presented the *Bottom-Up-Top-Down* procedure for training the Stacked Boltzmann Experts Network, a hybrid architecture that balances both discriminative and generative learning goals, in the context of semi-supervised text categorization. It combines a greedy, layer-wise bottom-up approach with a top-down fine-tuning method for pseudo-joint modification of parameters.

Models were evaluated using two text corpora: WebKB and 20NewsGroup. We compared results against several baseline models and found that our hybrid architecture outperformed the others in all settings investigated. We found that the SBEN, especially when trained with the full *Bottom-Up-Top-Down* learning procedure could in some cases improve classification error by as much 39% over the Pegasos SVM, and nearly 17% over the HRBM, especially when data is in very limited supply. While we were able to demonstrate the viability of our hybrid model when using only simple surface statistics of text, future work shall include application of our models to more semantic-oriented representations, such as those leveraged in building log-linear language models (Mikolov et al., 2013).

Acknowledgments

A.G.O. acknowledges support from The Pennsylvania State University and the National Science Foundation (DGE-1144860). D.R. acknowledges support from the National Science Foundation (SES-1528409).

References

Yoshua Bengio, Pascal Lamblin, Dan Popovici, Hugo Larochelle, et al. 2007. Greedy layer-wise training of deep networks. *Advances in neural information processing systems*, 19:153.

Yoshua Bengio. 2012. Deep learning of representations for unsupervised and transfer learning. *Journal of Machine Learning Research–Workshop and Conference Proceedings*, 27:17–37.

Avrim Blum and Tom Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, pages 92–100. ACM.

Roberto Calandra, Tapani Raiko, Marc Peter Deisenroth, and Federico Montesino Pouzols. 2012. Learning Deep Belief Networks from Non-stationary Streams. In *Artificial Neural Networks and Machine Learning - ICANN 2012*, number 7553 in Lecture Notes in Computer Science, pages 379–386. Springer Berlin Heidelberg.

Cornelia Caragea, Jian Wu, Kyle Williams, Sujatha Das, Madian Khabsa, Pradeep Teregowda, and C. Lee Giles. 2014. Automatic identification of research articles from crawled documents. In *Proceedings of the Workshop: Web-Scale Classification: Classifying Big Data from the Web*, New York, NY.

Noam Chomsky. 1980. Rules and representations. *Behavioral and brain sciences*, 3(01):1–15.

Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. 20(3):273–297.

Koby Crammer and Yoram Singer. 2002. On the algorithmic implementation of multiclass kernel-based vector machines. *The Journal of Machine Learning Research*, 2:265–292.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011a. Deep sparse rectifier networks. In *Proc. 14th International Conference on Artificial Intelligence and Statistics*, volume 15, pages 315–323.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011b. Domain Adaptation for Large-scale Sentiment Classification: A Deep Learning Approach. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 513–520.

Sujatha Das Gollapalli, Cornelia Caragea, Prasenjit Mitra, and C. Lee Giles. 2013. Researcher Homepage Classification Using Unlabeled Data. In *Proceedings of the 22Nd International Conference on World Wide Web*, WWW '13, pages 471–482. International World Wide Web Conferences Steering Committee.

Johan Hastad. 1987. *Computational limitations of small-depth circuits*. MIT press.

Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh. 2006. A Fast Learning Algorithm for Deep Belief Nets. *Neural computation*, 18(7):1527–1554.

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

Geoffrey E. Hinton. 2002. Training Products of Experts by Minimizing Contrastive Divergence. *Neural Computation*, 14(8):1771–1800.

Hugo Larochelle and Yoshua Bengio. 2008. Classification using Discriminative Restricted Boltzmann Machines. In *Proceedings of the 25th International Conference on Machine Learning*, pages 536–543, Helsinki, Finland.

Hugo Larochelle, Michael Mandel, Razvan Pascanu, and Yoshua Bengio. 2012. Learning Algorithms for the Classification Restricted Boltzmann Machine. *The Journal of Machine Learning Research*, 13:643–669.

Chen-Yu Lee, Saining Xie, Patrick Gallagher, Zhengyou Zhang, and Zhuowen Tu. 2014. Deeply-Supervised Nets. *arXiv:1409.5185 [cs, stat]*.

Dong-Hyun Lee. 2013. Pseudo-label: The Simple and Efficient Semi-supervised Learning Method for Deep Neural Networks. In *Workshop on Challenges in Representation Learning, ICML*, Atlanta, GA.

Tao Liu. 2010. A Novel Text Classification Approach Based on Deep Belief Network. In *Proceedings of the 17th International Conference on Neural Information Processing: Theory and Algorithms - Volume Part I*, ICONIP'10, pages 314–321. Springer-Verlag.

Zhengdong Lu and Hang Li. 2013. A Deep Architecture for Matching Short Texts. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 1367–1375. Curran Associates, Inc.

Shixiang Lu, Zhenbiao Chen, and Bo Xu. 2014. Learning new semi-supervised deep auto-encoder features for statistical machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 122–132, Baltimore, MD.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc., Lake Tahoe, NV.

Kamal Nigam, John Lafferty, and Andrew McCallum. 1999. Using maximum entropy for text classification. In *IJCAI-99 workshop on Machine Learning for Information Filtering*, volume 1, pages 61–67.

Alexander G. Ororbia II, David Reitter, Jian Wu, and C. Lee Giles. 2015. Online learning of deep hybrid architectures for semi-supervised categorization. In *ECML PKDD*, Porto, Portugal. Springer.

Geoffrey K Pullum and Barbara C Scholz. 2002. Empirical assessment of stimulus poverty arguments. *The linguistic review*, 18(1-2):9–50.

Ruslan Salakhutdinov and Geoffrey Hinton. 2009. Semantic hashing. *International Journal of Approximate Reasoning*, 50(7):969–978, July.

R. Sarikaya, G.E. Hinton, and A. Deoras. 2014. Application of Deep Belief Networks for Natural Language Understanding. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 22(4):778–784.

Robert E. Schapire. 1990. The Strength of Weak Learnability. *Machine learning*, 5(2):197–227.

Tanya Schmah, Geoffrey E. Hinton, Steven L. Small, Stephen Strother, and Richard S. Zemel. 2008. Generative versus Discriminative Training of RBMs for classification of fMRI images. In *Advances in neural information processing systems*, pages 1409–1416.

Jürgen Schmidhuber. 2015. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117.

Shai Shalev-Shwartz, Yoram Singer, Nathan Srebro, and Andrew Cotter. 2011. Pegasos: Primal Estimated Sub-gradient Solver for SVM. *Mathematical programming*, 127(1):3–30.

Richard Socher, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher D. Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 151–161. Association for Computational Linguistics.

Amarnag Subramanya and Jeff Bilmes. 2008. Soft-supervised learning for text classification. In *Empirical Methods in Natural Language Processing*, pages 1090–1099.

Michael Tomasello. 2001. Perceiving intentions and learning words in the second year of life. In Melissa Bowerman and Stephen Levinson, editors, *Language acquisition and conceptual development*, pages 132–158.

Jakub M. Tomczak. 2013. Prediction of Breast Cancer Recurrence using Classification Restricted Boltzmann Machine with Dropping. *arXiv preprint arXiv:1308.6324*.

Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. 2010. Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion. *The Journal of Machine Learning Research*, 11:3371–3408.

Junbo Zhang, Guangjian Tian, Yadong Mu, and Wei Fan. 2014. Supervised Deep Learning with Auxiliary Networks. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pages 353–361, New York City, New York. ACM.

Guanyu Zhou, Kihyuk Sohn, and Honglak Lee. 2012. Online Incremental Feature Learning with Denoising Autoencoders. In *International Conference on Artificial Intelligence and Statistics*, pages 1453–1461.

Joint Embedding of Query and Ad by Leveraging Implicit Feedback

Sungjin Lee and Yifan Hu
Yahoo Labs
229 West 43rd Street, New York, NY 10036, USA
{junion, yifanhu}@yahoo-inc.com

Abstract

Sponsored search is at the center of a multibillion dollar market established by search technology. Accurate ad click prediction is a key component for this market to function since the pricing mechanism heavily relies on the estimation of click probabilities. Lexical features derived from the text of both the query and ads play a significant role, complementing features based on historical click information. The purpose of this paper is to explore the use of word embedding techniques to generate effective text features that can capture not only lexical similarity between query and ads but also the latent user intents. We identify several potential weaknesses of the plain application of conventional word embedding methodologies for ad click prediction. These observations motivated us to propose a set of novel joint word embedding methods by leveraging implicit click feedback. We verify the effectiveness of these new word embedding models by adding features derived from the new models to the click prediction system of a commercial search engine. Our evaluation results clearly demonstrate the effectiveness of the proposed methods. To the best of our knowledge this work is the first successful application of word embedding techniques for the task of click prediction in sponsored search.

1 Introduction

Sponsored search is a multibillion dollar market (Easley and Kleinberg, 2010) that makes most search engine revenue and is one of the most successful ways for advertisers to reach their intended audiences. When search engines deliver results to a user, sponsored advertisement impressions (ads) are shown alongside the organic search results (Figure 1). Typically the advertiser pays the search engine based on the *pay-per-click* model. In this model the advertiser pays only if the impression that accompanies the search results is clicked. The price is usually set by a *generalized second-price* (GSP) auction (Edelman et al., 2005) that encourages advertisers to bid truthfully. An advertiser wins if the expected revenue for this advertiser, which is the bid price times the expected click probability (also know as *click through rate*, or CTR), is ranked the highest. The price the advertiser pays, known as *cost-per-click* (CPC), is the bid price for the second ranked advertiser times the ratio of the expected CTR between the second and first ranked advertisers. From this discussion it should be clear that CTR plays a key role in deciding both the ranking and the pricing of the ads. Therefore it is very important to predict CTR accurately.

Figure 1: Sponsored ads when "pizza" was searched at Yahoo! (*www.yahoo.com*).

The state of the art search engine typically uses a machine learning model to predict CTR by exploiting various features that have been found useful in practice. These include historical click performance features such as historical click probability for the query, the ad, the user, and a combination of these; contextual features such as temporal and geographical information; and text-based features such as query keywords or ad title and description. Among these, historical click performance features often have the most predictive power for queries, ads and users that have registered many impressions. For queries, ads and users that have not registered many impressions, however, historical CTR may have too high a variance to be useful. Hillard *et al.* (2011) observed that the number of impressions and clicks recorded on query-ad pairs have a very long tail: only 61% of queries has greater than three clicks. They also reported a drastic drop in the accuracy of the click prediction model when fewer historical observations are available. Furthermore, fine-grained historical CTR information takes a huge amount of space, which

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 482–491,
Lisbon, Portugal, 17-21 September 2015.

makes it costly to maintain. On the other hand, text features are always readily available, and thus are particularly useful for those cases for which there is insufficient historical information.

Multiple researchers, for example (Richardson, 2007; Cheng and Cantú-Paz, 2010), reported the usage of text features including simple lexical similarity scores between the query and ads, word or phrase overlaps and the number of overlapping words and characters. Such features rely on the assumption that query-ad overlap is correlated with perceived relevance. While this is true to a certain extent, the use of simple lexical similarity cannot capture semantic information such as synonyms, entities of the same type and strong relationships between entities (e.g. CEO-company, brand-model, part-of). Recently a host of studies on word embedding have been conducted; all map words into a vector space such that semantically relevant words are placed near each other in the space (Mikolov et al., 2013a; Pennington et al., 2014; Baroni et al., 2014). The use of continuous word vectors has been shown to be helpful for a wide range of NLP tasks by better capturing both syntactic and semantic information than simple lexical features (Socher et al., 2012a).

No previous research on sponsored search has successfully used word embeddings to generate text features. In this paper, we explore the use of word embeddings for click prediction. However, it is clear that conventional word embeddings (which solely rely on word co-occurrence in a context window) can only offer limited discriminative power because queries and ad text are typically very short. In addition, conventional word embeddings cannot capture user intents, preferences and desires. Wang et al. (2013) showed that specific frequently occurring lexical patterns, e.g., *x% off*, *guaranteed return in x days* and *official site*, are effective in triggering users desires, and thus lead to significant differences in CTR. Conventional word embeddings cannot capture these phenomena since they do not incorporate the implicit feedback users provide through clicks and non-clicks. These observations naturally lead us to leverage click feedback to infuse users' intentions and desires into the vector space.

The simplest way to harness click feedback is to train conventional word embedding models on a corpus that only includes clicked impressions, where each "sentence" is constructed by mixing the query and ad text. Having trained a word embedding model, we simply take the average of word vectors of the query and ads respectively to obtain sentence (or paragraph) vectors, which in turn are used to compute the similarity scores between the query and ads. Our experiments show that this method does improve click prediction performance. However, this method has several potential weaknesses. First, the use of only clicked impressions ignores the large amount of negative signals contained in the non-clicked ad impressions. Second, the use of indirect signals (word co-occurrences) can be noisy or even harmful to our ultimate goal (accurate click prediction) when it is combined with direct signals (impressions with click feedback). Third, without explicit consideration about the averaging step in the training process of word embedding models, a simple averaging scheme across word vectors may be a suboptimal. We therefore propose several joint word embedding models; all of these aim to put query vectors close to relevant ad vectors by explicitly utilizing both positive and negative click feedback. We evaluate all these models against a large sponsored search data set from a commercial search engine, and demonstrate that our proposed models significantly improve click prediction performance.

The rest of this paper is organized as follows. In Section 2 we present a brief summary of related work. In Section 3 we give some background information on ad click prediction in sponsored search. In Section 4 we describe our methods. In Section 5 we discuss our experiments. We finish with some conclusions and future directions in Section 6.

2 Related Work

Text features for predicting click probability There have been many studies on the use of text features for click prediction. For example, Dembczynski et al. (2008) used a decision rule-based approach involving such lexical features as the number of words in ad title and description, the number of segments and length of the ad URL, and individual words and terms in ad title and description. Cheng et al. (2010) used a logistic regression model that used both historical click performance features and simple lexical features such as word or phrase overlap between query and ad title and description. Trofimov et al. (2012) used a variant of boosted decision trees with similar features. Richardson et al. (2007) specifically considered new ads (which lack historical click prediction data) and proposed to use the CTR for ad terms, the frequency of certain unigrams (e.g., dollar signs) and general English usage patterns, and simple lexical distance between the query and ads. In all this previous work, text features consisted only of surface-level text features. To the best of our knowledge, there is no previous work adopting semantic-level text features for the purpose of click prediction, in particular word embeddings to measure query-ad relevance. In a similar vein of research, Grbovic et al. (2015) adopted word embeddings to the task of query rewriting for a better match between queries and keywords that advertisers entered into an auction. Using the embeddings, semantically similar queries are mapped into vectors close in the embedding space, which allows expansion of a query via K-nearest neighbor search.

Word embeddings for language processing Recently many NLP systems have obtained improved performance with less human engineering by adopting distributed word representations (Socher et al., 2012a).

In particular, neural word embedding techniques are now known to be effective in capturing syntactic and semantic relationships, and more computationally efficient than many other competitors (Mikolov et al., 2013a; Pennington et al., 2014; Baroni et al., 2014). On top of word embeddings, a new stream of research on sentence and paragraph embeddings has also emerged to tackle higher level tasks such as sentiment analysis, machine translation and semantic relatedness tasks. *Recursive Neural Networks* (RNNs) have been used for syntactic parsing (Socher et al., 2012b; Socher et al., 2013; Socher et al., 2014; Irsoy and Cardie, 2014). *Long Short-Term Memory* (LSTM) networks have been applied to machine translation (Bahdanau et al., 2014; Sutskever et al., 2014) and semantic processing (Tai et al., 2015). Interestingly *Convolutional Neural Networks* (CNNs), widely used for image processing, have recently emerged as a strong class of models for NLP tasks (Kim, 2014; Blunsom et al., 2014). As apposed to the models above, *PhraseVector* (Le and Mikolov, 2014) takes a less structured but unsupervised approach by treating a piece of text as a token and performing word embedding-like training with an unlimited context window. None of this previous work exactly fits the click prediction task. Since queries and ads are much less structured than usual text, it is not attractive to use models with complex structures, such as RNNs, at the cost of speed and scalability. *PhraseVector* is less structured but it does not support compositionality, suffering from sparseness or requiring to train new vectors for each unseen query and ad. Interestingly, as reported in (Tai et al., 2015), a simple averaging scheme (*mean vector*) was found to be very competitive to more complex models for high level semantic tasks despite its simplicity. These observations lead us to one of our models that aims to improve the *mean vector* method by directly optimizing mean vectors instead of word vectors.

Joint embedding to bridge multiple views. Multiple studies have explored the task of bringing multiple views into the same vector space. For example, there is now a large body of research on joint modeling of text and image information (Frome et al., 2013; Karpathy and Fei-Fei, 2014; Socher et al., 2014). The multimodal embedding space helps find appropriate alignments between image regions and corresponding pieces of text description. Joint embedding has also been applied to question answering (Wang et al., 2014) and semantic understanding (Yang et al., 2014). In contrast to the tasks above, there is no natural component-wise correspondence between queries and ads; instead the relationship is more implicit and pragmatic. Because of this, our methods rely on global rather than component-level signals for model training.

3 Baseline Click Prediction Model

We first present a high-level description of sponsored search. The process consists of several stages. First, given a user query, a list of candidate ads are retrieved, either by exactly matching query terms to the bid terms of the advertiser, or by first using query term expansion to obtain a longer list of matched ads. Some candidate ads may be filtered out based on metrics such as ad quality. Then, a *click prediction model* scores the candidate ads to estimate how likely it is that each will be clicked. This *click probability* serves a crucial role both in the user experience and in the revenue for the search engine. The ads with the highest click probabilities are placed in the search results page. The price-per-click for each ad shown is determined based on the click probabilities and the GSP auction.

Our baseline click prediction model is formulated as a supervised learning problem. Specifically we use *Logistic Regression* (LR) since LR is well suited for probability estimation. Given a variety of features, the probability of a click is expressed as:

$$p(c|q,a,u) = \frac{1}{1+exp\{\sum_i w_i f_i(q,a,u)\}}, \quad (1)$$

where $c \in \{1,0\}$ is the label (1: click or 0: non-click), $f_i(q,a,u)$ is the ith feature derived for query-ad-user triple (q,a,u) and w_i is the associated weight. The model is trained using a stochastic gradient descent algorithm on a per impression basis with l_1 regularization to avoid overfitting.

An accurate LR model relies greatly on the effectiveness of its features. Our baseline model is furnished with a rich set of features that are typically used in commercial search engines. The first feature type is based on the historical CTR of user, query, ad triples (if there is enough historical information on this). We use two groups of features of this type: COEC based features and user factor features. The second feature type is based on query and ad text.

Due to the significant decrease of CTR depending on the ad position, it has become common practice to use position-normalized CTR (a.k.a. *Clicks Over Expected Clicks*):

$$COEC = \frac{\sum_p c_p}{\sum_p i_p * CTR_p}, \quad (2)$$

where the numerator is the total number of clicks received by the configuration of interest; the denominator is the *expected clicks* (ECs) that an average ad would receive after i_p times impressions at position p, and CTR_p is the average CTR at position p, calculated over all queries, ads and users. We use user-independent features derived from COEC statistics for specific query-ad pairs. However, many impressions are needed for these statistics to be reliable and therefore data for specific query-ad pairs can be sparse and noisy (only around 70% of queries and about 50% of query-URL, query-bid term pairs have historical CTR). To alleviate this problem, additional COEC statistics over aggregations of queries or ads are also used. The exact description of these aggregations is beyond the

scope of this paper, but briefly we exploit the categorization of the ads in ad groups, campaigns, and accounts defined by advertisers.

Since it is well known that personalization features are crucial to obtain accurate click prediction models (Cheng and Cantú-Paz, 2010), we also use features that measure user factors relating to CTR. The user click feedback features capture the inclination of individual users to click on ads in general. The user-query click feedback features indicate the propensity of users to click for certain queries or groups of queries. Finally user-ad features dictate the user preferences on certain ads or advertisers. However the data sparseness problem becomes even more serious when it comes to user-specific features (we use a threshold of 100 for statistical confidence). For example, only about 5% of user-URL pairs and 1% of user, query, URL triples have historical CTR information. Therefore a set of segment-level features can be extracted as back-off features where users, queries and ads are clustered into groups, and group level historical CTRs are collected.

Our second major feature type involves the lexical similarity between query and ad text. These text features assume that users are more likely to click on ads that seem to be relevant to the query, and that perceived relevance is correlated with the degree of query-ad overlap. These features include the number of overlapping words and characters in query-ad URL, query-ad title, and query-ad description, and the number of words and characters in the query. The discrimination power of simple lexical features is relatively limited because query and ad text are typically very short.

Finally, we use other contextual features that are helpful in predicting the click probabilities.; for example, time of day, day of week, and geographic information. To model interactions among features, some features are selected by domain knowledge to be conjoined. All together, our baseline model utilize a comprehensive set of historical CTR, lexical, and contextual features (over a hundred features in total). The fact that this baseline model is highly optimized makes the subsequent performance improvement from our proposed algorithm meaningful. This baseline model is used in production in part of a major search platform.

4 Joint Embedding for Click Prediction

We now describe several methods that jointly embed words in both the query and the ads into the same vector space. In our experiments, we incorporate these methods as features in our click prediction model. We start by defining the notation used in this section. A sponsored search dataset $\mathcal{D}$ is a set of tuples for each ad impression (q, t, d, y) where $q \equiv \{q_j\}$ is a multiword query string, $t \equiv \{t_k\}$, $d \equiv \{d_l\}$ are multiword ad title and description strings, and y is a binary indicator for whether the ad is clicked. We have two choices in defining the vocabulary $\mathcal{V}$ from which words are drawn: we can use a unified vocabulary for both query and ads or define a separate vocabulary for each – $\mathcal{V} \equiv \mathcal{V}_q \oplus \mathcal{V}_a$. In our initial experiments the unified vocabulary constantly yielded better performances, thus we always use the unified vocabulary here. We use bold letters $\mathbf{q}_j, \mathbf{t}_k, \mathbf{d}_l$ to denote the corresponding embedding representations of $\{q_j, t_k, d_l\}$. Finally we use W to represent the vocabulary matrix; in W each column is a word vector.

4.1 Exploiting word2vec embedding

Typically word embeddings are learned from a given text corpus through implicit supervision of predicting the current word given a window of its surrounding text or predicting each word in the window of the current word. The former approach is known as *continuous bag-of-words* (CBOW) and the latter *Skip-gram*. For simplicity's sake we use negative-sampling for training word embedding models (Mikolov et al., 2013b). More formally we define the binary conditional probability for a pair of words (v, w): [1]

$$p(\mathbf{v}, \mathbf{w}) = \frac{1}{1 + \exp(-\mathbf{v}^T \mathbf{w})}, \tag{3}$$

The CBOW algorithm learns word embeddings by minimizing the following *logloss* of each impression i with regard to W:

$$\begin{aligned} CB_i(W) = &- \log p(\mathbf{w}_i, \boldsymbol{\mu}_C) \\ &- \sum_{v \in N(w_i)} \log\left(1 - p(\mathbf{v}, \boldsymbol{\mu}_C)\right), \end{aligned} \tag{4}$$

where the context C of a word w_i comes from a window of size k around the word in a sentence of n words $w_1,\ldots,w_n$: $C = w_{i-k},\ldots,w_{i-1},w_{i+1},\ldots,w_{i+k}$. $\boldsymbol{\mu}_C$ is the averaged context vector of w_i; $\boldsymbol{\mu}_C = \frac{1}{|C|}\sum_{v \in C} \mathbf{v}$. $N(w_i)$ is the set of negative examples which is drawn according to the unigram distribution of the corpus raised to the 3/4th power. Similarly to (Mikolov et al., 2013b), we adopt a dynamic window size – for each word the actual window size is sampled uniformly from $1, \ldots, k$.

Similarly the Skip-gram algorithm minimizes the *logloss* of each impression i with regard to W:

$$\begin{aligned} SK_i(W) = &- \sum_{v \in C} \log p(\mathbf{w}_i, \mathbf{v}) \\ &- \sum_{v \in N(w_i)} \log\left(1 - p(\mathbf{w}_i, \mathbf{v})\right). \end{aligned} \tag{5}$$

In our first word embedding model that incorporates click feedback, we construct a corpus by taking only clicked impressions from $\mathcal{D}$ and then mixing (q, t, d) of each impression into a sentence. Then we simply train CBOW and Skip-gram models on the corpus.

[1] We use only a single vector for a word unlike (Mikolov et al., 2013b) where two vectors ("input" and "output") for a word are used. This halves the required space to store vectors without performance loss.

4.2 Joint word embedding using click feedback

Although the CBOW and Skip-gram models trained on a specially constructed corpus can capture signals from both click feedback and word co-occurrence, they have a couple of drawbacks. First, by ingesting only clicked impressions we "waste" the large amount of negative signals contained in the non-clicked impressions. Second, the incorporation of indirect signals such as word co-occurrences can be rather harmful for achieving accurate click prediction as these are very noisy compared to direct signals such as click feedback.

For our second word embedding model that incorporates click feedback, we define a joint word embedding model that minimizes the following *weighted logloss*:

$$JW_i(W) = \eta(y_i)\Big\{l(y_i, q_i, t_i) + l(y_i, q_i, d_i)\Big\}, \quad (6)$$

where the component loss function $l(\cdot,\cdot,\cdot)$ is defined as follows:

$$l(y,a,b) = \sum_{k}^{|a|}\sum_{l}^{|b|} - y \log p(\mathbf{a}_k, \mathbf{b}_l) - (1-y)\log\left(1 - p(\mathbf{a}_k, \mathbf{b}_l)\right). \quad (7)$$

Here $\eta(y_i)$ is a function that returns a small weight η only to negative examples:

$$\eta(y_i) = \begin{cases} \eta, & \text{if } y_i = 0, \\ 1, & \text{otherwise.} \end{cases} \quad (8)$$

The fact that the user did not click on an ad does not necessarily mean that the ad is not what the user wanted; it often means that the ad is just less favored than the clicked ads (Rendle et al., 2009). Since the scope of this work is restricted to estimating click probability on a per-impression basis, we adopt a weighting scheme rather than optimizing rank-based loss over a set of related impressions.

4.3 Joint mean vector optimization

The joint word embedding models defined in the previous sections do not define how to aggregate a variable length sequence of word vectors into a sentence (or paragraph) vector to facilitate the computation of sentence-level similarity scores. One approach to this aggregation task is *mean vector*: simply average the word-level embeddings across the sentence or paragraph. As noted by (Tai et al., 2015), this approach is a strong competitor to more complex models such as RNNs or LSTMs despite its simple composition method. However, this method may generate suboptimal sentence vectors. With *weight logloss*, we aim to optimize sentence vectors instead of individual word vectors:

$$JM_i(W) = \eta(y_i)\Big\{\bar{l}(y_i, q_i, t_i) + \bar{l}(y_i, q_i, d_i)\Big\}, \quad (9)$$

where the component loss function $\bar{l}(\cdot,\cdot,\cdot)$ is defined as follows:

$$\bar{l}(y,a,b) = - y \log p(\boldsymbol{\mu}_a, \boldsymbol{\mu}_b) - (1-y)\log\left(1 - p(\boldsymbol{\mu}_a, \boldsymbol{\mu}_b)\right), \quad (10)$$

where $\boldsymbol{\mu}_s$ returns the average vector for the multiword string s, i.e. $\boldsymbol{\mu}_s = \frac{1}{|s|}\sum_k^{|s|} \mathbf{s}_k$.

5 Experiments

Data The data used in our experiments were collected from a random bucket of the Yahoo! sponsored search traffic logs for a period of 4 weeks in October 2014. In the data there are approximately 65 million unique users, 150 million unique queries and 12 million unique ads. There are approximately 985 million ad impression events in total. We split the data into 3 partitions with respect to time: the first 14 days' data are used for training word embedding models, the next 7 days' data for training click prediction models, and the last 7 days' data for testing. Table 1 presents more detailed statistics for the data.

Models We are interested in evaluating the usefulness of different word embedding models as features for click prediction, and already have a very good baseline system for this task. Consequently, in all experiments below, we used the same personalized historical CTRs, contextual and lexical features as the baseline system described in Section 3. We tested models with the following additional features derived from word embeddings:

1. cosine similarity between the mean vectors of the query and ad title
2. cosine similarity between the mean vectors of the query and ad description
3. sum of 1 and 2
4. sigmoid function value for the dot product of the mean vectors of the query and ad title
5. sigmoid function value for the dot product of the query and ad description
6. sum of 4 and 5

All these continuous features are quantized into 50 bins. We compared five different word embedding algorithms:

1. Skip-gram trained on Wikipedia (SK-WIKI)
2. Skip-gram trained on clicked impressions (SK-CI), see Section 4.1
3. CBOW trained on clicked impressions (CB-CI), see Section 4.1
4. Joint individual word vector embedding (JIWV), see Section 4.2

	Embedding	Train	Test	Total
Number of impressions	489M	252M	244M	985M
Number of unique queries	82M	46M	45M	150M
Number of unique ads	8M	6M	6M	12M
Number of unique users	40M	25M	25M	65M
Number of unique words in query	6757K	6039K	5774K	16028K
Number of unique words in ad title	3371K	2586K	2557K	4598K
Number of unique words in ad description	1993K	1583K	1589K	2628K
Number of words in query	1600M	830M	800M	3230M
Number of words in ad title	3260M	1690M	1635M	6585M
Number of words in ad description	4311M	2206M	2131M	8650M
Average number of words in query per impression	3.275	3.284	3.289	3.281
Average number of words in ad title per impression	6.667	6.708	6.706	6.687
Average number of words in ad description per impression	8.818	8.759	8.741	8.784

Table 1: Data description

5. Joint mean vector embedding (JMV), see Section 4.3

We used the skip-gram mode of word2vec [2] with a window size of 5 and negative sampling to train the SK-WIKI model. For all other models we used in-house implementation which employs *AdagradRDA* (Duchi et al., 2011) to minimize the loss functions introduced in Section 4. The dimension of a word vector is set to 100 for all algorithms[3]. We removed a set of prefixed stop-words and all words occurring fewer than 100 times; the resulting vocabulary comprised 126K unique words. To process our web-scale data, we implemented a multithread program where each thread randomly traverses over a partition of the data $\mathcal{D}$ to compute gradients and update the matrix W stored in a shared memory. For computational efficiency, the hogwild lock-free approach (Recht et al., 2011) is used. We set η in Eq. 8 and Eq. 9 to 0.2 through grid search based on two-fold cross-validation. This small value indeed verifies the idea of weak negative feedback for unclicked impressions.

In order to circumvent the severe influence of ad position on the click prediction model, only the impressions that are placed at the top position were used for training the click prediction model. Note that CTR at other positions can be derived from that of the ad at the top position through scaling. Dembczynski et al. (2008) showed that CTR can be decomposed as a product of the probability of an ad getting clicked given its being seen and the probability of an ad being seen at a particular position.

Results We ordered query-ad pairs by the predicted score to compute *AucLoss* (i.e. 1 - AUC where AUC is the area under the *Receiver Operating Characteristic* (ROC) curve). The ROC AUC is known to have a correlation with the quality of ranking by the predicted score (Fawcett, 2006); thus is one of the most important metrics for click prediction (McMahan et al., 2013).

	AucLoss Reduction (%)
Baseline + SK-WIKI	0.530
Baseline + SK-CI	0.595
Baseline + CB-CI	0.656
Baseline + JIWV	2.276
Baseline + JMV	**4.114**

Table 2: Comparative evaluation results in AucLoss reduction from the baseline system

Our experimental results (Table 2) show that the use of features derived from our proposed word embedding models significantly reduce AucLoss by up to 4.1%. For commercial search engines which have a very strong baseline *AucLoss*, a reduction of 1% can be considered large (McMahan et al., 2013). Moreover, as expected, the more issues (as identified in Section 4) an algorithm addresses, the better performance it achieves. From the comparison between the SK-WIKI model and the rest, we can recognize the importance of word embedding models specialized to domain text and supervision signals. Also the difference between the CB-CI (SK-CI) model and the JIWV model indicates the noisiness of indirect signals such as word co-occurrence compared to direct signals like click feedback. Finally the gap between the JIWV and JWV models highlights the significance of considering compositionality in the word embedding training process.

Eyeballing the most similar words to several queries in the vector space is often helpful for getting some sense about how different methods influence the resulting vector spaces. Table 3 lists the top 20 most similar words to the query "metal watch." The top words for the SK-WIKI model are not semantically interesting in terms of capturing the intent or desires. This clearly shows the limitation of word embedding methods that only rely on indirect signals (i.e. word co-occurrences) from a generic text corpus (e.g. Wikipedia) for sponsored search click prediction. Noticeably the methods

[2] Available at https://code.google.com/p/word2vec/

[3] Higher vector dimensions such as 200 were also tried but did not give a significant improvement.

SK-WIKI	CB-CI	JIWV	JMV
watch (0.733)	metal (0.718)	wwhl (0.711)	tacticalwatch.com (0.701)
grind (0.687)	previews (0.652)	cbs (0.704)	watchrepairsusa.com (0.695)
grease (0.682)	yidio.com (0.648)	station (0.668)	omegas (0.689)
kites (0.676)	episodes (0.633)	putlocker.com (0.662)	watchco.com (0.682)
hammer (0.675)	steel (0.626)	iwatch (0.659)	wach (0.670)
spinning (0.672)	whatch (0.615)	kidizoom (0.658)	station (0.656)
flashing (0.671)	bobcometal.com (0.586)	freesports360.com (0.658)	shockwarehouse.com (0.628)
trash (0.670)	ridiculousness (0.582)	tedtalks (0.655)	freesports360.com (0.618)
flame (0.669)	www.abc.com (0.574)	11/10c (0.650)	akribos (0.616)
flaming (0.665)	a&e (0.573)	movie2k (0.640)	18mm (0.616)
cigar (0.664)	utube (0.570)	nfl.com/now (0.637)	narutoget.com (0.614)
home-made (0.663)	instantly (0.565)	criminalsgonewilddvd (0.631)	watchstation.com (0.611)
glow (0.662)	khoobsurat (0.562)	espnnfllive.com (0.631)	authenticwatches.com (0.610)
bouncing (0.662)	outnumbered (0.561)	foxsports1 (0.623)	interrupted (0.603)
filler (0.662)	itv (0.559)	viooz (0.623)	criminalsgonewilddvd (0.601)
smoke (0.660)	premiere (0.556)	bubble (0.623)	whatch (0.600)
shoot (0.658)	films (0.555)	wewood (0.621)	$109.99 (0.597)
scoop (0.653)	stainless (0.554)	westclox (0.617)	tirebuyer.com (0.596)
noises (0.652)	fabricators (0.550)	potlocker (0.613)	skagen.com (0.588)
rocking (0.651)	fabrication (0.550)	nickelodeon (0.613)	wewood (0.584)

Table 3: Top 20 most similar words to "metal watch"

SK-WIKI	CB-CI	JIWV	JMV
costumes (0.762)	princess (0.833)	costumes (0.831)	new-costumes.com (0.815)
bride (0.723)	costumed (0.814)	wonder (0.784)	coustume (0.808)
princesses (0.722)	customes (0.808)	sweetiegames.com (0.782)	namefully.com (0.800)
serene (0.708)	costume (0.806)	cleopatra (0.753)	$35.90 (0.789)
highness (0.676)	costomes (0.806)	new-costumes.com (0.752)	2-days (0.781)
bess (0.674)	costimes (0.803)	girls.simple (0.749)	$36.90 (0.776)
princess]] (0.671)	m.buycostumes.com (0.800)	werewolf (0.749)	costume (0.766)
attire (0.670)	custumes (0.796)	yoshi (0.747)	$28.90 (0.764)
princess (0.662)	officialprincesscostumes.com (0.792)	leia (0.747)	princess (0.758)
dresses (0.662)	custume (0.789)	merida (0.742)	cistumes (0.756)
highness (0.658)	coustome (0.789)	$49.90 (0.735)	spider-woman (0.756)
jewels (0.652)	cosyumes (0.787)	low-budget (0.727)	the-wristband-factory.com (0.750)
wedding (0.651)	coustume (0.786)	babies (0.727)	$17.90 (0.750)
robes (0.648)	coustums (0.786)	fembot (0.726)	sugarsmascotcostumesċom (0.748)
prince (0.644)	buycostumes.com (0.785)	costums (0.722)	cotumes (0.748)
clothes (0.644)	codtume (0.784)	$3.90 (0.721)	coneheads (0.747)
dancing (0.644)	leia (0.784)	hermione (0.718)	$23.90 (0.739)
sophie (0.640)	coustumes (0.784)	supergirl (0.715)	$7.90 (0.739)
consorts (0.639)	costums (0.783)	toothless (0.713)	costomes (0.738)
glamorous (0.639)	coatumes (0.782)	starlord (0.709)	$19.90 (0.737)

Table 4: Top 20 most similar words to "princess costumes"

that incorporate click feedback find more words related to products, services or websites instead of just conceptuatlly related words. Given that real products or services can be regarded as the best possible surrogates to user intents and desires, this demonstrates the effectiveness of our methods. This tendency gets stronger as a method takes into account both positive and negative click feedback.

Another very interesting observation comes from the fact that none of the methods except JMV successfully captures the composite meaning of "metal watch"; they tend to either find related words separately for each query word (e.g. "watch" is strongly associated to the sense of watching something like movie or other types of video) or find totally unrelated words (particularly SK-WIKI). This demonstrates that it is crucial to address compositionality in the very process of learning word vectors.

Table 4 shows the top 20 most similar words to the query "princess costumes." In this example we can spot another surprising result. The JMV model pushes a lot of price related expressions to the top[4]. This may imply that many parents search for lower cost costumes, clearly showing a clear psychological desire in the financial dimension. This observation confirms the findings in (Wang et al., 2013) about the significant role of certain ad expressions in triggering users' psychological desires. We also note that the CB-CI model returns a lot of misspells for "costume(s)", which would not be possible with simple lexical features of the baseline system. A close look at this example generally confirms the observations we made for the previous ex-

[4]We have not tried any normalization for numbers but it might be worth doing given the important role they play.

SK-WIKI	CB-CI	JIWV	JMV
kids (0.724)	game (0.629)	kids (0.795)	program (0.764)
pac-man (0.708)	gams (0.615)	kidsfootlocker.com (0.788)	scorekeepers (0.757)
games (0.703)	games (0.613)	flight: (0.754)	gamingchairs.hayneedle.com (0.754)
pinball (0.687)	petrasplanet.com (0.601)	edit: (0.744)	teepees (0.752)
boy (0.680)	for (0.590)	raz (0.737)	nn2 (0.750)
adventure (0.664)	gamse (0.584)	program (0.733)	game (0.749)
roleplaying (0.664)	ganes (0.574)	abercrombiekids.com (0.729)	girls (0.746)
quests (0.658)	collectors (0.574)	sumdog (0.721)	cranium (0.746)
d&d (0.658)	awsome (0.551)	take20 (0.702)	ggg (0.738)
genie (0.655)	kid (0.550)	freegamesdownload.com (0.699)	pac-man (0.732)
solitaire (0.654)	game.com (0.537)	program; (0.695)	kidsfootlocker.com (0.730)
gamers (0.653)	g2u.com (0.536)	gromsocial.com (0.694)	equestriagirls.hasbro.com (0.721)
games=== (0.651)	bouncing (0.533)	softschools (0.693)	kids (0.718)
game; (0.649)	for.kids (0.532)	gapkids (0.691)	mahjong (0.718)
consoles]] (0.645)	catching (0.528)	downnload (0.682)	tvo (0.717)
in-game (0.645)	gamesfreak (0.519)	edheads (0.681)	sumdog (0.701)
rpg (0.644)	minecraft (0.516)	ruum.com (0.674)	cpm.wargaming.net (0.699)
wargames (0.644)	firetruck (0.514)	gamestop (0.672)	osgood-schlatter (0.698)
game]] (0.643)	games: (0.513)	a&f (0.668)	gamestop (0.697)
fast-paced (0.641)	todlers (0.512)	pac-man (0.663)	$4.01 (0.695)

Table 5: Top 20 most similar words to "game for kids"

ample. Finally Table 5 shows top the 20 most similar words for the query "game for kids." Once again we found the same analysis holds for this case.

6 Conclusions

In this paper we explored the use of word embedding techniques to overcome the shortcomings of traditional lexical features for ad click prediction in sponsored search. We identified several potential weaknesses of the plain application of conventional word embedding methodologies: the lack of the right machinery to harness both positive and negative click feedback, the limited utility of pure word co-occurrence signals, and no consideration of vector composition in the word embedding training process. We proposed a set of new implicit feedback-based joint word embedding methods to address those issues. We evaluated the new word embedding methods in the context of a very good baseline click prediction system, on a large scale data set collected from Yahoo! search engine logs. Our experimental results clearly demonstrate the effectiveness of the proposed methods. We also presented several examples for qualitative analysis to advance our understanding on how each algorithm really contributes to the improved performance. To the best of our knowledge this work is the first successful application of word embedding techniques for the sponsored search task.

There are multiple interesting research directions for future work. One of these directions is to extend the vocabulary by identifying significant phrases (as well as words) before training word vectors. Hillard et al. (2011) employed *Conditional Random Fields* to divide queries with multiple words into segments and collected historical CTR on the segment level. We also like to investigate more structured embedding methods such as RNNs (probably for ad descriptions). In case the computational cost of such methods are too high to be practical for sponsored search, we can employ them only for a small fraction of ads filtered by faster methods.

It may be possible to deal with the implicit negative feedback of unclicked ad impressions in a more principled way by adopting ranking-based loss functions. However, this is only possible with the extra cost of identifying and aggregating related ads into a single transaction.

Though not directly related to NLP, yet another promising direction is to jointly embed not only text data but also a variety of user activities (e.g., organic search results, mobile app usages, other daily activities) all together in the same vector space. Since many of the different sources contain their own unique information, we might be able to obtain a much better understanding about the user state and intent through this rich joint embedding space. Joint embedding with rich information can also help us to perform automatic clustering of users, eventually leading to powerful smoothing methods for personalized historical CTR statistics.

References

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

Marco Baroni, Georgiana Dinu, and Germán Kruszewski. 2014. Dont count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 238–247.

Phil Blunsom, Edward Grefenstette, Nal Kalchbrenner,

et al. 2014. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*. Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics.

Haibin Cheng and Erick Cantú-Paz. 2010. Personalized click prediction in sponsored search. In *Proceedings of the Third ACM International Conference on Web Search and Data Mining*, WSDM '10, pages 351–360, New York, NY, USA. ACM.

K. Dembczynski, W.Kotlowski, and D.Weiss. 2008. Predicting ads click-through rate with decision rules. In *Proceedings of WWW 08*.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *The Journal of Machine Learning Research*, 12:2121–2159.

David Easley and Jon Kleinberg. 2010. *Networks, Crowds, and Markets: Reasoning about a Highly Connected World*. Cambridge University Press.

Benjamin Edelman, Michael Ostrovsky, Michael Schwarz, Thank Drew Fudenberg, Louis Kaplow, Robin Lee, Paul Milgrom, Muriel Niederle, and Ariel Pakes. 2005. Internet advertising and the generalized second price auction: Selling billions of dollars worth of keywords. *American Economic Review*, 97.

Tom Fawcett. 2006. An introduction to ROC analysis. *Pattern Recognition Letters*, 27(8):861–874, June.

Andrea Frome, Greg S Corrado, Jon Shlens, Samy Bengio, Jeff Dean, Tomas Mikolov, et al. 2013. Devise: A deep visual-semantic embedding model. In *Advances in Neural Information Processing Systems*, pages 2121–2129.

Mihajlo Grbovic, Nemanja Djuric, Vladan Radosavljevic, Fabrizio Silvestri, and Narayan Bhamidipati. 2015. Context- and content-aware embeddings for query rewriting in sponsored search. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '15, pages 383–392, New York, NY, USA. ACM.

Dustin Hillard, Eren Manavoglu, Hema Raghavan, Chris Leggetter, Erick Cantú-Paz, and Rukmini Iyer. 2011. The sum of its parts: reducing sparsity in click estimation with query segments. *Inf. Retr.*, 14(3):315–336.

Ozan Irsoy and Claire Cardie. 2014. Deep recursive neural networks for compositionality in language. In *Advances in Neural Information Processing Systems*, pages 2096–2104.

Andrej Karpathy and Li Fei-Fei. 2014. Deep visual-semantic alignments for generating image descriptions. *arXiv preprint arXiv:1412.2306*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Quoc V Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. *arXiv preprint arXiv:1405.4053*.

H. Brendan McMahan, Daniel Golovin, Sharat Chikkerur, Dan Liu, Martin Wattenberg, Arnar Mar Hrafnkelsson, Tom Boulos, Jeremy Kubica, Gary Holt, D. Sculley, Michael Young, Dietmar Ebner, Julian Grady, Lan Nie, Todd Phillips, and Eugene Davydov. 2013. Ad Click Prediction: a View from the Trenches. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '13*, page 1222, New York, New York, USA, August. ACM Press.

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

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

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. *Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014)*, 12.

Benjamin Recht, Christopher Re, Stephen Wright, and Feng Niu. 2011. Hogwild: A Lock-Free Approach to Parallelizing Stochastic Gradient Descent. In *Advances in Neural Information Processing Systems*, pages 693–701.

Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. Bpr: Bayesian personalized ranking from implicit feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, UAI '09, pages 452–461, Arlington, Virginia, United States. AUAI Press.

Matthew Richardson. 2007. Predicting clicks: Estimating the click-through rate for new ads. In *In Proceedings of the 16th International World Wide Web Conference (WWW-07*, pages 521–530. ACM Press.

Richard Socher, Yoshua Bengio, and Christopher D. Manning. 2012a. Deep learning for nlp (without magic). In *Tutorial Abstracts of ACL 2012*, ACL '12, pages 5–5, Stroudsburg, PA, USA. Association for Computational Linguistics.

Richard Socher, Brody Huval, Christopher D Manning, and Andrew Y Ng. 2012b. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical*

Methods in Natural Language Processing and Computational Natural Language Learning, pages 1201–1211. Association for Computational Linguistics.

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, volume 1631, page 1642.

Richard Socher, Andrej Karpathy, Quoc V Le, Christopher D Manning, and Andrew Y Ng. 2014. Grounded compositional semantics for finding and describing images with sentences. *Transactions of the Association for Computational Linguistics*, 2:207–218.

Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*.

Ilya Trofimov, Anna Kornetova, and Valery Topinskiy. 2012. Using boosted trees for click-through rate prediction for sponsored search. In *Proceedings of the Sixth International Workshop on Data Mining for Online Advertising and Internet Economy*, ADKDD '12, pages 2:1–2:6, New York, NY, USA. ACM.

Taifeng Wang, Jiang Bian, Shusen Liu, Yuyu Zhang, and Tie-Yan Liu. 2013. Psychological advertising: exploring user psychology for click prediction in sponsored search. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 563–571. ACM.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph and text jointly embedding. In *Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014)*.

Min-Chul Yang, Nan Duan, Ming Zhou, and Hae-Chang Rim. 2014. Joint relational embeddings for knowledge-based question answering. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 645–650.

Automatic Extraction of Time Expressions Across Domains in French Narratives

Mike Donald Tapi Nzali
Université de Montpellier
LIRMM, I3M
Montpellier, France

Xavier Tannier
LIMSI-CNRS UPR 3251
Univ. Paris-Sud
91403 Orsay, France
xtannier@limsi.fr

Aurélie Névéol
LIMSI-CNRS UPR 3251
Rue John von Neuman
91403 Orsay, France
neveol@limsi.fr

Abstract

The prevalence of temporal references across all types of natural language utterances makes temporal analysis a key issue in Natural Language Processing. This work adresses three research questions: 1/is temporal expression recognition specific to a particular domain? 2/if so, can we characterize domain specificity? and 3/how can subdomain specificity be integrated in a single tool for unified temporal expression extraction? Herein, we assess temporal expression recognition from documents written in French covering three domains. We present a new corpus of clinical narratives annotated for temporal expressions, and also use existing corpora in the newswire and historical domains. We show that temporal expressions can be extracted with high performance across domains (best F-measure 0.96 obtained with a CRF model on clinical narratives). We argue that domain adaptation for the extraction of temporal expressions can be done with limited efforts and should cover pre-processing as well as temporal specific tasks.

1 Introduction

References to phenomena occurring in the world and their temporal characterization can be found in natural language utterances across domains, genres and languages. Temporal analysis is a key issue in natural language processing that has been receiving increasing attention in recent years. Many efforts in this direction focused on newswire text in English. The focus on this language and domain was in part guided by the availability of the TimeBank corpus (Pustejovsky et al., 2003) used in evaluation campaigns such as TempEval (Verhagen et al., 2007). More recent efforts have extended the initial work on English and addressed other languages such as Chinese (Li et al., 2014), French (Moriceau and Tannier, 2014), Arabic, Italian, Spanish, and Vietnamese (Strötgen et al., 2014a). A study of three domain corpora in English in addition to the newswire domain (SMS, historical narratives and clinical trial abstracts) yielded interesting insight to extend the normalized representation of temporal expressions (Strötgen and Gertz, 2012). This work was then applied to cover historical narratives in an additional seven languages. One key finding was that domain specificity could differ between languages (Strötgen et al., 2014b). This prompts the need to study temporal analysis across domains in a variety of languages in order to adequately characterize each domain and language pairs.

The clinical domain has been addressed during the 2012 i2b2 challenge (Sun et al., 2013b), with a task on temporal relation extraction from clinical narratives. This task used a corpus of clinical notes in English annotated with temporal information (Sun et al., 2013a) based on ISO-TimeML (Pustejovsky et al., 2010). It prompted further work in this domain in English (Jindal and Roth, 2013) and Swedish (Velupillai, 2014), including the release of detailed guidelines for creating temporal annotations of clinical text and a discussion of the clinical domain specificity related to temporal aspects (Styler IV et al., 2014). Finally, clinical TempEval 2015 brought the temporal information extraction tasks of past TempEval campaigns to the clinical domain (Bethard et al., 2015).

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 492–498,
Lisbon, Portugal, 17-21 September 2015.

In this paper, we continue to explore temporal expression identification across domains, with a focus on French narratives. We introduce a new corpus of French clinical narratives annotated with normalized time expressions. We characterize temporal expression recognition in three domains and discuss how the development of an automated temporal expression identification tool may be impacted.

2 Temporal Expression Extraction and Normalization

Rule-based methods were shown to be very efficient for the extraction and normalization of time expressions from news narratives in several languages[1]. In the latest SemEval campaign (UzZaman et al., 2013), the rule-based HeidelTime (Strötgen and Gertz, 2010) out performed machine-learning and hybrid counterparts by a large margin. However, statistical systems obtained promising results with respect to temporal entity extraction.

Based on these results, we chose to use the state-of-the-art rule-based system HeidelTime as well as an in-house statistical tool relying on the Wapiti (Lavergne et al., 2010) implementation of Conditional Random Fields (CRFs) (Lafferty et al., 2001).

Existing HeidelTime settings were used to customize it for the analysis of news and historical narratives in French. In addition, we developed a set of 14 rules to provide additional customization for the analysis of clinical narratives in French.

The CRF model was developed using part of the clinical corpus as a training set, with domain independant surface and lexical features for the text tokens:

- The original token from the text (word form);
- Surface features: capitalization of the token (all in upper/lower case, combination of both), presence of digit (YES, NO) and punctuation mark in the token (PUNCT, NO_PUNCT), temporal type of token according to HeidelTime;
- Lexical features: *n*-grams , number of words, number of digits, number of consecutive repeats. Token frequency was computed based on the entire training corpus.

For clinical text analysis, we experimented with standard tokenization (provided by TreeTagger (Schmid, 1994)) and a custom tokenization where punctuation marks are always considered as token separators, even in dates such as "10-02-2010" or "10.04.10".[2]

3 French Corpora with Temporal Annotations

For this study, we used two available French corpora with TIMEX3 annotations: the French TimeBank corpus (FTB called *news* in this paper) (Bittar et al., 2011) which covers the news domain and the AncientTimes corpus (ATC, called *historical* in this paper) (Strötgen et al., 2014b) which covers the historical domain.

To cover a third domain, we developed a corpus using a set of clinical notes where personal identifying information (PII) had been marked and replaced by surrogates (Grouin and Névéol, 2014). This included marking some temporal expressions such as dates, which were replaced by surrogate dates obtained by substracting a fixed number of days to the original dates. Manual review ensured that there were no format or other errors in the reintroduced surrogate dates. For compatibility with the sources that were already available, including our study corpora and HeidelTime, we chose the TIMEX3 standard for creating temporal annotations.

Three annotators (the authors of this paper) contributed towards the creation of gold-standard annotations for time expressions in the corpus. The annotation of time expressions was carried out in two phases: first, the time expressions and their values were annotated, and second the time expressions were normalized. At the beginning of the first phase, two initial samples of twenty documents were selected for all three annotators to work on. These documents were pre-annotated using the French version of HeidelTime and dates marked as PII. The annotators' task was then to revise the pre-annotations independently. This phase of the annotation process contributed to refining annotation guidelines and creating additional rules to improve on the pre-annotation. Subsequently, the rest of the corpus was divided between annotators, so that each document was annotated independently by two annotators. Final

[1] Normalization is the process of turning any reference to a date into an absolute, formated date.

[2] This tokenization script, adapted from TreeTagger, is available upon request.

gold standard annotations were created by adjudicating any disagreements during meetings between the pair of contributing annotators. Two annotators contributed to the second phase of annotations (normalization). A small sample of 20 documents was annotated independently by the two annotators. Inter-annotator agreement was then computed and found to be sufficiently high to allow for the rest of the corpus to be distributed between annotators.

The phasing of annotations allowed having all corpus documents reviewed several times, so that time expressions that might have been missed during the first phase could be identified and added to the gold standard in the second phase.

To visualize and create annotations, we used the BRAT Rapid Annotation Tool (Stenetorp et al., 2012). Inter-annotator agreement was computed in terms of F-measure, using the companion brateval tool (Verspoor et al., 2013), which we extended to compute inter-annotator agreement on normalized entities.

Table 1 shows the distribution of time expressions according to types in the three corpora used in our study. It suggests that domain specificity is reflected by the types of temporal expressions found in each of the corpora. While *Dates* are prevalent across domains, the news corpus stands out with a high proportion of *Times*, the clinical corpus with a high proportion of *Set* and the historical corpus with almost none of either type. Additional statistics on the clinical corpus are provided in section 4.

	FTB (news)		**ATC (hist.)**		**Clinical**	
	#	%	#	%	#	%
Date	227	53.41	124	81.05	2594	65.14
Dur.	52	12.24	25	16.34	343	8.61
Set	16	3.76	3	0.02	994	24.96
Time	130	30.59	1	0.01	51	1.28

Table 1: Distribution of Time Expressions in three French corpora

4 Results

4.1 French Clinical Corpus with Temporal Annotations

Figure 1 shows an excerpt of the training corpus annotated with temporal expressions. The blue boxes show the normalized value associated with each temporal expression. Due to the confidential nature of the corpus, we are currenlty not able to release it.

Table 2 presents detailed statistics on the clinical corpus. Inter-annotator agreement was .91 F-measure for temporal entity annotation (averaged over the three annotator pairs on the training corpus) and .99 F-measure for temporal normalization (computed on a sample of 20 documents from the training corpus).

	Training	**Test**	**All**
Documents	246	115	361
Tokens	97,008	44,803	141,811
DATE	1,659	935	2,594
DURATION	255	88	343
TEMPORAL SET (Frequency)	605	389	994
TIME	19	32	51

Table 2: Description of the gold standard clinical corpus

4.2 Extraction of Temporal Expressions across Domains

Table 3 presents the results of temporal expression extraction in French narratives across the three domains in our study. The model configurations used are either HeidelTime (H) or statistical (S), adapted to one of the three domains. For HeidelTime models, the adaptation consisted in selecting a domain specific set of rules. We also report results by Strötgen et al. (2014b) showing the difference between HeidelTime 1.5 and the improvements obtained by their new rules for historical French texts. For statistical models, the adaptation consisted in training the model on a corpus of the relevant domain. We studied the effect of corpus size by training a model using a portion of the clinical training data equivalent in size to that of FTB (marked *clin-* in Table 3). However, the ATC corpus was too small to train any usable models (results not shown). Experiments with our adapted, in-house tokenization tool are marked with a + in the models.

The results of the evaluation are reported in terms of precision, recall and F1-measure. We evaluate the extraction of temporal expressions associated to the correct TIMEX3 attribute type (DATE, DURATION, TIME, SET), with the 'strict' measure (only exact match is correct) and the 'relaxed' measures (overlaps are allowed).

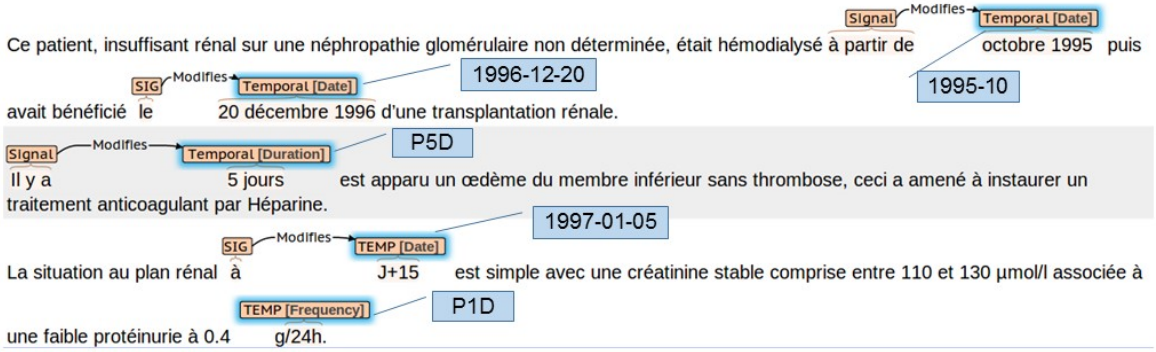


Figure 1: Excerpt from a sample document annotated with temporal expressions; dates and personal health information were replaced by plausible surrogates.

5 Discussion

Overall, our results show that while good performance can be achieved for the extraction of temporal expression on many specific domains, the task of automatically extracting temporal expressions is not solved across the board. Methods that are successfully developed for one specific domain do not carry very well over to other domains without any adaptation work. In our experiments, rule-based methods seem to fare somewhat better in terms of generalizability, but statistical methods can be better optimized for a particular domain, given enough training data. Similar insight resulted from a recent study of negation detection in the clinical domain (Wu et al., 2014). One additional issue highlighted in the negation study was that the definition of the entities that could fall under the scope of negation varied from domain to domain in the available negation corpora. This pitfall is avoided with the temporal task thanks to the use of the TimeML standard. The challenges of extracting temporal expressions across domains that we identified in this study on French correlate well with those described by (Strötgen and Gertz, 2012) on English. The performance of statistical models trained on in-domain data was significantly higher compared to out-domain data: *S-news+* yielded the highest performance on the News corpus (strict F-measure 0.74), and *S-clin+* yielded the highest performance on the Clinical corpus (strict F-measure 0.94).

Adaptation burden. The amount of effort to adapt to new domains was overall limited for the rule-based system: only a few rules needed to be added to the news-oriented HeidelTime (2 for historical, 14 for clinical) to reach comparable performance on other domains. The adaptation effort for the statistical model relies mostly on the availability of annotated corpora in the relevant domains. A competitive CRF model can be trained without using domain specific features. However, we find that adaptation effort can cover pre-processing: for clinical documents, using domain-specific tokenization yielded improved temporal expression extraction for both rule-based and statistical systems. This is in line with previous findings that pre-processing is essential for making experiments reproducible, and variations in pre-processing methods can result in significant difference in performance for different NLP tasks (Fokkens et al., 2013).

Finally, we can note that the difference between HeidelTime and CRFs is much bigger on the strict measure than on the relaxed measure, which may suggest that small variations could be better handled with more covering rules.

Impact of pre-processing, training corpus type and size. Tokenization had a real impact on the performance of statistical models. Using a baseline tokenization method can reduce the performance by several points in F-measure (strict F-measure of 0.68 for *S-news* vs. 0.74 for *S-news+* on the News corpus, strict F-measure of 0.88 for *S-clin* vs. 0.94 for *S-clin+* on the Clinical corpus – see Table 3). Corpus size had a comparably smaller impact. The training dataset used for *S-clin* is 10 times larger than that used for *S-clin-*, and yet it provides an improvement of about 4 points in F-measure for in-domain application. Conversely, for out-domain application, using the larger model is detrimental. Using in-domain training data (or more generally, training

Test	Model	relaxed			strict		
		P	R	F	P	R	F
Clinical	H-news	0.83	0.69	0.75	0.63	0.53	0.57
	H-clin	0.92	0.89	0.90	0.81	0.78	0.79
	S-news	0.86	0.72	0.79	0.60	0.49	0.54
	S-news+	0.86	0.70	0.77	0.64	0.51	0.57
	S-clin-	0.98	0.86	0.92	0.89	0.78	0.83
	S-clin-+	0.98	0.87	0.92	0.95	0.84	0.89
	S-clin	0.98	0.93	0.96	0.91	0.85	0.88
	S-clin+	**0.99**	**0.94**	**0.96**	**0.97**	**0.91**	**0.94**
Historical	H-news1.5*	0.97	0.43	0.59	0.71	0.31	0.43
	H-news*	0.98	**0.84**	**0.91**	**0.89**	**0.76**	**0.82**
	H-clin	0.93	0.44	0.59	0.61	0.40	0.40
	S-news	0.87	0.29	0.44	0.62	0.21	0.31
	S-news+	0.87	0.30	0.45	0.67	0.24	0.35
	S-clin-	0.94	0.28	0.43	0.56	0.17	0.26
	S-clin-+	0.94	0.25	0.40	0.56	0.15	0.24
	S-clin	0.96	0.30	0.46	0.61	0.19	0.29
	S-clin+	**0.99**	0.28	0.43	0.67	0.19	0.29
News	H-news	0.85	**0.79**	**0.82**	**0.83**	**0.78**	**0.81**
	H-clin	0.85	0.79	0.82	0,75	0.70	0.72
	S-news	0.83	0.66	0.68	0.77	0.61	0.68
	S-news+	**0.86**	0.69	0.77	0.83	0.68	0.74
	S-clin-	0.84	0.41	0.55	0.61	0.31	0.41
	S-clin-+	0.76	0.33	0.45	0.55	0.24	0.33
	S-clin	0.76	0.41	0.53	0.62	0.34	0.44
	S-clin+	0.77	0.38	0.51	0.65	0.33	0.43

Table 3: Evaluation of temporal expression extraction in French narratives across three domains. Values from models with a '*' come from Strötgen et al. (2014b). Models with a '+' used custom tokenization. Models with a '-' used the reduced training set.

data that is as close as possible to in-domain, such as News vs. Clinical for Historical) of any reasonable size yields better performance, even if still under the rule-based approach.

Limitations of this study. Size imbalance in the corpora used in our study was a limitation; the clinical corpus is much larger than the other two and the Historical corpus is really small, which limits the applicabiblity of statistical methods. In our work with the clinical corpus, more time was spent on annotating data and implementing statistical models vs. developping rules. Arguably, devoting additional efforts to rule development might improve the rule-based performance.

6 Conclusion and Future Work

This study contributes to a better understanding of temporal expression recognition accross domains. We found that an important part of domain specificity lies in the distribution of the types of temporal expressions accross domains. We also noticed that specific mentions of temporal expressions can be categorized as different types from one domain to another (e.g. *le soir* was generally considered a set in our clinical corpus – as in *every night* – and a time in the news corpus – as in *in the evening*). The results of our domain adaptation experiments suggest that the performance of temporal expression recognition is improved when domain specificity is taken into account by using in-domain training data or domain-specific rules.

In terms of adaptation strategy, our experiments show that the addition of a limited number of rules to the default (news-oriented) HeidelTime leads to matching the expected performance of HeidelTime on a new domain corpus. Furthermore, we show that more substantial efforts spent on annotating data can result in training data that will support a statistical model that outperforms simple rule adaptation. We can hypothesize that devoting equivalent efforts towards rule development may also result in increased performance. We believe that some amount of corpus annotation is necessary to gain adequate corpus knowledge to craft such rules.

Overall, we show that domain adaptation for the extraction of temporal expressions can be done with limited efforts, provided that an adequate corpus is available. We found that the tokenization method used in pre-processing was instrumental for improving statistical model performance across domains.

In future work, we will address the task of temporal expression normalization.

Acknowledgments

This work was supported by the French National Agency for Research under grant CABeRneT[3] ANR-13-JS02-0009-01

The authors thank the Biomedical Informatics Department at the Rouen University Hospital for providing access to the LERUDI corpus for this work.

[3]CABeRneT: Compréhension Automatique de Textes Biomédicaux pour la Recherche Translationnelle

References

Steven Bethard, Leon Derczynski, Guergana Savova, James Pustejovsky, and Marc Verhagen. 2015. SemEval-2015 Task 6: Clinical TempEval. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 806–814, Denver, USA, jun. Association for Computational Linguistics.

André Bittar, Pascal Amsili, Pascal Denis, and Laurence Danlos. 2011. French TimeBank: An ISO-TimeML Annotated Reference Corpus. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers - Volume 2*, HLT '11, pages 130–134, Stroudsburg, PA, USA. Association for Computational Linguistics.

Antske Fokkens, Marieke van Erp, Marten Postma, Ted Pedersen, Piek Vossen, and Nuno Freire. 2013. Offspring from Reproduction Problems: What Replication Failure Teaches Us. In *ACL (1)*, pages 1691–1701, Sofia, Bulgaria. The Association for Computer Linguistics.

Cyril Grouin and Aurélie Névéol. 2014. De-Identification of Clinical Notes in French: towards a Protocol for Reference Corpus Developpement. In *J Biomed Inform*, Aug.

P. Jindal and D. Roth. 2013. Extraction of Events and Temporal Expressions from Clinical Narratives. *Journal of Biomedical Informatics (JBI)*, 10.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Thomas Lavergne, Olivier Cappé, and Fran¸cois Yvon. 2010. Practical Very Large Scale CRFs. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 504–513, Uppsala, Sweden.

Hui Li, Jannik Strötgen, Julian Zell, and Michael Gertz. 2014. Chinese Temporal Tagging with HeidelTime. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, volume 2: Short Papers*, pages 133–137. Association for Computational Linguistics, April.

Véronique Moriceau and Xavier Tannier. 2014. French Resources for Extraction and Normalization of Temporal Expressions with HeidelTime. In *Proceedings of the 9th International Conference on Language Resources and Evaluation (LREC 2014)*, Reykjavík, Iceland, may.

J Pustejovsky, P Hanks, R Saur, A See, R Gaizauskas, A Setzer, D Radev, B Sundheim, D Day, L Ferro, and M Lazo. 2003. The TimeBank corpus. *Corpus Linguistics*, page 647–656.

James Pustejovsky, Kiyong Lee, Harry Bunt, and Laurent Romary. 2010. ISO-TimeML: An International Standard for Semantic Annotation. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation, (LREC'10)*, pages 394–7, La Valette, Malta, May.

Helmut Schmid. 1994. Probabilistic Part-of-Speech Tagging Using Decision Trees. In *International Conference on New Methods in Language Processing*, September.

Pontus Stenetorp, Sampo Pyysalo, Goran Topić, Tomoko Ohta, Sophia Ananiadou, and Jun'ichi Tsujii. 2012. BRAT: A Web-based Tool for NLP-assisted Text Annotation. In *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, EACL '12, pages 102–107, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jannik Strötgen and Michael Gertz. 2010. HeidelTime: High Quality Rule-based Extraction and Normalization of Temporal Expressions. In *Proceedings of the 5th International op on Semantic Evaluation*, SemEval '10, pages 321–324, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jannik Strötgen and Michael Gertz. 2012. Temporal Tagging on Different Domains: Challenges, Strategies, and Gold Standards. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Mehmet Ugur Dogan, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey, may. European Language Resources Association (ELRA).

Jannik Strötgen, Ayser Armiti, Tran Van Canh, Julian Zell, and Michael Gertz. 2014a. Time for More Languages: Temporal Tagging of Arabic, Italian, Spanish, and Vietnamese. *ACM Transactions on Asian Language Information Processing (TALIP)*, 13(1):1–21.

Jannik Strötgen, Thomas Bögel, Julian Zell, Ayser Armiti, Tran Van Canh, and Michael Gertz. 2014b. Extending HeidelTime for Temporal Expressions Referring to Historic Dates. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 2390–2397. European Language Resources Association (ELRA), May.

William F Styler IV, Steven Bethard, Sean Finan, Martha Palmer, Sameer Pradhan, Piet C de Groen, Brad Erickson, Timothy Miller, Chen Lin, Guergana Savova, and James Pustejovsky. 2014. Temporal Annotation in the Clinical Domain. *Transactions of the Association for Computational Linguistics*, page 143–154, Apr.

Weiyi Sun, Anna Rumshisky, and Özlem Uzuner. 2013a. Annotating temporal information in clinical narratives. *J Biomed Inform*, 46:Suppl:S5–12, Dec.

Weiyi Sun, Anna Rumshisky, and Özlem Uzuner. 2013b. Evaluating temporal relations in clinical text: 2012 i2b2 Challenge. *J Am Med Inform Assoc.*, 20:806–813, Sep-Oct.

Naushad UzZaman, Hector Llorens, Leon Derczynski, Marc Verhagen, JF Allen, and James Pustejovsky. 2013. SemEval-2013 Task 1: TempEval-3: Evaluating Time Expressions, Events, and Temporal Relations. In *Proceedings of the 7th International Workshop on Semantic Evaluation (SemEval 2013)*, pages 1–9. ACL, Jun.

Sumithra Velupillai. 2014. Temporal Expressions in Swedish Medical Text – A Pilot Study. In *Proceedings of BioNLP 2014*, pages 88–92, Baltimore, Maryland, June. Association for Computational Linguistics.

M Verhagen, R Gaizauskas, F Schilder, M Hepple, and J Pustejovsky. 2007. Semeval-2007 task 15: TempEval temporal relation identification. In *SemEval-2007: 4th International Workshop on Semantic Evaluations*.

Karin Verspoor, Antonio Jimeno Yepes, Lawrence Cavedon, Tara McIntosh, Asha Herten-Crabb, Zoë Thomas, and John-Paul Plazzer. 2013. Annotating the biomedical literature for the human variome. *Database (Oxford)*, page bat019–bat019.

S Wu, T Miller, J Masanz, M Coarr, S Halgrim, D Carrell, and C Clark. 2014. Negation's not solved: generalizability versus optimizability in clinical natural language processing. *PLoS One*, 9:e112774, Nov.

Semi-Supervised Bootstrapping of Relationship Extractors with Distributional Semantics

David S. Batista **Bruno Martins** **Mário J. Silva**
INESC-ID, Instituto Superior Técnico, Universidade de Lisboa
{david.batista,bruno.g.martins,mario.gaspar.silva}@ist.utl.pt

Abstract

Semi-supervised bootstrapping techniques for relationship extraction from text iteratively expand a set of initial seed relationships while limiting the semantic drift. We research bootstrapping for relationship extraction using word embeddings to find similar relationships. Experimental results show that relying on word embeddings achieves a better performance on the task of extracting four types of relationships from a collection of newswire documents when compared with a baseline using TF-IDF to find similar relationships.

1 Introduction

Relationship Extraction (RE) transforms unstructured text into relational triples, each representing a relationship between two named-entities. A bootstrapping system for RE starts with a collection of documents and a few seed instances. The system scans the document collection, collecting occurrence contexts for the seed instances. Then, based on these contexts, the system generates extraction patterns. The documents are scanned again using the patterns to match new relationship instances. These newly extracted instances are then added to the seed set, and the process is repeated until a certain stop criteria is met.

The objective of bootstrapping is thus to expand the seed set with new relationship instances, while limiting the semantic drift, i.e. the progressive deviation of the semantics for the extracted relationships from the semantics of the seed relationships.

State-of-the-art approaches rely on word vector representations with TF-IDF weights (Salton and Buckley, 1988). However expanding the seed set by relying on TF-IDF representations to find similar instances has limitations, since the similarity between any two relationship instance vectors of TF-IDF weights is only positive when the instances share at least one term. For instance, the phrases *was founded by* and *is the co-founder of* do not have any common words, but they have the same semantics. Stemming techniques can aid in these cases, but only for variations of the same root word (Porter, 1980).

We propose to address this challenge with an approach based on word embeddings (Mikolov et al., 2013a). By relying on word embeddings, the similarity of two phrases can be captured even if no common words exist. The word embeddings for *co-founder* and *founded* should be similar, since these words tend to occur in the same contexts. Word embeddings can nonetheless also introduce semantic drift. When using word embeddings, phrases like *studied history at* can, for instance, have a high similarity with phrases like *history professor at*. In our approach, we control the semantic drift by ranking the extracted relationship instances, and by scoring the generated extraction patterns.

We implemented these ideas in BREDS, a bootstrapping system for RE based on word embeddings. BREDS was evaluated with a collection of 1.2 million sentences from news articles. The experimental results show that our method outperforms a baseline bootstrapping system based on the ideas of Agichtein and Gravano (2000) which relies on TF-IDF representations.

2 Bootstrapping Relationship Extractors

Brin (1999) developed DIPRE, the first system to apply bootstrapping for RE, which represents the occurrences of seeds as three contexts of strings: words before the first entity (BEF), words between the two entities (BET), and words after the second entity (AFT). DIPRE generates extraction patterns by grouping contexts based on string matching, and controls semantic drift by limiting the number of instances a pattern can extract.

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 499–504,
Lisbon, Portugal, 17-21 September 2015.

Agichtein and Gravano (2000) developed Snowball, which is inspired on DIPRE's method of collecting three contexts for each occurrence, but computing a TF-IDF representation for each context. The seed contexts are clustered with a single-pass algorithm based on the cosine similarity between contexts using the three vector representations:

$$\begin{aligned}\text{Sim}(S_n, S_j) = \alpha \cdot \cos(BEF_i, BEF_j) \\ + \beta \cdot \cos(BET_i, BET_j) \\ + \gamma \cdot \cos(AFT_i, AFT_j)\end{aligned} \quad (1)$$

In the formula, the parameters α, β and γ weight each vector. An extraction pattern is represented by the centroid of the vectors that form a cluster. The patterns are used to scan the text again, and for each segment of text where any pair of entities with the same semantic types as the seeds co-occur, three vectors are generated. If the similarity from the context vectors towards an extraction pattern is greater than a threshold τ_{sim}, the instance is extracted.

Snowball scores the patterns and ranks the extracted instances to control the semantic drift. A pattern is scored based on the instances that it extracted, which can be included in three sets: P, N, and U. If an extracted instance contains an entity e_1, which is part of a seed, and if the associated entity e_2 in the instance is the same as in in the seed, then the extraction is considered positive (included in set P). If the relationship contradicts a relationship in the seed set (i.e., e_2 does not match), then the extraction is considered negative (included in a set N). If the relationship is not part of the seed set, the extraction is considered unknown (included in a set U). A score is assigned to each pattern p according to:

$$\text{Conf}_\rho(p) = \frac{|P|}{|P| + W_{ngt} \cdot |N| + W_{unk} \cdot |U|} \quad (2)$$

W_{ngt} and W_{unk} are weights associated to the negative and unknown extractions, respectively. The confidence of an instance is calculated based on the similarity scores towards the patterns that extracted it, weighted by the pattern's confidence:

$$\text{Conf}_\iota(i) = 1 - \prod_{j=0}^{|\xi|} (1 - \text{Conf}_\rho(\xi_j) \times \text{Sim}(C_i, \xi_j)) \quad (3)$$

where, ξ is the set of patterns that extracted i, and C_i is the textual context where i occurred. Instances with a confidence above a threshold τ_t are used as seeds in the next iteration.

3 Bootstrapping Relationship Extractors with Word Embeddings

BREDS follows the architecture of Snowball, having the same processing phases: find seed matches, generating extraction patterns, finding relationship instances, and detecting semantic drift. It differs, however, in that it attempts to find similar relationships using word embeddings, instead of relying on TF-IDF representations.

3.1 Find Seed Matches

BREDS scans the document collection and, if both entities of a seed instance co-occur in a text segment within a sentence, then that segment is considered and BREDS extracts the three textual contexts as in Snowball: BEF, BET, and AFT.

In the BET context, BREDS tries to identify a relational pattern based on a shallow heuristic originally proposed in ReVerb (Fader et al., 2011). The pattern limits a relation context to a verb (e.g., *invented*), a verb followed by a preposition (e.g., *located in*), or a verb followed by nouns, adjectives, or adverbs ending in a preposition (e.g., *has atomic weight of*). These patterns will nonetheless only consider verb mediated relationships. If no verbs exist between two entities, BREDS extracts all the words between the two entities, to build the representations for the BET context.

Each context is transformed into a single vector by a simple compositional function that starts by removing stop-words and adjectives and then sums the word embedding vectors of each individual word. Representing small phrases by summing each individual word's embedding results in good representations for the semantics in small phrases (Mikolov et al., 2013b).

A relationship instance i is represented by three embedding vectors: V_{BEF}, V_{BET}, and V_{AFT}. Considering the sentence:

The tech company Soundcloud *is based in* Berlin, *capital of Germany.*

BREDS generates the relationship instance with:

$$V_{BEF} = \text{E}(\text{“tech”}) + \text{E}(\text{“company”})$$
$$V_{BET} = \text{E}(\text{“is”}) + \text{E}(\text{“based”})$$
$$V_{AFT} = \text{E}(\text{“capital”})$$

where, E(x) is the word embedding for word x.

BREDS also tries to identify the passive voice using part-of-speech (PoS) tags, which can help to detect the correct order of the entities in a relational triple. BREDS identifies the presence of the passive voice by considering any form of the verb *to be*, followed by a verb in the past tense or the past participle, and ending in the word *by*.

For instance, the seed <Google, owns, DoubleClick> states that the organisation Google owns the organisation DoubleClick. Using this seed, if BREDS detects a pattern like *agreed to be acquired by* it will swap the order of the entities when producing a relational triple, outputting the triple <ORG$_2$, *owns*, ORG$_1$>, instead of the triple <ORG$_1$, *owns*, ORG$_2$>.

3.2 Extraction Patterns Generation

As Snowball, BREDS generates extraction patterns by applying a single-pass clustering algorithm to the relationship instances gathered in the previous step. Each resulting cluster contains a set of relationship instances, represented by their three context vectors.

Algorithm 1 describes the clustering approach taken by BREDS, which takes as input a list of relationship instances and assigns the first instance to a new empty cluster. Next, it iterates through the list of instances, computing the similarity between an instance i_n and every cluster Cl_j. The instance i_n is assigned to the first cluster whose similarity is higher or equal to a threshold τ_{sim}. If all the clusters have a similarity lower than a threshold τ_{sim}, a new cluster C_m is created, containing the instance i_n.

The similarity function Sim(i_n, Cl_j), between an instance i_n and a cluster Cl_j, returns the maximum of the similarities between an instance i_n and any of the instances in a cluster Cl_j, if the majority of the similarity scores is higher than a threshold τ_{sim}. A value of zero is returned otherwise. The similarity between two instances is computed according to Formula (1). As a result, clustering in Algorithm 1 differs from the original Snowball method, which instead computes similarities towards cluster centroids.

3.3 Find Relationship Instances

After the generation of extraction patterns, BREDS finds relationship instances with Algorithm 2. It scans the documents once again, collecting all segments of text containing entity pairs whose semantic types are the same as those in the seed instances. For each segment, an instance i is generated as described in Section 3.1, and the similarity towards all previously generated extraction patterns (i.e., clusters) is computed. If the similarity between i and a pattern Cl_j is equal or above τ_{sim}, then i is considered a candidate instance, and the confidence score of the pattern is updated, according to Formula (2). The pattern which has the highest similarity ($pattern_{best}$) is associated with i, along with the corresponding similarity score (sim_{best}). This information is kept in a history of $Candidates$. Note that the histories of $Candidates$ and $Patterns$ are kept through all the bootstrap iterations, and new patterns or instances can be added, or the scores of existing patterns or instances can change.

Algorithm 1: Single-Pass Clustering.

Input: $Instances = \{i_1, i_2, i_3, ..., i_n\}$
Output: $Patterns = \{\}$
$Cl_1 = \{i_1\}$
$Patterns = \{Cl_1\}$
for $i_n \in Instances$ **do**
  for $Cl_j \in Patterns$ **do**
    if $Sim(i_n, Cl_j) >= \tau_{sim}$ **then**
      $Cl_j = Cl_j \cup \{i_n\}$
    else
      $Cl_m = \{i_n\}$
      $Patterns = Patterns \cup \{Cl_m\}$

Algorithm 2: Find Relationship Instances.

Input: $Sentences = \{s_1, s_2, s_3, ..., s_n\}$
Input: $Patterns = \{Cl_1, Cl_2, ..., Cl_n\}$
Output: $Candidates$
for $s_i \in Sentences$ **do**
  $i = create\_instance(s_i)$
  $sim_{best} = 0$
  $p_{best} = None$
  for $Cl_i \in Patterns$ **do**
    $sim = \text{Sim}(i, Cl_i)$
    if $sim >= \tau_{sim}$ **then**
      $\text{Conf}_\rho(C_i)$
      if $sim >= sim_{best}$ **then**
        $sim_{best} = sim$
        $P_{best} = Cl_i$
  $Candidates[i].patterns[p_{best}] = sim_{best}$

3.4 Semantic Drift Detection

As Snowball, BREDS ranks the candidate instances at the end of each iteration, based on the scores computed with Formula (3). Instances with a score equal or above the threshold τ_t are added to the seed set, for use in the next iteration of the bootstrapping algorithm.

4 Evaluation

In our evaluation we used a set of 5.5 million news articles from AFP and APW (Parker et al., 2011).

Our pre-processing pipeline is based on the models provided by the NLTK toolkit (Bird et al., 2009): sentence segmentation[1], tokenisation[2], PoS-tagging[3] and named-entity recognition (NER). The NER module in NLTK is a wrapper over the Stanford NER toolkit (Finkel et al., 2005).

We performed weak entity-linking by matching entity names in sentences with FreebaseEasy (Bast et al., 2014). FreebaseEasy is a processed version of Freebase (Bollacker et al., 2008), which contains a unique meaningful name for every entity, together with canonical binary relations. For our experiments, we selected only the sentences containing at least two entities linked to FreebaseEasy, which corresponded to 1.2 million sentences.

With the full articles set, we computed word embeddings with the skip-gram model[4] using the *word2vec*[5] implementation from Mikolov et. al. (2013a). The TF-IDF representations used by Snowball were calculated over the same articles set. We adopted a previously proposed framework for the evaluation of large-scale RE systems by Bronzi et al. (2012), to estimate precision and recall, using FreebaseEasy as the knowledge base.

We considered entity pairs no further away than 6 tokens, and a window of 2 tokens for the BEF and AFT contexts, ignoring the remaining of the sentence. We discarded the clusters with only one relationship instances, and ran a maximum of 4 bootstrapping iterations. The W_{unk} and W_{ngt} parameters were set to 0.1 and 2, respectively, based on the results reported by Yu et al. (2003).

We compared BREDS against Snowball in four relationship types, shown in Table 1. For each relationship type we considered several bootstrapping configurations by combining different values for the τ_{sim} and τ_t thresholds, all within the interval [0.5,1.0].

Relationship	Seeds
acquired	{Adidas, Reebok}
	{Google, DoubleClick}
founder-of	{CNN, Ted Turner}
	{Amazon, Jeff Bezos}
headquartered	{Nokia, Espoo}
	{Pfizer, New York}
affiliation	{Google, Marissa Mayer}
	{Xerox, Ursula Burns}

Table 1: Relationship types and used seeds.

We bootstrapped each relationship with two context weighting configurations in Formula (1):

- Conf_1: $\alpha = 0.0$, $\beta = 1.0$, $\gamma = 0.0$
- Conf_2: $\alpha = 0.2$, $\beta = 0.6$, $\gamma = 0.2$

where Conf_1 only considers the BET context and Conf_2 uses the three contexts, while giving more importance to the BET context.

Table 2 shows, for each relationship type, the best F_1 score and the corresponding precision and recall, for all combinations of τ_{sim} and τ_t values, and considering only extracted relationship instances with confidence scores equal or above 0.5. Table 2a shows the results for the BREDS system, while Table 2b shows the results for Snowball (ReVerb), a modified Snowball in which a relational pattern based on ReVerb is used to select the words for the BET context. Finally, Table 2c shows the results for Snowball, implemented as described in the original paper.

Overall, BREDS achieves better F_1 scores than both versions of Snowball. The F_1 score of BREDS is higher, mainly as a consequence of much higher recall scores, which we believe to be due to the relaxed semantic matching caused by using the word embeddings. For some relationship types, the recall more than doubles when using word embeddings instead of TF-IDF. For the *acquired* relationship, when considering Conf_1, the precision of BREDS drops compared with the other versions of Snowball, but without affecting the F_1 score, since the higher recall compensates for the small loss in precision.

Regarding the context weighting configurations, Conf_2 produces a lower recall when compared to Conf_1. This might be caused by the

[1] nltk.tokenize.punkt.PunktSentenceTokenizer
[2] nltk.tokenize.treebank.TreebankWordTokenizer
[3] taggers/maxent_treebank_pos_tagger/english.pickle
[4] skip length of 5 tokens and vectors of 200 dimensions
[5] https://code.google.com/p/word2vec/

sparsity of both BEF and AFT, which contain many different words that do not contribute to capture the relationship between the two entities. Although, sometimes, the phrase or word that indicates a relationship occurs on the BEF or AFT contexts, it is more often the case that these phrases or words occur in the BET context.

The performance results of Snowball (Classic) and Snowball (ReVerb) suggest that selecting words based on a relational pattern to represent the BET context, instead of using all the words, works better for TF-IDF representations.

The results also show that word embeddings can generate more extraction patterns. For instance, for the *founder-of* relationship, BREDS learns patterns based on words such as *founder, co-founder, co-founders* or *founded*, while Snowball only learns patterns that have the word *founder*, like *CEO and founder* or *founder and chairman*.

The implementations of BREDS and Snowball, as described in this paper, are available on-line[6].

5 Conclusions and Future Work

This paper reports on a novel bootstrapping system for relation extraction based on word embeddings. In our experiments, bootstrapped RE achieved better results when using word embeddings to find similar relationships than with similarities between TF-IDF weighted vectors.

We have identified two main sources of errors: NER problems and incorrect relational patterns extraction due to the use of a shallow heuristic that only captures local relationships.

In future work, more robust entity-linking approaches, as proposed by Hoffart et al. (2011), could be included in our pre-processing pipeline. This could alleviate NER errors and enable experimentation with other relationship types.

Gabbard et al. (2011) have shown that coreference resolution can increase bootstrapping RE performance, and the method of Durrett and Klein (2014) could also be included in our preprocessing pipeline.

Finally, we could explore richer compositional functions, combining word embeddings with syntactic dependencies (SD) (Yu et al., 2014). The shortest path between two entities in an SD tree supports the extraction of local and long-distance relationships (Bunescu and Mooney, 2005).

[6] https://github.com/davidsbatista/BREDS

BREDS			
Relationship	**Precision**	**Recall**	**F_1**
	$Conf_1$		
acquired	0.73	**0.77**	**0.75**
founder-of	**0.98**	**0.86**	**0.91**
headquartered	0.63	**0.69**	**0.66**
affiliation	**0.85**	**0.91**	**0.88**
	$Conf_2$		
acquired	**1.00**	0.15	0.26
founder-of	0.97	0.79	0.87
headquartered	**0.64**	0.61	0.62
affiliation	0.84	0.60	0.70

(a) Precision, Recall and F_1 results obtained with different configurations of BREDS.

Snowball (ReVerb)			
Relationship	**Precision**	**Recall**	**F_1**
	$Conf_1$		
acquired	0.83	0.61	0.70
founder-of	0.96	0.77	0.86
headquartered	0.48	0.63	0.55
affiliation	0.52	0.29	0.37
	$Conf_2$		
acquired	0.73	0.22	0.34
founder-of	0.97	0.75	0.85
headquartered	0.55	0.42	0.47
affiliation	0.36	0.05	0.08

(b) Precision, Recall and F_1 results obtained with different configurations of Snowball (ReVerb).

Snowball (Classic)			
Relationship	**Precision**	**Recall**	**F_1**
	$Conf_1$		
acquired	0.87	0.54	0.67
founder-of	0.97	0.76	0.85
headquartered	0.52	0.61	0.57
affiliation	0.49	0.29	0.36
	$Conf_2$		
acquired	0.77	0.54	0.63
founder-of	0.98	0.73	0.84
headquartered	0.53	0.54	0.54
affiliation	0.42	0.08	0.13

(c) Precision, Recall and F_1 results obtained with different configurations of Snowball (Classic).

Table 2: Precision, Recall and F_1 scores over the four relationships for the three different systems.

Acknowledgements

This work was supported by Fundação para a Ciência e a Tecnologia, under grants SFRH/BD/70478/2010, UID/CEC/50021/2013 and EXCL/EEI- ESS/0257/2012 (DataStorm).

References

Eugene Agichtein and Luis Gravano. 2000. Snowball: Extracting Relations from Large Plain-Text Collections. In *Proceedings of the ACM Conference on Digital libraries*, DL'00.

Hannah Bast, Florian Bäurle, Björn Buchhold, and Elmar Haußmann. 2014. Easy Access to the Freebase Dataset. In *Companion Publication of the 23rd International Conference on World Wide Web*, WWW Companion '14.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python*. O'Reilly Media, Inc.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: A Collaboratively Created Graph Database for Structuring Human Knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08.

Sergey Brin. 1999. Extracting Patterns and Relations from the World Wide Web. In *Selected Papers from the International Workshop on The World Wide Web and Databases*, WebDB'98.

Mirko Bronzi, Zhaochen Guo, Filipe Mesquita, Denilson Barbosa, and Paolo Merialdo. 2012. Automatic Evaluation of Relation Extraction Systems on Large-scale. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*, AKBC-WEKEX '12.

Razvan Bunescu and Raymond Mooney. 2005. A shortest path dependency kernel for relation extraction. In *Proceedings of the Human Language Technology Conference and the Conference on Empirical Methods in Natural Language Processing*, HLT/EMNLP'05.

Greg Durrett and Dan Klein. 2014. A Joint Model for Entity Analysis: Coreference, Typing, and Linking. *Transactions of the Association for Computational Linguistics*, 2.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying Relations for Open Information Extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, ACL'05.

Ryan Gabbard, Marjorie Freedman, and Ralph M. Weischedel. 2011. Coreference for Learning to Extract Relations: Yes Virginia, Coreference Matters. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, ACL'11.

Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. 2011. Robust Disambiguation of Named Entities in Text. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient Estimation of Word Representations in Vector Space. In *Proceedings of the Workshop at the International Conference on Learning Representations*, ICLR'13.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed Representations of Words and Phrases and their Compositionality. In *Proceedings of the Conference on Neural Information Processing Systems*, NIPS'13.

Robert Parker, David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2011. English Gigaword Fifth Edition LDC2011T07.

M.F. Porter. 1980. An algorithm for suffix stripping. *Program*, 14(3).

Gerard Salton and Christopher Buckley. 1988. Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5).

Hong Yu and Eugene Agichtein. 2003. Extracting Synonymous Gene and Protein Terms from Biological Literature. *Bioinformatics*, 19(suppl 1).

Mo Yu, Matthew R. Gormley, and Mark Dredze. 2014. Factor-based Compositional Embedding Models. In *Proceedings of the Conference on Neural Information Processing Systems*, NIPS'14.

Extraction and generalisation of variables from scientific publications

Erwin Marsi, Pinar Öztürk
Department of Computer and Information Science
Norwegian University of Science and Technology (NTNU)
{emarsi,pinar}@idi.ntnu.no

Abstract

Scientific theories and models in Earth science typically involve changing *variables* and their complex interactions, including correlations, causal relations and chains of positive/negative feedback loops. Variables tend to be complex rather than atomic entities and expressed as noun phrases containing multiple modifiers, e.g. *oxygen depletion in the upper 500 m of the ocean or timing* and *magnitude of surface temperature evolution in the Southern Hemisphere in deglacial proxy records*. Text mining from Earth science literature is therefore significantly different from biomedical text mining and requires different approaches and methods. Our approach aims at automatically locating and extracting variables and their direction of variation: *increasing*, *decreasing* or just *changing*. Variables are initially extracted by matching tree patterns onto the syntax trees of the source texts. Next, variables are generalised in order to enhance their similarity, facilitating hierarchical search and inference. This generalisation is accomplished by progressive pruning of syntax trees using a set of tree transformation operations. Text mining results are presented as a browsable variable hierarchy which allows users to inspect all mentions of a particular variable type in the text as well as any generalisations or specialisations. The approach is demonstrated on a corpus of 10k abstracts of Nature publications in the field of Marine science. We discuss experiences with this early prototype and outline a number of possible improvements and directions for future research.

1 Introduction

Text mining of scientific literature originates from efforts to cope with the ever growing flood of publications in biomedicine (Swanson, 1986; Swanson, 1988; Swanson and Smalheiser, 1997; Hearst, 1999; Ananiadou et al., 2006; Zweigenbaum et al., 2007; Cohen and Hersh, 2005; Krallinger et al., 2008; Rodriguez-Esteban, 2009; Zweigenbaum and Demner-Fushman, 2009; Ananiadou et al., 2010; Simpson and Demner-Fushman, 2012; Ananiadou et al., 2014). Consequently the resulting approaches, methods, tools and applications – as well as data, corpora and evaluation tasks – are rooted in the paradigm of biomedical research and its conceptual framework. Typical source text consists of abstracts from PubMed or full-text articles from PubMed Central. Standard tasks include recognition, normalisation and mapping of biological entities (e.g., genes, proteins, drugs, symptoms and diseases), extraction of biological relations (e.g., protein-protein interaction, disease-gene associations or drug-drug interaction) or bio-event extraction (e.g., regulation or inhibition events and their participants). There are extensive ontologies like the Gene Ontology (Consortium, 2001), annotated corpora like the GENIA (Kim et al., 2003) and BioInfer (Pyysalo et al., 2007) corpora and dedicated shared tasks including BioCreative (Hirschman et al., 2005) and BioNLP (Pyysalo et al., 2012). In short, there is a whole infrastructure supporting biomedical text mining (Cohen and Hunter, 2008).

Text mining is now spreading out to other scientific disciplines, notably in the humanities and social sciences (O'Connor et al., 2011), holding the promise for knowledge discovery from large text collections. Our own research targets text mining in the field of Earth science, more specifically in Oceanography or Marine science, with a focus on climate change. As text mining efforts in this

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 505–511,
Lisbon, Portugal, 17-21 September 2015.

area are extremely rare (Ekstrom and Lau, 2008; Vossen et al., 2010; Zhang et al., 2013; Marsi et al., 2014; Aamot, 2014), it is not surprising that a corresponding infrastructure is mostly lacking. In addition, however, we found that due to significant differences between the conceptual frameworks of biomedicine and marine science, simply "porting" the biomedical text mining infrastructure to another domain will not suffice.

One major difference is that the biomedical entities of interest are relatively well defined – genes, proteins, organisms, species, drugs, diseases, etc. – and typically expressed as proper nouns. In contrast, defining the entities of interest in marine science turns out to be much harder. Not only does it seem to be more open-ended in nature, the entities themselves tend to be complex and expressed as noun phrases containing multiple modifiers, giving rise to examples like *oxygen depletion in the upper 500 m of the ocean* or *timing and magnitude of surface temperature evolution in the Southern Hemisphere in deglacial proxy records*.

Given the difficulties with entities, we propose to concentrate first on text mining of events, leaving entities underspecified for the time being. Theories and models in marine science are characterised by changing variables and their complex interactions, including correlations, causal relations and chains of positive/negative feedback loops. Many marine scientists are interested in finding evidence – or counter-evidence – in the literature for events of change and their relations. Here we present ongoing work to automatically locate and extract *variables* and their direction of variation: *increasing*, *decreasing* or just *changing*. Examples are given in Table 1.

Since many of these changing variables are long and complex expressions, their frequency of occurrence tends to be low, making the discovery of relations among different variables harder. As a partial solution to this problem, we propose progressive pruning of syntax trees using a set of tree transformation operations. For example, generalising *oxygen depletion in the upper 500 m of the ocean* to *oxygen depletion in the ocean* and subsequently to the much more frequent *oxygen depletion*. Text mining results are then presented as a browsable variable hierarchy which allows users to inspect all mentions of a particular variable type in the text as well as any generalisations or specialisations.

2 Variable extraction

Our text material consists of 10k abstracts from journals published by Nature Publishing Group. Search terms obtained from domain experts were used to query Nature's OpenSearch API[1] for publications in a limited range of relevant journals, after 1997, retrieving records including title and abstract. The top-10k abstracts matching most search terms were selected for further processing with CoreNLP (Manning et al., 2014), including tokenisation, sentence splitting, POS tagging, lemmatisation and parsing. Lemmatised parse trees were obtained by substituting terminals with their lemmas. The resulting new corpus contains 9,586 article abstracts, 59,787 sentences and approximately 4M tokens.

Methods for information extraction broadly rely on either knowledge-based pattern matching or supervised machine learning (Sarawagi, 2008). Although ML approaches are currently dominant in IE research, rule-based systems have several advantages, including: (a) the rules are interpretable and thus suitable for rapid development and domain transfer; and (b) humans and machines can contribute to the same model (Valenzuela-Escárcega et al., 2015). In our case, patterns offered more flexibility in exploring the domain, whereas the manual annotation required for ML demands more commitment to a precise definition of entities, relations and events, which we found hard to achieve at this stage. Tree pattern matching is applied to lemmatised syntax trees using the Tregex engine (Levy and Andrew, 2006), which supports a compact language for writing regular expressions over trees; see Table 1 for examples of patterns and matching phrases. For instance, the pattern for a decreasing variable is defined as a noun phrase (NP) that is immediately dominated (>) by a verb phrase (VP), which in turn is headed by (<<#) the lemma *reduce*. Similarly, the pattern for increase describes an NP dominated by a prepositional phrase (PP) that is headed by the preposition *in* or *of*; in addition, this PP must be preceded by an NP sister node (`$,`) headed by the lemma *increase*.

Patterns were generated by instantiating a small set of hand-written pattern templates, drawing from manually created lists of verbs and nouns ex-

[1] `http://www.nature.com/developers/documentation/api-references/opensearch-api`

Table 1: Examples of tree patterns and matching variables

Direction:	**Tree pattern:**	**Matched variable in sentence:**
Change	`NP <- (/NN/=d1 < variability $ /NN/) !$. PP`	Thus **the annual, Milankovitch and continuum temperature variability** together represent the response to deterministic insolation forcing.
Increase	`NP > (PP <<# (in|of) $, (NP <<# increase))`	The record reveals a linear increase in **annual temperature between 1958 and 2010** by 2.4 +/-1.2 degreesC ...
Decrease	`NP > (VP <<# reduce )`	Some researchers have observed that abundant natural gas substituting for coal could reduce **carbon dioxide (CO2) emissions**.

pressing change, increase or decrease. The patterns cover expression as a main verb (*X increases, something increases X)*, attributive use of verbs (*increasing temperature, temperature is increasing*), head of NP (*a temperature increase*) or NP with PP modifier (*increase in temperature*). The total number of patterns is 320: 90 for change, 122 for increase, 108 for decrease (see supplements for a full list). The total number of matched variables in the corpus is 21,817: 9,352 for change, 7,400 for increase and 5,065 for decrease.

Some variables do not exactly correspond to a node, i.e., not every variable is a valid syntactic phrase. For instance, the pattern for Change in Table 1 matches the NP *the annual, Milankovitch and continuum temperature variability*, whereas the actual variable is *the annual, Milankovitch and continuum temperature*. This is corrected in a post-processing step that deletes the *variability* node from the extracted subtree and substring. For this purpose, the pattern contains an assignment of the name d1 to the node directly dominating the lemma *variability* (`/NN/=d1 < variability`)), allowing a corresponding tree operation to delete this node, which is implemented using the Tsurgeon counterpart of Tregex.

3 Variable generalisation

Since many of the extracted variables are long and complex expressions, their frequency is low. The most frequent variables are generic terms (*climate* 1207, *temperature* 156, *global climate* 73), but over 66% is unique. This evidently impedes the discovery of relations among variables. As a partial solution to this problem, variables are generalised by progressive pruning of syntax trees using a set of tree transformation operations.

Figure 1 shows an example of generalisation by iterative tree pruning. The first transformation STRIP INIT DT strips the initial determiner from the NP. Next, COORD 3.1 deletes everything but the first conjunct from a coordinated structure of three NPs, resulting in *annual temperature* , which is finally reduced to just *temperature* by stripping the premodifier (STRIP PREMOD 1). An analogous procedure is applied to the other two conjuncts of the coordinated structure.

Tree transformations are implemented using Tsurgeon (Levy and Andrew, 2006): Tregex patterns match the syntactic structures of interest, whereas an associated Tsurgeon operation deletes selected nodes (see supplements for details). The transformations are ordered in four groups. The first group handles coordination of two to four conjuncts (cf. Figure 1) – at the phrase level or the lexical level – as well as cases of ellipsis (e.g. *hailstorm frequency and intensity* into *hailstorm frequency* and *hailstorm intensity*). The second group strips bracketed material in parenthetical and list structures. The third group deletes non-restrictive relative clauses and other non-restrictive modifiers preceded by a comma. The final group progressively strips premodifiers (mainly adjectives) from left to right and postmodifiers (PPs, relative clauses) from right to left . Since different transformation may arrive at the same generalisation (e.g. *temperature* in Figure 1), duplicates are filtered out. After filtering, 150,716 variables remained, which is 4.86 times the number of originally extracted variables.

As mentioned, the point of generalisation is to find relations among variables. In Table 1, for example, both *the annual, Milankovitch and continuum temperature variability* and *annual temperature between 1958 and 2010* are generalised to *annual temperature*. However, many generalised variables are unique and thus serve no purpose in relating variables. Retaining only original variables and generalised variables with at least two mentions yields a total of 17,613 variable types.

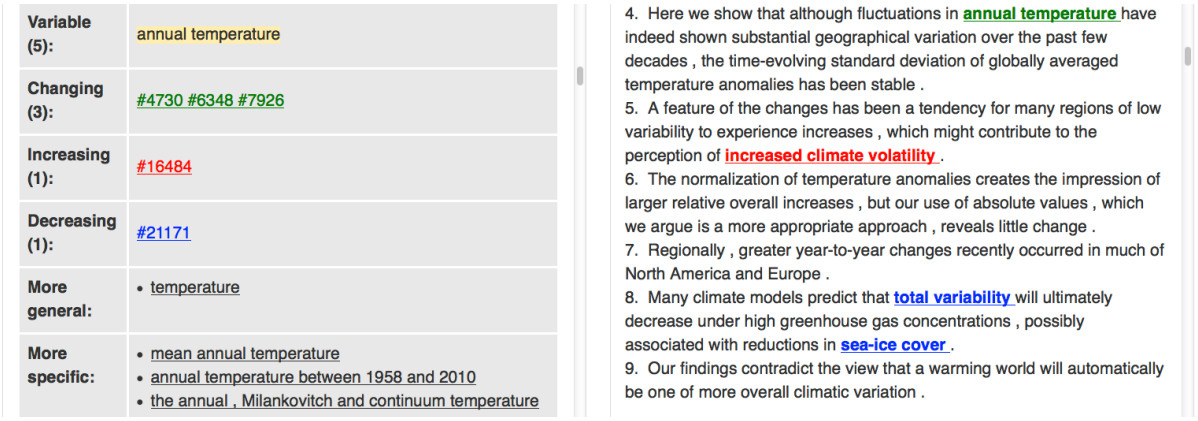


Figure 2: Partial screenshot of user interface showing variable type hierarchy (left) and linked variable mentions in text (right) where colour encodes change (green), increase (red) or decrease (blue)

the annual , Milankovitch and continuum temperature
  STRIP INIT DT → annual , Milankovitch and continuum temperature
    COORD 3.1 → annual temperature
      STRIP PREMOD 1 → temperature
    COORDI 3.2 → Milankovitch temperature
      STRIP PREMOD 1 → temperature
    COORD 3.3 → continuum temperature
      STRIP PREMOD 1 → temperature

Figure 1: Example of generalisation by iterative tree pruning

4 User interface

The output of the text mining step can be regarded as a directed graph where the nodes are variable *types* and the edges point from a more specific variable to a more general variable (as a result of a particular tree transformation). Each variable type is also linked to a set of tokens, i.e. variable mentions in the text which are either changing, increasing or decreasing. Figure 2 shows how this information is presented to the user in a browser (see supplements for full version). The left panel lists the variable types, ordered from most general to most specific and, secondary, on decreasing token frequency. Links point to more specific/general variables types, as well as to changing/increasing/decreasing variable mentions in the text. The right panel shows the source text, where colour encodes changing (green), increasing (red) or decreasing (blue) variable mentions, which are linked to their most specific variable type. This setup allows users to quickly explore variables, for example, finding abstracts containing a variable of interest and from there to related variables.

5 Discussion

We have argued that the paradigm established in biomedical text mining does not transfer directly to other scientific domains like Earth science. A new approach was proposed for extracting variables and their direction of variation (increasing, decreasing or just changing), focusing on events rather than entities. A generic system based on syntactic pattern matching and tree transformations was described for extraction and subsequent generalisation of variable events. Text mining results are presented in an innovative way as a browsable hierarchy ranging from most general to most specific variables, with links to their textual instances. In addition, a first text corpus in marine science was produced, including automatically annotated change events. Our corpus as well as the extracted variables are publicly available[2]. We think our approach to extraction is generalisable to other domains where the entities of interest are common nouns or complex noun phrases rather the proper nouns, e.g. in nanotechnology & nanoscience (Kostoff et al., 2007).

To the best of our knowledge, there are currently no other systems for text mining in Earth science which we can compare our results with, nor are there any benchmark data sets for our task. Most related is (Marsi et al., 2014), but their definition of variables is more restricted and their pilot corpus is too small for evaluation purposes. Reporting on our ongoing work now, future work will include an evaluation by asking domain exports to judge the correctness of extracted variables as well

[2] https://dl.dropboxusercontent.com/u/2370516/emnlp15_corpus.zip

as their generalisations in the given context.

Preliminary observations indicate that most problems originate from syntactic parsing errors, in particular well-known ambiguities in coordination and PP-attachment. As a result, patterns may either fail to match or match unintentionally, yielding incomplete or incoherent variables. Since many sentences are long, complex and domain-specific, it comes as no surprise that the parser often fails to correctly resolve well-known ambiguities in coordination and PP-attachment. However, with pattern matching on strings and/or POS tags instead of syntax trees, determining boundaries of variables would be problematic. False positives also occur because of different semantics of the same pattern, e.g. *change in western Europe* is unlikely to mean literally that the European continent is changing, neither does *changes in less than a few thousand years* imply that past years are changing.

At the same time, certain false negatives are beyond the power of pattern matching. For instance, variation may be entailed rather than explicitly stated: *ocean acidification* entails increasing acidity of ocean water and *Arctic warming* entails increasing temperature in the Arctic region. This is closely related to textual entailment (Androutsopoulos and Malakasiotis, 2010; Dagan et al., 2006), requiring inference in combination with domain knowledge. A related matter is negation (*no increase in global temperature*), which can even be expressed in non-trivial ways (*temperature remained constant*) (Morante and Daelemans, 2009). Variables were also found to be recursive or embedded, expressing "a change of a change". For example, *reduce subseasonal temperature variance* implies both a change in temperature as well as a decrease of this temperature change. The current visualisation falls short in these cases, as HTML browsers cannot render a link in a link.

Generalisation by tree pruning appears to work quite well as long as the parse is correct. However, pruning by itself is insufficient and should be supplemented with other methods. For instance, linking named entities like species, chemicals or locations to unique concepts in appropriate ontologies/taxonomies would support generalisations such as *iron* is a *metal* or a *diatom* is a *plankton*. Generalisation also bears a strong resemblance to other text-to-text generation tasks such as paraphrasing (Androutsopoulos and Malakasiotis, 2010), sentence compression (Jing, 2000) and sentence simplification (Shardlow, 2014). Given suitable training data, ML approaches may therefore be applied, e.g. (Knight and Marcu, 2002; Cohn and Lapata, 2009).

The most general variables are probably too generic to be of much help to a user, e.g. *concentration*, *rate*, *level*, etc. Likewise, *climate* is by far the most frequent changing variable due to the frequently occurring collocation *climate change*. In addition, variables often contain references to previously mentioned entities – anaphoric *it* being the ultimate example of this – suggesting a need for co-reference resolution (Miwa et al., 2012).

Yet another future direction is to structurally model variables as opposed to a possibly over-simplified generalisation. Similar to nominal SRL, one can define relevant arguments including frequency (e.g. annual), temporal scope (between 1958 and 2010), location, etc. The most generic variables mentioned earlier in fact provide a good basis for such modelling.

Extraction and generalisation of variables provides a basis for building systems supporting knowledge discovery. One approach is mining associations between variables frequently co-occurring in the same sentence or abstract (Jenssen et al., 2001; Hashimoto et al., 2012)) More precise results can be expected by extracting causal relations between change events (Chang and Choi, 2005; Blanco et al., 2008; Raja et al., 2013). Pairs of change events – causally or otherwise associated – obtained from different publications can be chained together, possibly in combination with domain knowledge, in order to generate new hypotheses, as pioneered in the work on literature-based knowledge discovery (Swanson, 1986; Swanson, 1988; Swanson and Smalheiser, 1997). Automatic extraction and generalisation of variables from scientific publications thus paves the way for future research on text mining in Earth science.

Acknowledgments

Financial aid from the European Commission (OCEAN-CERTAIN, FP7-ENV-2013-6.1-1; no: 603773) is gratefully acknowledged. We thank Murat Van Ardelan for sharing his knowledge of Marine science and the anonymous reviewers for their valuable comments.

References

Elias Aamot. 2014. Literature-based discovery for oceanographic climate science. In *Proceedings of the Student Research Workshop at the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 1–10, Gothenburg, Sweden, April. Association for Computational Linguistics.

Sophia Ananiadou, Douglas B. Kell, and Jun I. Tsujii. 2006. Text mining and its potential applications in systems biology. *Trends Biotechnol*, 24(12):571–579, December.

Sophia Ananiadou, Sampo Pyysalo, Jun'ichi Tsujii, and Douglas B. Kell. 2010. Event extraction for systems biology by text mining the literature. *Trends in biotechnology*, 28(7):381–390, July.

Sophia Ananiadou, Paul Thompson, Raheel Nawaz, John McNaught, and Douglas B Kell. 2014. Event-based text mining for biology and functional genomics. *Briefings in functional genomics*, page elu015.

Ion Androutsopoulos and Prodromos Malakasiotis. 2010. A survey of paraphrasing and textual entailment methods. *Journal of Artificial Intelligence Research*, 38:135–187, May.

Eduardo Blanco, Nuria Castell, and Dan I Moldovan. 2008. Causal relation extraction. In *LREC*, pages 310–313.

Du-Seong Chang and Key-Sun Choi. 2005. Causal relation extraction using cue phrase and lexical pair probabilities. In *Natural Language Processing–IJCNLP 2004*, pages 61–70. Springer.

Aaron M. Cohen and William R. Hersh. 2005. A survey of current work in biomedical text mining. *Briefings in Bioinformatics*, 6(1):57–71, March.

Kevin Bretonnel Cohen and Lawrence Hunter. 2008. Getting Started in Text Mining. *PLoS Comput Biol*, 4(1):e20+, January.

Trevor Cohn and Mirella Lapata. 2009. Sentence compression as tree transduction. *J. Artif. Int. Res.*, 34(1):637–674.

The Gene Ontology Consortium. 2001. Creating the gene ontology resource: Design and implementation. *Genome Research*, 11(8):1425–1433, August.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The PASCAL Recognising Textual Entailment challenge. *Machine Learning Challenges*, pages 177–190.

Julia A Ekstrom and Gloria T Lau. 2008. Exploratory text mining of ocean law to measure overlapping agency and jurisdictional authority. In *Proceedings of the 2008 international conference on Digital government research*, pages 53–62. Digital Government Society of North America.

Chikara Hashimoto, Kentaro Torisawa, Stijn De Saeger, Jong H. Oh, and Jun'ichi Kazama. 2012. Excitatory or inhibitory: A new semantic orientation extracts contradiction and causality from the web. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 619–630, Stroudsburg, PA, USA. Association for Computational Linguistics.

Marti A. Hearst. 1999. Untangling text data mining. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics on Computational Linguistics*, ACL '99, pages 3–10, Stroudsburg, PA, USA. Association for Computational Linguistics.

Lynette Hirschman, Alexander Yeh, Christian Blaschke, and Alfonso Valencia. 2005. Overview of BioCreAtIvE: critical assessment of information extraction for biology. *BMC Bioinformatics*, 6(Suppl 1):S1+.

T. K. Jenssen, A. Laegreid, J. Komorowski, and E. Hovig. 2001. A literature network of human genes for high-throughput analysis of gene expression. *Nat Genet*, 28:21–28.

Hongyan Jing. 2000. Sentence reduction for automatic text summarization. In *Proceedings of the sixth conference on Applied natural language processing*, pages 310–315. Association for Computational Linguistics.

J. D. Kim, T. Ohta, Y. Tateisi, and J. Tsujii. 2003. GENIA corpus—a semantically annotated corpus for bio-textmining. *Bioinformatics*, 19(suppl 1):i180–i182, July.

Kevin Knight and Daniel Marcu. 2002. Summarization beyond sentence extraction: A probabilistic approach to sentence compression. *Artificial Intelligence*, 139(1):91–107.

Ronald N. Kostoff, Raymond G. Koytcheff, and Clifford G.Y. Lau. 2007. Global nanotechnology research literature overview. *Technological Forecasting and Social Change*, 74(9):1733 – 1747. Three Special Sections: Assessment of China's and India's Science and Technology Literature Nanotechnology Policy Minding the Gap: Previewing the Potential of Breakthrough Technologies.

Martin Krallinger, Alfonso Valencia, and Lynette Hirschman. 2008. Linking genes to literature: text mining, information extraction, and retrieval applications for biology. *Genome Biology*, 9(Suppl 2):S8+, September.

Roger Levy and Galen Andrew. 2006. Tregex and Tsurgeon: tools for querying and manipulating tree data structures. In *Proceedings of the fifth international conference on Language Resources and Evaluation*, pages 2231–2234.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60.

Erwin Marsi, Pinar Oztürk, Elias Aamot, Gleb Sizov, and Murat V Ardelan. 2014. Towards text mining in climate science: Extraction of quantitative variables and their relations. In *Proceedings of the Fourth Workshop on Building and Evaluating Resources for Health and Biomedical Text Processing*, Reykjavik, Iceland.

Makoto Miwa, Paul Thompson, and Sophia Ananiadou. 2012. Boosting automatic event extraction from the literature using domain adaptation and coreference resolution. *Bioinformatics*, 28(13):1759–1765.

Roser Morante and Walter Daelemans. 2009. Learning the scope of hedge cues in biomedical texts. In *Proceedings of the Workshop on Current Trends in Biomedical Natural Language Processing*, pages 28–36. Association for Computational Linguistics.

Brendan O'Connor, David Bamman, and Noah Smith. 2011. Computational text analysis for social science: Model assumptions and complexity. In *Proceedings of the Second Workshop on Computational Social Science and the Wisdom of the Crowds (NIPS 2011)*.

Sampo Pyysalo, Filip Ginter, Juho Heimonen, Jari Bjorne, Jorma Boberg, Jouni Jarvinen, and Tapio Salakoski. 2007. BioInfer: a corpus for information extraction in the biomedical domain. *BMC Bioinformatics*, 8:50.

Sampo Pyysalo, Tomoko Ohta, Rafal Rak, Dan Sullivan, Chunhong Mao, Chunxia Wang, Bruno Sobral, Jun'ichi Tsujii, and Sophia Ananiadou. 2012. Overview of the ID, EPI and REL tasks of BioNLP shared task 2011. *BMC Bioinformatics*, 13(Suppl 11):S2+, June.

Kalpana Raja, Suresh Subramani, and Jeyakumar Natarajan. 2013. Ppinterfinder–a mining tool for extracting causal relations on human proteins from literature. *Database (Oxford)*, 2013:bas052.

Raul Rodriguez-Esteban. 2009. Biomedical Text Mining and Its Applications. *PLoS Comput Biol*, 5(12):e1000597+, December.

Sunita Sarawagi. 2008. Information extraction. *Found. Trends databases*, 1(3):261–377, March.

Matthew Shardlow. 2014. A survey of automated text simplification. *International Journal of Advanced Computer Science and Applications*, 4(1).

Matthew S. Simpson and Dina Demner-Fushman. 2012. Biomedical Text Mining: A Survey of Recent Progress. In Charu C. Aggarwal and ChengXiang Zhai, editors, *Mining Text Data*, pages 465–517. Springer US.

Don R. Swanson and Neil R. Smalheiser. 1997. An interactive system for finding complementary literatures: a stimulus to scientific discovery. *Artificial Intelligence*, 91(2):183–203, April.

Don R. Swanson. 1986. Fish oil, raynaud's syndrome, and undiscovered public knowledge. *Perspectives in biology and medicine*, 30(1):7–18.

Don R. Swanson. 1988. Migraine and magnesium: eleven neglected connections. *Perspectives in Biology and Medicine*, 31(4):526–557.

Marco A. Valenzuela-Escárcega, Gustave Hahn-Powell, Thomas Hicks, and Mihai Surdeanu. 2015. A domain-independent rule-based framework for event extraction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Assian Federation of Natural Language Processing: Software Demonstrations (ACL-IJCNLP)*.

Piek Vossen, German Rigau, Eneko Agirre, Aitor Soroa, Monica Monachini, and Roberto Bartolini. 2010. KYOTO: an open platform for mining facts. In *Proceedings of the 6th Workshop on Ontologies and Lexical Resources* , pages 1–10, Beijing, China, August. Coling 2010 Organizing Committee.

Ce Zhang, Vidhya Govindaraju, Jackson Borchardt, Tim Foltz, Christopher Ré, and Shanan Peters. 2013. GeoDeepDive: Statistical inference using familiar data-processing languages. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, SIGMOD '13, pages 993–996, New York, NY, USA. ACM.

Pierre Zweigenbaum and Dina Demner-Fushman. 2009. Advanced Literature-Mining Tools. In David Edwards, Jason Stajich, and David Hansen, editors, *Bioinformatics*, pages 347–380. Springer New York.

Pierre Zweigenbaum, Dina Demner-Fushman, Hong Yu, and Kevin B. Cohen. 2007. Frontiers of biomedical text mining: current progress. *Briefings in Bioinformatics*, 8(5):358–375, September.

Named entity recognition with document-specific KB tag gazetteers

Will Radford **Xavier Carreras** **James Henderson**
Xerox Research Centre Europe
6 chemin de Maupertuis
38240 Meylan, France
firstname.lastname@xrce.xerox.com

Abstract

We consider a novel setting for Named Entity Recognition (NER) where we have access to *document-specific knowledge base tags*. These tags consist of a canonical name from a knowledge base (KB) and entity type, but are not aligned to the text. We explore how to use KB tags to create document-specific gazetteers at inference time to improve NER. We find that this kind of supervision helps recognise organisations more than standard wide-coverage gazetteers. Moreover, augmenting document-specific gazetteers with KB information lets users specify fewer tags for the same performance, reducing cost.

1 Introduction

NER is the task of identifying names in text and assigning them a type (e.g. person, location, organisation, miscellaneous). State-of-the-art supervised approaches use models that incorporate a name's form, its linguistic context and its compatibility with known names. These models rely on large manually-annotated corpora, specifying name spans and types. These are vital for training models, but it is laborious and expensive to label every occurrence of a name in a document.

We consider a non-standard setting where, for each document, we have metadata in the form of *document-specific knowledge base tags*. A KB tag is a canonical name, that is an identifier in a KB (e.g. a Wikipedia title), and an entity type. While these tags have a correct type assigned for at least one context, they are not aligned to phrases in the text, and may not share the same form as all of their mentions (e.g. we may see the tag `United Nations` for the mention UN). We also assume that each tag matches at least one mention in the document, but do not specify where in the document the mention is.

There are many sources of KB tags, such as manual entity indexing for news stories or data extracted from personalised knowledge stores. For example, the New York Times Annotated Corpus (Sandhaus, 2008) contains more than 1.5M articles "manually tagged by library scientists with tags drawn from a normalized indexing vocabulary of people, organizations, locations and topic descriptors". Names and types are also present in large quantities of financial news stories from Bloomberg (Bradesko et al., 2015), in the form of linked names of companies and people.

Document-level tags may be quicker for annotators to apply than the usual method of marking spans in text, and are thus a cheap form of supervision. It is hard to make strong comparisons to the standard NER task, as KB tags can be considered partial, unaligned gold-standard supervision – so fully supervised models *should* perform better, the question is by how much and why.

This paper explores effective ways to use KB tags for improving NER. We use the CoNLL 2003 English NER dataset (Tjong Kim Sang and De Meulder, 2003), annotated with Wikipedia links (Hoffart et al., 2011). This allows us to simulate a set of KB tags for each document in the TRAIN, TESTA and TESTB splits of the dataset. We use a document's KB tags to build a document-specific gazetteers which we use in addition to standard features for a conditional random field (CRF) model (Lafferty et al., 2001).

We compare against wide-coverage gazetteers, which score 89.85% F-score on TESTA. Assuming access to all possible KB tags, the upper bound for KB tag models is substantially better at 92.85% F-score. KB tags help NER accuracy across all entity types, but provide relatively better supervision for organisation entities than wide-coverage gazetteers. The benefit of KB tags comes from their type information, which is required for good performance. We also examine how performance

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 512–517,
Lisbon, Portugal, 17-21 September 2015.

degrades as we use fewer KB tags, simulating the use-case where a busy knowledge worker spends less time annotating. We find that KB augmentation means we require fewer tags to reach the same performance, which reduces the cost of obtaining KB tags. We show how KB tags can be exploited as a useful complement to traditional NER supervision.

2 Background

Gazetteers have long been used to augment statistical NER models, adding general evidence of tokens used in names (Nadeau and Sekine, 2007). These are usually drawn from wide-coverage sources like Wikipedia and census lists (Ratinov and Roth, 2009) and can be incorporated into sequence models by designing binary features that indicate whether a token appears in a gazetteer entry. Features can be refined by specifying which part of an entry a token matches using tag encoding schemes such as IOB (Kazama and Torisawa, 2007). Using multiple gazetteers allows feature weights to capture different name types and sources. Given their purpose to increase coverage beyond names included in training data, gazetteers are usually large, general and static, remaining the same during training and prediction time.

Beyond their use as sources for gazetteers, the link structure in and around KBs has been used to create training data. A prominent technique is to follow links back from KB articles to documents that mention the subject of the article, heuristically labelling high-precision matches to create training data. This has been used for genetic KBs (Morgan et al., 2003; Vlachos and Gasperin, 2006), and Wikipedia (Kazama and Torisawa, 2007; Richman and Schone, 2008; Nothman et al., 2013). These works do not consider our setting where gold-standard entities are given at inference time as their goal is to generate training data.

KBs have also been used to help other natural language processing tasks such as coreference resolution (Rahman and Ng, 2011), topic modelling (Kataria et al., 2011) and named entity linking (Cucerzan, 2007; Ratinov et al., 2011). Finally, it may be that supervised data is only available in some circumstances, for example in the case of personalising NER models. Jung et al. (2015) query a user's smartphone data services to create user-specific gazetteers of personal information. The background NER model is initially trained without access to the user-specific information and later adapted on the users's smartphone.

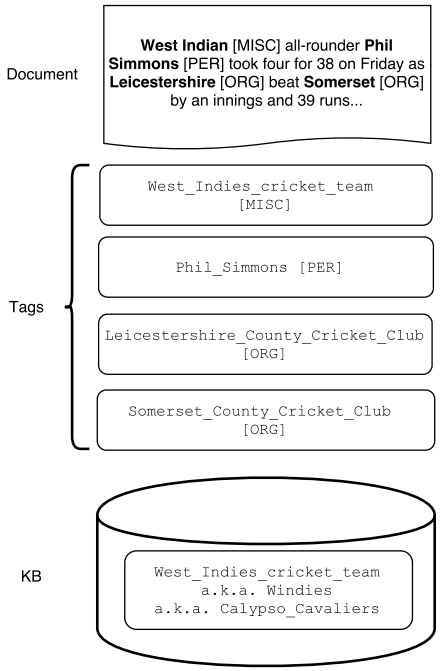


Figure 1: An entity-tagged document, KB tags with canonical name and type and KB with aliases.

3 Document-level KB tags

We incorporate information from KB tags by building document-specific gazetteers. Figure 1 shows an example with a document in which the **names** need to be recognised and typed (in square brackets). We are also given a list of KB tags, each of which is a canonical name and a type. These are linked to a KB which we use to extract aliases, in our case each canonical name is a Wikipedia article, and redirects to that article are considered aliases. Our goal is that knowing that a document mentions the entity `West Indies cricket team` can help us identify Windies or Calypso Cavaliers.

To create gazetteers from a document's KB tags, we preprocess the canonical name from each KB tag, tokenising by underscore, lowercasing and removing parenthesised suffixes (e.g. `Chris_Lewis_(cricketer)` becomes chris lewis). We use an encoding scheme to incorporate the type information from the KB tag. Inspired by Kazama and Torisawa (2007), who applied IOB encoding to gazetteers, we apply the BMEOW (a.k.a. BILOU), a scheme that also distinguishes between **b**eginning, **m**iddle, **e**nd, **o**utside

and single word positions.[1] For example, this allows us to map chris lewis to B-PER E-PER, and we can aggregate gazetteers of tokens for each encoded type, such that the gazetteer for B-PER contains chris.

Our CRF gazetteer features are calculated from an input token from the text that we wish to label. Having created a document's KB tag gazetteers, we can define binary features that are active if an input token matches (case-insensitively) with a particular gazetteer. This models both the part of the KB tag name that the token matched, and its type. The input token Chris thus activates the feature fB-PER and the token cricket would activate the fI-MISC and fI-ORG, as it matches inside entries of the two types.

4 Methodology

We define several configurations to investigate KB tags. The first four baselines either do not use KB tags, or do not integrate them into the CRF. The second four configurations use KB tag features in the CRF model.

4.1 Baselines

KB tag matching (MATCH) We find the longest full match from the document gazetteer and apply the known type. This will not match partial or non-canonical names, but should be high-precision. This is similar to the CoNLL 2003 baseline system (Tjong Kim Sang and De Meulder, 2003).

Baseline (CRF) We train a CRF model using CRFsuite (Okazaki, 2007) with a standard set of features that encode lexical context, token shape, but no external knowledge features such as gazetteers. All following configurations build on the CRF with standard features.

KB tag repair ($\text{CRF}_{+\text{REPAIR}}$) We label the text using the baseline CRF, then find the longest full match from the document gazetteer and assign the known type. When a gazetteer match overlaps with a CRF match, we prefer the gazetteer and remove the latter. Although we do not consider partial matches, this may recognise longer names that can be difficult for CRF models.

Wide-coverage gazetteers ($\text{CRF}_{+\text{WIDE}}$) This uses gazetteers distributed with the Illinois NER system (Ratinov and Roth, 2009). We encode each phrase using the BMEOW scheme described above, and use the filename of each gazetteer as its type. There are 33 gazetteers drawn from many sources with approximately 2 million entries.

4.2 Using KB tags as CRF features

KB tag names ($\text{CRF}_{+\text{NAME}}$) We generate document-specific gazetteer features, but use the same type for each entry.

KB tag names and types ($\text{CRF}_{+\text{NAME+TYPE}}$) This is equivalent to $\text{CRF}_{+\text{NAME}}$, but includes known types. Since type varies with context, this may not be correct, but is hopefully informative.

KB tag names, types and KB aliases ($\text{CRF}_{+\text{NAME+TYPE+AKA}}$) This builds on the above, but uses the KB to augment the document-specific gazetteer with known aliases of the KB tags, for example adding UN for United_Nations with the known type.

KB tag names, types, KB aliases and large gazetteers ($\text{CRF}_{+\text{NAME+TYPE+AKA+WIDE}}$) This combines all KB tag features with the wide-coverage gazetteers.

We fetch and cache KB information using a Wikipedia API client.[2] We assume the tag set of person (PER), organisation (ORG), location (LOC) and miscellaneous (MISC), and report precision, recall and F-score from the conlleval evaluation script. The median proportion of mentions in a document that are linked to the KB is 81% in TRAIN and TESTB, and 85% in TESTA. Augmenting the gazetteer with aliases produces, on average, 26 times the number of gazetteer entries than KB tags alone in TESTA, and 23 times in TESTB.

5 Results

Table 1 shows the performance of different configurations – we focus first on TESTA overall F-scores. Matching against KB tag names results in high-precision but low recall with an F-score of 55.35%, far worse than the baseline CRF at 87.68%. Despite its naïve assumptions, repairing the CRF tags using longest matches in the document gazetteer performs surprisingly well at 89.76%, just lower than using wide coverage gazetteers, with an F-score of 89.85%.

The first setting that uses KB tags as CRF features is $\text{CRF}_{+\text{NAME}}$, which includes typeless names

[1] We omit the O tag as all gazetteer tokens are inside.

[2] https://github.com/goldsmith/Wikipedia adapted to allow access to Redirect pages.

Method	TESTA							TESTB		
	P	R	F	F_{LOC}	F_{MSC}	F_{ORG}	F_{PER}	P	R	F
MATCH	94.93	39.06	55.35	76.90	22.01	29.47	59.24	94.57	37.62	53.83
CRF	88.52	86.86	87.68	90.92	85.18	81.44	90.06	81.87	80.93	81.40
+REPAIR	89.28	90.24	89.76	91.68	87.44	84.44	92.68	84.06	86.54	85.28
+WIDE	90.45	89.26	89.85	92.63	85.99	84.21	93.00	85.10	83.80	84.44
+NAME	89.96	88.64	89.29	92.21	85.57	82.71	92.89	84.26	82.72	83.48
+NAME+TYPE	93.29	92.12	92.70	95.42	88.19	88.18	95.38	89.46	87.92	88.69
+NAME+TYPE+AKA	93.42	92.29	92.85	95.63	88.35	88.27	95.54	89.90	88.74	89.32
+NAME+TYPE+AKA+WIDE	93.13	92.01	92.57	95.48	88.34	87.96	94.97	89.86	88.85	89.35

Table 1: Results for CoNLL 2003 TESTA and TESTB. We report P/R/F for all tags and per-type F-scores. Methods starting with "+" build on the standard CRF by repairing or adding features.

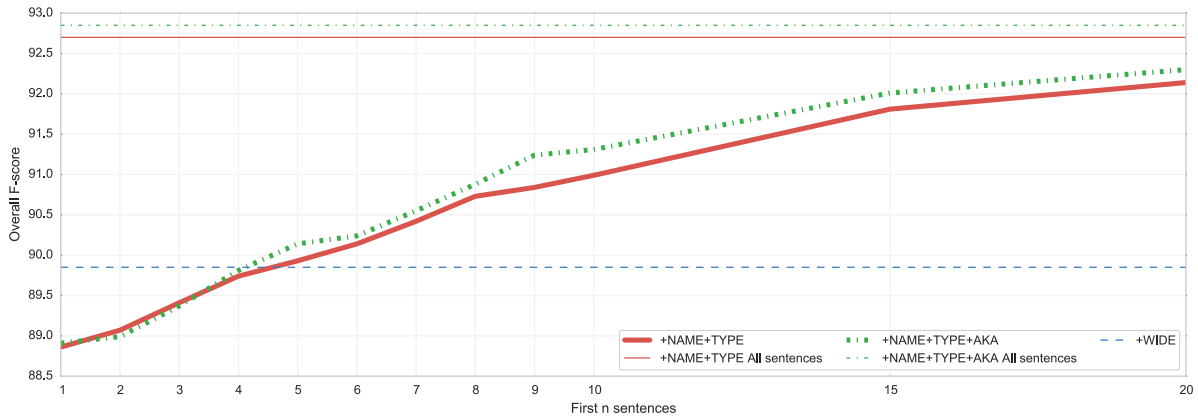


Figure 2: How many sentences should an annotator check for KB tags? TESTA results for $CRF_{+NAME+TYPE}$ and $CRF_{+NAME+TYPE+AKA}$ where KB tags are drawn from the first n sentences. This is compared to the F-score for CRF_{+WIDE} and versions of the models with access to all sentences in the document (horizontal, thin lines).

and has an F-score of 89.29%. Precision and recall are lower than wide coverage gazetteers, suggesting that, without type information, bigger gazetteers are better. Adding type features ($CRF_{+NAME+TYPE}$) results in better performance than either CRF or CRF_{+WIDE} at 92.7% F-score.[3] Augmenting the document gazetteers using aliases from the KB further improves F-score for aliases (92.85%). Adding wide-coverage gazetteers to KB tags slightly decreases F-score at 92.57%. These results indicate that type information is critical and, to confirm this, we ran experiments that used only name and alias information from KB tags. This scores 89.45% F-score on TESTA and 83.62% F-score on TESTB. While aliases help, type information is required to improve performance beyond wide coverage gazetteers.

To give some insight into why KB tag types are effective, consider the name West Indian. This appears 65 times across 11 of the 33 wide-coverage gazetteers, including those that also contain people, locations, organisations, songs. A document-specific gazetteer is able to constrain this type ambiguity, producing a cleaner signal for the model. Aliases, on the other hand, allow the model to capture non-canonical variants of a name, but this depends on type information for good NER performance.

We also examine the per-tag F-scores for TESTA to investigate whether KB tags help some types of entities more than others. Using $CRF_{+NAME+TYPE+AKA}$ we obtain around 95.5% F-score for PER and LOC entities. As with CRF_{+WIDE}, MISC entities remain hard to tag correctly. However, if we consider the percentage F-score gain by type over the CRF baseline, CRF_{+WIDE} gazetteers improve performance most for PER (+2.94%), then ORG (2.77%) entities. The top two are reversed for $CRF_{+NAME+TYPE+AKA}$, with ORG (+6.83%), then PER (+5.48%). This suggests that KB tags are particularly well-suited for helping recognise organisations names.

[3]We tried to manually map the 33 gazetteer filenames to the 4 NER types, but this reduced performance on TESTA.

We see similar trends in TESTB, except KB tags and $\text{CRF}_{+\text{WIDE}}$ are complementary. The experiments illustrate that if we are lucky enough to have KB tags, they improve NER. However, the models use all possible KB tags and should be considered an upper bound. To better model busy workers, we restrict the gazetteers to only KB tags from mentions in the first n sentences. This matches asking an annotator to only bother looking at the first n sentences. Figure 2 shows how the KB tag models perform on TESTA as we increase n. To achieve better performance than $\text{CRF}_{+\text{WIDE}}$, one should view the first 5 sentences for $\text{CRF}_{+\text{NAME+TYPE}}$. Aliases ($\text{CRF}_{+\text{NAME+TYPE+AKA}}$) reduce performance slightly when only using a few sentences, but with more than 4 sentences, aliases are consistently useful. This trend is also apparent in TESTB, showing that augmenting tags with KB information improves NER, especially when only a few tags are available.

6 Discussion and conclusion

There are several avenues to explore further. Asking annotators to specify types is not ideal and it would be better to predict them from the KB. We only use the KB to collect aliases, but we could use it to harvest related entities. Another challenge is appropriately modelling the interaction between a sentence-level task and document-level constraints. A KB tag might match a mention in one sentence and this should influence predictions there. However, its evidence should be less important elsewhere since that constraint has already been satisfied. This would improve robustness, however global constraints are hard to model in sentence-unit CRF models.

This paper presents a novel NER setting whereby we have access to some number of KB tags – canonical names and types – at training and inference time. We explore how best to use this information, finding that CRF models can indeed take advantage of this non-standard supervision. Moreover, models benefit from integration with the KB, in our case augmenting document gazetteers to maximise the benefit of KB tags.

Acknowledgements

Thank you to the anonymous reviewers whose comments helped us improve the paper.

References

Luka Bradesko, Janez Starc, and Stefano Pacifico. 2015. Isaac Bloomberg Meets Michael Bloomberg: Better Entity Disambiguation for the News. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15 Companion, pages 631–635, Republic and Canton of Geneva, Switzerland. International World Wide Web Conferences Steering Committee.

Silviu Cucerzan. 2007. Large-scale named entity disambiguation based on Wikipedia data. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 708–716, Prague, Czech Republic, June. Association for Computational Linguistics.

Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. 2011. Robust disambiguation of named entities in text. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 782–792, Edinburgh, Scotland, UK., July. Association for Computational Linguistics.

YoungHoon Jung, Karl Stratos, and Luca P. Carloni. 2015. LN-Annote: An alternative approach to information extraction from emails using locally-customized named-entity recognition. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15, pages 538–548, Republic and Canton of Geneva, Switzerland. International World Wide Web Conferences Steering Committee.

Saurabh S Kataria, Krishnan S Kumar, Rajeev R Rastogi, Prithviraj Sen, and Srinivasan H Sengamedu. 2011. Entity disambiguation with hierarchical topic models. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1037–1045. ACM.

Jun'ichi Kazama and Kentaro Torisawa. 2007. Exploiting Wikipedia as external knowledge for named entity recognition. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 698–707, Prague, Czech Republic, June. Association for Computational Linguistics.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Alex Morgan, Lynette Hirschman, Alexander Yeh, and Marc Colosimo. 2003. Gene name extraction using flybase resources. In *Proceedings of the ACL*

2003 Workshop on Natural Language Processing in Biomedicine, pages 1–8, Sapporo, Japan, July. Association for Computational Linguistics.

David Nadeau and Satoshi Sekine. 2007. A survey of named entity recognition and classification. *Linguisticae Investigationes*, 30(1):3–26, January. Publisher: John Benjamins Publishing Company.

Joel Nothman, Nicky Ringland, Will Radford, Tara Murphy, and James R. Curran. 2013. Learning multilingual named entity recognition from Wikipedia. *Artificial Intelligence*, 194(0):151 – 175. Artificial Intelligence, Wikipedia and Semi-Structured Resources.

Naoaki Okazaki. 2007. Crfsuite: a fast implementation of conditional random fields (crfs).

Altaf Rahman and Vincent Ng. 2011. Coreference resolution with world knowledge. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 814–824, Portland, Oregon, USA, June. Association for Computational Linguistics.

Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL-2009)*, pages 147–155, Boulder, Colorado, June. Association for Computational Linguistics.

Lev Ratinov, Dan Roth, Doug Downey, and Mike Anderson. 2011. Local and global algorithms for disambiguation to Wikipedia. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1375–1384, Portland, Oregon, USA, June. Association for Computational Linguistics.

Alexander E. Richman and Patrick Schone. 2008. Mining wiki resources for multilingual named entity recognition. In *Proceedings of ACL-08: HLT*, pages 1–9, Columbus, Ohio, June. Association for Computational Linguistics.

E. Sandhaus. 2008. The New York Times Annotated Corpus. *Linguistic Data Consortium, Philadelphia*, 6(12).

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In Walter Daelemans and Miles Osborne, editors, *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147.

Andreas Vlachos and Caroline Gasperin. 2006. Bootstrapping and evaluating named entity recognition in the biomedical domain. In *Proceedings of the HLT-NAACL BioNLP Workshop on Linking Natural Language and Biology*, pages 138–145, New York, New York, June. Association for Computational Linguistics.

"A Spousal Relation Begins with a Deletion of *engage* and Ends with an Addition of *divorce*": Learning State Changing Verbs from Wikipedia Revision History

Derry Tanti Wijaya
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA, 15213
dwijaya@cs.cmu.edu

Ndapandula Nakashole
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA, 15213
ndapa@cs.cmu.edu

Tom M. Mitchell
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA, 15213
tom.mitchell@cs.cmu.edu

Abstract

Learning to determine when the time-varying facts of a Knowledge Base (KB) have to be updated is a challenging task. We propose to learn state changing verbs from Wikipedia edit history. When a state-changing event, such as a marriage or death, happens to an entity, the infobox on the entity's Wikipedia page usually gets updated. At the same time, the article text may be updated with verbs either being added or deleted to reflect the changes made to the infobox. We use Wikipedia edit history to distantly supervise a method for automatically learning verbs and state changes. Additionally, our method uses constraints to effectively map verbs to infobox changes. We observe in our experiments that when state-changing verbs are added or deleted from an entity's Wikipedia page text, we can predict the entity's infobox updates with 88% precision and 76% recall. One compelling application of our verbs is to incorporate them as triggers in methods for updating existing KBs, which are currently mostly static.

1 Introduction

Extracting relational facts between entities and storing them in knowledge bases (KBs) has been a topic of active research in recent years. The resulting KBs are generally static and are not updated as the facts change (Suchanek et al., 2007; Carlson et al., 2010; Fader et al., 2011; Mitchell et al., 2015). One possible approach to updating KBs is to extract facts from dynamic Web content such as news (Nakashole and Weikum, 2012). In this paper, we propose to predict state changes caused by verbs acting on entities in text. This is different from simply applying the same text extraction pipeline, that created the original KB, to dynamic Web content.

In particular, our approach has the following advantages: (1) Consider for example the SPOUSE relation, both *marry* and *divorce* are good patterns for extracting this relation. In our work, we wish to learn that they cause different state changes. Thus, we can update the entity's fact *and* its temporal scope (Wijaya et al., 2014a). (2) Learning state changing verbs can pave the way for learning the ordering of verbs in terms of their pre- and post-conditions.

Our approach learns state changing verbs from Wikipedia revision history. In particular, we seek to establish a correspondence between infobox edits and verbs edits in the same article. The infobox of a Wikipedia article is a structured box that summarizes an entity as a set of facts (attribute-value pairs) . Our assumption is that when a state-changing event happens to an entity e.g., a marriage, its Wikipedia infobox is updated by adding a new SPOUSE value. At approximately the same time, the article text might be updated with verbs that express the event, e.g., *X is now **married** to Y*. Figure 1 is an example of an infobox of an entity changing at the same time as the article's main text to reflect a marriage event.

Wikipedia revision history of many articles can act as distant supervision data for learning the correspondence between text and infobox changes. However, these revisions are very *noisy*. Many infobox slots can be updated when a single event happens. For example, when a death happens, slots regarding birth e.g., *birthdate*, *birthplace*, may also be updated or added if they were missing before. Therefore, our method has to handle these sources of noise. We leverage logical constraints to rule out meaningless mappings between

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 518–523,
Lisbon, Portugal, 17-21 September 2015.

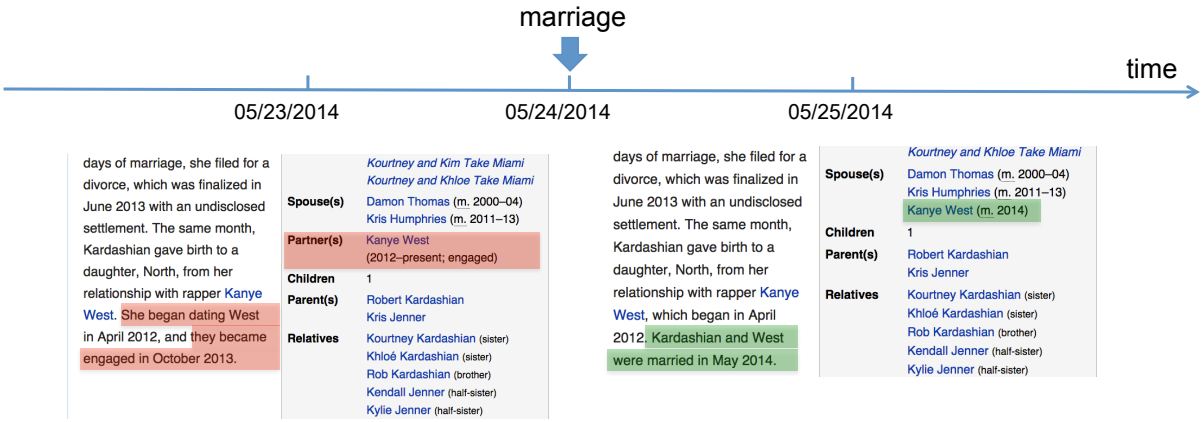


Figure 1: A snapshot of Kim Kardashian's Wikipedia revision history, highlighting text and infobox changes. In red (and green) are the differences between the page on 05/25/2014 and 05/23/2014: things that are deleted from (and added to) the page.

infobox and text changes.

In summary, our contributions are as follows: (1) we present an algorithm that uses Wikipedia edit histories as distantly labeled data to learn which verbs result in which state changes to entities, and experimentally demonstrate its success, (2) we make available this set of distantly labeled training data on our website[1], and (3) we also make available our learned mappings from verbs to state changes, as a resource for other researchers, on the same website.

2 Method

2.1 Data Construction

We construct a dataset from Wikipedia edit histories of person entities whose facts change between the year 2007 and 2012 (i.e., have at least one fact in YAGO KB (Suchanek et al., 2007) with a start or end time in this period). We obtain Wikipedia URLs of this set of entities P from YAGO and crawl their article's revision history. Given a person p, his/her Wikipedia revision history R_p has a set of ordered dates T_p on which revisions are made to his/her Wikipedia page (we consider date granularity). Each revision $r_{p,t} \in R_p$ is his/her Wikipedia page at date t where $t \in T_p$.

Each Wikipedia revision $r_{p,t}$ is a set of infobox slots $S_{p,t}$ and textual content $C_{p,t}$. Each infobox slot $s \in S_{p,t}$ is a quadruple, $\langle s_{att}, s_{value}, s_{start}, s_{end} \rangle$ containing the attribute name (non-empty), the attribute value, and the start and end time for which this attribute-value pair holds in reality.

A document $d_{p,t}$ in our data set is the *difference*[2] between any two consecutive revisions separated by more than 24 hours i.e., $d_{p,t} = r_{p,t+2} - r_{p,t}$, where $r_{p,t+2}$ is the *first* revision on date $t+2$ and $r_{p,t}$ is the *last* revision on date t (as a page can be revised many times in a day).

A document $d_{p,t}$ is therefore a set of infobox changes $\Delta S_{p,t}$ and textual changes $\Delta C_{p,t}$. Each slot change $\delta s \in \Delta S_{p,t} = \langle s_{att}, \delta s_{value}, \delta s_{start}, \delta s_{end} \rangle$ is prefixed with $+$ or $-$ to indicate whether they are added or deleted in $r_{p,t+2}$. Similarly, each text change $\delta c \in \Delta C_{p,t}$ is prefixed with $+$ or $-$ to indicate whether they are added or deleted.

For example, in Figure 1, a document $d_{kim,\ 05/23/2014} = r_{kim,05/25/2014} - r_{kim,05/23/2014}$ is a set of slot changes: $\langle$SPOUSE, $+$"Kanye West", $+$"2014", " "$\rangle$, $\langle$PARTNER, $-$"Kanye West", $-$"2012-present; engaged", " "$\rangle$ and a set of text changes: $+$"Kardashian and West were married in May 2014", $-$"She began dating West", $-$"they became engaged in October 2013".

For each $d_{p,t}$, we use $\Delta S_{p,t}$ to label the document and $\Delta C_{p,t}$ to extract features for the document. We label $d_{p,t}$ that has a new value or start time added to its infobox: $\langle s_{att}, +\delta s_{value}, *, * \rangle \in \Delta S_{p,t}$ or $\langle s_{att}, *, +\delta s_{start}, * \rangle \in \Delta S_{p,t}$ with the label *begin-*s_{att} and label $d_{p,t}$ that has a new end time added to its infobox: $\langle s_{att}, *, *, +\delta s_{end} \rangle \in \Delta S_{p,t}$ with the label *end-*s_{att}.

The label represents the state change that

[1] http://www.cs.cmu.edu/ dwijaya/postcondition.html

[2] a HTML document obtained by "compare selected revisions" functionality in Wikipedia

happens in $d_{p,t}$. For example, in Figure 1, $d_{kim,\ 05/23/2014}$ is labeled with *begin-spouse*.

The revision history dataset that we make available for future research consists of all documents $d_{p,t}$, labeled and unlabeled, $\forall t \in T_p,\ t \in [01/01/2007,\ 12/31/2012]$, and $\forall p \in P$; a total of 288,184 documents from revision histories of 16,909 Wikipedia entities. Using our labeling process, we find that out of 288,184 documents, only 41,139 have labels (i.e., have their infobox updated with new values/start/end time). The distribution of labels in the dataset is skewed towards birth and death events as these are life events that happen to almost all person entities in Wikipedia. The distribution of labels in the dataset that we release can be seen in Figure 2. We show only labels that we evaluate in our task.

For our task of learning state changing verbs from this revision history dataset, for each labeled $d_{p,t}$, we extract as features, verbs (or verbs+prepositions) $v \in \Delta C_{p,t}$ of which its subject (or object) matches the Wikipedia entity p *and* its object (or subject resp.) matches an infobox value, start or end time: $(v_{subject}, v_{object}) = (_{arg1}, _{arg2})$ or $(v_{subject}, v_{object}) = (_{arg2}, _{arg1})$, where $_{arg1}= p$ and $\langle s_{att},_{arg2}, *, * \rangle$ or $\langle s_{att}, *,_{arg2}, * \rangle$ or $\langle s_{att}, *, *,_{arg2} \rangle \in \Delta S_{p,t}$. We use Stanford CoreNLP (Manning et al., 2014) to dependency parse sentences and extract the subjects and objects of verbs. We find that 27,044 out of the 41,139 *labeled* documents contain verb edits, but only 4,735 contain verb edits with two arguments, where one argument matches the entity and another matches the value of the infobox change. We use the latter for our task, to improve the chance that the verb edits used as features are related to the infobox change.

2.2 Model

We use a Maximum Entropy (MAXENT) classifier[3] given a set of training data = $\{(\mathbf{v}_{d_\ell},\ y)\}$ where $\mathbf{v}_{d_\ell} = (v_1,\ v_2,\ ...\ v_{|V|}) \in R^{|V|}$ is the $|V|$-dimensional representation of a labeled document d_ℓ where V is the set of all verbs in our training data, and y is the label of d_ℓ as defined in 2.1.

These training documents are used to estimate a set of weight vectors $\mathbf{w} = \{\mathbf{w}_1,\ \mathbf{w}_2,\ ...\ \mathbf{w}_{|Y|}\}$, $\mathbf{w}_y \in R^{|V|}$, one for each label $y \in Y$, the set of all labels in our training data. The classifier can then be applied to classify an unlabeled document d_u using:

$$p(y|\mathbf{v}_{d_u}) = \frac{\exp(\mathbf{w}_y \cdot \mathbf{v}_{d_u})}{\sum_{y'} \exp(\mathbf{w}_{y'} \cdot \mathbf{v}_{d_u})} \quad (1)$$

2.3 Feature Selection using Constraints

While feature weights from the MAXENT model allow us to identify verbs that are good features for predicting a particular state change label, our distantly supervised training data is inherently noisy. Changes to multiple infoboxes can happen within our revision. We therefore utilize constraints among state changes to select consistent verb features for each type of state change.

We use two types of constraints: (1) mutual exclusion (*Mutex*) which indicate that mutex state changes do not happen at the same time e.g., update on *birthdate* should not *typically* happen with update on *deathcause*. Hence, their state changing verbs should be different. (2) Simultaneous (*Sim*) constraints which indicate that simultaneous state changes should *typically* happen at the same time e.g., update on *birthdate* should *typically* happen with other birth-related updates such as *birthplace*, *birthname*, etc. We manually specified these two types of constraints to all pairs infoboxes where they apply. We have 10 mutex constraints and 23 simultaneously updated constraints. The full list of constraints can be found in our website.

Given a set of constraints, a set of labels Y, and a set of base verbs[4] B in our training data, we solve a Mixed-Integer Program (MIP) for each base verb $b \in B$ to estimate whether b should be a feature for state change $y \in Y$.

We obtain label membership probabilities $\{P(y|b) = count(y,b)/\sum_{y'} count(y',b)\}$ from our training data. The MIP takes the scores $P(y|b)$ and constraints as input and produces a bit vector of labels $\mathbf{a}_b$ as output, each bit $a_b^y \in \{0,\ 1\}$ represents whether or not b should be a feature for y.

The MIP formulation for a base verb b is presented by Equation 2. For each b, this method tries to maximize the sum of scores of selected labels, after penalizing for violation of label constraints. Let $\zeta_{y,y'}$ be slack variables for *Sim* constraints, and $\xi_{y,y'}$ be slack variables for *Mutex* constraints.

[3]We use MALLET implementation of MAXENT: http://mallet.cs.umass.edu/

[4]The verb root or base form of a verb (after removing preposition)

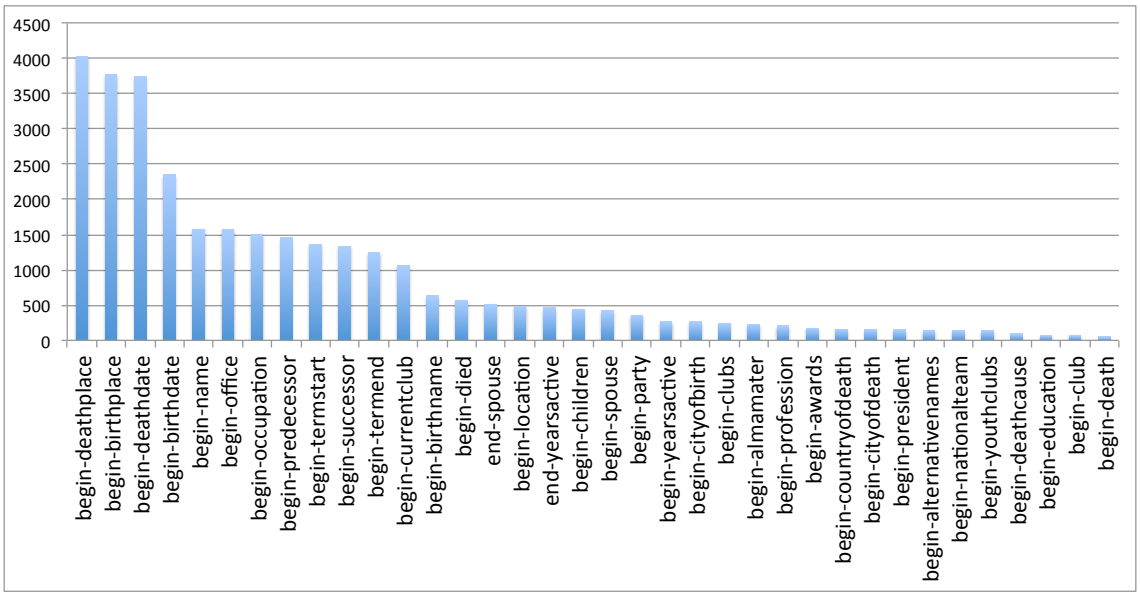


Figure 2: Distribution of labels we evaluate in our task in the revision history dataset.

Solving MIP per base verb is fast; we reduce the number of labels considered per base verb i.e., we only consider a label y to be a candidate for b if $\exists\, v_i \in V$ s.t. $w_y^i > 0$ and b = base form of v_i.

After we output $\mathbf{a}_b$ for each b, we select features for each label. We only select a verb v_i to be a feature for y if the learned weight $w_y^i > 0$ and $a_b^y = 1$, where b = the base form of v_i. Essentially for each label, we select verb features that have positive weights and are consistent for the label.

$$\begin{aligned} \underset{\mathbf{a}_b,\, \zeta_{y,y'},\, \xi_{y,y'}}{\text{maximize}} \quad & \Big(\sum_y a_b^y * P(y|b) - \sum_{\langle y,y' \rangle \in Sim} \zeta_{y,y'} - \\ & \sum_{\langle y,y' \rangle \in Mutex} \xi_{y,y'} \Big) \\ \text{subject to} \quad & (a_b^y - a_b^{y'})^2 \leq \zeta_{y,y'}, \quad \forall \langle y, y' \rangle \in Sim \\ & a_b^y + a_b^{y'} \leq 1 + \xi_{y,y'}, \quad \forall \langle y, y' \rangle \in Mutex \\ & \zeta_{y,y'},\, \xi_{y,y'} \geq 0,\; a_b^y \in \{0, 1\}, \quad \forall y, y' \end{aligned} \tag{2}$$

3 Experiments

We use 90% of our labeled documents that have verb edits as features (section 2.1) as training data and test on the remaining 10%. Since revision history data is noisy, we manually go through our test data to discard documents that have incorrect infobox labels by looking the text that changed. The task is to predict for each document (revision), the label (infobox slot change) of the document given its verbs features. We compute precision, recall, and F1 values of our predictions and compare the values before and after feature selection (Fig. 3).

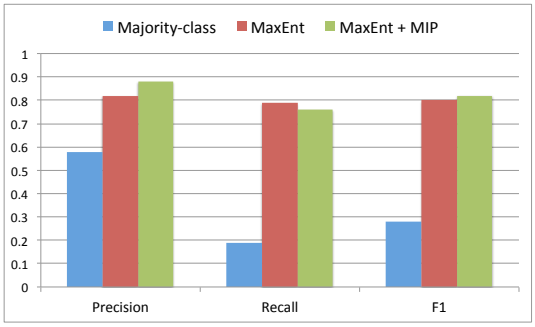


Figure 3: Results of predicting state change labels (infobox types) using verb features.

To the best of our knowledge, the task to learn state-changing verbs in terms of states defined in existing knowledge bases and learning it from Wikipedia edit histories is novel. There is no previous approach that can be used as baseline; therefore we have compared our structured prediction using MIP and MaxEnt with a majority class baseline. Both our approaches (MaxEnt and MaxEnt + MIP) perform better than the majority class baseline (Figure 3).

We observe the value of doing feature selection by asserting constraints in an MIP formulation. Feature selection improves precision; resulting in a better F1. By asserting constraints, some of the inconsistent verb features for the la-

Label	Verb
begin-deathdate	+(arg1) die on (arg2), +(arg1) die (arg2), +(arg1) pass on (arg2)
begin-birthplace	+(arg1) be born in (arg2), +(arg1) bear in (arg2), +(arg1) be born at (arg2)
begin-predecessor	+(arg1) succeed (arg2), +(arg1) replace (arg2), +(arg1) join cabinet as (arg2), +(arg1) join as (arg2)
begin-successor	+(arg1) lose **seat** to (arg2), +(arg1) resign on (arg2), +(arg1) resign from post on (arg2)
begin-termstart	+(arg1) be appointed on (arg2), +(arg1) serve from (arg2), +(arg1) be elected on (arg2)
begin-spouse	+(arg1) marry on (arg2), +(arg1) marry (arg2), +(arg1) be married on (arg2), -(arg1) be engaged to (arg2)
end-spouse	+(arg1) file **for divorce** in (arg2), +(arg1) die on (arg2), +(arg1) divorce in (arg2)
begin-youthclubs	+(arg1) start career with (arg2), +(arg1) begin **career** with (arg2), +(arg1) start with (arg2)

Table 1: Comparison of verb phrases learned before and after feature selection for various labels (infobox types). The texts in bold are (preposition+) noun that occur most frequently with the ⟨verb phrase, label⟩ pair in the training data.

bels were removed. For example, before feature selection, the verbs: "marry", and "be married to" were high-weighted features for both *begin-spouse* and *end-spouse*. After asserting constraints that *begin-spouse* is mutex with *end-spouse*, these verbs (whose base form is "marry") are filtered out from the features of *end-spouse*. We show some of the learned verb features (after feature selection) for some of the labels in (Table 1). In average, we have about 18 verbs per infobox state change in our state changing verb resource that we make available for future research.

4 Related Work

Learning from Wikipedia Revision History. Wikipedia edit history has been exploited in a number of problems. A popular task in this regard is that of Wikipedia edit history categorization (Daxenberger and Gurevych, 2013). This task involves characterizing a given edit instance as one of many possible categories such as spelling error correction, paraphrasing, vandalism, and textual entailment (Nelken and Yamangil, 2008; Cahill et al., 2013; Zanzotto and Pennacchiotti, 2010; Recasens et al., 2013). Prior methods target various tasks different from ours.

Learning State Changing Verbs. Very few works have studied the problem of learning state changing verbs. (Hosseini et al., 2014) learned state changing verbs in the context of solving arithmetic word problems. They learned the effect of words such as *add, subtract* on the current state. The VerbOcean resource was automatically generated from the Web (Chklovski and Pantel, 2004). The authors studied the problem of fine-grained semantic relationships between verbs. They learn relations such as if someone has bought an item, they may sell it at a later time. This then involves capturing empirical regularities such as "X buys Y" happens before "X sells Y". Unlike the work we present here, the methods of (Chklovski and Pantel, 2004; Hosseini et al., 2014) do not make a connection to KB relations such as Wikipedia infoboxes. In a vision paper, (Wijaya et al., 2014b) give high level descriptions of a number of possible methods for learning state changing methods. They did not implement any of them.

5 Conclusion

In this paper we presented a method that learns state changing verb phrases from Wikipedia revision history. We first constructed and curated a novel dataset from Wikipedia revision history that is tailored to our task. We showed that this dataset is useful for learning verb phrase features that are effective for predicting state changes in the knowledge base (KB), where we considered the KB to be infoboxes and their values. As future work we wish to explore the usefulness of our verb resource to other KBs to improve KB freshness. This is important because existing KBs are mostly static. We wish to also explore the application of the learned verb resource to domains other than Wikipedia infobox and text e.g., for predicting state changes in the knowledge base from news text.

In this paper, we learned post-conditions of verbs: state changes that occur when an event expressed by a verb happens. As future work we would also explore the feasibility of learning preconditions of verbs from Wikipedia revisions. Additionally, most Wikipedia revisions only have text changes without the associated infobox change. Therefore, another line of future work is to also learn from these unlabeled documents.

Acknowledgments

We thank members of the NELL team at CMU for their helpful comments. This research was supported by DARPA under contract number FA87501320005.

References

Aoife Cahill, Nitin Madnani, Joel Tetreault, and Diane Napolitano. 2013. Robust systems for preposition error correction using wikipedia revisions. In *In Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R Hruschka Jr, and Tom M Mitchell. 2010. Toward an architecture for never-ending language learning. In *AAAI*, volume 5, page 3.

Timothy Chklovski and Patrick Pantel. 2004. Verbocean: Mining the web for fine-grained semantic verb relations. In *Proceedings of EMNLP 2004*, pages 33–40.

Johannes Daxenberger and Iryna Gurevych. 2013. Automatically classifying edit categories in wikipedia revisions. In *EMNLP*, pages 578–589.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1535–1545. Association for Computational Linguistics.

Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. 2014. Learning to solve arithmetic word problems with verb categorization. In *EMNLP*, pages 523–533. ACL.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60.

Tom M. Mitchell, William W. Cohen, Estevam R. Hruschka Jr., Partha Pratim Talukdar, Justin Betteridge, Andrew Carlson, Bhavana Dalvi Mishra, Matthew Gardner, Bryan Kisiel, Jayant Krishnamurthy, Ni Lao, Kathryn Mazaitis, Thahir Mohamed, Ndapandula Nakashole, Emmanouil Antonios Platanios, Alan Ritter, Mehdi Samadi, Burr Settles, Richard C. Wang, Derry Tanti Wijaya, Abhinav Gupta, Xinlei Chen, Abulhair Saparov, Malcolm Greaves, and Joel Welling. 2015. Never-ending learning. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA.*, pages 2302–2310.

Ndapandula Nakashole and Gerhard Weikum. 2012. Real-time population of knowledge bases: opportunities and challenges. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*, pages 41–45. Association for Computational Linguistics.

Rani Nelken and Elif Yamangil. 2008. Mining wikipedias article revision history for training computational linguistics algorithms.

Marta Recasens, Cristian Danescu-Niculescu-Mizil, and Dan Jurafsky. 2013. Linguistic models for analyzing and detecting biased language. In *ACL (1)*, pages 1650–1659.

Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*, pages 697–706. ACM.

Derry Wijaya, Ndapa Nakashole, and Tom Mitchell. 2014a. CTPs: Contextual temporal profiles for time scoping facts via entity state change detection. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Derry Tanti Wijaya, Ndapandula Nakashole, and Tom M Mitchell. 2014b. Mining and organizing a resource of state-changing verbs. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*.

Fabio Massimo Zanzotto and Marco Pennacchiotti. 2010. Expanding textual entailment corpora from wikipedia using co-training.

Improving Distant Supervision for Information Extraction Using Label Propagation Through Lists

Lidong Bing[§] **Sneha Chaudhari**[§] **Richard C. Wang**[♮] **William W. Cohen**[§]
[§]Carnegie Mellon University, Pittsburgh, PA 15213
[♮]US Development Center, Baidu USA, Sunnyvale, CA 94089
[§]{lbing@cs, sschaudh@andrew, wcohen@cs}.cmu.edu
[♮]richardwang@baidu.com

Abstract

Because of polysemy, distant labeling for information extraction leads to noisy training data. We describe a procedure for reducing this noise by using label propagation on a graph in which the nodes are entity mentions, and mentions are coupled when they occur in coordinate list structures. We show that this labeling approach leads to good performance even when off-the-shelf classifiers are used on the distantly-labeled data.

1 Introduction

In distantly-supervised information extraction (IE), a knowledge base (KB) of relation or concept instances is used to train an IE system. For instance, a set of facts like *adverseEffectOf(meloxicam, stomachBleeding)*, *interactsWith(meloxicam, ibuprofen)*, might be used to train an IE system that extracts these relations from documents. In distant supervision, instances are first matched against a corpus, and the matching sentences are then used to generate training data consisting of labeled entity mentions. For instance, matching the KB above might lead to labeling passage 1 from Table 1 as support for the fact *adverseEffectOf(, stomachBleeding)*.

A weakness of distant supervision is that it produces noisy training data: for instance, matching the adverse effect *weakness* might lead to incorrectly-labeled mention examples. Distant supervision is often coupled with learning methods that allow for this sort of noise by introducing latent variables for each entity mention (e.g., (Hoffmann et al., 2011; Riedel et al., 2010; Surdeanu et al., 2012)); by carefully selecting the entity mentions from contexts likely to include specific KB facts (Wu and Weld, 2010); by careful filtering of the KB strings used as seeds (Movshovitz-Attias and Cohen, 2012); or by making use of named-entity linking methods and co-reference to improve the matching phase of distant learning (Koch et al., 2014).

1. "Avoid drinking alcohol. It may increase your risk of stomach bleeding."
2. "Get emergency medical help if you have chest pain, weakness, shortness of breath, slurred speech, or problems with vision or balance."

Table 1: Passages from a page discussing the drug meloxicam.

Here we explore an alternative approach of ***D***istant ***IE*** using coordinate-term ***L***ists (DIEL) based on detection of *lists* in text, such as the one illustrated in passage 2 in Table 1. Since list items are usually of the same type, the unambiguous mention *chest pain* here disambiguates the mention *weakness*. Label propagation methods (Zhu et al., 2003; Lin and Cohen, 2010) can be used to exploit this intuition, by propagating the low-confidence labels associated with distance supervision through an appropriate graph.

Here we describe a pipelined system which (1) identifies lists of semantically-related items using lexico-syntactic patterns (2) uses distant supervision, in combination with a label-propagation method, to find entity mentions that can be confidently labeled and (3) from this data, uses ordinary classifier learners to classify entity mentions by their semantic type. We show that this approach outperforms a naive distant-supervision approach.

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 524–529,
Lisbon, Portugal, 17-21 September 2015.

2 DIEL: *D*istant *IE* Using Coordinate *L*ists

2.1 Corpus and KB

We consider extending the coverage of Freebase in the medical domain, which is currently fairly limited: e.g., a Freebase snapshot from April 2014 has (after filtering noise with simple rules such as length greater than 60 characters and containing comma) only 4,605 disease instances and 4,383 drug instances, whereas `dailymed.nlm.nih.gov` contains data on over 74k drugs, and `malacards.org` lists nearly 10k diseases. We use a corpus downloaded from `dailymed.nlm.nih.gov` which contains 28,590 XML documents, each of which describes a drug that can be legally prescribed in the United States. We focus here on extracting instances of four semantic types, without explicitly extracting relationships between them.

We used the GDep parser (Sagae and Tsujii, 2007), a dependency parser trained on the GENIA Treebank, to parse this corpus. We used a simple POS-tag based noun-phrase (NP) chunker, and extract a list for each coordinating conjunction that modifies a nominal. For each NP we extract features (described below); and for each identified coordinate-term list, we extract its items, and a similar feature set describing the list.

The extracted lists and their items, as well as entity mentions and their corresponding NPs, can be viewed as a bipartite graph, where one set of vertices are identifiers for the lists and entity mentions, and the other set of vertices are the strings that occur as items of those lists, or as NPs of those mentions. Note that list items are also NPs. A mention can be regarded as a singleton list that contains only one item, and a list can be regarded as a complexus mention that contains a few mentions. If an item is contained by a list, an edge between the item vertex and the list vertex is included in the graph. An example bipartite graph is given in Figure 1, in which there are nine symptoms from three lists and three mentions. Some symptoms are common, such as vomiting, while some others are not, such as epigastric pain.

2.2 Label Propagation

It seems intuitive to assume that if two items co-occur in a coordinate-term list, they are very likely to have the same type, so it seems plausible to use label propagation on this graph to propagate types from NPs with known types (e.g., that match entities in the KB) to lists, and then from lists to NPs, across this graph.

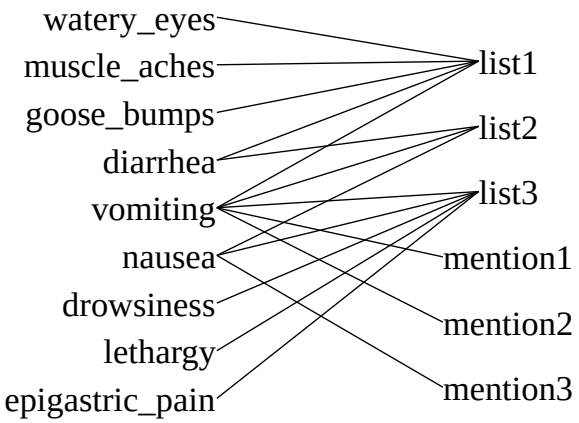


Figure 1: A bipartite graph example.

This can be viewed as semi-supervised learning (SSL) of the NPs that may denote a type (e.g., diseases or adverse effects). We adopt an existing multi-class label propagation method, namely, MultiRankWalk (MRW) (Lin and Cohen, 2010), to handle our task, which is a graph-based SSL related to personalized PageRank (PPR) (Haveliwala et al., 2003) (aka random walk with restart (Tong et al., 2006)). MRW can be described as simply computing one personalized PageRank vector for each class, where each vector is computed using a personalization vector that is uniform over the seeds, and finally assigning to each node the class associated with its highest-scoring vector. MRW's final scores depend on centrality of nodes, as well as proximity to the seeds, and in this respect MRW differs from other label propagation methods (e.g., (Zhu et al., 2003)): in particular, it will not assign identical scores to all seed examples. The MRW implementation we use is based on ProPPR (Wang et al., 2013).

2.3 Classification

One could imagine using the output of MRW to extend a KB directly. However, the process described above cannot be used conveniently to label new documents as they appear. Since this is also frequently a goal, we use the MRW output to train a classifier, which can be then used to classify the entity mentions (singleton lists) and coordinate lists in any new document.

We use the same feature generator for both entity mentions and lists. Shallow features include: tokens in the NPs, and character prefixes/suffixes of these tokens; tokens from the sentence contain-

Run	1	2	3	4	5	6	7	8	9	10	Avg.
DIEL	0.418	0.390	0.404	0.420	0.400	0.397	0.404	0.403	0.409	0.404	0.405
DS-baseline	0.394	0.342	0.400	0.373	0.342	0.297	0.326	0.332	0.330	0.352	0.349
LP-baseline	0.167	0.167	0.162	0.155	0.150	0.159	0.157	0.165	0.167	0.167	0.162
Upper bound	0.617	0.616	0.615	0.619	0.620	0.614	0.618	0.617	0.615	0.618	0.617

Table 2: Recall on the held-out set.

ing the NP; and tokens and bigrams from a window around the NPs. From the dependence parse, we also find the verb which is the closest ancestor of the head of the NP, all modifiers of this verb, and the path to this verb. For a list, the dependency features are computed relative to the head of the list. We used an SVM classifier (Chang and Lin, 2001) and discard singleton features, and also the frequent 5% of all features (as a stop-wording variant). We train a binary classifier on the top N lists (including entity mentions and coordinate lists) of each type, as scored by MRW. A linear kernel and defaults for all other parameters are used. If a new list or mention is not classified as positive by all binary classifiers, it is predicted as "other".

3 Experimental Results

3.1 Results of Recovering KB

In this experiment, we examine the capability of our approach in recovering KB type instances. The targeted types are diseases, symptoms treated and adverse effects (symptom for short), drugs, and drug ingredients.

3.1.1 Baselines

We implemented a distant-supervision-based baseline (DS-baseline). It attempts to classify each NP in the input corpus into one of the four types or "other" with the training seeds as distance supervision. Each sentence is processed with the same reprocessing pipeline to detect NPs. Then, these NPs are labeled with the training seeds. The features are defined and extracted in the same way as we did for DIEL, and four binary classifiers are trained with the same method. Another baseline is developed with the output of MRW LP (LP-baseline) that contains labeled lists and mentions. Specifically, the labeled coordinate lists are broken into items each of which has the list class, and evaluation is conducted with these items together with the labeled mentions as positive predictions.

3.1.2 Settings

We extracted the seeds of these types from Freebase, and got 4,605, 1,244, 4,383, and 4,066 instances, respectively. The seeds are split into development set and held-out evaluation set. The development set is further split into a training set and a validating set in the ratio of 4:1. The validating set will be used in the next subsection to validate different parameter settings, and the training set is used in this experiment as MRW seeds and the distant supervision of DS-baseline.

For getting the development set, the polysemous instances (i.e., "headache", belonging to multiple classes: disease and symptom) are discarded since such instances will bring in ambiguity to the training examples of DS-baseline and MRW LP. After that, we randomly take half of the single-type instances as development set, and the remaining single-type instances together with the polysemous instances are used as held-out set. We report the performance of 10 runs, and each run has its own randomly generated training set (containing 1,980 diseases, 310 symptoms, 1,066 drugs, and 911 ingredients on average) and held-out set (containing 2,130 diseases, 857 symptoms, 3,051 drugs, and 2,927 ingredients on average). For DS-baseline, 35,689 training examples are collected on average for each run. For evaluation metric, we calculate recall on the instances in the held-out set.[1]

3.1.3 Results

DIEL can classify both NPs and coordinate lists, and the lists are also broken into items for recall calculation. The training examples of DIEL are prepared with the top 20,000 lists of each type, as scored by MRW with the training seeds. After removing the multiple-class ones, we obtained, on average, 58,614 training examples for each run.

The results are given in Table 2. DIEL outperforms the baselines in all runs. It shows that

[1]Because the outputs contain many correct instances that are not in Freebase, it is not suitable to calculate precision.

		2.5%		7.5%		12.5%		25%		50%		75%		100%	
	N	P	R	P	R	P	R	P	R	P	R	P	R	P	R
DIEL	1,000	0.581	0.119	0.680	0.150	0.706	0.154	0.754	0.216	0.768	0.277	0.749	0.308	0.769	0.331
	2,000	0.587	0.168	0.675	0.183	0.705	0.192	0.746	0.245	0.760	0.292	0.747	0.317	0.777	0.347
	4,000	0.551	0.205	0.661	0.225	0.682	0.254	0.726	0.295	0.763	0.334	0.763	0.356	0.766	0.375
	6,000	0.500	0.205	0.636	0.244	0.650	0.281	0.680	0.314	0.757	0.362	0.758	0.380	0.768	0.392
	8,000	0.518	0.219	0.615	0.251	0.641	0.300	0.662	0.331	0.727	0.393	0.764	0.407	0.762	0.409
	10,000	0.520	0.219	0.610	0.260	0.643	0.309	0.664	0.349	0.708	0.406	0.753	0.436	0.765	0.429
	20,000	0.520	0.216	0.590	0.277	0.634	0.335	0.661	0.375	0.688	0.435	0.723	0.473	0.746	0.494
	30,000	0.520	0.216	0.584	0.284	0.624	0.341	0.649	0.392	0.688	0.454	0.717	0.486	0.737	0.496
LP-baseline		0.442	0.024	0.604	0.074	0.673	0.108	0.722	0.134	0.797	0.173	0.838	0.187	0.855	0.212
DS-	Training NP#	678		2,442		5,983		9,831		18,230		27,549		35,689	
baseline	Performance	0.317	0.139	0.384	0.161	0.643	0.244	0.714	0.252	0.747	0.322	0.767	0.347	0.742	0.351

Table 4: The classification performance of the three methods with different parameters.

Type	Disease	Drug	Ingredient	Symptom
DIEL	0.369	**0.312**	**0.489**	**0.555**
DS-baseline	**0.375**	0.300	0.408	0.224
LP-baseline	0.185	0.101	0.117	0.453
Upper bound	0.445	0.652	0.700	0.697

Table 3: Recall on individual types.

our result is consistently better. The reason is twofold. First, DIEL can avoid the effect of noisy training data by disambiguation with the coordinate relation in the list, so that the training examples are of high quality. Second, with label propagation, we have a larger number of training examples, which helps the recall. Compared with DS-baseline, DIEL's performance is more stable in different runs. It is because DS-baseline suffers from the noisy training data and training seed sets of different runs may bring in different levels of noisy data. Thus, its run 3 achieves 0.400, while run 6 only achieves 0.297. We also examined the upper bound recall that a system can achieve on our corpus. The results are given in the last row of Table 2. On average, the best performance of a system can achieve is 0.617.

The results for individual types are given in Table 3. DIEL and DS-baseline achieve similar results for disease and drug. Especially, both systems cover more than 80% of the held-out disease instances that exist in the corpus. DS-baseline performs poorly for symptom. The reason is that symptom instances are more ambiguous than other types, and they lead to more incorrectly-labeled mention examples. LP-baseline achieves an encouraging recall for symptom, which shows that coordinate lists are very helpful for disambiguating those symptom mentions.

3.2 Classification Results and Parameters

We present another experiment to examine the precision of the systems, and investigate the effect of training size and top N numbers on the results.

3.2.1 Setting

The evaluation data is generated with the validating set of each run. Specifically, for DIEL and LP-baseline, the evaluation data is prepared with the top 500 lists (singleton and coordinate lists) of each type, as scored by MRW with the validating instances as seeds. 2,000 testing examples are collected in total since no multiple-class ones are found. Manual checking on 200 random examples shows that 99% of them are correct. The systems are evaluated by their performance for classifying these example lists. DIEL predicts the class of a list with its feature vector, while LP-baseline determines the class of a list by checking its predicted class in the result of MRW with the corresponding training instances as seeds. For DS-baseline, the testing NP examples are obtained by distant labeling with validating instances, and on average 8,270 examples are collected for each run.

3.2.2 Results

The precision and recall are given in Table 4. For each system, different portions of the training set are used to train the system, as shown in the first row. For DIEL, the top N number varies for generating training examples, as shown in the second column. F1 values of DIEL with different settings are given in Figure 2, and F1 comparison of three systems is given in Figure 3. All results are the average of 10 runs. Each run has its own randomly generated development set, which is split into training set and validating set.

In general, for all systems, larger number of training seeds leads to better performance. For

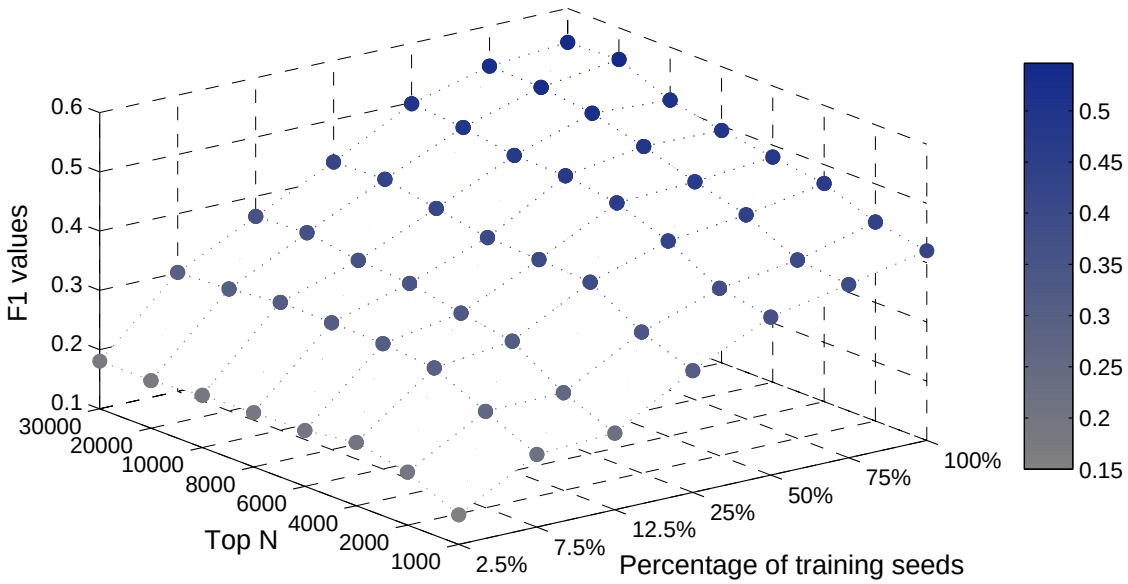


Figure 2: F1 values of DIEL under different settings.

DIEL, smaller N values achieve higher precision, but lower recall. For smaller seed numbers, the precision value is more sensitive to N. This is because the quality of training examples drops faster compared with that from larger seed numbers. For larger seeds numbers, the recall values are improved more significantly when the N value is larger. The reported results of DIEL in the previous experiment are obtained with top 20,000 examples from 100% seeds as training data, since this setting achieves the highest F1 value as shown in Figure 2.

For the DS-baseline, the number of training NPs obtained with different portions of the training set is given in the penultimate row. The recall values of this baseline are low. The reason is that it only uses the training examples that are distantly labeled with training seeds, thus, the trained classifier may not have good generalization on the testing examples labeled with validating seeds. In addition, its performance is more sensitive on the amount of training data. When the percentage is lower than 25%, its precision and recall drop significantly. Its F1 values are 0.381, 0.399 and 0.402 for 50%, 75%, and 100%, respectively. LP-baseline achieves the highest precision when using all training instances. It shows that MRW does label the testing lists very accurately in condition that the lists are traversed in the propagation with the training instances as seeds. However, its recall is much lower than DIEL. It is because, with the

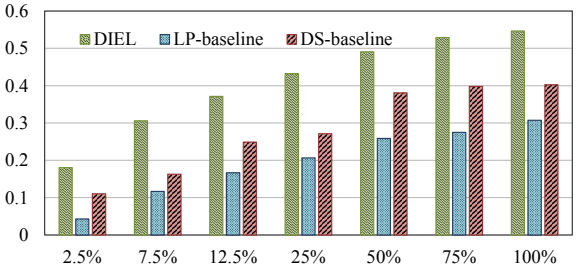


Figure 3: F1 comparison under different portions of the training seeds. For DIEL, top N = 20,000.

training seeds, MRW cannot effectively walk to testing lists that are generated with the validating set, having no intersection with the training set.

4 Conclusions

We explored an alternative approach to distant supervision, based on detection of lists in text, to overcome the weakness of distant supervision resulted by noisy training data. It uses distant supervision and label propagation to find mentions that can be confidently labeled, and uses them to train classifiers to label more entity mentions. The experimental results show that this approach consistently and significantly outperforms a naive distant-supervision approach.

Acknowledgments

This work was funded by a grant from Baidu USA and by the NSF under research grant IIS-1250956.

References

Chih-Chung Chang and Chih-Jen Lin, 2001. *LIBSVM: a library for support vector machines*. Software available at http://www.csie.ntu.edu.tw/˜cjlin/libsvm.

Taher Haveliwala, Sepandar Kamvar, Ar Kamvar, and Glen Jeh. 2003. An analytical comparison of approaches to personalizing pagerank. Technical report, Stanford University.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 541–550. Association for Computational Linguistics.

Mitchell Koch, John Gilmer, Stephen Soderland, and Daniel S. Weld. 2014. Type-aware distantly supervised relation extraction with linked arguments. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1891–1901.

Frank Lin and William W. Cohen. 2010. Semi-supervised classification of network data using very few labels. In Nasrullah Memon and Reda Alhajj, editors, *ASONAM*, pages 192–199. IEEE Computer Society.

Dana Movshovitz-Attias and William W. Cohen. 2012. Bootstrapping biomedical ontologies for scientific text using nell. In *Proceedings of the 2012 Workshop on Biomedical Natural Language Processing*, BioNLP '12, pages 11–19, Stroudsburg, PA, USA. Association for Computational Linguistics.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *Machine Learning and Knowledge Discovery in Databases*, pages 148–163. Springer.

K. Sagae and J. Tsujii. 2007. Dependency parsing and domain adaptation with lr models and parser ensembles. In *Proceedings of the CoNLL 2007 Shared Task in the Joint Conferences on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL'07 shared task)*, pages 1044–1050.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D. Manning. 2012. Multi-instance multi-label learning for relation extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 455–465, Stroudsburg, PA, USA. Association for Computational Linguistics.

Hanghang Tong, Christos Faloutsos, and Jia-Yu Pan. 2006. Fast random walk with restart and its applications. In *ICDM*, pages 613–622. IEEE Computer Society.

William Yang Wang, Kathryn Mazaitis, and William W Cohen. 2013. Programming with personalized pagerank: a locally groundable first-order probabilistic logic. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*, pages 2129–2138. ACM.

Fei Wu and Daniel S Weld. 2010. Open information extraction using wikipedia. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 118–127. Association for Computational Linguistics.

X. Zhu, Z. Ghahramani, and J. Lafferty. 2003. Semi-supervised learning using Gaussian fields and harmonic functions. In *Proceedings of ICML-03, the 20th International Conference on Machine Learning*.

An Entity-centric Approach for Overcoming Knowledge Graph Sparsity

Manjunath Hegde
Indian Institute of Science
manjunath@ssl.serc.iisc.in

Partha Talukdar
Indian Institute of Science
ppt@serc.iisc.in

Abstract

Automatic construction of knowledge graphs (KGs) from unstructured text has received considerable attention in recent research, resulting in the construction of several KGs with millions of entities (nodes) and facts (edges) among them. Unfortunately, such KGs tend to be severely sparse in terms of number of facts known for a *given* entity, i.e., have low *knowledge density*. For example, the NELL KG consists of only 1.34 facts per entity. Unfortunately, such low knowledge density makes it challenging to use such KGs in real-world applications. In contrast to *best-effort* extraction paradigms followed in the construction of such KGs, in this paper we argue in favor of ENTIty Centric Expansion (ENTICE), an *entity-centric* KG population framework, to alleviate the low knowledge density problem in existing KGs. By using ENTICE, we are able to increase NELL's knowledge density by a factor of 7.7 at 75.5% accuracy. Additionally, we are also able to extend the ontology discovering new relations and entities.

1 Introduction

Over the last few years, automatic construction of knowledge graphs (KGs) from web-scale text data has received considerable attention, resulting in the construction of several large KGs such as NELL (Mitchell et al., 2015), Google's Knowledge Vault (Dong et al., 2014). These KGs consist of millions of entities and facts involving them. While measuring size of the KGs in terms of number of entities and facts is helpful, they don't readily capture the volume of knowledge needed in real-world applications. When such a KG is used in an application, one is often interested in known facts for a *given* entity, and not necessarily the overall size of the KG. In particular, knowing the average number of facts per entity is quite informative. We shall refer to this as the *knowledge density* of the KG.

	Known Target **E**ntity	**N**ew Target **E**ntity
Known **R**elation	KR-KE	KR-NE
New **R**elation	NR-KE	NR-NE

Table 1: Any new fact involving a source entity from a Knowledge Graph (i.e., facts of the form *entity1-relation-entity2* where *entity1* is already in the KG) can be classified into one of the four extraction classes shown above. Most KG population techniques tend to focus on extracting facts of the KR-KE class. ENTICE, the entity-centric approach proposed in this paper, is able to extract facts of all four classes.

Low knowledge density (or high sparsity) in automatically constructed KGs has been recognized in recent research (West et al., 2014). For example, NELL KG has a knowledge density of 1.34. Such low knowledge density puts significant limitations on the utility of these KGs. Construction of such KGs tend to follow a batch paradigm: the knowledge extraction system makes a full pass over the text corpus extracting whatever knowledge it finds, and finally aggregating all extractions into a graph. Clearly, such *best-effort* extraction paradigm has proved to be inadequate to address the low knowledge density issue mentioned above. We refer to such paradigm as best-effort since its attention is divided equally among all possible entities.

Recently, a few *entity-centric* methods have

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 530–535,
Lisbon, Portugal, 17-21 September 2015.

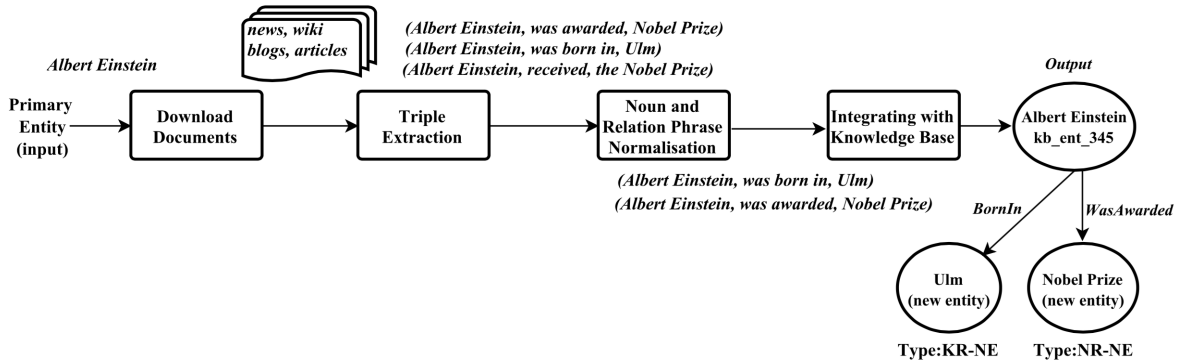


Figure 1: Dataflow and architecture and of ENTICE. See Section 3 for details.

been proposed to increase knowledge density in KGs (Gardner et al., 2013; Gardner et al., 2014). In contrast to the best-effort approaches mentioned above, these entity-centric approaches aim at increasing knowledge density for a *given* entity. A new fact involving the given entity can belong to one of the four types shown in Table 1. Unfortunately, these densifying techniques only aim at identifying instances of known relations among entities already present in the KG, i.e., they fall in the KR-KE type of Table 1.

In this paper we propose ENTIty Centric Expansion (ENTICE), an entity-centric knowledge densifying framework which, given an entity, is capable of extracting facts belonging to all the four types shown in Table 1. By using ENTICE, we are able to increase NELL's knowledge density by a factor of 7.7[1], while achieving 75.4% accuracy. Our goal here is to draw attention to the effectiveness of entity-centric approaches with bigger scope (i.e., covering all four extraction classes in Table 1) towards improving knowledge density, and that even relatively straightforward techniques can go a long way in alleviating low knowledge density in existing state-of-the-art KGs. ENTICE code is available at: `https://github.com/malllabiisc/entity-centric-kb-pop`

2 Related Work

Open Information Extraction (OIE) systems (Yates et al., 2007; Fader et al., 2011; Schmitz et al., 2012) aim at extracting textual triples of the form noun phrase-predicate-noun phrase. While such systems aim for extraction coverage, and because they operate in an ontology-free setting, they don't directly address the problem of improving knowledge density in ontological KGs such as NELL. However, OIE extractions provide a suitable starting point which is exploited by ENTICE.

(Galárraga et al., 2014) addresses the problem of normalizing (or canonicalizing) OIE extractions which can be considered as one of the components of ENTICE (see Section 3.3).

As previously mentioned, recent proposals for improving density of KGs such as those reported in (Gardner et al., 2013; Gardner et al., 2014) focus on extracting facts of one of the four extraction classes mentioned in Table 1, viz., KR-KE. The KBP challenge (Surdeanu, 2013) also focuses on extracting facts while keeping the relation set fixed, i.e., it addresses the KR-KE and KR-NE extraction classes.

A method to improve knowledge density in KGs by using search engine query logs and a question answering system is presented in (West et al., 2014). The proprietary nature of datasets and tools used in this approach limits its applicability in our setting.

ENTICE aims to improve knowledge density by extracting facts from all four extraction classes, i.e., for a given entity, it extracts facts involving known relations, identifies potentially new relations that might be relevant for this entity, establishes such relations between the given entity and other known as well as new entities – all in a single system. While various parts of this problem have been studied in isolation in the past, ENTICE is

[1]Measured with respect to the five categories experimented with in the paper. See Section 4 for details.

the first system to the best of our knowledge that addresses the complete problem as a single framework.

3 ENTIty Centric Expansion (ENTICE)

Overall architecture and dataflow within ENTICE is shown in Figure 1. We describe each of the components in the sections below.

3.1 Data Preprocessing

Given the source entity, documents relevant to it are downloaded by issues queries against Google. In order to make the query specific, especially in case of ambiguous entities, a few keywords are also added to the query. For the experiments in this paper, the category is used as the keyword. For example, for the entity *Albert Einstein* from the *scientist* category, the query will be *"Albert Einstein scientist"*. Top 20 documents returned by the search engine are downloaded and processed further. Text is extracted from the raw downloaded documents using regex patters, HTML tag matching, and by using the Boilerpipe tool[2].

3.2 Triple Extraction

Text of each document obtained in the previous step is processed through the Stanford CoreNLP toolkit (Manning et al., 2014) for tokenization, coreference resolution, and dependency parsing. Tokenized and coreference-resolved sentences are then passes through OpenIEv4 system [3] to extract *(noun phrase, predicate, noun phrase)* triples. Multiple and overlapping triples from the sentence was permitted. Length filter is applied on the noun phrase and the predicate of the triple extracted. This eliminates triples whose predicate is more than 6 tokens and noun phrase more than 7 tokens.

3.3 Noun and Relation Phrase Normalization

Noun phrases (NPs) and relation phrases obtained from the previous step are normalized (or canonicalized) in this step. Canopy clustering technique as proposed in (Galárraga et al., 2014) was used for noun phrase as well relation phrase clustering. Initial clustering is done over the *unlinked* noun phrases in the triples. Please note that since we are working in an entity-centric manner, one of the two NPs present in the triple is already connected to the knowledge graph, and hence is considered *linked*. To cluster noun phrases, we first construct canopies corresponding to each word in the noun phrase. For example, for noun phrase *Albert Einstein*, we create two canopies, viz., a canopy for *Albert* and another canopy for *Einstein*, and add *Albert Einstein* to both canopies. Grouping of noun phrases inside the canopy is the next step of clustering phase. Noun phrase similarity is calculated based on similarity of words in the noun phrases. Word similarity is either direct string matching or Gensim similarity score[4], which internally uses word2vec embeddings (Mikolov et al., 2013). After calculating pairwise similarity of noun phrases, hierarchical clustering is carried out to group noun phrases inside each canopy. A threshold score is used to stop hierarchical clustering. At the end of this process, we have canopies and groups of noun phrases inside them. A noun phrase can be in more than one canopy, hence those groups across canopies are merged if the similarity is greater than certain threshold. After this, each group will contain facts which have similar noun phrases and different (or same) relation phrase. Again the facts are clustered based on the similarity of the relation phrase. Relation phrase similarity calculation step resembles the one used for noun phrases as described above.

After this triple clustering step, the best representative triple from each cluster is selected based on a few rules. We consider the structure of POS tags in noun phrases of a triple as one of the criteria. Secondly, if both noun phrases in the triple are linked to the knowledge graph, then it makes the triple more likely to become a representative tuple of the cluster. Also, if the NPs present in the triple are frequent in the cluster, then it makes the corresponding triple more like to become a representative.

[2]Boilerpipe: http://code.google.com/p/boilerpipe
[3]OpenIEv4: http://knowitall.github.io/openie/
[4]https://github.com/piskvorky/gensim/

Category	Knowledge Density in NELL	Knowledge Density after ENTICE	# Facts Evaluated	# Correct Facts	Accuracy
Scientist	1.27	18.5	164	141	85.97
Universities	1.17	9	197	141	71.57
Books	1.34	4.49	202	165	81.68
Birds	1.27	6.69	194	136	70.10
Cars	1.5	11.61	201	140	69.65
Overall	1.3	10.05	958	723	75.46

Table 2: Knowledge densities of five categories in NELL and after application of ENTICE, along with resulting accuracy. We observe that overall, ENTICE is able to increase knowledge density by a factor of 7.7 at 75.5% accuracy. This is our main result.

Entity Name	All facts in NELL	Sample facts extracted by ENTICE	Extraction Class
George Paget Thomson	*(George Paget Thomson, isInstanceOf, scientist)*	*(Sir George Thomson, isFellowOf, Royal Society)*	NR-KE
		(George Thomson, hasSpouse, Kathleen Buchanan Smith)	KR-NE
		(George Paget Thomson, diedOn, September 10)	KR-KE

Table 3: Facts corresponding to an entity from the *scientists* domain in NELL as well as those extracted by ENTICE. While NELL contained only one fact for this entity, ENTICE was able to extract 15 facts for this entity, only 3 of which are shown above.

Category	KR - KE			KR - NE			NR - KE			NR - NE		
	correct facts	wrong facts	acc.	correct facts	wrong facts	acc.	correct facts	wrong facts	acc.	correct facts	wrong facts	acc.
Scientists	57	10	85.07	61	8	88.40	14	3	82.35	9	2	81.81
Cars	68	35	66.01	58	21	73.41	9	5	64.28	5	0	100
Universities	52	30	63.41	68	20	77.27	9	2	81.81	12	4	75
Books	78	24	76.47	79	12	86.81	2	0	100	6	1	85.71
Birds	67	29	69.79	46	19	70.76	15	4	78.94	8	6	57.14
Overall	322	128	71.55	312	80	79.59	49	14	77.77	40	13	75.47

Table 4: Accuracy breakdown over ENTICE extractions for each of the four extraction classes in Table 1. For each category, approximately 200 extractions were evaluated using Mechanical Turk.

3.4 Integrating with Knowledge Graph

The set of normalized triples from the previous step are linked with the Knowledge Graph, whenever possible, in this step. For a given normalized triple, following steps are performed as part of linking. First, category of each noun phrase in the triple is obtained based on string matching. In case of no match, refinements like dropping of adjectives, considering only noun phrases are done to for rematching. Now, the relation phrase is mapped to an existing predicate in the KG based on the extraction patterns in the metadata of the target relation (e.g., NELL and many other KGs have such metadata available). Candidate predicates are chosen from the above mapped predicates based on category signature of the two noun phrases (i.e. entity1 and entity2). This is possible since the all the predicates in NELL have the type signature defined in the metadata. Frequency of the relation phrase in the metadata is used as a criteria to select a candidate from multiple predicates. If such category-signature based mapping is not possible, then the predicate is listed as a new relation, and the corresponding triple marked to belong to either NR-KE or NE-NE extraction class, depending on whether the target entity is already present in the KG or not.

4 Experiments

In order to evaluate effectiveness of ENTICE, we apply it to increase knowledge density for 100 randomly selected entities from each of the following five NELL categories: *Scientist, Universities, Books, Birds*, and *Cars*. For each category, a random subset of extractions in that category was evaluated using Mechanical Turk. To get a better accuracy of the evaluation, each fact was evaluated by 3 workers. Workers were made to classify each fact as correct, incorrect or can't say. Only those facts classified as correct by 2 or more evaluators were considered as correct facts.

Main Result: Experimental results comparing knowledge densities in NELL and after application of ENTICE, along with the accuracy of extractions, are presented in Table 2. From this, we observe that ENTICE is able to improve knowledge density in NELL by a factor of 7.7 while maintaining 75.5% accuracy. Sample extraction examples and accuracy per-extraction class are presented in Table 3 and Table 4, respectively.

Noun and Relation Phrase Normalization: We didn't perform any intrinsic evaluation of the entity and relation normalization step. However, in this section, we provide a few anecdotal examples to give a sense of the output quality from this step. We observe that the canopy clustering algorithm for entity and normalization is able to cluster together facts with somewhat different surface representations. For example, the algorithm came up with the following cluster with two facts: {*(J. Willard Milnor, was awarded, 2011 Abel Prize); (John Milnor, received, Abel Prize)*}. It is encouraging to see that the system is able to put *J. Willard Milnor* and *John Milnor* together, even though they have somewhat different surface forms (only one word overlap). Similarly, the relation phrases *was awarded* and *received* are also considered to be equivalent in the context of these beliefs.

Integrating with Knowledge Graph: Based on evaluation over a random-sampling, we find that entity linking in ENTICE is 92% accurate, while relation linking is about 70% accurate.

In the entity linking stage, adjectives present in a noun phrase (NP) were ignored while matching the noun phrase to entities in the knowledge graph (NELL KB in this case). In case the whole NP didn't find any match, part of the NP was used to retrieve its category, if any. For example, in *(Georg Waldemar Cantor, was born in, 1854)*, the NP *Georg Waldemar Cantor* was mapped to category *person* using his last name and *1854* to category *date*. The relation phrase *"was born in"* maps to many predicates in NELL relational metadata. NELL predicate *AtDate* was selected based on the rule that category signature of the predicate matches the category of the noun phrases present in the triple. It also has the highest frequency count for the relational phrase in the metadata.

We observed that relation mapping has lesser accuracy due to two reasons. Firstly, error in determining right categories of NPs present in a triple; and secondly, due to higher ambiguity involving relation phrases in general, i.e., a single relation phrase usually matches many relation predicates in the ontology.

5 Conclusion

This paper presents ENTICE, a simple but effective entity-centric framework for increasing knowledge densities in automatically constructed knowledge graphs. We find that ENTICE is able to significantly increase NELL's knowledge density by a factor of 7.7 at 75.5% accuracy. In addition to extracting new facts, ENTICE is also able to extend the ontology. Our goal in this paper is twofold: (1) to draw attention to the effectiveness of entity-centric approaches with bigger scope (i.e., covering all four extraction classes in Table 1) towards improving knowledge density; and (2) to demonstrate that even relatively straightforward techniques can go a long way in alleviating low knowledge density in existing state-of- the-art KGs. While these initial results are encouraging, we hope to apply ENTICE on other knowledge graphs, and also experiment with other normalization and entity linking algorithms as part of future work.

Acknowledgment

This work is supported in part by a gift from Google. Thanks to Uday Saini for carefully reading a draft of the paper.

References

Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. 2014. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1535–1545. Association for Computational Linguistics.

Luis Galárraga, Geremy Heitz, Kevin Murphy, and Fabian M Suchanek. 2014. Canonicalizing open knowledge bases. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 1679–1688. ACM.

Matt Gardner, Partha Pratim Talukdar, Bryan Kisiel, and Tom Mitchell. 2013. Improving learning and inference in a large knowledge-base using latent syntactic cues.

Matt Gardner, Partha Pratim Talukdar, Jayant Krishnamurthy, and Tom Mitchell. 2014. Incorporating vector space similarity in random walk inference over knowledge bases.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*.

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

T Mitchell, W Cohen, E Hruschka, P Talukdar, J Betteridge, A Carlson, B Dalvi, M Gardner, B Kisiel, J Krishnamurthy, et al. 2015. Never-ending learning. In *Proceedings of AAAI*.

Michael Schmitz, Robert Bart, Stephen Soderland, Oren Etzioni, et al. 2012. Open language learning for information extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 523–534. Association for Computational Linguistics.

Mihai Surdeanu. 2013. Overview of the tac2013 knowledge base population evaluation: English slot filling and temporal slot filling. In *Proceedings of the Sixth Text Analysis Conference (TAC 2013)*.

Robert West, Evgeniy Gabrilovich, Kevin Murphy, Shaohua Sun, Rahul Gupta, and Dekang Lin. 2014. Knowledge base completion via search-based question answering. In *Proceedings of the 23rd international conference on World wide web*.

Alexander Yates, Michael Cafarella, Michele Banko, Oren Etzioni, Matthew Broadhead, and Stephen Soderland. 2007. Textrunner: open information extraction on the web. In *Proceedings of Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 25–26. Association for Computational Linguistics.

Semantic Relation Classification via Convolutional Neural Networks with Simple Negative Sampling

Kun Xu[1], **Yansong Feng**[*1], **Songfang Huang**[2] **and Dongyan Zhao**[1]
[1]ICST, Peking University, Beijing, China
[2]IBM China Research Lab, Beijing, China
{xukun,fengyansong,zhaodongyan}@pku.edu.cn
huangsf@cn.ibm.com

Abstract

Syntactic features play an essential role in identifying relationship in a sentence. Previous neural network models directly work on raw word sequences or constituent parse trees, thus often suffer from irrelevant information introduced when subjects and objects are in a long distance. In this paper, we propose to learn more robust relation representations from shortest dependency paths through a convolution neural network. We further take the relation directionality into account and propose a straightforward negative sampling strategy to improve the assignment of subjects and objects. Experimental results show that our method outperforms the state-of-the-art approaches on the SemEval-2010 Task 8 dataset.

1 Introduction

The relation extraction (RE) task can be defined as follows: given a sentence S with a pair of nominals e_1 and e_2, we aim to identify the relationship between e_1 and e_2. RE is typically investigated in a classification style, where many features have been proposed, e.g., Hendrickx et al. (2010) designed 16 types of features including POS, WordNet, FrameNet, dependency parse features, etc. Among them, syntactic features are considered to bring significant improvements in extraction accuracy (Bunescu and Mooney, 2005a). Earlier attempts to encode syntactic information are mainly kernel-based methods, such as the convolution tree kernel (Qian et al., 2008), subsequence kernel (Bunescu and Mooney, 2005b), and dependency tree kernel (Bunescu and Mooney, 2005a).

With the recent success of neural networks in natural language processing, different neural network models are proposed to learn syntactic features from raw sequences of words or constituent parse trees (Zeng et al., 2014; Socher et al., 2012), which have been proved effective, but, often suffer from irrelevant subsequences or clauses, especially when subjects and objects are in a longer distance. For example, in the sentence, "The [singer]$_{e1}$, who performed three of the nominated songs, also caused a [commotion]$_{e2}$ on the red carpet", the *who* clause is used to modify subject e_1, but is unrelated to the *Cause-Effect* relationship between *singer* and *commotion*. Incorporating such information into the model will hurt the extraction performance. We therefore propose to learn a more robust relation representation from a convolution neural network (CNN) model that works on the simple dependency path between subjects and objects, which naturally characterizes the relationship between two nominals and avoids negative effects from other irrelevant chunks or clauses.

Our second contribution is the introduction of a negative sampling strategy into the CNN models to address the relation directionality, i.e., properly assigning the subject and object within a relationship. In the above *singer* example, (*singer*, *commotion*) hold the *Cause-Effect* relation, while (*commotion*, *singer*) not. Previous works (Liu et al., 2015) do not fully investigate the differences between subjects and objects in the utterance, and simply transform a (K+1)-relation task into a ($2\times K$+1) classification task, where 1 is the *other* relation. Interestingly, we find that dependency paths naturally offer the relative positions of subjects and objects through the path directions. In this paper, we propose to model the relation directionality by exploiting the dependency path to learn the assignments of subjects and objects using a straightforward negative sampling method, which adopts the shortest dependency path from the object to the subject as a negative sample. Experimental results show that the negative sampling method significantly improves the performance,

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 536–540,
Lisbon, Portugal, 17-21 September 2015.

and our model outperforms the-state-of-the-art methods on the SemEval-2010 Task 8 dataset.

2 The Shortest Path Hypothesis

If e_1 and e_2 are two nominals mentioned in the same sentence, we assume that the shortest path between e_1 and e_2 describes their relationship. This is because (1) if e_1 and e_2 are arguments of the same predicate, then their shortest path should pass through that predicate; (2) if e_1 and e_2 belong to different predicate-argument structures, their shortest path will pass through a sequence of predicates, and any consecutive predicates will share a common argument. Note that, the order of the predicates on the path indicates the proper assignments of subjects and objects for that relation. For example, in Figure 1, the dependency path consecutively passes through *carried* and *receives*, which together implies that in the *Instrument-Agency* relation, the subject and object play a sender and receiver role, respectively.

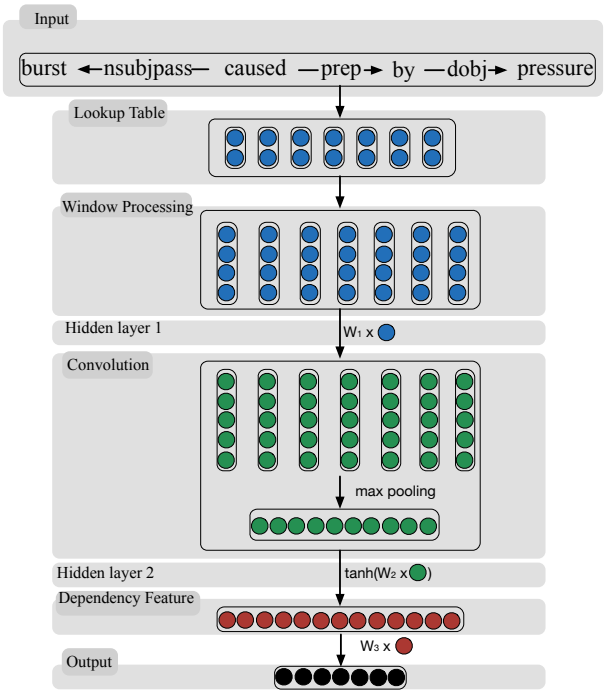


Figure 2: Architecture of the convolution neural network.

3 A Convolutional Neural Network Model

Our model successively takes the shortest dependency path (i.e, the words, dependency edge directions, and dependency labels) from the subject to the object as input, passes it through the lookup table layer, produces local features around each node on the dependency path, and combines these features into a global feature vector that are then fed to a softmax classifier. Each dimension of the output vector indicates the confidence score of the corresponding relation.

In the *lookup table* step, each node (i.e. word, label or arrow) in the dependency path is transformed into a vector by looking up the embedding matrix $W_e \in \mathbb{R}^{d \times |\mathbb{V}|}$, where d is the dimension of a vector and $\mathbb{V}$ is a set of all nodes we consider.

Convolution To capture the local features around each node of the dependency path, we consider a fixed size window of nodes around each node in the *window processing* component, producing a matrix of node features of fixed size $d_w \times 1$, where $d_w = d \times w$ and w is the window size. This matrix can be built by concatenating the vectors of nodes within the window, which are then fed to the convolutional layer.

In the convolutional layer, we use a linear transformation $W_1 \in \mathbb{R}^{n_1 \times d_w}$ to extract local features around each window of the given sequence, where n_1 is the size of hidden layer 1. The resulting matrix Z has size of $n_1 \times t$, where t is the number of nodes in the input dependency path.

We can see that Z captures local contextual information in the dependency path. However, identifying a relationship requires considerations for the *whole* dependency path. We therefore perform a max pooling over Z to produce a global feature vector in order to capture the most useful local features produced by the convolutional layer (Collobert et al., 2011), which has a fixed size of n_1, independent of the dependency path length.

Dependency based Relation Representation To extract more meaningful features, we choose hyperbolic tanh as the non-linearity function in the second hidden layer, which has the advantage of being slightly cheaper to compute, while leaving the generalization performance unchanged. $W_2 \in \mathbb{R}^{n_2 \times n_1}$ is the linear transformation matrix, where n_2 is the size of hidden layer 2. The output vector can be considered as higher level syntactic features, which is then fed to a softmax classifier.

Objective Function and Learning The softmax classifier is used to predict a K-class distribution $d(x)$, where K is the size of all possible relation types, and the transformation matrix is $W_3 \in \mathbb{R}^{K \times n_2}$. We denote $t(x) \in \mathbb{R}^{K \times 1}$ as the target

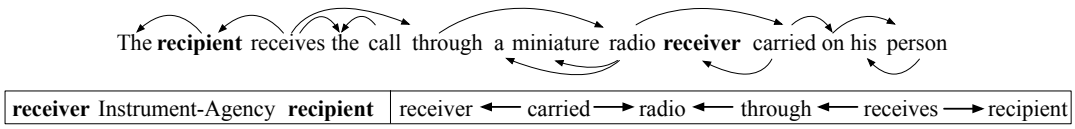


Figure 1: The shortest dependency path representation for an example sentence from SemEval-08.

Train Strategy	Test Strategy	F1(%)
Blind	Blind	79.3
Sighted	Blind	81.3
Sighted	Sighted	89.2

Table 1: Performances on the development set with different train and testing strategies.

distribution vector[1]: the entry $t_k(x)$ is the probability that the dependency path describes the *k*-th relation. We compute the cross entropy error between $t(x)$ and $d(x)$, and further define the objective function over all training data:

$$J(\theta) = -\sum_{x}\sum_{k=1}^{K} t_k(x)\log d_k(x) + \lambda||\theta||^2$$

where $\theta = (W_e, W_1, W_2, W_3)$ is the set of model parameters to be learned, and λ is a vector of regularization parameters. The model parameters θ can be efficiently computed via backpropagation through network structures. To minimize $J(\theta)$, we apply stochastic gradient descent (SGD) with AdaGrad (Duchi et al., 2011) in our experiments.

4 Negative Sampling

We start by presenting three pilot experiments on the development set. In the first one, we assume that the assignment of the subject and object for a relation is not given (blind), we simply extract features from e_1 to e_2, and test it in a blind setting as well. In the second one, we assume that the assignment is given (sighted) during training, but still blind in the test phase. The last one is assumed to give the assignment during both training and test steps. The results are listed in Table 1.

The third experiment can be seen as an upper bound, where we do not need to worry about the assignments of subjects and objects. By comparing the first and the second one, we can see that when adding assignment information during training, our model can be significantly improved, indicating that our dependency based representation can be used to learn the assignments of subjects/objects, and injecting better understandings of such assignments during training is crucial to the performance. We admit that models with more complex structures can better handle these considerations. However, we find that this can be achieved by simply feeding typical negative samples to the model and let the model learn from such negative examples to correctly choose the right assignments of subjects and objects. In practice, we can treat the opposite assignments of subjects and the objects as negative examples. Note that, the dependency path of the wrong assignment is different from that of the correct assignment, which essentially offers the information for the model to learn to distinguish the subject and the object.

[1]Note that, there may be more than one relation existing between two nominals. A dependency path thus may correspond to multiple relations.

5 Experimental Evaluation

We evaluate our model on the SemEval-2010 Task 8 (Hendrickx et al., 2010), which contains 10,717 annotated examples, including 8,000 instances for training and 2,717 for test. We randomly sampled 2,182 samples from the training data for validation.

Given a sentence, we first find the shortest dependency path connecting two marked nominals, resulting in two dependency paths corresponding to two opposite subject/object directions, and then make predictions for the two paths, respectively. We choose the relation *other* **if and only if** both predictions are *other*. And for the rest cases, we choose the non-*other* relation with highest confidence as the output, since ideally, for a non-*other* instance, our model will output the correct label for the right subject/object direction and an *other* label for the wrong direction. We evaluate our models by macro-averaged F1 using the official evaluation script.

We initialized W_e with 50-dimensional word vectors trained by Turian et al. (2010). We tuned the hyper parameters using the development set for each experimental setting. The hyper parameters include w, n_1, n_2, and regularization parameters

Method	Feature Sets	F1
SVM	16 types of features	82.2
RNN	-	74.8
	+POS, NER, WordNet	77.6
MVRNN	-	79.1
	+POS, NER, WordNet	82.4
CNN	-	78.9
(Zeng et al., 2014)	+WordNet,words around nominals	82.7
DepNN	+NER	83.6
depCNN	-	81.3
depLCNN	-	81.9
depLCNN	+WordNet,words around nominals	83.7
depLCNN+NS	-	**84.0**
	+WordNet,words around nominals	**85.6**

Table 2: Comparisons of our models with other methods on the SemEval 2010 task 8.

Negative sampling schemes	F1
No negative examples	81.3
Randomly sampled negative examples from NYT	83.5
Dependency paths from the object to subject	**85.4**

Table 3: Comparisons of different negtive sampling methods on the development set.

for W_e, W_1, W_2 and W_3. The best setting was obtained with the values: 3, 200, 100, 10^{-4}, 10^{-3}, 10^{-4} and 2×10^{-3}, respectively.

Results and Discussion Table 2 summarizes the performances of our model, depLCNN+NS(+), and state-of-the-art models, SVM (Hendrickx et al., 2010), RNN, MV-RNN (Socher et al., 2012), CNN (Zeng et al., 2014) and DepNN (Liu et al., 2015). For fair comparisons, we also add two types of lexical features, WordNet hypernyms and words around nominals, as part of input vector to the final `softmax` layer.

We can see that our vanilla depLCNN+NS, without extra lexical features, still outperforms, by a large margin, previously reported best systems, MVRNN+ and CNN+, both of which have taken extra lexical features into account, showing that our treatment to dependency path can learn a robust and effective relation representation. When augmented with similar lexical features, our depLCNN+NS further improves by 1.6%, significantly better than any other systems.

Let us first see the comparisons among plain versions of depLCNN (taking both dependency directions and labels into account), depCNN (considering the directions of dependency edges only), MVRNN and CNN, which all work in a $2\times K$+1 fashion. We can see that the both of our depCNN and depLCNN outperforms MVRNN and CNN by at least 2.2%, indicating that our treatment is better than previous conventions in capturing syntactic structures for relation extraction. And note that depLCNN, with extra considerations for dependency labels, performs even better than depCNN, showing that dependency labels offer more discriminative information that benefits the relation extraction task.

And when we compare plain depLCNN and depLCNN+NS (without lexical features), we can see that our Negative Sampling strategy brings an improvement of 2.1% in F1. When both of the two models are augmented with extra lexical features, our NS strategy still gives an improvement of 1.9%. These comparisons further show that our NS strategy can drive our model to learn proper assignments of subjects/objects for a relation.

Next, we will have a close look at the effect of our Negative Sampling method. We conduct additional experiments on the development set to compare two different negative sampling methods. As a baseline, we randomly sampled 8,000 negative examples from the NYT dataset (Chen et al., 2014). For our proposed NS, we create a negative example from each *non-other* instance in the training set, 6,586 in total. As shown in Table 2, it is no doubt that introducing more negative examples improves the performances. We can see that our model still benefits from the randomly sampled negative examples, which may help our model learn to refine the margin between the positive and negative examples. However, with similar amount of negative examples, treating the reversed dependency paths from objects to subjects as negative examples can achieve a better performance (85.4% F1), improving random samples by 1.9%. This again proves that dependency paths provide useful clues to reveal the assignments of subjects and objects, and a model can learn from such reversed paths as negative examples to make correct assignments. Beyond the relation extraction task, we believed the proposed Negative Sampling method has the potential to benefit other NLP tasks, which we leave for future work.

6 Conclusion

In this paper, we exploit a convolution neural network model to learn more robust and effective relation representations from shortest dependency paths for relation extraction. We further propose a simple negative sampling method to help make correct assignments for subjects and objects

within a relationship. Experimental results show that our model significantly outperforms state-of-the-art systems and our treatment to dependency paths can well capture the syntactic features for relation extraction.

Acknowledgments

We would like to Sheng Zhang, Weiwei Sun and Liwei Chen for their helpful discussions. This work was supported by National High Technology R&D Program of China (Grant No. 2015AA015403, 2014AA015102), Natural Science Foundation of China (Grant No. 61202233, 61272344, 61370055) and the joint project with IBM Research. Any correspondence please refer to Yansong Feng.

References

Razvan C. Bunescu and Raymond J. Mooney. 2005a. A shortest path dependency kernel for relation extraction. In *HLT/EMNLP 2005, Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference, 6-8 October 2005, Vancouver, British Columbia, Canada*.

Razvan C. Bunescu and Raymond J. Mooney. 2005b. Subsequence kernels for relation extraction. In *Advances in Neural Information Processing Systems 18 [Neural Information Processing Systems, NIPS 2005, December 5-8, 2005, Vancouver, British Columbia, Canada]*, pages 171–178.

Liwei Chen, Yansong Feng, Songfang Huang, Yong Qin, and Dongyan Zhao. 2014. Encoding relation requirements for relation extraction via joint inference. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 1: Long Papers*, pages 818–827.

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

John C. Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159.

Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. 2010. Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In *Proceedings of SemEval-2*, Uppsala, Sweden.

Yang Liu, Furu Wei, Sujian Li, Heng Ji, Ming Zhou, and Houfeng WANG. 2015. A dependency-based neural network for relation classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 285–290, Beijing, China, July. Association for Computational Linguistics.

Longhua Qian, Guodong Zhou, Fang Kong, Qiaoming Zhu, and Peide Qian. 2008. Exploiting constituent dependencies for tree kernel-based semantic relation extraction. In *COLING 2008, 22nd International Conference on Computational Linguistics, Proceedings of the Conference, 18-22 August 2008, Manchester, UK*, pages 697–704.

Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, EMNLP-CoNLL 2012, July 12-14, 2012, Jeju Island, Korea*, pages 1201–1211.

Joseph P. Turian, Lev-Arie Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *ACL 2010, Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, July 11-16, 2010, Uppsala, Sweden*, pages 384–394.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation classification via convolutional deep neural network. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 2335–2344, Dublin, Ireland, August. Dublin City University and Association for Computational Linguistics.

A Baseline Temporal Tagger for all Languages

Jannik Strötgen and **Michael Gertz**
Institute of Computer Science, Heidelberg University
Im Neuenheimer Feld 348, 69120 Heidelberg, Germany
{stroetgen,gertz}@informatik.uni-heidelberg.de

Abstract

Temporal taggers are usually developed for a certain language. Besides English, only few languages have been addressed, and only the temporal tagger HeidelTime covers several languages. While this tool was manually extended to these languages, there have been earlier approaches for automatic extensions to a single target language. In this paper, we present an approach to extend HeidelTime to all languages in the world. Our evaluation shows promising results, in particular considering that our approach neither requires language skills nor training data, but results in a baseline tagger for 200+ languages.

1 Introduction & Related Work

Temporal tagging (the extraction and normalization of temporal expressions) is a crucial task in NLP, and many applications can benefit from temporal information, e.g., in the context of question answering and information retrieval (Campos et al., 2014). Thus, there has been a lot of research during the last years addressing temporal information extraction, as reflected by manually annotated corpora, approaches to temporal tagging, and research challenges such as the TempEval series (Verhagen et al., 2010; UzZaman et al., 2013).

While, in the meantime, the range of annotated corpora covers several languages different from English, e.g., the French, Portuguese, Italian, and Romanian TimeBank corpora (Bittar et al., 2011; Costa and Branco, 2012; Caselli et al., 2011; Forascu and Tufis, 2012), most approaches to temporal tagging focus on processing English text, e.g., DANTE (Mazur and Dale, 2009; Mazur, 2012) and SUTime (Chang and Manning, 2012). Of course, exceptions exist, e.g., TipSem for English and Spanish (Llorens et al., 2010), and HeidelTime (Strötgen and Gertz, 2013) even covers 13 languages. However, it required a lot of manual effort to extend HeidelTime, and researchers of different institutes and countries have been involved for German (Strötgen and Gertz, 2011), Dutch (van de Camp and Christiansen, 2012), Spanish (Strötgen et al., 2013), French (Moriceau and Tannier, 2014), Croatian (Skukan et al., 2014), Vietnamese and Arabic (Strötgen et al., 2014), Italian (Manfredi et al., 2014), Chinese (Li et al., 2014), Russian, Estonian, and Portuguese.

Obviously, it is possible to manually extend HeidelTime and thus a temporal tagger in general. However, each language is typically addressed separately, and the extension process is time- and labor-intensive, so that a method to automate this process is desirable. While there have been earlier approaches for automatic extensions of temporal taggers to further languages (Saquete et al., 2004; Negri et al., 2006; Spreyer and Frank, 2008), these were limited to a few languages and the results were considered less successful, in particular for the normalization subtask. In contrast, Angeli and Uszkoreit (2013) presented an approach to language-independent parsing of temporal expressions, however, addressing only the normalization and not the extraction subtask.

In this paper, we re-address the task of automatically extending a temporal tagger to further languages. In contrast to previous work, we address both subtasks of temporal tagging, i.e., the extraction and the normalization, and we do not limit our work to a few selected languages, but we aim at extending HeidelTime to cover all languages in the world. In addition of making use of automatic pattern translations based on Wiktionary,[1] we allow for fuzzy matching in the newly addressed languages. This strategy results in promising temporal tagging quality for many languages, although our evaluations show that the quality of a manually extended temporal tagger is not reached, and

[1] http://www.wiktionary.org/.

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 541–547,
Lisbon, Portugal, 17-21 September 2015.

that there are open issues that need to be addressed for specific language characteristics.

By making our newly developed resources publicly available, we establish a baseline temporal tagger for basically all languages in the world – for comparison or as starting point for improvements.

2 HeidelTime

HeidelTime is a rule-based, domain-sensitive, and publicly available temporal tagger that was developed with multilinguality in mind so that the source code and language-dependent resources are strictly separated, and it can be extended to new languages without modifying its source code (Strötgen and Gertz, 2013).

2.1 Input and Output Format

As input, HeidelTime requires linguistic preprocessed documents with sentence, token, and part-of-speech (pos) information – although it is possible to create language resources not making use of pos constraints. For processing news-style documents, a publication date is additionally needed.

Temporal expressions are annotated following the temporal markup language TimeML (Pustejovsky et al., 2005) using TIMEX3 tags with normalization attributes, most importantly *type* (date, time, duration, set) and *value* to represent the main semantics of expressions in standard format.

2.2 Language-dependent Resources

Each language requires its own language resources consisting of patterns, normalization information and rules. Pattern files contain terms that are frequently used to form temporal expressions, e.g., for names of months and weekdays or numbers that may refer to a day or year. Regular expressions can be used in the pattern files, and each pattern file is interpreted as a regular expression pattern, which can be called by HeidelTime's rules.

Normalization files contain normalized information for patterns defined in the pattern files, e.g., that the value of "January" is "01" and that "3" has to be normalized to "03" in case it refers to a day or a month and to "3" in case it refers to a duration.

Finally, for the four types of temporal expressions (date, time, duration, and set), a rule file contains all rules for respective expressions. In addition to a rule name, the extraction part and the value normalization part are obligatory. The former is a regular expression that may contain any number of references to pattern files, and in the latter, normalization functions are called to normalize respective parts of the extracted expressions. In addition, part-of-speech constraints can be specified, negative rules can be formulated, and offset information can be set, e.g., if parts of the extracted patterns are required context information during the extraction process but not part of the temporal expression itself.[2]

2.3 Manual Extension to a New Language

So far, HeidelTime resources for newly supported languages have been manually created (cf. references is Section 1). For this, the following steps had to be completed: (i) linguistic preprocessing for the new language has been provided, (ii) all patterns in the pattern and normalization files have been manually translated, and (iii) rules have been developed iteratively. Starting with simple rules of the source language (typically English), rules have been adapted and added based on false negatives, false positives, partial matches and incorrect value normalizations on the training data for the new language. Then, more complex rules of the source language have been addressed step-by-step.

Note that tasks (ii) and (iii) have been carried out by language experts, and task (iii) was performed until no further improvements could be achieved on the training data. In addition to the fact that this process is time- and labor intensive, further disadvantages are that a language expert and temporally annotated training data are required. Finally, languages are typically addressed by different researchers with differing effort so that quality and completeness varies.

3 Automatic Extension to All Languages

In this section, we present the steps of our automatic extension approach in detail. A graphical summary of our approach is depicted in Figure 1.

3.1 Linguistic Preprocessing

While part-of-speech (pos) information can be helpful to write more general rules and to improve temporal tagging quality, there is no pos tagger covering all languages in the world. In addition,

[2]While these explanations are necessary to understand our automated resource development approach that will be presented in Section 3, for further details about HeidelTime and its language-dependent resources we refer to Strötgen and Gertz (2013) and `https://github.com/HeidelTime/heideltime`.

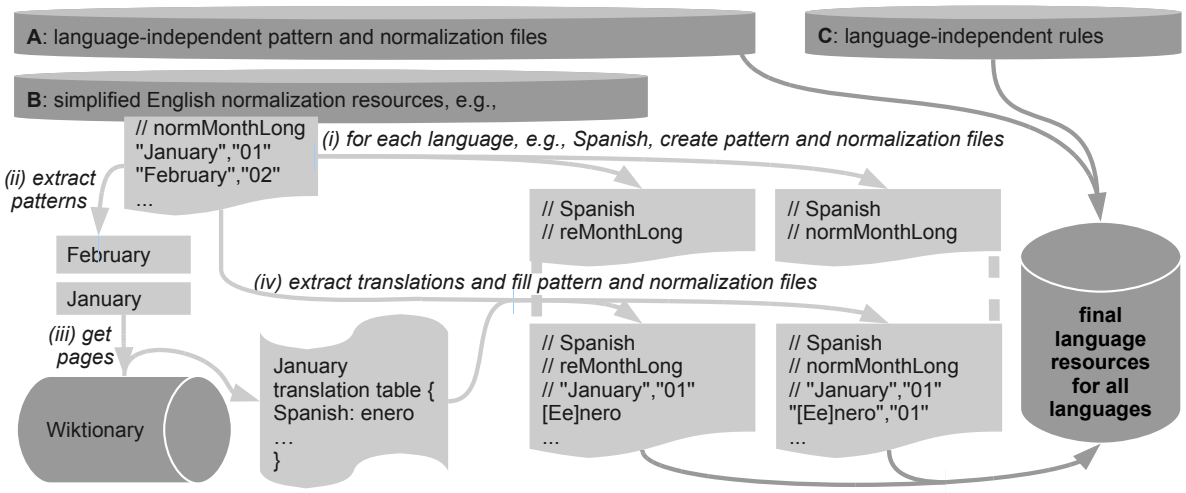


Figure 1: Graphical overview of the automated resource development process. Resources A and C are directly added to the resources of all languages; resource B is processed following steps (i) to (iv).

the only obligatory linguistic preprocessing steps are sentence splitting and tokenization. Thus, we write a simple and generic, language-independent sentence splitter and tokenizer, which we use for all languages. Note that our tokenization is based on a white-space token boundary assumption so that we have to take care of languages without such token boundaries separately (cf. Section 4).

3.2 Language-independent Resources

Some patterns and normalization information are valid for all (many) languages, e.g., numbers for days and months. For such patterns, we write language-independent pattern and normalization resources (Figure 1, A). These do not contain language-dependent terms and will be part of the resources for each of the languages in the world.

3.3 Simplified English Resources

The main goal of the simplified English resources is to make HeidelTime's original English resources amenable to automatic translation. Thus, rules are addressed in a language-independent fashion (cf. Section 3.4), and the simplified English resources (Figure 1, B) are written as normalization files, from which pattern and normalization files for all languages are derived (Figure 1, i-iv).

For terms directly occurring in English rules, new pattern files are created, and all English patterns are written without regular expressions. While this makes the resources longer (original patterns with regular expressions are expanded to many patterns), these simplified resources are necessary for a smooth translation process.

3.4 Language-independent Rules

Based on HeidelTime's English resources, we also write completely language-independent rules for date, time, duration, and set expressions. In addition, we add "creative" rules that are not necessary for English expressions but might match expressions in other languages. In particular, we allow several orderings of patterns and sometimes even random tokens in between. For instance, instead of just a "month day" pattern (e.g., to match "January 13"), we add "day month", "day X month" and "month X day", with "X" matching any token (e.g., for the Spanish translation "13 de enero").

In the extraction and normalization parts of the rules, we refer to the names of the language-independent and simplified English pattern and normalization files. Note that in contrast to the original rules, these rules do neither contain pos constraints nor any English terms to guarantee language independence. However, fuzzy pattern matching is allowed at the end of some patterns to try to take care of morphology-rich languages.

3.5 Creating Resources for All Languages

As shown in Figure 1, the patterns of the simplified English resources are translated to all languages in the world using Wiktionary as translation resource. For each normalization file, the respective pattern and normalization files for each language in Wiktionary are created (i).[3] Then,

[3]Note that Wiktionary exists in about 170 languages but contains translations to many more languages; for details, see `http://en.wiktionary.org/wiki/Wiktionary:Statistics`.

	HeidelTime 1.9 (manual)					HeidelTime – automatic				
	relaxed extr			value		relaxed extr			value	
language: corpus / domain	P	R	F1	F1	acc.	P	R	F1	F1	acc.
English: TE-3 TimeBank / news (UzZaman et al., 2013)	93.1	90.8	91.9	79.6	86.5	95.6	49.2	64.9	54.7	84.3
English: TE-3 platinum / news (UzZaman et al., 2013)	93.1	88.4	90.7	78.1	86.1	98.7	56.5	71.9	54.4	75.7
English: WikiWars / narrative (Mazur and Dale, 2010)	98.3	86.1	91.8	83.1	90.5	97.9	58.4	73.2	53.4	73.0
Arabic: Arabic test-50* / news (Strötgen et al., 2014)	90.9	90.9	90.9	82.2	90.4	91.7	31.8	47.2	38.0	80.5
Chinese: TE-2 test impro. / news (Li et al., 2014)	95.8	89.3	92.4	79.5	86.0	100	9.5	17.3	7.6	44.0
Croatian: WikiWarsHR / narrative (Skukan et al., 2014)	92.6	90.5	91.5	80.8	88.3	87.3	6.8	12.6	9.7	77.0
French: FR-TimeBank / news (Bittar et al., 2011)	91.9	90.1	91.0	73.6	80.9	87.2	59.5	70.8	54.6	77.1
German: WikiWarsDE / narrative (Strötgen and Gertz, 2011)	98.7	89.3	93.8	83.0	88.5	98.4	64.7	78.1	59.7	76.4
Italian: EVALITA'14 test / news (Caselli et al., 2014)	92.7	86.1	89.3	75.0	84.0	98.5	41.2	58.1	49.3	84.9
Spanish: TempEval-3 test / news (UzZaman et al., 2013)	96.0	84.9	90.1	85.3	94.7	95.5	53.8	68.8	58.5	85.0
Vietnamese: WikiWarsVN / narrative (Strötgen et al., 2014)	98.2	98.2	98.2	91.4	93.1	84.0	45.5	59.0	27.1	45.9
Portuguese: PT-TimeBank test / news (Costa and Branco, 2012)	87.3	75.9	81.2	63.5	78.2	91.5	59.3	72.0	59.4	82.5
Portuguese: PT-TimeBank train / news (Costa and Branco, 2012)	83.3	73.1	77.9	54.5	70.0	88.2	51.0	64.6	50.4	78.0
Romanian: Ro-TimeBank / news (Forascu and Tufis, 2012)	–	–	–	–	–	31.9	11.4	16.9	7.8	46.2

Table 1: Evaluation results for several languages on public corpora. HeidelTime 1.9 results as reported on `https://github.com/HeidelTime/heideltime/wiki/Evaluation-Results`.

the patterns are extracted from the simplified English normalization resources (ii) and the respective Wiktionary pages are accessed (iii). These contain translation information that is then added to the pattern and normalization files of the occurring languages by extracting respective translations (iv).

Note that for ambiguous words, Wiktionary contains one translation table for each meaning. For instance, there are two translation tables for "November" for the meanings "eleventh month of the Gregorian calendar" and "N in the ICAO spelling alphabet".[4] Obviously, it is crucial to use only the translations of the intended meaning. Thus, in the case of multiple meanings of English patterns, we provide the names of Wiktionary's correct translation tables to our translation script, so that only translations of the correct meaning are generated. In addition, if patterns in the new language start with a lower case character, we allow upper case and lower case for the respective character as shown in Figure 1, (iv).

Unfortunately, Wiktionary is of course not complete and does not contain translations to all languages for all English patterns. However, to allow backtracking, we add the English information to the resource files of all languages. If, at a later point, the automatically created resources are used as basis for developing better resources for specific languages, missing translations can be easily detected and manually added.

[4] `http://en.wiktionary.org/wiki/November#Translations`.

3.6 Improving and Finalizing the Resources

In particular for the rule development process, we also create the automatic resources for English. These are repeatedly evaluated on the temporally annotated English corpora TimeBank (Pustejovsky et al., 2003), TempEval-3 platinum (UzZaman et al., 2013), and WikiWars (Mazur and Dale, 2010). Analyzing the errors on these corpora, we iteratively improve the simplified English resources and the rules. In Table 1, we compare HeidelTime's original English resources (HeidelTime 1.9) with the simplified resources (HeidelTime – automatic). While details about evaluation measures will be described in Section 4, the results of the automatic approach are obviously lower, as expected due to the simplification process.

Resources for all languages are finally built by combining the language-independent resources, the automatically created pattern and normalization files, and the language-independent rules. Note that depending on the completeness of Wiktionary, the coverage of the resources – and thus also their quality – varies between languages.

4 Evaluation

For a subset of languages, temporally annotated corpora exist. Thus, we can directly evaluate our approach for these languages. For German, Spanish, Italian, French, Portuguese, Croatian, Arabic, Vietnamese, and Chinese, we compare our automatic approach with HeidelTime's manually developed resources. In addition, first results are reported for Romanian.

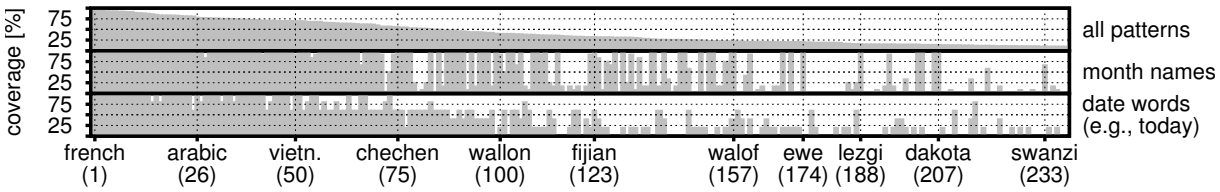


Figure 2: Language completeness statistics showing how many of the simplified English patterns have translations in Wiktionary for the 238 languages with at least 10% coverage (ordered based on "all patterns" coverage). A detailed plot with all language names is available on HeidelTime's GitHub page.

For all other languages, it is obviously difficult to judge the temporal tagging quality of the newly developed resources. However, for an estimation of the quality for languages without temporally annotated corpora, we provide completeness statistics of the pattern translations. The fewer translations are available for a language, the more likely it is that the temporal tagging quality is rather low.

4.1 Evaluation Measures

Since the task of temporal tagging is two-fold, we report precision, recall, and F1-score for the extraction task, and for the full task (relaxed extraction plus value normalization) *value F1*. Similar as the TempEval-3 organizers (UzZaman et al., 2013), we consider *value F1* with relaxed matching as most important. However, when processing large amounts of data, *value accuracy* becomes also important. Since it directly shows independent of the recall if extracted expressions are normalized correctly, it is reported, too.

4.2 Evaluation Results and Discussion

As shown in Table 1, the results of our automatically created resources are worse than those of HeidelTime with manually developed resources. For Spanish, German, French, and Portuguese, high precision, moderate recall, and good normalization results are achieved. In contrast, recall for Chinese, Croatian, and Romanian are quite low, precision is low for Romanian, and normalization accuracy for Chinese, Vietnamese and Romanian. Clearly, more work is necessary to achieve temporal tagging quality that can be considered as applicable for other applications. In particular, issues with morphology-rich languages and those without white-space tokenization have to be addressed.

Nevertheless, the results look promising for several languages taking into account that the resources are developed without any language-specific information, and that our approach has been the first step towards more sophisticated temporal tagging of all languages in the world. Note that recall is much worse than precision for all languages and that the normalization of extracted expressions works quite well (*value acc.*).

4.3 Completeness Statistics

To estimate the quality of our automatically created resources for languages without annotated corpora, we create completeness statistics. As Figure 2 shows, the completeness differs significantly between languages. However, Wiktionary translations exist for at least 75% and 50% of the patterns for 34 and 83 languages, respectively, and even for many languages with low overall coverage, important patterns such as month names and date words are well-covered. Note that even for languages with few translations, our approach can be considered as baseline since temporal tagging of most languages has not been addressed before.

5 Conclusions and Ongoing Work

In this paper, we presented our approach to automatically extend the temporal tagger HeidelTime to all languages in the world. We thus establish new temporal tagging baselines for many languages, and anyone working on any of those languages in the future can either use our resources as a starting point or as a baseline for comparison. The resources for 200+ languages are available with HeidelTime 2.0 at `http://github.com/HeidelTime/heideltime`.

Currently, we are improving the simplified English resources and the translation process by exploiting further resources such as Wikipedia. In the future, we thus plan to constantly update HeidelTime's automatically created resources.

Acknowledgments

We thank the anonymous reviewers for their valuable comments and helpful suggestions to improve the paper and to exploit the additional page.

References

Gabor Angeli and Jakob Uszkoreit. 2013. Language-Independent Discriminative Parsing of Temporal Expressions. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL '13)*, pages 83–92. ACL.

André Bittar, Pascal Amsili, Pascal Denis, and Laurence Danlos. 2011. French TimeBank: an ISO-TimeML Annotated Reference Corpus. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL '11)*, pages 130–134. ACL.

Ricardo Campos, Gaël Dias, Alípio M. Jorge, and Adam Jatowt. 2014. Survey of Temporal Information Retrieval and Related Applications. *ACM Computing Surveys*, 47(2):15:1–15:41.

Tommaso Caselli, Valentina Bartalesi Lenzi, Rachele Sprugnoli, Emanuele Pianta, and Irina Prodanof. 2011. Annotating Events, Temporal Expressions and Relations in Italian: the It-TimeML Experience for the Ita-TimeBank. In *Proceedings of the 5th Linguistic Annotation Workshop (LAW '11)*, pages 143–151. ACL.

Tommaso Caselli, Rachele Sprugnoli, Manuela Speranza, and Monica Monachine. 2014. EVENTI. EValuation of Events and Temporal INformation at Evalita 2014. In *Proceedings of the Forth International Workshop EVALITA*, pages 27–34.

Angel X. Chang and Christopher D. Manning. 2012. SUTime: A Library for Recognizing and Normalizing Time Expressions. In *Proceedings of the 8th International Conference on Language Resources and Evaluation (LREC '12)*, pages 3735–3740. ELRA.

Francisco Costa and António Branco. 2012. TimeBankPT: A TimeML Annotated Corpus of Portuguese. In *Proceedings of the 8th International Conference on Language Resources and Evaluation (LREC '12)*, pages 3727–3734. ELRA.

Corina Forascu and Dan Tufis. 2012. Romanian TimeBank: An Annotated Parallel Corpus for Temporal Information. In *Proceedings of the 8th International Conference on Language Resources and Evaluation (LREC '12)*, pages 3762–3766. ELRA.

Hui Li, Jannik Strötgen, Julian Zell, and Michael Gertz. 2014. Chinese Temporal Tagging with HeidelTime. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics (EACL '14)*, pages 133–137. ACL.

Hector Llorens, Estela Saquete, and Borja Navarro. 2010. TIPSem (English and Spanish): Evaluating CRFs and semantic roles in TempEval-2. In *Proceedings of the 5th International Workshop on Semantic Evaluation (SemEval '10)*, pages 284–291. ACL.

Giulio Manfredi, Jannik Strötgen, Julian Zell, and Michael Gertz. 2014. HeidelTime at EVENTI: Tuning Italian Resources and Addressing TimeML's Empty Tags. In *Proceedings of the Forth International Workshop EVALITA*, pages 39–43.

Pawel Mazur and Robert Dale. 2009. The DANTE Temporal Expression Tagger. In *Proceedings of the 3rd Language and Technology Conference (LTC '07)*, pages 245–257. Springer.

Pawel Mazur and Robert Dale. 2010. WikiWars: A New Corpus for Research on Temporal Expressions. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing (EMNLP '10)*, pages 913–922. ACL.

Pawel Mazur. 2012. *Broad-coverage Rule-based Processing of Temporal Expressions*. Ph.D. thesis, Macquarie University and Wroclaw University of Technology.

Véronique Moriceau and Xavier Tannier. 2014. French Resources for Extraction and Normalization of Temporal Expressions with HeidelTime. In *Proceedings of the 9th International Conference on Language Resources and Evaluation (LREC '14)*, pages 3239–3243. ELRA.

Matteo Negri, Estela Saquete, Patricio Martínez-Barco, and Rafael Muñoz. 2006. Evaluating Knowledge-based Approaches to the Multilingual Extension of a Temporal Expression Normalizer. In *Proceedings of the Workshop on Annotating and Reasoning about Time and Events (ARTE '06)*, pages 30–37. ACL.

J. Pustejovsky, P. Hanks, R. Saurí, A. See, R. Gaizauskas, A. Setzer, D. Radev, B. Sundheim, D. Day, L. Ferro, and M. Lazo. 2003. The TIMEBANK Corpus. In *Proceedings of Corpus Linguistics 2003*, pages 647–656.

James Pustejovsky, Robert Knippen, Jessica Littman, and Roser Saurí. 2005. Temporal and Event Information in Natural Language Text. *Language Resources and Evaluation*, 39(2-3):123–164.

Estela Saquete, Patricio Martínez-Barco, and Rafael Muñoz. 2004. Automatic Multilinguality for Time Expression Resolution. In *Proceedings of the Third Mexican International Conference on Artificial Intelligence (MICAI '04)*, pages 458–467.

Luka Skukan, Goran Glavaš, and Jan Šnajder. 2014. HeidelTime.Hr: Extracting and Normalizing Temporal Expressions in Croatian. In *Proceedings of the 9th Language Technologies Conference (LTC '14)*, pages 99–103. Information Society.

Kathrin Spreyer and Anette Frank. 2008. Projection-based Acquisition of a Temporal Labeller. In *Proceedings of the 3rd International Joint Conference on Natural Language Processing (IJCNLP '08)*, pages 489–496. ACL.

Jannik Strötgen and Michael Gertz. 2011. WikiWarsDE: A German Corpus of Narratives Annotated with Temporal Expressions. In *Proceedings of the Conference of the German Society for Computational Linguistics and Language Technology (GSCL '11)*, pages 129–134.

Jannik Strötgen and Michael Gertz. 2013. Multilingual and Cross-domain Temporal Tagging. *Language Resources and Evaluation*, 47(2):269–298.

Jannik Strötgen, Julian Zell, and Michael Gertz. 2013. HeidelTime: Tuning English and Developing Spanish Resources for TempEval-3. In *Proceedings of the 7th International Workshop on Semantic Evaluation (SemEval '13)*, pages 15–19. ACL.

Jannik Strötgen, Ayser Armiti, Tran Van Canh, Julian Zell, and Michael Gertz. 2014. Time for More Languages: Temporal Tagging of Arabic, Italian, Spanish, and Vietnamese. *ACM Transactions on Asian Language Information Processing (TALIP)*, 13(1):1:1–1:21.

Naushad UzZaman, Hector Llorens, Leon Derczynski, James F. Allen, Marc Verhagen, and James Pustejovsky. 2013. SemEval-2013 Task 1: TempEval-3: Evaluating Time Expressions, Events, and Temporal Relations. In *Proceedings of the 7th International Workshop on Semantic Evaluation (SemEval '13)*, pages 1–9. ACL.

Matje van de Camp and Henning Christiansen. 2012. Resolving Relative Time Expressions in Dutch Text with Constraint Handling Rules. In *Proceedings of the 7th International Workshop on Constraint Solving and Language Processing (CSLP '12)*, pages 74–85. Springer.

Marc Verhagen, Roser Saurí, Tommaso Caselli, and James Pustejovsky. 2010. SemEval-2010 Task 13: TempEval-2. In *Proceedings of the 5th International Workshop on Semantic Evaluation (SemEval '10)*, pages 57–62. ACL.

Named Entity Recognition for Chinese Social Media with Jointly Trained Embeddings

Nanyun Peng and **Mark Dredze**
Human Language Technology Center of Excellence
Center for Language and Speech Processing
Johns Hopkins University, Baltimore, MD, 21218
npeng1@jhu.edu, mdredze@cs.jhu.edu

Abstract

We consider the task of named entity recognition for Chinese social media. The long line of work in Chinese NER has focused on formal domains, and NER for social media has been largely restricted to English. We present a new corpus of Weibo messages annotated for both name and nominal mentions. Additionally, we evaluate three types of neural embeddings for representing Chinese text. Finally, we propose a joint training objective for the embeddings that makes use of both (NER) labeled and unlabeled raw text. Our methods yield a 9% improvement over a state-of-the-art baseline.

1 Introduction

Named entity recognition (NER), and more generally the task of mention detection[1], is an essential component of information extraction technologies: the first step before tasks such as relation extraction (Bunescu and Mooney, 2005) and entity linking (Dredze et al., 2010; Ratinov et al., 2011). A long line of work has focused on NER in both formal and informal domains (Collins and Singer, 1999; McCallum and Li, 2003; Nadeau and Sekine, 2007; Jin and Chen, 2008; He et al., 2012a), with recent efforts turning towards social media (Finin et al., 2010; Liu et al., 2011; Ritter et al., 2011; Fromreide et al., 2014; Li et al., 2012; Liu et al., 2012). While NER has included work on several languages, work on social media NER has largely focused on English language data.[2]

We consider NER on Chinese social media from the popular Sina Weibo service, both because of the popularity of the service (comparable in size to Twitter and previously used in NLP research (Ling et al., 2013)) and the challenges faced in processing Chinese language data. One approach is to utilize lexical embeddings to improve NER systems (Collobert and Weston, 2008; Turian et al., 2010; Passos et al., 2014), including for Twitter (Cherry and Guo, 2015). However, the use of embeddings for Chinese remains a challenge. Unlike most languages, we cannot easily assign an embedding to each Chinese word without automated segmentation, which may be unreliable, especially when we want to model informal text.[3] For this reason, state-of-the-art NER systems for Chinese do not tag words; they instead tag characters directly (Mao et al., 2008). While work has explored different embeddings for Chinese (Liu et al., 2014; Sun et al., 2014; Qiu et al., 2014; Chen et al., 2015), their inclusion in downstream tasks, such as NER, remains untested.

We explore several types of embeddings for Chinese text and their effect on Chinese social media NER. Specifically, we make the following contributions. 1) We present the first system for NER on Chinese social media using a new corpus based on Weibo messages. We consider both name and nominal mentions, with the goal of supporting downstream systems, such as coreference resolution. Notably, our results reveal that the gap between social media and traditional text for Chinese is much larger than similar corpora for English, suggesting this task as an interesting area of future work.[4] 2) We evaluate three types of embeddings for Chinese text based on their inclusion in a downstream task. We include results with and without fine-tuning. 3) We present a joint ob-

[1] Since we consider name and nominals, our work is closer to mention detection. For simplicity, we use the term NER.

[2] Etter et al. (2013) considered Spanish Twitter, which is quite similar to English from the standpoint of building models and features.

[3] Word segmentation performance is much worse on social media compared to formal text (Duan et al., 2012).

[4] Consider the overall F1 scores from Ritter et al. (2011), Cherry and Guo (2015) and Fromreide et al. (2014) compared to our best results in Table 2. This is despite the fact that Chinese NER performance on formal texts is similar to English.

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 548–554,
Lisbon, Portugal, 17-21 September 2015.

jective that trains embeddings simultaneously for both NER and language modeling. Joint training yields better results than post-hoc fine-tuning.

2 NER for Chinese Social Media

Several SIGHAN shared tasks have focused on Chinese NER (Zhang et al., 2006; Jin and Chen, 2008; He et al., 2012b; Zhu et al., 2003; Fang et al., 2004; Zhang et al., 2006), though they have been restricted to formal text, e.g. news. NER for Chinese social media remains unexplored.[5]

As is the case for other languages, social media informality introduces numerous problems for NLP systems, such as spelling errors, novel words, and ungrammatical constructions. Chinese presents additional challenges, since it uses logograms instead of alphabets, and lacks many of the clues that a word is a name, e.g. capitalization and punctuation marks. The lack of explicit word boundaries further confuses NER systems. These problems are worse in social media, which has worse word segmentation. Additionally, typical Chinese corpora use exclusively traditional or simplified characters, whereas social media mixes them. Figure 1 demonstrates some challenges.

The baseline system for our task is our own implementation of Mao et al. (2008), which is the current state-of-the-art on the SIGHAN 2008 shared task (Jin and Chen, 2008). They use a CRF tagger with a BIOSE (begin, inside, outside, singleton, end) encoding that tags individual characters, not words, since word segmentation errors are especially problematic for NER (Zhang et al., 2006). Features include many common English NER features, e.g. character unigrams and bigrams, with context windows of size 5. See Mao et al. (2008) for complete details on their system.

Mao et al. (2008) use a two pass approach, training a CRF first for mention detection and using the resulting predictions as a feature for an NER system. Furthermore, they make extensive use of gazetteer features. For simplicity, we exclude the first pass mention detection and the gazetteer features, which make only small improvements to their overall performance. We note that other implementations of this system (Zhang et al., 2013) have been unable to match the performance reported in Mao et al. (2008). Similarly, our implementation yields results on SIGHAN 2008 similar

[5]Yang et al. (2014) consider a related problem of identifying product mentions in Weibo messages.

有好多好多的话想对你说李巾凡想要瘦瘦瘦成李帆我是想切开云朵的心
Have many many words to say to you Jinfan Li wanna thin thin thin to Fan Li I am a heart that want to cut the cloud

美得呀～顾天池苦逼青年杨素晗闵日记肖立伟嘻嘻嘻嘻嘻嘻美啊
Beautiful Tianchi Gu bitter youth Suhan Yang Riji Min Liwei Xiao hahahahahaha beautiful

看见前女友和她的新欢走在一起的时候，已经无处可躲了，只好硬着 头皮上去打招呼哎呀，好久不见，你儿子都这么高了。
When saw ex-girl friend and her new partner coming across, nowhere to hide, have to say hello, long time no see, your son grown up.

Figure 1: Examples of Weibos messages and translations with named (red) and nominal (blue) mentions.

to those reported in Zhang et al. (2013).[6] Overall, we take this tagger as representative of state-of-the-art for Chinese NER.

3 Embeddings for Chinese Text

Lexical embeddings represent words in a continuous low dimensional space, which can capture semantic or syntactic properties of the lexicon: similar words would have similar low dimensional vector representations. Embeddings have been used to gain improvements in a variety of NLP tasks. In NER specifically, several papers have shown improvements by using pre-trained neural embeddings as features in standard NER systems (Collobert and Weston, 2008; Turian et al., 2010; Passos et al., 2014). More recently, these improvements have been demonstrated on Twitter data (Cherry and Guo, 2015). Embeddings are especially helpful when there is little training data, since they can be trained on a large amount of unlabeled data. This is the case for new languages and domains, the task we face in this paper.

However, training embeddings for Chinese is not straightforward: Chinese is not word segmented, so embeddings for each word cannot be trained on a raw corpus. Additionally, the state-of-the-art systems for downstream Chinese tasks, such as NER, may not use words.

We present three types of Chinese embeddings that will be trained on a large corpus of Weibo messages. These embeddings will be used as features in the NER system by adding a (real valued) feature for each dimension of the embedding for the current word/character.

Word Embeddings We train an embedding for each word type, the standard approach in other languages. We run a Chinese word segmentation

[6]Our implementation obtains an F1 of 88.63%.

system[7] over the raw corpus of Weibo messages. To create features, we first segment the NER data, and then lookup the embedding that matches the segmented word. Since the NER system tags characters, we add the same word embedding features to each character in the word.

Character Embeddings We learn an embedding for each character in the training corpus (Sun et al., 2014; Liu et al., 2014).This removes the dependency on pre-processing the text, and better fits our intended use case: NER tagging over characters. Since there are many fewer characters than words, we learn many fewer embeddings. On the one hand, this means fewer parameters and less over-fitting. However, the reduction in parameters comes with a loss of specificity, where we may be unable to learn different behaviors of a character in different settings. We explore a compromise approach in the next section. These embeddings are directly incorporated into the NER system by adding embedding features for each character.

Character and Position Embeddings Character embeddings cannot distinguish between uses of the same character in different contexts, whereas word embeddings fail to make use of characters or character n-grams that are part of many words. A compromise is to use character embeddings that are sensitive to the character's position in the word (Chen et al., 2015). We first word segment the corpus. For each character in each word, we add a positional tag, e.g. the first/second/etc. character in the word, yielding multiple embeddings per character. We learn separate embeddings for each positionally tagged character. To use these embeddings as features, we segment the NER text, obtain position tags for each character, and add features for the corresponding embedding.

These three methods lead to 179,809 word embeddings, 10,912 character embeddings, and 24,818 character with position embeddings.

3.1 Fine Tuning

For each of the embeddings, we fine-tune pre-trained embeddings in the context of the NER task. This corresponds to initializing the embeddings parameters using a pre-trained model, and then modifying the parameters during gradient updates of the NER model by back-propogating gradients. This is a standard method that has been previously explored in sequential and structured prediction problem (Collobert et al., 2011; Zheng et al., 2013; Yao et al., 2014; Pei et al., 2014).

3.2 Joint Training Objectives

Fine-tuning has a disadvantage: it can arbitrarily deviate from the settings obtained from training on large amounts of raw text. Recent work has instead tuned embeddings for a specific task, while maintaining information learned from raw text. Yu and Dredze (2014) use multi-part objectives that include both standard unlabeled objectives, such as skip-gram models in word2vec, and task specific objectives. Jointly training the embeddings with the multi-part objectives allows the fine-tuned embeddings to further influence other embeddings, even those that do not appear in the labeled training data. This type of training can help improve OOVs (Yu and Dredze, 2015), an important aspect of improving social media NER.

We propose to jointly learn embeddings for both language models and the NER task. The modified objective function (log-likelihood) for the CRF is given by:

$$\mathcal{L}_s(\boldsymbol{\lambda}, \boldsymbol{e_w}) = \frac{1}{K} \sum_k \left[\log \frac{1}{Z(x)^k} + \sum_j \lambda_j F_j(\boldsymbol{y}^k, \boldsymbol{x}^k, \boldsymbol{e_w}) \right],$$

where K is the number of instances, $\boldsymbol{\lambda}$ is the weight vector, $\boldsymbol{x}^k$ and $\boldsymbol{y}^k$ are the words and labels sequence for each instance, $\boldsymbol{e_w}$ is the embedding for a word/character/character-position representation w, $Z(x)^k$ is the normalization factor for each instance, and $F_j(\boldsymbol{y}^k, \boldsymbol{x}^k, \boldsymbol{e_w}) = \sum_{i=1}^{n} f_j(y_{i-1}^k, y_i^k, \boldsymbol{x}^k, \boldsymbol{e_w}, i)$ represents the feature function in which j denotes different feature templates and i denotes the position index in a sentence. This differs from a traditional CRF in that the feature function depends on the additional variables $\boldsymbol{e_w}$, which are the embeddings (as defined above). As a result, the objective is no longer log-linear, but log-bilinear [8].

[7] We use Jieba for segmentation: `https://github.com/fxsjy/jieba`

[8] It is log-bilinear because the log-likelihood takes the form $f(x, y) = axy + bx + cy$, where x, y are variables and a, b, c are coefficients. In this case, x is the feature weight and y is the embedding; both of them are vectors. Taking the partial derivative with respect to any one of the variables, one gets a constant (wrt that variable). This satisfies the definition of log-bilinear functions.

The second term is the standard skip-gram language model objective (Mikolov et al., 2013):

$$\mathcal{L}_u(\boldsymbol{e_w}) = \frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0} \log p(w_{t+j}|w_t), \quad (1)$$

where

$$p(w_i|w_j) = \frac{\exp\left(\boldsymbol{e}_{\boldsymbol{w_i}}^T \boldsymbol{e}_{\boldsymbol{w_j}}\right)}{\sum_{i'} \exp\left(\boldsymbol{e}_{\boldsymbol{w_{i'}}}^T \boldsymbol{e}_{\boldsymbol{w_j}}\right)}.$$

The first objective is notated $\mathcal{L}_s$ for "supervised" (trained on labeled NER data), and the second is $\mathcal{L}_u$, "unsupervised" (trained on raw text.) Both objectives share the same variables $\boldsymbol{e_w}$. The overall goal is to maximize their weighted sum:

$$\arg\max_{e_w} = \mathcal{L}_s(\boldsymbol{\lambda}, \boldsymbol{e_w}) + C\mathcal{L}_u(\boldsymbol{e_w}) \quad (2)$$

where C is a tradeoff parameter.

3.3 Parameter Estimation

We pre-trained embeddings using word2vec (Mikolov et al., 2013) with the skip-gram training objective and NEC negative sampling. Unless otherwise stated, we used word2vec's default parameter settings. All embeddings were 100-dimensional, and we used the same embeddings for the input and output parameters in the skip-gram objective. We optimized the joint objective (2) using an alternative optimization strategy: we alternated 30 iterations of CRF training on the NE labeled data and 5 multi-threaded passes through both the labeled and unlabeled data for the skip-gram objective. We avoided over-fitting using early-stopping. For simplicity, we set $C = 1$ for (2). The CRF was trained using stochastic gradient descent with an L2 regularizer. All model hyper-parameters were tuned on dev data.

We use the off-the-shelf tool word2vec (Mikolov et al., 2013) to do skip-gram training for language model, and implement our own CRF model to modify the embeddings. We optimize (2) by alternating the optimzation of each of the two objectives.

4 Weibo NER Corpus

We constructed a corpus of Weibo messages annotated for NER. We followed the DEFT ERE (Linguistics Data Consortium, 2014) [9] annotation guidelines for entities, which includes four major semantic types: person, organization, location and geo-political entity. We annotated both name and nominal mentions. Chinese pronoun mentions can be easily recognized with a regular expression. We used Amazon Mechanical Turk, using standard methods of multiple annotators and including gold examples to ensure high quality annotations (Callison-Burch and Dredze, 2010).

	Mentions		
Entity Type	**Name**	**Nominal**	**Total**
Geo-political	243	0	243
Location	88	38	126
Organization	224	31	255
Person	721	636	1,357

Table 1: Mention statistics for the Weibo NER corpus.

Our corpus includes 1,890 messages sampled from Weibo between November 2013 and December 2014. Rather than selecting messages at random, which would yield a small number of messages with entities, we selected messages that contained three or more (segmented) words that were not in a fixed vocabulary of common Chinese words. Initial experiments showed this gave messages more likely to contain entities.

Table 1 shows statistics of the final corpus. We divided the corpus into 7 folds, each with 127 messages, where each message corresponds to a single instance. We use the first 5 folds for train, the 6th for development, and the 7th for test. We make our code and the annotated corpus available.[10]

We constructed an additional corpus of unlabeled messages for training the embeddings. We randomly selected 2,259,434 messages from the same time period as above.

5 Experiments

We evaluate our methods under two settings: training on only name mentions, and training on both name and nominal mentions. We re-train the Stanford NER system (Finkel et al., 2005) as a baseline; besides, we also evaluate our implementation of the CRF from Mao et al. (2008) as described in §2 as *Baseline Features*. To this baseline, we add each of our three embedding models: *word*, *character*, *character+position* (as described in §3), and report results on the modified

[9] See Aguilar et al. (2014) for a comparison of DEFT ERE with other common standards.

[10] https://github.com/hltcoe/golden-horse

Method	Dev						Test					
	Without Fine Tuning			With Fine Tuning			Without Fine Tuning			With Fine Tuning		
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
Stanford	63.51	23.27	34.06				55.70	22.86	33.06			
Baseline Features	63.51	27.17	38.06				56.98	25.26	35.00			
+ word	65.71	26.59	37.86	70.97	25.43	37.45	56.82	25.77	35.46	64.94	25.77	36.90
+ character	53.54	30.64	38.97	58.76	32.95	42.22	56.48	31.44	40.40	57.89	34.02	42.86
+ character+position	60.87	32.37	42.26	61.76	36.42	**45.82**	61.90	33.51	43.48	57.26	34.53	43.09
Joint (cp)				57.41	35.84	44.13				57.98	35.57	**44.09**
Stanford	72.39	31.80	44.19				63.96	22.19	32.95			
Baseline Features	71.94	33.22	45.45				60.16	23.87	34.18			
+ word	69.66	33.55	45.29	70.67	35.22	47.01	59.40	25.48	35.67	60.68	22.90	33.26
+ character	58.76	32.95	42.22	66.88	35.55	46.42	58.28	28.39	38.18	55.15	29.35	38.32
+ character+position	73.43	34.88	47.30	69.38	36.88	48.16	65.91	28.06	39.37	62.33	29.35	39.91
Joint (cp)				72.55	36.88	**48.90**				63.84	29.45	**40.38**

Table 2: NER results for name mentions (top) and name + nominal mentions (bottom).

CRF model with and without fine-tuning. We also report results for the joint method trained with the character+position model (cp), which performed the best on dev data for joint training.

General Results Table 2 shows results for both dev (tuned) and test (held out) splits. First, we observe that the results for the baseline are significantly below those for SIGHAN shared tasks as well as the reported results on Twitter NER, showing the difficulty of this task. In particular, recall is especially challenging. Second, all embeddings improve the baseline on test data, but the character + position model gets the best results. Fine-tuning improves embedding results, but seems to overfit on dev data. Finally, our joint model does the best in both conditions (name and name+nominal) on test data, improving over fine-tuning, yielding up to a 9% (absolute) improvement over a strong baseline.

Effect of Embeddings We expect improvements from embeddings to be larger when there is less training data. Figure 2 shows F1 on dev data for different amounts of training data, from 200 instances up to 1400, for the character + position embeddings versus the baseline model. We see that for both settings, we see larger improvements from embeddings for smaller training sets.

Error Analysis Since the results are relatively low, we conducted an error analysis by randomly sampling 150 error items and manually looking through them. Among the 150 examples, 65 are annotation errors, majorly cause by annotators neglecting some mentions, this contributes 43% of the errors. The second largest error source are the person names: Chinese person names are very flexible and nearly every character can be used in given names, this makes recognizing person names challenging and contributes to 9% of our errors. The following largest source of error are transliterated foreign names, which contributes to 7% of the errors. Other sources including boundary error, type error, name abbreviation, nicknames, etc.

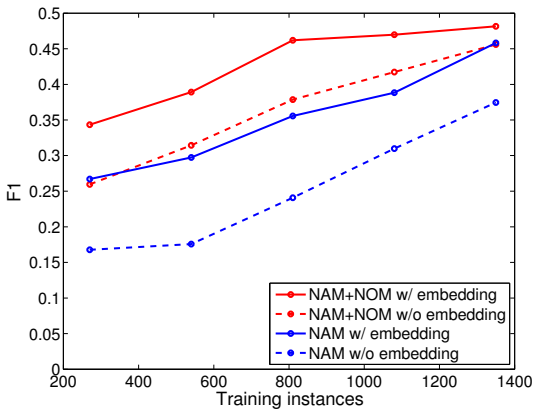


Figure 2: Dev F1 for varying number of training instances.

6 Conclusion

Our results show that NER for Chinese social media remains a challenging task, results lag behind both formal Chinese text and English Twitter. Nevertheless, our embeddings, combined with our joint training objective, provide a large improvement over a state-of-the-art model.

Acknowledgement We would like to thank the reviewers for their helpful comments and perspectives. We thank Mo Yu, Kevin Duh, Jiang Guo and Wenzhe Pei for the insightful discussions and Xuezhe Ma for help running the Stanford baseline.

References

Jacqueline Aguilar, Charley Beller, Paul McNamee, Benjamin Van Durme, Stephanie Strassel, Zhiyi Song, and Joe Ellis. 2014. A comparison of the events and relations across ACE, ERE, TAC-KBP, and FrameNet annotation standards. In *ACL Workshop: EVENTS*.

Razvan C Bunescu and Raymond J Mooney. 2005. A shortest path dependency kernel for relation extraction. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 724–731. Association for Computational Linguistics.

Chris Callison-Burch and Mark Dredze. 2010. Creating speech and language data with Amazon's Mechanical Turk. In *NAACL Workshop on Creating Speech and Language Data With Mechanical Turk*.

Xinxiong Chen, Lei Xu, Zhiyuan Liu, Maosong Sun, and Huanbo Luan. 2015. Joint learning of character and word embeddings. In *International Joint Conference on Artificial Intelligence (IJCAI'15)*.

Colin Cherry and Hongyu Guo. 2015. The unreasonable effectiveness of word representations for twitter named entity recognition. In *North America Chapter of Association for Computational Linguistics (NAACL)*. Association for Computational Linguistics.

Michael Collins and Yoram Singer. 1999. Unsupervised models for named entity classification. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 100–110. Citeseer.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *International Conference on Machine Learning (ICML)*, pages 160–167. ACM.

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

Mark Dredze, Paul McNamee, Delip Rao, Adam Gerber, and Tim Finin. 2010. Entity disambiguation for knowledge base population. In *Conference on Computational Linguistics (Coling)*.

Huiming Duan, Zhifang Sui, Ye Tian, and Wenjie Li. 2012. The cips-sighan clp 2012 chineseword segmentation onmicroblog corpora bakeoff. In *Second CIPS-SIGHAN Joint Conference on Chinese Language Processing*, pages 35–40, Tianjin, China, December. Association for Computational Linguistics.

David Etter, Francis Ferraro, Ryan Cotterell, Olivia Buzek, and Benjamin Van Durme. 2013. Nerit: Named entity recognition for informal text. Technical report, Technical Report 11, Human Language Technology Center of Excellence, Johns Hopkins University, July.

Xiaoshan Fang, Jianfeng Gao, and Huanye Sheng. 2004. A semi-supervised approach to build annotated corpus for chinese named entity recognition. In Oliver Streiter and Qin Lu, editors, *ACL SIGHAN Workshop 2004*, pages 129–133, Barcelona, Spain, July. Association for Computational Linguistics.

Tim Finin, William Murnane, Anand Karandikar, Nicholas Keller, Justin Martineau, and Mark Dredze. 2010. Annotating named entities in twitter data with crowdsourcing. In *NAACL Workshop on Creating Speech and Language Data With Mechanical Turk*.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Association for Computational Linguistics (ACL)*, pages 363–370. Association for Computational Linguistics.

Hege Fromreide, Dirk Hovy, and Anders Søgaard. 2014. Crowdsourcing and annotating NER for Twitter# drift. In *LREC*.

Zhengyan He, Houfeng Wang, and Sujian Li. 2012a. The task 2 of cips-sighan 2012 named entity recognition and disambiguation in chinese bakeoff. In *Second CIPS-SIGHAN Joint Conference on Chinese Language Processing*, pages 108–114, Tianjin, China, December. Association for Computational Linguistics.

Zhengyan He, Houfeng Wang, and Sujian Li. 2012b. The task 2 of cips-sighan 2012 named entity recognition and disambiguation in chinese bakeoff. In *Second CIPS-SIGHAN Joint Conference on Chinese Language Processing*, pages 108–114, Tianjin, China, December. Association for Computational Linguistics.

Guangjin Jin and Xiao Chen. 2008. The fourth international chinese language processing bakeoff: Chinese word segmentation, named entity recognition and chinese pos tagging. In *Sixth SIGHAN Workshop on Chinese Language Processing*, page 69. Citeseer.

Chenliang Li, Jianshu Weng, Qi He, Yuxia Yao, Anwitaman Datta, Aixin Sun, and Bu-Sung Lee. 2012. Twiner: Named entity recognition in targeted twitter stream. In *SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '12, pages 721–730, New York, NY, USA. ACM.

Wang Ling, Guang Xiang, Chris Dyer, Alan Black, and Isabel Trancoso. 2013. Microblogs as parallel corpora. In *Association for Computational Linguistics (ACL)*. Association for Computational Linguistics.

Linguistics Data Consortium. 2014. DEFT ERE Annotation Guidelines: Entities.

Xiaohua Liu, Shaodian Zhang, Furu Wei, and Ming Zhou. 2011. Recognizing named entities in tweets. In *Association for Computational Linguistics (ACL)*, pages 359–367. Association for Computational Linguistics.

Xiaohua Liu, Ming Zhou, Furu Wei, Zhongyang Fu, and Xiangyang Zhou. 2012. Joint inference of named entity recognition and normalization for tweets. In *Association for Computational Linguistics (ACL)*, ACL '12, pages 526–535, Stroudsburg, PA, USA. Association for Computational Linguistics.

Xiaodong Liu, Kevin Duh, Yuji Matsumoto, and Tomoya Iwakura. 2014. Learning character representations for chinese word segmentation. In *NIPS 2014 Workshop on Modern Machine Learning and Natural Language Processing*.

Xinnian Mao, Yuan Dong, Saike He, Sencheng Bao, and Haila Wang. 2008. Chinese word segmentation and named entity recognition based on conditional random fields. In *IJCNLP*, pages 90–93.

Andrew McCallum and Wei Li. 2003. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In *North America Chapter of Association for Computational Linguistics (NAACL)*.

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

David Nadeau and Satoshi Sekine. 2007. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26.

Alexandre Passos, Vineet Kumar, and Andrew McCallum. 2014. Lexicon infused phrase embeddings for named entity resolution. *CoRR*, abs/1404.5367.

Wenzhe Pei, Tao Ge, and Chang Baobao. 2014. Max-margin tensor neural network for chinese word segmentation. In *Association for Computational Linguistics (ACL)*.

Siyu Qiu, Qing Cui, Jiang Bian, Bin Gao, and Tie-Yan Liu. 2014. Co-learning of word representations and morpheme representations. In *Conference on Computational Linguistics (Coling)*.

Lev Ratinov, Dan Roth, Doug Downey, and Mike Anderson. 2011. Local and global algorithms for disambiguation to wikipedia. In *Association for Computational Linguistics (ACL)*, pages 1375–1384. Association for Computational Linguistics.

Alan Ritter, Sam Clark, Oren Etzioni, et al. 2011. Named entity recognition in tweets: an experimental study. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1524–1534. Association for Computational Linguistics.

Yaming Sun, Lei Lin, Nan Yang, Zhenzhou Ji, and Xiaolong Wang. 2014. Radical-enhanced chinese character embedding. In *Neural Information Processing Systems (NIPS)*, pages 279–286. Springer.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Association for Computational Linguistics (ACL)*, pages 384–394. Association for Computational Linguistics.

Xianxiang Yang, Heyan Huang, Xin Xin, Quanchao Liu, and Xiaochi Wei. 2014. Domain-specific product named entity recognition from chinese microblog. In *Computational Intelligence and Security (CIS), 2014 Tenth International Conference on*, pages 218–222. IEEE.

Kaisheng Yao, Baolin Peng, Geoffrey Zweig, Dong Yu, Xiaolong Li, and Feng Gao. 2014. Recurrent conditional random field for language understanding. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 4077–4081. IEEE.

Mo Yu and Mark Dredze. 2014. Improving lexical embeddings with semantic knowledge. In *Association for Computational Linguistics (ACL)*, pages 545–550.

Mo Yu and Mark Dredze. 2015. Learning composition models for phrase embeddings. *Transactions of the Association for Computational Linguistics*, 3:227–242.

Suxiang Zhang, Ying Qin, Juan Wen, and Xiaojie Wang. 2006. Word segmentation and named entity recognition for sighan bakeoff3. In *Fifth SIGHAN Workshop on Chinese Language Processing*, pages 158–161, Sydney, Australia, July. Association for Computational Linguistics.

Longkai Zhang, Houfeng Wang, Xu Sun, and Mairgup Mansur. 2013. Exploring representations from unlabeled data with co-training for chinese word segmentation. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Xiaoqing Zheng, Hanyang Chen, and Tianyu Xu. 2013. Deep learning for chinese word segmentation and pos tagging. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 647–657.

Xiaodan Zhu, Mu Li, Jianfeng Gao, and Chang-Ning Huang. 2003. Single character chinese named entity recognition. In *Second SIGHAN Workshop on Chinese Language Processing*, pages 125–132, Sapporo, Japan, July. Association for Computational Linguistics.

Inferring Binary Relation Schemas for Open Information Extraction

Kangqi Luo[1] and **Xusheng Luo**[2] and **Kenny Q. Zhu**[3]
Department of Computer Science & Engineering
Shanghai Jiao Tong University, Shanghai, China
[1]luokangqi@sjtu.edu.cn [2]freefish_6174@sjtu.edu.cn [3]kzhu@cs.sjtu.edu.cn

Abstract

This paper presents a framework to model the semantic representation of binary relations produced by open information extraction systems. For each binary relation, we infer a set of preferred types on the two arguments simultaneously, and generate a ranked list of type pairs which we call schemas. All inferred types are drawn from the Freebase type taxonomy, which are human readable. Our system collects 171,168 binary relations from ReVerb, and is able to produce top-ranking relation schemas with a mean reciprocal rank of 0.337.

1 Introduction

Open information extraction (or Open IE) is a task of extracting all sorts of relations between named entities or concepts from open-domain text corpora, without restraining itself to specific relations or patterns. State-of-the-art Open IE systems (Carlson et al., 2010; Fader et al., 2011; Schmitz et al., 2012; Nakashole et al., 2012) extract millions of binary relations with high precision from the web corpus. Each extracted relation instance is a triple of the form $\langle arg_1,\ rel,\ arg_2\rangle$, where the relation rel is a lexical or syntactic pattern, and both arguments are multi-word expressions representing the argument entities or concepts.

Whereas Open IE provides concrete relation instances, we are interested in generalizing these instances into more abstract semantic representations. In this paper, we focus on inferring the schemas of binary relations.

For example, given the binary relation "play in", an Open IE system extracts many triples of the form $\langle X,\ play\ in,\ Y\rangle$. The following relation triples are extracted in ReVerb:

$$\langle \text{Goel Grey}, played\ in, \text{Cabaret}\rangle$$
$$\langle \text{Tom Brady}, play\ in, \text{National Football League}\rangle$$

Informally, the goal of our system is to automatically infer a set of schemas such as $\langle t_1,\ play\ in,\ t_2\rangle$, where t_1 and t_2 are two semantic types drawn from a standard knowledge base such as WordNet (Miller, 1995), Yago (Suchanek et al., 2007), Freebase (Bollacker et al., 2008), and Probase (Wu et al., 2012), and each such schema can be used to represent a set of "play in" relation instances. For the above example, two possible schemas for "play in" are:

$$\langle \textbf{film actor},\ play\ in,\ \textbf{film}\rangle$$
$$\langle \textbf{athlete},\ play\ in,\ \textbf{sports league}\rangle$$

The schema of a binary relation is useful information in NLP tasks, such as context-oriented entity recognition and open domain question answering. Suppose we are to recognize the entities in the sentence "*Granger* played in *the NBA*". "*Granger*" is a highly ambiguous term, while "*the NBA*" is probably a sports league. Then with the the above relation schemas for "play in", the entity recognizer knows that "*Granger*" is more likely to be an athlete, which results in the correct linking to "Danny Granger", who is an NBA player, even though the Open IE has never extracted such fact before.

One relevant technique to achieve our goal is *selectional preference* (SP) (Resnik, 1996; Erk, 2007; Ritter et al., 2010), which computes the most appropriate types for a specific argument of a predicate. SP is based on the idea of mutual information (Erk, 2007), which tends to select types which are *unique* to the relation. In other words, common types which can be used for many different relations are less preferred. However, in Open IE, many relations are related or even similar, e.g., *play in*, *take part in* and *be involved in*. There's no reason for these relations not to share schemas. Therefore in this paper, our problem is, given a re-

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 555–560,
Lisbon, Portugal, 17-21 September 2015.

lation and its instances, identify the smallest types that can cover as many instances as possible. Our approach first attempts to link the arguments in the relation instances to a set of possible entities in a knowledge base, hence generate a set of $\langle e_1, e_2 \rangle$ entity pairs. Then we select a pair of types $\langle t_1, t_2 \rangle$ that covers maximum number of entity pairs. We resolve ties by selecting the smaller (more specific) types according to a type taxonomy inferred from knowledge base.

This paper makes the following contributions: i) we defined the schema inference problem for binary relations from Open IE; ii) we developed a prototype system based on Freebase and entity linking (Lin et al., 2012; Ratinov et al., 2011; Hoffart et al., 2011; Rao et al., 2013; Cai et al., 2013), which simultaneously models the type distributions of two arguments for each binary relation; iii) our experiment on ReVerb triples showed that the top inferred schemas receive decent mean reciprocal rank (MRR) of 0.337, with respect to the human labeled ground truth.

2 Problem Definition

A knowledge base K is a 5-tuple $\langle E, Alist, T, P, IsA \rangle$, where:

- E is a finite set of entities $e \in E$,
- $Alist(e) = \{n_1, n_2, ...\}$ is a function which returns a set of names (or aliases) of an entity,
- T is a finite set of types $t \in T$,
- P is a finite set of relation instances $p(e_1, e_2)$, where p is a predicate in K.
- IsA is a finite set entity-type pairs (e, t), representing the isA relation between entities and types. An entity belongs to at least one type.

An Open IE triple set S contains all relation instances extracted by the IE system, of the form $\langle a_1,\ rel,\ a_2 \rangle$, where a_1 and a_2 are the arguments of extracted relation pattern rel. The set of argument pairs sharing the same relation pattern rel is denoted by S_{rel}.

The problem is, for each S_{rel}, return a set of type pairs (or schemas) from T, $\langle t_1, t_2 \rangle$, ordered by the number of argument pairs covered in S_{rel}. If two schemas cover the same number of argument pairs from S_{rel}, the schema covering smallest number of entities wins.

3 System

The workflow of our system is shown in Figure Figure 1. The system takes Open IE relation tuples as the input, then performs entity linking, relation grouping and schema ranking to translate them into final ranked list of schemas.

(1) Entity Linking: Relation arguments are linked to entities in the knowledge base by fuzzy string matching. Each entity in the knowledge base has a unique identifier.

(2) Relation Grouping: Linked tuples sharing similar relation patterns are grouped together. Besides, each group has a representative relation pattern, which is generated from all the patterns within the group.

(3) Schema Ranking: For each linked tuple in one relation group, argument entities are transformed into types drawn from the knowledge base. Then this procedure ranks type pairs (schemas) in terms of how much Open IE tuples a type pair can cover and how specific a type concept is.

3.1 Entity Linking

In the entity linking step, by matching arguments to entities in the knowledge base, each relation tuple is transformed into linked tuples, $ltup = \langle e_1,\ rel,\ e_2 \rangle$, with linking scores.

We aim to support fuzzy matching between arguments and entity aliases, so we take all the aliases into consideration, and build an inverted index from words to aliases. Different words in one alias cannot be treated equally. Intuitively, a word is more important if it occurs in fewer aliases (n), and vice versa. Based on the inverted index, we use inverted document frequency score to approximately model the weight of a word w:

$$idf(w) = 1 \,/\, \log(|\{n : w \in n\}|) \qquad (1)$$

Besides, stop words are removed from aliases, treating their idf scores as 0. In order to measure the probability of fuzzy matching from an argument (a) to an alias (n), we introduce the weighted overlap score:

$$overlap(a, n) = \frac{\sum_{w \in a \cap n} idf(w)}{\sum_{w \in a \cup n} idf(w)} \qquad (2)$$

We merge all the aliases of an entity together to producing a similarity score of fuzzy matching between an entity and an argument:

$$sim(e, a) = \max_{n \in Alist(e)} overlap(a, n) \qquad (3)$$

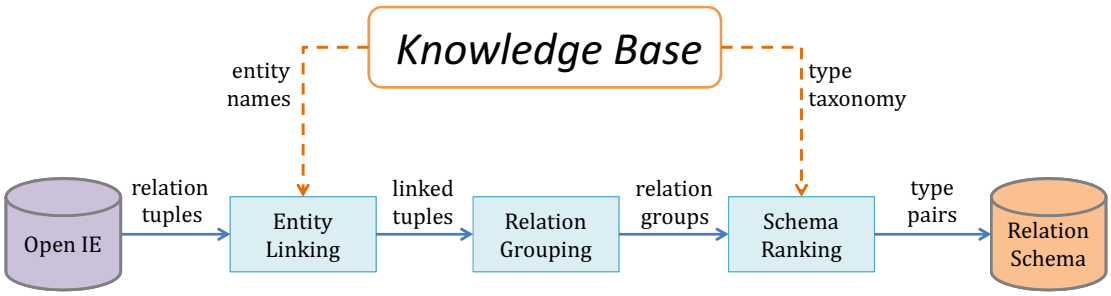


Figure 1: System Architecture

In order to control the quality of candidate entities, for an argument having m words (with stop words removed), we only keep entities that have at least one alias matching $m-1$ words in the argument, and have a similarity score larger than a threshold, τ. With similarity score computed, we generate 10 best entity candidates respectively for both the subject and the object of rel.

Next, we model the joint similarity score (F) of the relation tuple $\langle a_1, rel, a_2 \rangle$ with each entity pair combination $\langle e_1, e_2 \rangle$ in two ways. One is a **naive method** which only considers the similarity between arguments and corresponding entities:

$$F(a_1, e_1, a_2, e_2, rel) = sim(e_1, a_1) \times sim(e_2, a_2). \quad (4)$$

The other method takes predicate paths between e_1 and e_2 into consideration. Let $\vec{w}$ be the word vector of rel, and $\vec{p}$ be a path of predicates connecting e_1 and e_2 in at most 2 hops. Here we say two entities e_1 and e_2 are connected in 1 hop, if there exists a predicate p, such that $p(e_1, e_2)$ (or $p(e_2, e_1)$) is in the knowledge base.

Similarly, e_1 and e_2 are connected in 2 hops, if there exists two predicates p_1, p_2 and a transition entity e', such that $p_1(e_1, e')$ (or $p_1(e', e_1)$) and $p_2(e', e_2)$ (or $p_2(e_2, e')$) are in the knowledge base.

We hence define the relatedness between $\vec{p}$ and $\vec{w}$ in the form of a conditional probability according to the Naive Bayes model:

$$P(\vec{p}\,|\,\vec{w}) \approx \prod_p P(p\,|\,\vec{w}) \propto \prod_p P(p) \prod_w P(w\,|\,p), \quad (5)$$

and we follow the IBM alignment Model 1 (Yao and Van Durme, 2014) to calculate the conditional probability between predicates and relation words $P(\vec{p}\,|\,\vec{w})$. Based on the information above, we define a richer joint similarity score, considering all valid paths between e_1 and e_2:

$$F(a_1, e_1, a_2, e_2, rel) = sim(e_1, a_1) \times sim(e_2, a_2) \times \sum_{\vec{p}} P(\vec{p}|\vec{w}). \quad (6)$$

Due to the multiplications, the value of $P(\vec{p}|\vec{w})$ varies a lot among different entity pair candidates. The large deviation makes $P(\vec{p}|\vec{w})$ the most important term in Eq. (6), especially in the case when none of predicate paths are similar enough to the relation words. Therefore, we trust the factor of $P(\vec{p}|\vec{w})$ only when there exists a similar predicate path. In practice, we use a threshold ρ to control whether to use Eq. (6) or Eq. (4). We call this an **ensemble method**. For each case of entity linking, if there exists one candidate entity pair satisfying $P(\vec{p}\,|\,\vec{w}) > \rho$, we use the ensemble method, otherwise we fall back to the naive method for the current case.

3.2 Relation Grouping

In the step of relation grouping, linked tuples with similar relation patterns form a group. Each linked tuple belongs to one unique group.

The idea is to simplify relation patterns by syntactic transformations. If two patterns share the same simplified pattern, we treat them as being equivalent and put them into one group. First, since adjectives, adverbs and modal verbs can hardly change the type distribution of arguments in a relation, we remove these words from a pattern. Second, many relations from Open IE contain verbs, which come in different tenses. We transform all tenses into present tense. In addition, passive voice in a pattern, if any, is kept in the transformed pattern. A simple example below shows a group of relations:

⟨X, *resign from*, Y⟩
⟨X, *had resigned from*, Y⟩
⟨X, *finally resigned from*, Y⟩

All linked tuples with the same simplified pattern form a group. This pattern is selected as the representative pattern, like the pattern "*resign from*" in the above example.

3.3 Schema Ranking

Given a relation group, the step of schema ranking produces a ranked list of relation schemas with two constraints. Take "play in" as an example, the ideal schemas will contain the pair $\langle actor,\ film\rangle$ and $\langle athlete,\ sports\ league\rangle$

Each linked tuple $\langle e_1,\ rel,\ e_2\rangle$ supports the type pair $\langle t_1,\ t_2\rangle$ where $(e_1, t_1),\ (e_2, t_2) \in IsA$ in the knowledge base. We treat these pairs equally, since it's not trivial to tell which type is more related to the argument given the relation tuple as context. Combining all tuples in one group, we define the support of a type pair tp in a group (using the representative pattern r to stand for the group):

$$sup_r(\langle t_1, t_2\rangle) = \{(e_1, t_1) \in IsA,\ (e_2, t_2) \in IsA\} \quad (7)$$

A simple intuition is to rank schemas by the size of the support. Since one entity belongs to multiple types, relation schemas with general types will be ranked higher. However, two different schemas may share the same support. For instance, given the relation "*X die in Y*", suppose Open IE extractions and entity linking step returns correct results, the schema $\langle person,\ location\rangle$ and $\langle deceased\ person,\ location\rangle$ have identical supports. The latter one shows a more concrete representation of the relation, because *deceased person* covers small entities than *person* in the knowledge base.

Therefore, the schemas cannot be ranked by using the support alone. Next, we aim to extract the subsumption relations between types in the knowledge base, building the taxonomy of types.

We first define all entities in t as

$$cover(t) = \{e \mid (e, t) \in IsA\}. \quad (8)$$

Intuitively, type t_1 is subsumed in t_2, if all entities in t_1 also belong to t_2, that is, $cover(t_1) \subseteq cover(t_2)$. This uses the idea of **strict set inclusion**. For example, we can learn that the type *person* subsumes types such as *actor, politician* and *deceased person*.

However, strict set inclusion doesn't always hold in the knowledge base. For example, entities in type *award winner* are mostly *person*, but there still has some organizations in it. The strict method fails to find the subsumption relation between *award winner* and *person*, while this subsumption actually holds with a large confidence.

To resolve this problem, we use a **relaxed set inclusion**, where the set $cover(t_1)$ can be a subset of another set $cover(t_2)$ to a certain degree. We define the degree of the subsumption as the ratio between the number of entities in the two sets:

$$deg(t_1 \subseteq t_2) = \frac{|cover(t_1) \cap cover(t_2)|}{|cover(t_1)|}. \quad (9)$$

If $deg(t_1 \subseteq t_2) > \epsilon$, then t_1 is subsumed by t_2, and ϵ is a confidence parameter determined by weight tuning. By scanning all types in the knowledge base, all subsumption relations with enough confidence are extracted, forming our type taxonomy.

With a type hierarchy computed by above relaxed set inclusion, we can define a schema $\langle t_1,\ t_2\rangle$ subsumes another schema $\langle t_3,\ t_4\rangle$ if i) t_1 subsumes t_3 and t_2 subsumes t_4; ii) t_1 subsumes t_3 and $t_2 = t_4$; or iii) t_2 subsumes t_4 and $t_1 = t_3$.

If a schema (type pair) tp_1 subsumes another schema tp_2, and their supports ($|sup_r(tp)|$) are approximately equal, we give the more specific schema tp_2 a higher rank in the output list. Here two supports are roughly equal if:

$$\frac{|sup_r(tp_1)| - |sup_r(tp_2)|}{max(|sup_r(tp_1)|,\ |sup_r(tp_2)|)} < \lambda \quad (10)$$

Where λ is a threshold determined in the experiments.

4 Evaluation

Freebase (Bollacker et al., 2008) is a collaboratively generated knowledge base, which contains more than 40 million entities, and more than 1,700 real types [1]. In our experiment, We use the 16 Feb. 2014 dump of Freebase as the knowledge base.

ReVerb (Fader et al., 2011) is an Open IE system which aims to extract verb based relation instances from web corpora. The release ReVerb dataset contains more than 14 millions of relation tuples with high quality. We observed that in ReVerb, some argument is unlikely to be an entity in Freebase, for example:

$\langle Metro\ Manila,\ consists\ of,$ **12 cities**$\rangle$,

[1] Freebase types are identified by type id, for example, $sports.pro\_athlete$ stands for "professional athlete".

where the object argument is not an entity but a type. Since types are usually represented by lowercase common words, we remove the tuple if one argument is lowercase, or if it is made up completely of common words in WordNet. In addition, because date/time such as "Jan. 16th, 1981" often occurs in the object argument while Freebase does not have any such specific dates as entities, we use SUTime (Chang and Manning, 2012) to recognize dates as an virtual entity. After cleaning, the system collects 3,234,208 tuples and 171,168 relation groups.

The following parameters are tuned using a development set: $\tau = 0.667$, $\epsilon = 0.6$, $\lambda = 5\%$ and $\rho = e^{-50}$. For relation grouping, we use Stanford Parser (Klein and Manning, 2003) to perform POS tagging, lemmatizing and parsing on relations.

We first evaluate the results of entity linking. We randomly pick 200 relation instances from ReVerb, and manually labeled arguments with Freebase entities. For both naive and ensemble strategy, we evaluate the precision, recall, F1 and MRR score on the labeled set. An output entity pair is correct, if and only if both arguments are correctly linked. Experimental results are listed in Table 1.

Table 1: Entity Linking Result

Strategy	P	R	F1	MRR
Naive	0.371	0.327	0.348	0.377
Ensemble	0.386	0.340	0.361	0.381

For the evaluation of relation schema, we first randomly pick 50 binary relations with support larger than 500 from the system. For each relation, we selected top 100 type pairs with the largest support, as what we evaluated. We assigned 3 human annotators to label the fitness score of type pair for the relation. The labeled score ranges from 0 to 3. Then we merge these 3 label sets, forming 50 gold standard rankings. When evaluating a relation schema list from our system, we calculate the MRR score (Liu, 2009) by the top schemas in the gold rankings.

For comparison, we use Pointwise Mutual Information (Church and Hanks, 1990) as our baseline model, which is used in other selectional preference tasks (Resnik, 1996). We define the association score between relation and type pair as:

$$PMI(r, tp) = p(r, tp) \log \frac{p(r, tp)}{p(r, *)p(*, tp)} \quad (11)$$

Where $p(r, tp)$ is the joint probability of relation and type pair in the whole linked tuple set, and $*$ stands for any relations or type pairs.

Table 2 shows the MRR scores by using both baseline model (PMI) and our approach. As the result shows, our approach improves the MRR score by 10.1%.

Table 2: End-to-end Schema Inference Results

Approach	MRR Score
PMI Baseline	0.306
Our Approach	0.337

Finally, Table 3 shows some example binary relations, and their schemas inferred by our system. We can see that with a well-defined type hierarchy, our system is able to extract both coarse-grained and fine-grained type information from entities, resulting in a informative type lists.

Table 3: Sample Relation Schemas

Relation	Top Schemas
be find at	$\langle location,\ location\rangle$ $\langle employer,\ location\rangle$ $\langle organization,\ location\rangle$
appear on	$\langle person,\ tv\ program\rangle$ $\langle person,\ nominated\ work\rangle$ $\langle person,\ winning\ work\rangle$
be the writer of	$\langle person,\ nominated\ work\rangle$ $\langle person,\ film\rangle$ $\langle person,\ book\ subject\rangle$

5 Conclusion

In summary, our work describes a data driven approach of relation schema inference. By maximizing the support of both arguments simultaneously, our system is able to generate human-readable type pairs for a binary relation from Open IE systems. Our experiments shows that the top ranked relation schemas for each relation are accurate according to human judges. The proposed framework can be integrated with future Open IE systems.

Acknowledgement

Kenny Q. Zhu is the contact author and was supported by NSFC grant 61373031 and NSFC-NRF Joint Research Program No. 61411140247.

References

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring

human knowledge. In *ACM SIGMOD*, pages 1247–1250. ACM.

Zhiyuan Cai, Kaiqi Zhao, Kenny Q. Zhu, and Haixun Wang. 2013. Wikification via link co-occurrence. In *ACM CIKM'13*, pages 1087–1096.

Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R Hruschka Jr, and Tom M Mitchell. 2010. Toward an architecture for never-ending language learning. In *AAAI*, volume 5, page 3.

Angel X Chang and Christopher D Manning. 2012. Sutime: A library for recognizing and normalizing time expressions. In *LREC*, pages 3735–3740.

Kenneth Ward Church and Patrick Hanks. 1990. Word association norms, mutual information, and lexicography. *Computational linguistics*, 16(1):22–29.

Katrin Erk. 2007. A simple, similarity-based model for selectional preferences. In *ANNUAL MEETING-ASSOCIATION FOR COMPUTATIONAL LINGUISTICS*, page 216.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *EMNLP*, pages 1535–1545.

Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. 2011. Robust disambiguation of named entities in text. In *Proceedings of EMNLP*, pages 782–792.

Dan Klein and Christopher D Manning. 2003. Accurate unlexicalized parsing. In *ACL*, pages 423–430.

Thomas Lin, Oren Etzioni, et al. 2012. Entity linking at web scale. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*, pages 84–88.

Tie-Yan Liu. 2009. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3):225–331.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Ndapandula Nakashole, Gerhard Weikum, and Fabian Suchanek. 2012. Patty: a taxonomy of relational patterns with semantic types. In *EMNLP*, pages 1135–1145.

Delip Rao, Paul McNamee, and Mark Dredze. 2013. Entity linking: Finding extracted entities in a knowledge base. In *Multi-source, multilingual information extraction and summarization*, pages 93–115. Springer.

Lev Ratinov, Dan Roth, Doug Downey, and Mike Anderson. 2011. Local and global algorithms for disambiguation to wikipedia. In *Proceedings of ACL:HLT*, pages 1375–1384.

Philip Resnik. 1996. Selectional constraints: An information-theoretic model and its computational realization. *Cognition*, 61(1):127–159.

Alan Ritter, Oren Etzioni, et al. 2010. A latent dirichlet allocation method for selectional preferences. In *ACL*, pages 424–434.

Michael Schmitz, Robert Bart, Stephen Soderland, Oren Etzioni, et al. 2012. Open language learning for information extraction. In *EMNLP*, pages 523–534.

Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: A Core of Semantic Knowledge. In *16th international World Wide Web conference (WWW 2007)*, New York, NY, USA. ACM Press.

Wentao Wu, Hongsong Li, Haixun Wang, and Kenny Q. Zhu. 2012. Probase: a probabilistic taxonomy for text understanding. In *SIGMOD Conference*, pages 481–492.

Xuchen Yao and Benjamin Van Durme. 2014. Information extraction over structured data: Question answering with freebase. In *Proceedings of ACL*.

LDTM: A Latent Document Type Model for Cumulative Citation Recommendation

Jingang Wang[1]*, Dandan Song[1]†, Zhiwei Zhang[2], Lejian Liao[1], Luo Si[2], Chin-Yew Lin[3]
[1]School of Computer Science, Beijing Institute of Technology
[2]Dept. of Computer Science, Purdue University
[3]Knowledge Mining Group, Microsoft Research
{bitwjg, sdd, liaolj}@bit.edu.cn
{zhan1187, lsi}@purdue.edu
cyl@microsoft.com

Abstract

This paper studies Cumulative Citation Recommendation (CCR) - given an entity in Knowledge Bases, how to effectively detect its potential citations from volume text streams. Most previous approaches treated all kinds of features indifferently to build a global relevance model, in which the prior knowledge embedded in documents cannot be exploited adequately. To address this problem, we propose a latent document type discriminative model by introducing a latent layer to capture the correlations between documents and their underlying types. The model can better adjust to different types of documents and yield flexible performance when dealing with a broad range of document types. An extensive set of experiments has been conducted on TREC-KBA-2013 dataset, and the results demonstrate that this model can yield a significant performance gain in recommendation quality as compared to the state-of-the-art.

1 Introduction

Knowledge Bases (KBs), like Wikipedia, are playing increasingly important roles in numerous entity-based information retrieval tasks. Nevertheless, most KBs are hard to be up-to-date due to their manual maintenances by human editors. As reported in (Frank et al., 2012), there exists a median time lag of 356 days between the day a news article is published and the time that the news is cited in a Wikipedia article dedicated to the entity concerned by the news. The time lag would be reduced if relevant documents could be automatically detected as soon as they are published online and then recommended to the editors. This task is studied as Cumulative Citation Recommendation (CCR). Formally, given a set of KB entities, CCR is to filter relevant documents from a stream corpus and evaluate their citation-worthiness to the target entities.

A variety of supervised approaches (e.g., classification, learning to rank) have been employed and achieved promising results (Wang et al., 2013; Balog and Ramampiaro, 2013; Balog et al., 2013). Nevertheless, most of them leverage all features indiscriminately to build a global relevance model, which leads to unsatisfactory performance. The documents can offer some prior knowledge, which is named as **type** in this paper. The type is the prior knowledge embedded in the document that impacts on the probability of its being recommended to KBs. For instance, when dealing with a document on "music" topic, we would like to have less weights put on a politician entity because this document is not likely to related to it, but more often related to musicians or musical bands. Besides, the source of a document impacts on the recommendation strategies too. A document from news agencies is more reliable and citable than the one from social websites even if they state an identical story about the target KB entity. Hence we consider two kinds of document features to model the prior type knowledge: (1) topic-based features, and (2) source-based features.

This paper proposes a latent document type discriminative mixture model for CCR. We introduce an intermediate latent layer to model latent document types and define a joint distribution over the document-entity pairs and latent document-types on the observation data. The aim is to achieve a discriminative mixture model that is expected to outperform the global relevance model.

To the best of our knowledge, this is the first research work that leverages prior knowledge embedded in documents to improve CCR perfor-

*This work was partially performed when the first author was visiting Purdue University and Microsoft Research Asia.
†Corresponding Author

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 561–566,
Lisbon, Portugal, 17-21 September 2015.

mance. An extensive set of experiments conducted on TREC-KBA-2013 dataset has demonstrated the effectiveness of the proposed mixture model.

2 Discriminative Models for CCR

Given a set of KB entities $\mathcal{E} = \{e_u | u = 1, \cdots, M\}$ and a document collection $\mathcal{D} = \{d_v | v = 1, \cdots, N\}$, our objective is to estimate the conditional probability of relevance $P(r|e,d)$ with respect to an entity-document pair (e,d). Each (e,d) is represented as a feature vector $\mathbf{f}(e,d) = (f_1(e,d), \cdots, f_K(e,d))$, where K is the dimension of the entity-document feature vector. Moreover, to model the hidden document type, each document is represented as an document-type feature vector $\mathbf{g(d)} = (g_1(d), \cdots, g_L(d))$, where L indicates the dimension of the document-type feature vector.

2.1 Global Model

This paper utilizes logistic regression to estimate the conditional probability $P(r|e,d)$, where $r(r \in \{1,-1\})$ is a binary label indicating the relevance of an entity-document pair (e,d). The value of r is 1 if the document d is related to the entity e, otherwise $r = -1$. Formally, the parametric form of $P(r{=}1|e,d)$ is expressed as $P(r{=}1|e,d) = \delta(\sum_{i=1}^{K} \omega_i f_i(e,d))$, where $\delta(x)$ is the standard logistic function, ω_i is the combination parameter for the ith feature. It is easy to derive that for different values of r, the only difference in $P(r|e,d)$ is the sign within the logistic function. Therefore, we adopt the general representation of $P(r|e,d){=}\delta(r \sum_{i=1}^{K} \omega_i f_i(e,d))$. This model is denoted as **GM** in this paper. Several previous approaches can be deemed as global models adopting different classification functions such as decision trees (Wang et al., 2013) and Support Vector Machine (SVM) (Bonnefoy et al., 2013).

2.2 Latent Document Type Model

In GM, a fixed set of combination weights (i.e., ω) are learned to optimize the overall performance for all entity-document pairs. However, the best combination strategy for a given pair is not always the best for the others since both the documents and entities are heterogeneous. Therefore, we may benefit from developing a document type dependent model in which we choose the combination strategy individually for each document type to optimize the performance for specific document types. Since it is not feasible to determine the proper combination strategy for each document type, we need to classify documents into one of several types. The combination strategy is then tuned to optimize average performance for documents within the same type.

We propose a latent document type model (**LDTM**) by introducing an intermediate layer to capture the underlying type information in documents. A latent variable z is utilized to indicate which type the combination weights ω_z are drawn from. The choice of z is determined by the document d. The joint probability of relevance r and the latent variable z is represented as $P(r,z|e,d;\alpha,\omega){=}P(z|d;\alpha)P(r|e,d,z;\omega)$, where $P(z|d;\alpha)$ is the mixing coefficient, denoting the probability of choosing the hidden type z given document d, and α is the corresponding parameter. $P(r|e,d,z;\omega)$ denotes the discriminative component which takes a logistic function. By marginalizing out z, we obtain

$$P(r|e,d;\alpha,\omega) = \sum_{z}^{N_z} P(z|d;\alpha)\delta\Big(r\sum_{i=1}^{K}\omega_{zi}f_i(e,d)\Big) \quad (1)$$

where ω_{zi} is the weight for the ith entry in the feature vector under the hidden variable z. We adopt a soft-max function $\frac{1}{Z_d}\exp(\sum_{j=1}^{L}\alpha_j^z g_j(d))$ to model $P(z|d;\alpha)$, and Z_d is the normalization factor that scaled the exponential function to be a probability distribution. In this representation, each document d is denoted by a bag of document type features $(g_1(d), \cdots, g_L(d))$. By plugging the soft-max function into Equation (1), we have

$$P(r|e,d;\alpha,\omega){=} \frac{1}{Z_d}\sum_{z=1}^{N_z}\exp\Big(\sum_{j=1}^{L_z}\alpha_{zj}g_j(d)\Big)\delta\Big(r\sum_{i=1}^{K}\omega_{zi}f_i(e,d)\Big) \quad (2)$$

Suppose entity-document pairs in training set are represented as $\mathcal{T}{=}\{(d_u,e_v)\}$, and $\mathcal{R}{=}\{r_{uv}\}$ denotes the corresponding relevance judgment of (d_u,e_v), where $u = 1, \cdots, M$ and $v = 1, \cdots, N$. Assume training instances in $\mathcal{T}$ are independently generated, the conditional likelihood of training data is written as

$$P(\mathcal{R}|\mathcal{T}) = \prod_{u=1}^{M}\prod_{v=1}^{N} P(r_{uv}|e_u,d_v) \quad (3)$$

2.3 Parameter Estimation

The parameters (i.e., ω and α) can be estimated by maximizing the data log-likelihood $\mathcal{L}(\omega, \alpha)$, which is the form of logarithm of Equation (3). A typical parameter estimation method is to use Expectation-Maximization (EM) algorithm by iterating E-step and M-step continuously until convergence. The E-step is derived by computing the posterior probability of z given d_u and e_v, which is denoted as $P(z|d_u, e_v)$.

$$P(z|d_u, e_v) = \frac{\exp\Big(\sum_{j=1}^{L_z}\alpha_{zj}g_j(d_u)\Big)\delta\Big(r_{uv}\sum_{i=1}^{K}\omega_{zi}f_i(d_u, e_v)\Big)}{\sum_z \exp\Big(\sum_{j=1}^{L_z}\alpha_{zj}g_j(d_u)\Big)\delta\Big(r_{uv}\sum_{i=1}^{K}\omega_{zi}f_i(d_u, e_v)\Big)} \tag{4}$$

In M-step, we can obtain the following parameter update rules.

$$\omega_z^* = \arg\max_{\omega_z} \sum_{uv} P(z|d_u, e_v) \log\Big(\delta\Big(\sum_{i=1}^{K}\omega_{zi}f_i(d_u, e_v)\Big)\Big)$$

$$\alpha_z^* = \arg\max_{\alpha_z} \sum_u \Big(\sum_v P(z|d_u, e_v)\Big) \log\Big(\frac{1}{Z_{d_u}}\exp\Big(\sum_{j=1}^{L_z}\alpha_{zj}g_j(d_u)\Big)\Big) \tag{5}$$

To optimize Equation (5), we employ the `minFunc` toolkit[1] using Quasi-Newton strategy. We adopt Akaike Information Criteria (AIC) to determine the number of latent variables (Fang et al., 2010), which is calculated as $2m - 2\mathcal{L}(\omega, \alpha)$, where m is the number of parameters in the model.

LDTM holds two advantages over GM. (1) The combination parameters vary across various document types and hence lead to a gain of flexibility; (2) It offers probabilistic semantics for the latent document types and thus documents can be associated with multiple types.

3 Features

This section presents the two types of features used in the discriminative models. Entity-document features (i.e., $\mathbf{f}(e, d)$) are used in the discriminative components of GM and LDTM. In addition, LDTM requires document-type features (i.e., $\mathbf{g}(e)$) to learn the mixing coefficients in the mixture component.

Since our goal is not to develop new entity-document features, we adopt the identical entity-document feature set proposed in our previous work (Wang et al., 2013; Wang et al., 2015a; Wang et al., 2015b), which has been proved effective.

In terms of document-type features, we consider two kinds of prior knowledge embedded in documents to model the correlations between documents and their latent types.

Topic-based features One prior knowledge to model a document's latent type is its intrinsic topics. As we have claimed, documents with one or more obvious topics are more likely to be recommended to KB than those without any explicit topic. We capture the underlying topics in documents with word co-occurrences. After removing stop words, we represent each document as a feature vector with the bag-of-words model, where word weights are determined by TF-IDF scheme.

Source-based features The source of a document is another prior knowledge to evaluate the probability of the document's being recommended to KBs. We leverage a "bag-of-sources" model to represent each document as source-based feature vector, and term weights are determined by binary occurrence scheme. Please note that the sources are organized hierarchically. For example, *mainstream_news* is a sub-source of *news*.

4 Experiments

4.1 Dataset

We utilize TREC-KBA-2013 dataset[2] as our experimental dataset. The dataset is composed of a temporally stream corpus and a target KB entity set. The stream corpus contains nearly 1 billion documents crawled from 10 sources: **news**, **mainstream_news**, **social**, **weblog**, **linking**, **arxiv**, **classified**, **reviews**, **forum** and **memetracker**[3]. The corpus has been split with documents from October 2011 to February 2012 as training instances and the remainder for evaluation. We adopt the same training/test range setting in our experiments. The entity set is composed of 121 Wikipedia entities and 20 Twitter entities.

[1] `http://www.cs.ubc.ca/~schmidtm/Software/minFunc.html`

[2] `http://trec-kba.org/kba-stream-corpus-2013.shtml`

[3] `http://www.memetracker.org/`

Each entity-document pair is labeled as one of the 4 relevance levels: (i) **Vital**, timely information about the entity's current state, actions, or situation. This would motivate a change to an already up-to-date KB article. (ii) **Useful**, possibly citable but not timely, e.g., background biography, secondary source information. (iii) **Neural**, informative but not citable, e.g., tertiary source like Wikipedia article itself. and (iv) **Garbage**, no information about the target entity could be learned from the document, e.g., spam. Annotation details of the dataset are presented in Table 1.

	Range	Vital	Useful	Neutral	Garbage
Train	2011.10 $\sim$ 2012.02	1696	2121	1030	1702
Test	2012.03 $\sim$ 2013.02	5630	11579	3379	10543

Table 1: Annotation details of TREC-KBA-2013 dataset.

4.2 Evaluation Scenarios

According to different granularity settings, we evaluate the proposed models in two scenarios: (i) **Vital Only**. Only vital entity-document pairs are treated as positive instances. (ii) **Vital + Useful**. Both vital and useful entity-document pairs are treated as positive instances.

4.3 Comparison Methods

We conduct extensive comparisons with the following methods.

- Global Model (**GM**). The global discriminative model introduced in section 2.1.
- Source-based Latent Document Type Model (**src_LDTM**). A variant of LDTM that utilizes source-based features as document-type features.
- Topic-based Latent Document Type Model (**topic_LDTM**). A variant of LDTM that utilizes topic-based features as document-type features.
- Combination Latent Document Type Model (**combine_LDTM**). This approach utilizes source-based and topic-based features together as document-type features. In our experimental setting, we simply union the two feature vectors together into an integral feature vector.

For reference, we also include three top-ranked approaches in TREC-KBA-2013 track.

- **BIT-MSRA** (Wang et al., 2013). A global random forests classification method, the first place approach in TREC-KBA-2013 track.
- **UDEL** (Liu et al., 2013). An entity-centric query expansion approach, the second place approach in TREC-KBA-2013 track.
- **Official Baseline** (Frank et al., 2013). A strong baseline in which human annotators go through target entities and came up with a list of keywords for filtering vital documents.

4.4 Results and Discussion

Improving precision is harder than improving recall for CCR (Frank et al., 2013). Therefore, we care more about recommendation quality of CCR. Precision and overall accuracy are adopted as metrics to evaluate different approaches. All the metrics are computed in the test pool of all entity-document pairs. The results are reported in Table 2. In comparison to the baselines listed in the 2nd block of Table 2, our mixture models achieve higher or competitive precision and accuracy in both scenarios considerably. Compared with the official baseline, our best mixture model improves precision about 28%. In both scenarios, the variants of LDTM outperform GM on precision and accuracy, which validates our motivations that (i) introducing document latent types in mixture model can enhance the recommendation quality, and (ii) source-based and topic-based features can capture the hidden type information of documents.

Methods	Vital Only		Vital + Useful	
	P	Accu	P	Accu
Official Baseline	.171	.175	.540	.532
BIT-MSRA	.214	.445	.589	**.615**
UDEL	.169	.259	.573	.579
GM	.218	.587	.604	.565
src_LDTM	.273	**.763**	.626	.607
topic_LDTM	.293	.755	**.643**	.609
combine_LDTM	**.299**	.751	.633	**.611**

Table 2: Overall results of evaluated methods. Best scores are typeset boldface.

Moreover, topic_LDTM generally performs better than src_LDTM in both scenarios, which meets our expectation because topic-based features have far more dimensions than source-based features. However, even if source-based feature vector holds a few dimensions (10 in our experiments),

src_LDT improves the precision on the basis of GM. Thus, the precision can be enhanced further if we can develop more valuable features to represent the underlying document types. The combination variant of LDTM achieve the best precision in **Vital Only** scenario and the best accuracy in **Vital + Useful** scenario. The naïve combination strategy of two types of features can improve the performance but not stable, so we need find better combination strategies.

For all variant of the LDTM, the number of latent types determined by AIC are reported in Table 3. The optimal number of latent types in **Vital + Useful** is more than that in **Vital Only**. This reveals that the types of **Vital** documents for entities have more restrictions than **Useful** documents, either by topics or by sources. In addition, the optimal number of latent topics is more than that of latent sources, which also follows our intuition that topic-based features holding more dimensions than source-based features. Since we employ a naïve combination strategy for the two types of features, the number of latent types of combine_LDTM is more close to topic_LDTM, which possesses more features than src_LDTM.

Model	Vital	Vital + Useful
src_LDTM	6	7
topic_LDTM	9	15
combine_LDTM	14	15

Table 3: Number of latent types determined by AIC.

5 Related Work

There are three kinds of approaches developed for CCR in previous work: query expansion (Liu et al., 2013; Wang et al., 2013), classification such as SVM (Kjersten and McNamee, 2012) and Random Forest classifier (Bonnefoy et al., 2013; Balog et al., 2013), and learning to rank approaches (Wang et al., 2013; Balog and Ramampiaro, 2013). Transfer learning is utilized to transfer the keyword importance learned from training pairs to query pairs (Zhou and Chang, 2013).

However, some highly supervised methods require training instances for each entity to build a relevance model, limiting their scalabilities. A compromised solution is to build a global discriminative model with all features indifferently. We spotlight document-type features and study the impacts of them in discriminative mixture models. Mixture model has been applied and proved effective in multiple information retrieval tasks, such as expert search (Fang et al., 2010) and federated search (Hong and Si, 2012). By learning flexible combination weights for different types of training instances, mixture model can outperform global models with fixed weights for all instances.

6 Conclusion

Cumulative Citation Recommendation (CCR) is an important task to automatically detect citation-worthy documents from volume text streams for knowledge base entities. We study CCR as a classification problem and propose a latent document type model (LDTM) through introducing a latent layer in a discriminative model to capture the correlations between documents and their intrinsic types. Two variants of LDTM are implemented by modeling the latent types with document source-based and topic-based features respectively. Experimental results on TREC-KBA-2013 dataset demonstrate that our mixture model can improve CCR performance significantly, especially on precision and accuracy, revealing the advantage of LDTM in enhancing recommendation quality of citation-worthy documents.

For future work, we wish to explore more useful document-type features and apply more proper combination strategies to improve the latent document type model.

Acknowledgement

The authors would like to thank Fei Sun, Qifan Wang and Chen Shao for their valuable suggestions and the anonymous reviewers for their helpful comments. This work is funded by the National Program on Key Basic Research Project (973 Program, Grant No. 2013CB329600), National Natural Science Foundation of China (NSFC, Grant Nos. 61472040 and 60873237), and Beijing Higher Education Young Elite Teacher Project (Grant No. YETP1198).

References

Krisztian Balog and Heri Ramampiaro. 2013. Cumulative citation recommendation: classification vs. ranking. In *SIGIR*, pages 941–944. ACM.

Krisztian Balog, Heri Ramampiaro, Naimdjon Takhirov, and Kjetil Nørvåg. 2013. Multi-step classification approaches to cumulative citation recommendation. In *OAIR*, pages 121–128.

Ludovic Bonnefoy, Vincent Bouvier, and Patrice Bellot. 2013. A weakly-supervised detection of entity central documents in a stream. In *SIGIR*, pages 769–772.

Yi Fang, Luo Si, and Aditya P. Mathur. 2010. Discriminative models of integrating document evidence and document-candidate associations for expert search. In *SIGIR*, pages 683–690. ACM.

J. R. Frank, M. Kleiman-Weiner, D. A. Roberts, F. Niu, C. Zhang, C. Re, and I. Soboroff. 2012. Building an Entity-Centric Stream Filtering Test Collection for TREC 2012. In *TREC*.

John Frank, Steve J. Bauer, Max Kleiman-Weiner, Daniel A. Roberts, Nilesh Triouraneni, Ce Zhang, and Christopher Rè. 2013. Evaluating stream filtering for entity profile updates for trec 2013. In *TREC*.

Dzung Hong and Luo Si. 2012. Mixture model with multiple centralized retrieval algorithms for result merging in federated search. In *SIGIR*, pages 821–830. ACM.

Brain Kjersten and Paul McNamee. 2012. The hltcoe approach to the trec 2012 kba track. In *TREC*.

Xitong Liu, Jeffrey Darko, and Hui Fang. 2013. A related entity based approach for knowledge base acceleration. In *TREC*.

Jingang Wang, Dandan Song, Chin-Yew Lin, and Lejian Liao. 2013. Bit and msra at trec kba ccr track 2013. *TREC*.

Jingang Wang, Lejian Liao, Dandan Song, Lerong Ma, Chin-Yew Lin, and Yong Rui. 2015a. Resorting relevance evidences to cumulative citation recommendation for knowledge base acceleration. In *Web-Age Information Management*, volume 9098 of *Lecture Notes in Computer Science*, pages 169–180. Springer International Publishing.

Jingang Wang, Dandan Song, Qifan Wang, Zhiwei Zhang, Luo Si, Lejian Liao, and Chin-Yew Lin. 2015b. An entity class-depedent discriminative mixture model for cumulative citation recommendation. In *SIGIR*. ACM.

Mianwei Zhou and Kevin Chen-Chuan Chang. 2013. Entity-centric document filtering: boosting feature mapping through meta-features. In *CIKM*, pages 119–128. ACM.

Online Sentence Novelty Scoring for Topical Document Streams

Sungjin Lee
Yahoo Labs
229 West 43rd Street, New York, NY 10036, USA
junion@yahoo-inc.com

Abstract

The enormous amount of information on the Internet has raised the challenge of highlighting new information in the context of already viewed content. This type of intelligent interface can save users time and prevent frustration. Our goal is to scale up novelty detection to large web properties like Google News and Yahoo News. We present a set of lightweight features for online novelty scoring and fast nonlinear feature transformation methods. Our experimental results on the TREC 2004 shared task datasets show that the proposed method is not only efficient but also very powerful, significantly surpassing the best system at TREC 2004.

1 Introduction

The Internet supplies a wealth of news content with a corresponding problem: finding the right content for different users. Search engines are helpful if a user is looking for something specific that can be cast as a keyword query. If a user does not know what to look for, recommendation engines can make personalized suggestions for stories that may interest the user. But both types of systems frequently represent content that the user has already consumed, leading to delay and frustration. Consequently, identifying novel information has been an essential aspect of studies on news information retrieval. Newsjunkie (Gabrilovich et al., 2004), for instance, describes a system that personalizes a newsfeed based on a measure of information novelty: the user can be presented custom tailored news feeds that are novel in the context of documents that have already been reviewed. This will spare the user from hunting through duplicate and redundant content for new nuggets of information. Identifying genuinely novel information is also an essential aspect of *update summarization* (Nenkova and McKeown, 2012; Gao et al., 2013; Guo et al., 2013; Wang and Li, 2010; Bentivogli et al., 2011). But the temporal dynamics of a document stream are not generally the focus. Novelty detection has also been studied in *Topic Detection and Tracking* field for the *First Story Detection* task (Allan, 2002; Karkali1 et al., 2013; Karkali et al., 2014; Tsai and Zhang, 2011) where the aim is to detect novel documents given previously seen documents. In this paper, we examine a slightly different problem; we perform novelty detection at the sentence level to highlight sentences that contain novel information.

The novelty track in TREC was designed to serve as a shared task for exactly this type of research: finding novel, on-topic sentences from a news stream (Harman, 2002). There were four tasks in the novelty track but we only focus on task 2 in this paper: "given relevant sentences in all documents, identify all novel sentences." The track changed slightly from year to year. The data of the first run in 2002 (Harman, 2002) used old topics and judgments which proved to be problematic due to the small percentage of relevant sentences. TREC 2003 (Soboroff and Harman, 2003) included 50 new topics with an improved balance of relevant and novel sentences and chronologically ordered documents. TREC 2004 (Soboroff and Harman, 2005) used the same task settings and the same number of topics, but made a major change through the inclusion of irrelevant documents.

Although the participants in the novelty track of TREC and many followup studies have investigated a wide ranging set of features and algorithms (Soboroff and Harman, 2005), almost none were specifically focused on scalability. However, modern news aggregators are usually visited by millions of unique users and consume millions of stories each day. Moreover, every few minutes item churn takes place and the stories of interest are likely to be the ones that appeared in the last couple of hours. As real-time processing on a large scale gains more attention (Osborne et al., 2014), we investigate features that are both effective and efficient, and so could be used in a scalable online novelty scoring engine for making personalized newsfeeds on large web properties like Google News and Yahoo News.

To achieve this goal, our contributions are two-fold. First, we present a set of effective, light-weight features: KL divergence with asymmetric smoothing, nonlinear transformation of unseen word count, relative sentence position and word embedding-based similarity. Note that we restrict ourselves to only surface-level text features and algorithms that have time complexity of $O(W)$ where W is the number of unique words seen so far (previous studies often employed quite expensive

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 567–572,
Lisbon, Portugal, 17-21 September 2015.

features and algorithms that have time complexity of at least $O(WT)$ where T is the number of sentences so far). To fully comply with the online setting, we also exclude very popular methods for measuring similarity such as tf-idf, since we are not allowed to see the entire corpus. Second, we propose efficient feature transformation methods: recursive feature averaging and *Deep Neural Network* (DNN)-based nonlinear transformation. We evaluate our system on task 2 of the 2004 TREC novelty track. Interestingly, our experiment results indicate that our light-weight features are actually very powerful when used in conjunction with the proposed feature transformation; we obtain a significant performance improvement over the best challenge system.

The rest of this paper is structured as follows: Section 2 presents a brief summary of related work. Section 3 describes our algorithm and features. Section 4 outlines the experimental setup and reports the results of comparative analysis with challenge systems. We finish with some conclusions and future directions in Section 5.

Algorithm 1: Novelty scoring for a topical document stream

```
Data: a document stream
Result: a document stream with novelty annotation
Initialize a context C_0;
while not at end of the document stream do
    read a document;
    split the document into sentences;
    while not at end of the document do
        read a sentence S_t;
        perform preprocessing on S_t;
        compute novelty score as the posterior probability of a binary novelty random variable N_t, p(N_t|S_t, C_{t-1});
        update the context C_t with C_{t-1} and S_t;
    end
    compute a document-level score (e.g. average out all sentence-level scores)
end
```

2 Related Work

There were 13 groups and 54 submitted entries for the 2004 TREC novelty track task 2. The participants used a wide range of methods which can be roughly categorized into statistical and linguistic methods. Statistical methods included traditional information retrieval models such as tf-idf and Okapi, and metrics such as importance value, new sentence value, conceptual fuzziness, scarcity measure, information weakness, unseen item count with a threshold optimized for detecting novel sentences (Blott et al., 2004; Zhang et al., 2004; Abdul-Jaleel et al., 2004; Eichmann et al., 2004; Erkan, 2004; Schiffman and McKeown, 2004). Thresholds are either learned on the 2003 data or determined in an ad hoc manner. Some groups also used machine learning algorithms such as SVMs by casting the problem as a binary classification (Tomiyama et al., 2004). Many groups adopted a variety of preprocessing steps including expansion of the sentences using dictionaries, ontologies or corpus-based methods and named entity recognition. Graph-based analysis has also been applied where directed edges are established by cosine similarity and chronological order. After this graph is constructed, the eigenvector centrality score for each sentence was computed by using a power method. The sentences with low centrality scores were considered as new (Erkan, 2004). Graph-based approaches were further pursued by Gamon (2006) that drew a richer set of features from graph topology and its changes, resulting in a system that ties with the best system at TREC 2004 (i.e. Blott et al. (2004)). On the other hand, deep linguistic methods included parsing, coreference resolution, matching discourse entities, searching for particular verbs and verb phrases, standardizing acronyms, building a named-entity lexicon, and matching concepts to manually-constructed ontology for topic-specific concepts (Amrani et al., 2004). The difficulty of the novelty detection task is evident from the relatively low score achieved by even the best systems at TREC 2004 (Soboroff and Harman, 2005). The top scoring systems were mostly based on statistical methods while deep linguistic approaches achieved the highest precision at the cost of poor recall.

3 Method

For the purpose of this paper, we formulate task 2 of the TREC novelty detection track as an online probabilistic inference task. More specifically, we compute the novelty score as the posterior probability of a binary novelty random variable N:

$$p(N_t|S_t, C_{t-1}) = \frac{1}{Z} \exp \sum_i w_i f_i(N_t, S_t, C_{t-1}) \quad (1)$$

in which the f_i are feature functions, w_i model parameters, S_t the sentence in focus and C_{t-1} a context containing information about previously seen sentences S_1 through S_{t-1} across documents.

The overall procedure is listed in Algorithm 1. The algorithm takes as input documents which have been clustered by topic and chronologically ordered. For each sentence S_t in each document, basic preprocessing is performed (e.g. simple tokenization, stopword filtering and stemming (Porter, 1980)), then the inference is made whether S_t is novel given the context C_{t-1}. Without the use of the context, the time complexity of our algorithm would depend on the number of sentences so far. Thus, the features and the model for the context are important for efficiency. Note that our method only takes time complexity of $O(W)$ for

both context update and feature generation.

3.1 Features

KL divergence with asymmetric smoothing. KL divergence has been successfully adopted to measure the distance between a document and a set of documents (Gabrilovich et al., 2004; Gamon, 2006). We use it to measure the distance between context C and sentence S:

$$\sum_w p_C(w) \log \frac{p_C(w)}{p_S(w)} \quad (2)$$

The intuition is that the more distant the distributions are, the more likely it is that the sentence is novel. Since KL divergence is asymmetric, both directions are used as features, with and without scale normalization. The computation of KL divergence requires both p_C and p_S to be non-zero; while simple add-one smoothing is employed in previous work, we adopt novel asymmetric smoothing. We add a larger smoothing factor s for already seen words than the factor u for unseen words. The rationale behind this is that we intensify the difference caused by unseen words and attenuate the difference caused by seen words (Figure 1.) Asymmetric smoothing with various smoothing factors consistently showed better performance than symmetric smoothing in our experiments.

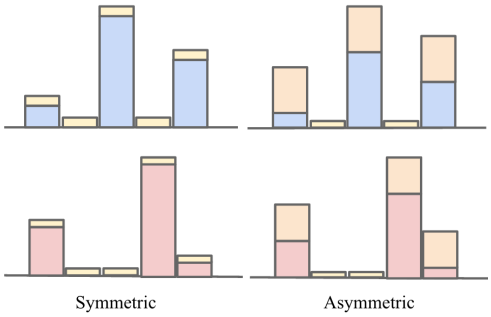


Figure 1: KL divergence with symmetric (left) and asymmetric (right) smoothing. Pink and blue correspond to two distributions while light yellow and orange to smoothing factors.

Nonlinear transformation of unseen word count. One of the simplest metrics to measure novelty is the plain count of unseen words. This measure, however, does not necessarily reflect human perception of novelty given the prevalence of nonlinearity in human perception (Kingdom and Prins, 2009). Thus, we explored the use of a simple nonlinear transformation of unseen word counts instead of the plain count (Figure 2):

$$T(n) = (\alpha n + \beta)^\gamma \quad (3)$$

where n is the number of new words and α, β and γ are parameters. In our experiments, the use of a nonlinear transformation helped yield better results.

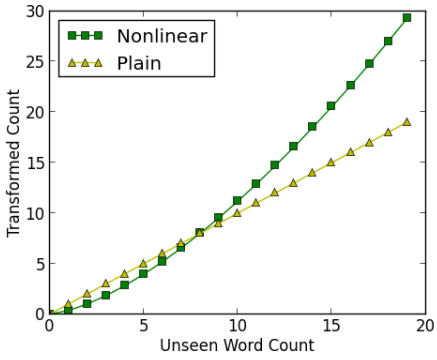


Figure 2: Nonlinear transformation of unseen word count with parameters set via cross-validation on the TREC training data: $\alpha = 0.5$, $\beta = 0$ and $\gamma = 1.5$.

Relative position in a document. Relative position of a sentence in a document is simple yet has been proven effective for summarization. Relative position is also closely related to novelty detection as follows: 1) There is in general a good chance that earlier sentences are more novel than the later ones. 2) We found a pattern that news articles coming in later are apt to present novel information first and then a summary of old information.

Word embedding-based similarity. Neural word embedding techniques can be effective in capturing syntactic and semantic relationships, and more computationally efficient than many other competitors (Socher et al., 2012; Mikolov et al., 2013). As reported in (Tai et al., 2015), a simple averaging scheme was found to be very competitive to more complex models for representing a sentence vector. These observations lead us to adopt the following additional features derived from word embeddings: 1) cosine similarity between the mean vectors of the context C and sentence S, 2) sigmoid function value for the dot product of the mean vectors of the context C and sentence S. The mean vectors of C and S are computed by taking the average of the word vectors of each unique word in C and S, respectively. We use word embedding with 100 dimensions trained on Wikipedia using the word2vec toolkit (https://code.google.com/p/word2vec).

3.2 Feature transformation

Recursive feature averaging. A large portion of the novel sentences in the TREC 2004 data appear in consecutive runs of two or more (Schiffman and McKeown, 2004). Sequential labeling would be a natural approach to take advantage of this characteristic of the problem, but the use of sequential labeling will make time complexity depend on the number of sentences T. Thus we came up with another way to exploit this characteristic, recursively averaging over previous feature vectors and augmenting the current feature vector with

the average:

$$R_t = \eta F_{t-1} + (1-\eta) R_{t-1} \quad (4)$$

$$F'_t = F_t :: R_t \quad (5)$$

where F is a feature vector, R the average vector of previous ones, F' the augmented feature vector, η the weight of the last feature vector in averaging and :: means concatenation.

DNN-based feature transformation. In order to better capture non-trivial interactions between the features described above, we adopt a DNN with a bottleneck. DNNs with a bottleneck have been successfully explored for nonlinear feature transformation (Grézl et al., 2007). The feature transformation is normally achieved from narrow hidden layers that retain only the information useful to classification. This leads us to introduce bottleneck hidden layers between the input layer and the *Logistic Regression* output layer (Figure 3.)

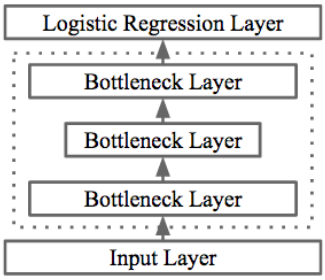


Figure 3: Flowchart for a bottleneck DNN. The dotted-box represents bottleneck generating hidden layers.

4 Experiments and Results

Following the guidelines of task 2 for the TREC 2004 novelty detection track, we used the TREC 2003 dataset as training data and the TREC 2004 dataset as test data. The training data includes 10,226 novel sentences out of 15,557 sentences. The test data includes 3,454 novel ones out of 8,343 sentences. We trained a DNN-based classifier and several logistic regression classifiers (which are the same model with the DNN model except without the hidden layers) using the Theano toolkit (Bergstra et al., 2010) to verify the effectiveness of each feature and feature transformation. We optimized all models by minimizing *logloss* with the stochastic gradient decent algorithm with momentum. We classified a sentence as novel if the posterior probability is greater than 0.5. We performed a search based on five-fold cross validation to identify optimal values for the parameters defined in Section 3, and obtained the following values: $s = 10$, $u = 0.1$, $\alpha = 0.5$, $\beta = 0$, $\gamma = 1.5$ and $\eta = 0.5$. For the DNN classifier, we used a set of five bottleneck hidden layers. The number of nodes for each hidden layer were set to 10, 5, 3, 5 and 10, respectively.

Comparative evaluation results in F-score (following the TREC protocol) are shown in Table 1. In Table 1, the first four entries refer to the best top systems from TREC 2004 and followup studies, *KLdiv* to a system using only KL divergence features, *TransCount* to a system using only nonlinear transformation of unseen word count features, *RelPos* to a system using only relative position features, *Word2Vec* to a system using only word embedding features, *All* to a system using all features, *All + Recursive* to *All* with recursive feature averaging applied, *All + DNN* to *All* with DNN-based feature transformation applied and *All + Recursive + DNN* to *All + Recursive* with DNN-based feature transformation applied. The best result (in bold) is significantly better than the best system results from TREC 2004, while still being very computationally efficient and therefore scalable. In terms of individual features, *KLdiv* (ties for 5th place at TREC 2004) and *TransCount* (outperforms the 6th entry) showed very strong results. Although *RelPos* and *Word2Vec* did not yield good results, we found them complementary to other features; performance was degraded to 0.621 and 0.624, respectively, when they were excluded from *All + Recursive + DNN*. The DNN-based feature transformation generally yielded better results. In particular, it becomes very effective in conjunction with recursive feature averaging. This result indicates that the DNN-based transformation allows the system to capture the non-trivial interactions between previous sentences and the current one.

Systems	F-score
Blott et al. (2004) / Gamon (2006)	0.622
Tomiyama et al. (2004)	0.619
Abdul-Jaleel et al. (2004)	0.618
Schiffman and McKeown (2004)	0.617
KLdiv	0.614
TransCount	0.611
RelPos	0.577
Word2Vec	0.577
All	0.615
All + Recursive	0.615
All + DNN	0.617
All + Recursive + DNN	**0.625**

Table 1: Performance breakdown. The best result is significantly better than the other configurations ($p < 0.01$) based on the McNemar test. Since the systems' output is not available, we are not able to calculate statistical significance against TREC systems.

5 Conclusions

We explored the space of light-weight features and their nonlinear transformation with the goal of supporting online web-scale sentence novelty detection. The experiment results show that these features are not only efficient but also very powerful; a combination of these

features with a simple, scalable classification approach significantly surpassed the best challenge system at TREC 2004. For future work, it would be interesting to see if more sophisticated DNN training techniques (e.g. unsupervised pre-training and different optimization algorithms) would yield a better performance.

References

Nasreen Abdul-Jaleel, James Allan, W Bruce Croft, Fernando Diaz, Leah Larkey, Xiaoyan Li, Mark D Smucker, and Courtney Wade. 2004. UMass at TREC 2004: Novelty and HARD. In *Proceedings of TREC*.

James Allan. 2002. Introduction to topic detection and tracking. In *Topic detection and tracking*, pages 1–16. Springer.

Ahmed Amrani, Jérôme Azé, Thomas Heitz, Yves Kodratoff, and Mathieu Roche. 2004. From the texts to the concepts they contain: a chain of linguistic treatments. In *In Proceedings of TREC*.

Luisa Bentivogli, Peter Clark, Ido Dagan, Hoa Dang, and Danilo Giampiccolo. 2011. The seventh pascal recognizing textual entailment challenge. *Proceedings of TAC*, 2011.

James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. 2010. Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*, June. Oral Presentation.

Stephen Blott, Oisin Boydell, Fabrice Camous, Paul Ferguson, Georgina Gaughan, Cathal Gurrin, Gareth JF Jones, Noel Murphy, Noel E O'Connor, and Alan F Smeaton. 2004. Experiments in terabyte searching, genomic retrieval and novelty detection for TREC 2004.

David Eichmann, Yi Zhang, Shannon Bradshaw, Xin Ying Qiu, Li Zhou, Padmini Srinivasan, Aditya Kumar Sehgal, and Hudon Wong. 2004. Novelty, question answering and genomics: The University of Iowa response. In *Proceedings of TREC*.

Günes Erkan. 2004. The University of Michigan in novelty 2004. In *Proceedings of TREC*.

Evgeniy Gabrilovich, Susan Dumais, and Eric Horvitz. 2004. Newsjunkie: providing personalized newsfeeds via analysis of information novelty. In *Proceedings of WWW*.

Michael Gamon. 2006. Graph-based text representation for novelty detection. In *Proceedings of the First Workshop on Graph Based Methods for Natural Language Processing*.

Dehong Gao, Wenjie Li, and Renxian Zhang. 2013. Sequential summarization: A new application for timely updated Twitter trending topics. In *Proceedings of the ACL*.

Frantisek Grézl, Martin Karafiát, Stanislav Kontár, and Jan Cernocky. 2007. Probabilistic and bottle-neck features for lvcsr of meetings. In *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*, volume 4, pages IV–757. IEEE.

Qi Guo, Fernando Diaz, and Elad Yom-Tov. 2013. Updating users about time critical events. In *Advances in Information Retrieval*, pages 483–494. Springer.

Donna Harman. 2002. Overview of the trec 2002 novelty track. In *TREC*.

Margarita Karkali, François Rousseau, Alexandros Ntoulas, and Michalis Vazirgiannis. 2014. Using temporal IDF for efficient novelty detection in text streams. *CoRR*, abs/1401.1456.

Margarita Karkali1, Alexandros Ntoulas, Franois Rousseau, and Michalis Vazirgiannis. 2013. Efficient online novelty detection in news streams. In *Proceedings of Web Information Systems Engineering*.

Frederick Kingdom and Nicolaas Prins. 2009. *Psychophysics: a practical introduction*. Academic Press.

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

Ani Nenkova and Kathleen McKeown. 2012. A survey of text summarization techniques. In *Mining Text Data*, pages 43–76. Springer.

Miles Osborne, Sean Moran, Richard McCreadie, Alexander Von Lunen, Martin D Sykora, Elizabeth Cano, Neil Ireson, Craig Macdonald, Iadh Ounis, Yulan He, et al. 2014. Real-time detection, tracking, and monitoring of automatically discovered events in social media.

Martin F Porter. 1980. An algorithm for suffix stripping. *Program*, 14(3):130–137.

Barry Schiffman and Kathleen McKeown. 2004. Columbia University in the novelty track at TREC 2004. In *Proceedings of TREC*.

Ian Soboroff and Donna Harman. 2003. Overview of the trec 2003 novelty track. In *TREC*, pages 38–53.

Ian Soboroff and Donna Harman. 2005. Novelty detection: the TREC experience. In *Proceedings of HLT-EMNLP*.

Richard Socher, Yoshua Bengio, and Christopher D. Manning. 2012. Deep learning for nlp (without magic). In *Tutorial Abstracts of ACL 2012*, ACL '12, pages 5–5, Stroudsburg, PA, USA. Association for Computational Linguistics.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*.

Tomoe Tomiyama, Kosuke Karoji, Takeshi Kondo, Yuichi Kakuta, Tomohiro Takagi, Akiko Aizawa, and Teruhito Kanazawa. 2004. Meiji University web, novelty and genomic track experiments. In *Proceedings of TREC*.

FloraS. Tsai and Yi Zhang. 2011. D2s: Document-to-sentence framework for novelty detection. *Knowledge and Information Systems*, 29(2):419–433.

Dingding Wang and Tao Li. 2010. Document update summarization using incremental hierarchical clustering. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*.

Huaping Zhang, Hongbo Xu, Shuo Bai, Bin Wang, and Xueqi Cheng. 2004. Experiments in TREC 2004 novelty track at CAS-ICT. In *Proceedings of TREC*.

Global Thread-Level Inference for Comment Classification in Community Question Answering

Shafiq Joty, Alberto Barrón-Cedeño, Giovanni Da San Martino, Simone Filice, Lluís Màrquez, Alessandro Moschitti, and **Preslav Nakov**,
Qatar Computing Research Institute, HBKU
{sjoty,albarron,gmartino,sfilice,
lmarquez,amoschitti,pnakov}@qf.org.qa

Abstract

Community question answering, a recent evolution of question answering in the Web context, allows a user to quickly consult the opinion of a number of people on a particular topic, thus taking advantage of the wisdom of the crowd. Here we try to help the user by deciding automatically which answers are good and which are bad for a given question. In particular, we focus on exploiting the output structure at the thread level in order to make more consistent global decisions. More specifically, we exploit the relations between pairs of comments at any distance in the thread, which we incorporate in a graph-cut and in an ILP frameworks. We evaluated our approach on the benchmark dataset of SemEval-2015 Task 3. Results improved over the state of the art, confirming the importance of using thread level information.

1 Introduction

Community question answering (CQA) is a recent evolution of question answering, in the Web context, where users pose questions and then receive answers from other users. This setup is very attractive, as the anonymity on the Web allows users to ask just about anything and then hope to get some honest answers from a number of people. On the negative side, there is no guarantee about the quality of the answers as people of very different background, knowledge, and with different motivation contribute answers to a given question.

Unlike traditional question answering (QA), in CQA answering takes the form of commenting in a forum. Thus, many comments are only loosely connected to the original question, and some are not answers at all, but are rather interactions between users.

As question-comment threads can get quite long, finding good answers in a thread can be time-consuming. This has triggered research in trying to automatically determine which answers might be good and which ones are likely to be bad or irrelevant. One early work going in this direction is that of Qu and Liu (2011), who tried to determine whether a question is "solved" or not, given its associated thread of comments. As a first step in the process, they performed a comment-level classification, considering four classes: *problem*, *solution*, *good feedback*, and *bad feedback*.

More recently, the shared task at SemEval 2015 on Answer Selection in CQA (Nakov et al., 2015), whose benchmark datasets we will use below, tackled the task of identifying *good*, *potentially* useful, and *bad* comments within a thread. In that task, the top participating systems used thread-level features, in addition to the usual local features that only look at the question–answer pair. For example, the second-best team, HITSZ-ICRC, used as a feature the position of the comment in the thread (Hou et al., 2015). Similarly, our participation, which achieved the third-best postition, used features that try to describe a comment in the context of the entire comment thread, focusing on user interaction (Nicosia et al., 2015). Finally, the fifth-best team, ICRC-HIT, treated the answer selection task as a sequence labeling problem and proposed recurrent convolution neural networks to recognize good comments (Zhou et al., 2015b).

In a follow-up work, Zhou et al. (2015a) included a long-short term memory in their convolution neural network to learn the classification sequence for the thread. In parallel, in our recent work (Barrón-Cedeño et al., 2015), we tried to exploit the dependencies between the thread comments to tackle the same task. We did it by designing features that look globally at the thread and by applying structured prediction models, such as Conditional Random Fields (Lafferty et al., 2001).

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 573–578,
Lisbon, Portugal, 17-21 September 2015.

Our goal in this paper goes in the same direction: we are interested in exploiting the output structure at the thread level to make more consistent global assignments.

To the best of our knowledge, there is no work in QA that identifies good answers based on the selection of the other answers retrieved for a question. This is mainly due to the loose dependencies between the different answer passages in standard QA. In contrast, we postulate that in a CQA setting, the answers from different users in a common thread are strongly interconnected and, thus, a joint answer selection model should be adopted to achieve higher accuracy. In particular, we focus on the relations between two comments *at any distance in the thread*. This is more general than previous approaches, which were either limited to sequential interactions or considered conversational interactions only at the level of features.

We propose a model based on the idea that similar comments should have similar labels. Below, we apply graph-cut and we compare it to an integer linear programming (ILP) formulation for decoding under global constraints; we also provide results with a linear-chain CRF. We show that the CRF is ineffective due to long-distance relations, e.g., a conversation in a thread can branch and then come back later. On the contrary, the global inference models (either graph-cut or ILP) using the similarity between pairs of comments manage to significantly improve a strong baseline performing local comment-based classifications.

2 The Task

We use the CQA-QL corpus from Subtask A of SemEval-2015 Task 3 on Answer Selection in CQA. The corpus contains data from the *Qatar Living* forum,[1] and is publicly available on the task's website.[2] The dataset consists of questions and a list of answers for each question, i.e., *question-answer threads*. Each question, and each answer, consist of a short title and a more detailed description. There is also meta information associated with both, e.g., ID of the user asking/answering the question, timestamp, category. The task asks participants to determine for each answer in the thread whether it is *Good*, *Bad*, or *Potential*ly useful for the given question.

[1] http://www.qatarliving.com/forum

[2] http://alt.qcri.org/semeval2015/task3/

Q: Can I obtain Driving License my QID is written Employee

$\mathbf{A}_1$ the word employee is a general term that refers to all the staff in your company either the manager, secretary up to the lowest position or whatever positions they have. you are all considered employees of your company.

$\mathbf{A}_2$ your qid should specify what is the actual profession you have. i think for me, your chances to have a drivers license is low.

$\mathbf{A}_3$ dear richard, his asking if he can obtain. means he have the driver license

$\mathbf{A}_4$ Slim chance ...

Figure 1: **Example from SemEval-2015 Task 3.**

A simplified example is shown in Figure 1,[3] where answers 2 and 4 are good, answer 1 is potentially useful, and answer 3 is bad. In this paper, we focus on a *2-class* variant of the above Subtask A, which is closer to a real CQA application. We merge *Potential* and *Bad* labels into *Bad* and we focus on the 2-class problem: Good-vs-Bad. Table 1 shows some statistics about the resulting dataset used for development, training and testing.

Category	Train	Dev	Test
Questions	2,600	300	329
Comments	16,541	1,645	1,976
Good	8,069	875	997
Bad	8,472	770	979

Table 1: **Statistics about the CQA-QL dataset:** after merging *Bad* and *Potential* into *Bad*.

3 Our Proposed Solution

We model the pairwise relations between the comments in the answer thread ($\{c_i\}_{i=1}^{n}$) to produce a better global assignment: we combine the predictions of a Good-vs-Bad classifier at the comment level with the output of a pairwise classifier, Same-vs-Different, which takes two comments and predicts whether they should have the same label.

Each comment c_i has an individual score s_{iK}, provided by the Good-vs-Bad classifier, for being in class $K \in \{G, B\}$ (i.e., G for *Good* and B for *Bad*). Moreover, for each pair of comments (c_i, c_j), we have an association score s_{ij}, an estimate by the pairwise classifier about how likely it is that the comments c_i and c_j will have the same label. Next, we define two ways of doing global inference using these two sources of information.

[3] http://www.qatarliving.com/moving-qatar/posts/can-i-obtain-driving-license-my-qid-written-employee

3.1 Graph Partition Approach

Here our goal is to find a partition $P = (G, B)$ that *minimizes* the following cost:

$$C(P) = \lambda\Big[\sum_{c_i \in G} s_{iB} + \sum_{c_i \in B} s_{iG}\Big] + (1-\lambda) \sum_{c_i \in G, c_j \in B} s_{ij}$$

The first part of the cost function discourages misclassification of individual comments, while the second part encourages similar comments to be in the same class. The mixing parameter $\lambda \in [0, 1]$ determines the relative strength of the two components. Our approach is inspired by Pang and Lee (2004), where they model the proximity relation between sentences for finding subjective sentences in product reviews, whereas we are interested in global inference based on local classifiers.

The optimization problem can be efficiently solved by finding a *minimum cut* of a weighted undirected graph $G = (V, E)$. The set of nodes $V = \{v_1, v_2, \cdots, v_n, s, t\}$ represent the n comments in a thread, the *source* and the *sink*. We connect each comment node v_i to the source node s by adding an edge $w(v_i, s)$ with capacity s_{iG}, and to the sink node t by adding an edge $w(v_i, t)$ with capacity s_{iB}. Finally, we add edges $w(v_i, v_j)$ with capacity s_{ij} to connect all pairs of comments.

Minimizing $C(P)$ amounts to finding a partition (S, T), where $S = \{s\} \cup S'$ and $T = \{t\} \cup T'$ for $s \notin S', t \notin T'$, that minimizes the cut capacity, i.e., the net flow crossing from S to T. One crucial advantage of this approach is that we can use *max-flow* algorithms to find the exact solution in polynomial time — near-linear in practice (Cormen et al., 2001; Boykov and Kolmogorov, 2004).

3.2 Integer Linear Programming Approach

Here we follow the *inference with classifiers* approach by Roth and Yih (2004), solved with Integer Linear Programming (ILP). We have one ILP problem per question–answer thread. We define a set of binary variables, whose assignment will univocally define the classification of all comments in the thread. In particular, we define a pair of variables for each answer: x_{iG} and x_{iB}, $1 \leq i \leq n$. Assigning 1 to x_{iG} means that comment c_i in the thread is classified as *Good*; assigning it 0 means that c_i is not classified as *Good*. The same applies to the other classes (here, only *Bad*). Also, we have a pair of variables for each pair of comments (to capture the pairwise relations): x_{ijS} and x_{ijD}, $1 \leq i < j \leq n$. Assigning 1 to x_{ijS} means that c_i and c_j have the same label; assigning 0 to x_{ijS} means that c_i and c_j do not have the same label. The same interpretation holds for the other possible classes (in this case only *Different*).[4]

Let c_{iG} be the cost of classifying c_i as *Good*, c_{ijS} be the cost of assigning the same labels to c_i and c_j, etc. Following (Roth and Yih, 2004), these costs are obtained from local classifiers by taking log probabilities, i.e., $c_{iG} = -\log s_{iG}$, $c_{ijS} = -\log s_{ij}$, etc. The goal of the ILP problem is to find an assignment A to all variables x_{iG}, x_{iB}, x_{ijS}, x_{ijD} that *minimizes* the cost function:

$$C(A) = \lambda \cdot \sum_{i=1}^{N} (c_{iG} \cdot x_{iG} + c_{iB} \cdot x_{iB}) + (1-\lambda) \cdot \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} (c_{ijS} \cdot x_{ijS} + c_{ijD} \cdot x_{ijD})$$

subject to the following constraints: (*i*) All variables are binary; (*ii*) One and only one label is assigned to each comment or pair of comments; (*iii*) The assignments to the comment variables and to the comment-pair variables are consistent: $x_{ijD} = x_{iG} \oplus x_{jG}, \forall i, j \quad 1 \leq i < j \leq n$. $\lambda \in [0, 1]$ is a parameter used to balance the contribution of the two sources of information.

4 Local Classifiers

For classification, we use Maximum Entropy, or MaxEnt, (Murphy, 2012), as it yields a probability distribution over the class labels, which we then use directly for the graph arcs and the ILP costs.

4.1 Good-vs-Bad Classifier

Our most important features measure the similarity between the question (q) and the comment (c). We compare lemmata and POS [1-4]-grams using Jaccard (1901), containment (Lyon et al., 2001), and cosine, as well as using some similarities from DKPro (Bär et al., 2012) such as longest common substring (Allison and Dix, 1986) and greedy string tiling (Wise, 1996). We also compute similarity using partial tree kernels (Moschitti, 2006) on shallow syntactic trees.

Forty-three Boolean features express whether (*i*) c includes URLs or emails, the words "yes", "sure", "no", "neither", "okay", etc., as well as '?' and '@' or starts with "yes" (12 features); (*ii*) c includes a word longer than fifteen characters (1);

[4] Setting a binary variable for each class label is necessary to have an objective function that is linear on the labels.

(*iii*) q belongs to each of the forum categories (26); and (*iv*) c and q were posted by the same user (4). An extra feature captures the length of c.

Four features explore whether c is close to a comment by the user who asked the question, u_q: (*i-ii*) there is a comment by u_q following c and (not) containing an acknowledgment or (*iii*) containing a question, or (*iv*) among the comments preceding c there is one by u_q asking a question. We model dialogues by identifying conversation chains between two users with three features: whether c is at the beginning/middle/end of a chain. There are copies of these features for chains in which u_q participates. Another feature for c_{u_i} checks whether the user u_i wrote more than one comment in the current thread. Three more features fire for the first/middle/last comment by u_i. One extra feature counts the total number of comments written by u_i. Finally, there is a feature modeling the position of c in the thread.

4.2 Same-vs-Different Classifier

We use the following types of features for a pair of comments (c_i, c_j): (*i*) all the features from the Good-vs-Bad classifier (i.e., we subtracted the feature vectors representing the two comments, $|v_i - v_j|$)[5]; (*ii*) the similarity features between the two comments, $sim(c_i, c_j)$; and (*iii*) the prediction from the Good-vs-Bad classifiers on c_i and c_j (i.e., the scores for c_i and c_j, the product of the two scores, and five boolean features specifying whether any of c_i and c_j are predicted as *Good*, *Bad*, and whether their predictions are identical).

5 Experiments and Evaluation

We performed standard pre-processing, and we further filtered user's signatures. All parameters (e.g., Gaussian prior for MaxEnt and the mixing λ for the graph-cut and ILP) were tuned on the development set. We also trained a second-order linear-chain CRF to check the contribution of the sequential relations between comments. We report results on the official SemEval test set for all methods. For the Same-vs-Different problem, we explored a variant of training with three classes, by splitting the *Same* class into *Same-Good* and *Same-Bad*. At test time, the probabilities of these two subclasses are added to get the probability of *Same* and all the algorithms are run unchanged.

[5]Subtracting vectors is standard in preference learning (Shen and Joshi, 2003). The absolute value is necessary to emphasize comment differences instead of preferences.

Classifier	P	R	F_1	Acc
baseline: *Same*				69.26
MaxEnt-2C	73.95	90.99	81.59	71.56
MaxEnt-3C	77.15	80.42	78.75	69.94

Table 2: **Same-vs-Different classification.** P, R, and F_1 are calculated with respect to *Same*.

Table 2 shows the results for the Same-vs-Different classification. We can see that the two-class MaxEnt-2C classifier works better than the three-class MaxEnt-3C. MaxEnt-3C has more balanced P and R, but loses in both F_1 and accuracy. MaxEnt-2C is very skewed towards the majority class, but performs better due to the class imbalance. Overall, it seems very difficult to learn with the current features, and both methods only outperform the majority-class baseline by a small margin. Yet, while the overall accuracy is low, note that the graph-cut/ILP inference allows us to recover from some errors, because if nearby utterances are clustered correctly, the wrong decisions should be outvoted by correct ones.

The results for Good-vs-Bad are shown in Table 3. On the top are the best systems at SemEval-2015 Task 3. We can see that our MaxEnt classifier is competitive: it shows higher accuracy than two of them, and the highest F_1 overall.[6]

System	P	R	F_1	Acc
Top-3 at SemEval-2015 Task 3				
JAIST	80.23	77.73	78.96	79.10
HITSZ-ICRC	75.91	77.13	76.52	76.11
QCRI	74.33	83.05	78.45	76.97
Instance Classifiers				
MaxEnt	75.67	84.33	79.77	78.43
Linear Chain Classifiers				
CRF	74.89	83.45	78.94	77.53
Global Inference Classifiers				
ILP	77.04	83.53	80.15	79.14‡
Graph-cut	78.30	82.93	**80.55**	**79.80**‡
ILP-3C	78.07	80.42	79.23	78.73
Graph-cut-3C	78.26	81.32	79.76	79.19†

Table 3: **Good-vs-Bad classification.** ‡ and † mark statistically significant differences in accuracy compared to the baseline MaxEnt classifier with confidence levels of 99% and 95%, respectively (randomized test).

[6]This comparison is not strictly fair as the SemEval systems were trained to predict three classes, and here we remapped them to two. We just want to show that our baseline system is very strong.

The CRF model is worse than MaxEnt on all measures, which suggests that the sequential information does not help. This can be because many interactions between comments are long-distance and there are gaps in the threads due to the annotation procedure at SemEval (Nakov et al., 2015).

However, global inference with graph-cut and ILP improves both F_1 and accuracy, mostly due to better recall. Graph-cut works better than ILP as it has higher precision, which helps F_1 and accuracy.

Both yield statistically significant improvements over the MaxEnt classifier; they also improve over the state-of-the-art JAIST system. Note that the devtest-tuned values of λ for graph-cut and ILP put much lower weight to the Same-vs-Different component (values are 0.95 and 0.91, respectively). Finally, as expected, using the predictions of MaxEnt-2C in the global classifiers is better than using those from MaxEnt-3C.

Q: I have a female friend who is leaving for a teaching job in Qatar in January. What would be a useful portable gift to give her to take with her?

$\mathbf{A}_1$ A couple of good best-selling novels. It's hard to find much here in Doha in the way of books.
Local: Good, Global: Good, Human: Good

$\mathbf{A}_2$ ipod to entertain herself in case of boredom... a lot of patience for her students...
Local: Good, Global: Good, Human: Good

$\mathbf{A}_3$ Thanks, please keep suggestions coming, would like to send her off with a useful gift.
Local: Bad, Global: Bad, Human: Bad

$\mathbf{A}_6$ Bacon. Nice bread, bacon, bacon, errmmm bacon and a pork joint..
Local: Bad, Global: Good, Human: Good

$\mathbf{A}_9$ Couple of good novels, All time favorite movies, ..
Local: Bad, Global: Good, Human: Good

$\mathbf{A}_{11}$ Ditto on the books and dvd's. Excedrin.
Local: Bad, Global: Bad, Human: Good

$\mathbf{A}_{12}$ Ditto on the bacon, pork sausage, pork chops, ham,..can you tell we miss pork! I think getting a care package together: her favorite perfume; shampoo; conditioner; glycerin soaps; set of DVDs of her favorite TV series.. Oh, and did I mention she should pack PATIENCE?
Local: Bad, Global: Good, Human: Good

Figure 2: **An excerpt of a thread** with decisions by local and global classifiers, and humans.

6 Discussion

We manually examined a number of examples where our global classfier could successfully recover from the errors made by the local classifier, and where it failed to do so. In Figure 2, we show the classification decisions of our local and global (graph-cut) classifiers along with the human annotations for an excerpt of a thread.

For example, consider answers A_6, A_9, and A_{12}, which were initially misclassified as *Bad* by the local classifier, but later recovered by the global classifier exploiting the *pairwise* information. In this case, the votes received by these answers from other *Good* answers in the thread for being in the *same* class won against the votes received from other *Bad* answers.

Now consider A_{11}, which our method failed to classify correctly as *Good*. Our investigation revealed that in this case the votes from the *Bad* answers won against the votes from the *Good* ones. The accuracy of the pairwise classifier has proven to be crucial for the performance of our overall framework. We probably need more informative features (e.g., textual entailment and semantic similarity to capture the relation between books and novels, movies and DVDs, etc.) in order to improve the pairwise classification performance.

7 Conclusion and Future Work

We have investigated the use of thread-level information for answer selection in CQA. We have shown that using a pairwise classifier that predicts whether two comments should get the same label, followed by a graph-cut (or ILP) global inference improves significantly over a very strong baseline as well as over the state of the art. We have further shown that using a linear-chain CRF model does not help, probably because many interactions between comments are long distance.

In future work, we would like to improve the pairwise classifiers with richer features, as this is currently the bottleneck for improving the performance in the global model. We further plan to test our framework on other CQA datasets, including on other languages.[7] Last but not least, we are interested in extending this research with even more global information, e.g., by modeling global decision consistency across multiple threads.

Acknowledgments

This research was performed by the Arabic Language Technologies (ALT) group at the Qatar Computing Research Institute (QCRI), HBKU, part of Qatar Foundation. It is part of the Interactive sYstems for Answer Search (Iyas) project, which is developed in collaboration with MIT-CSAIL.

[7] SemEval-2015 Task 3 had an Arabic subtask, but there the answers were not coming from the same thread.

References

Lloyd Allison and Trevor Dix. 1986. A bit-string longest-common-subsequence algorithm. *Inf. Process. Lett.*, 23(6):305–310, December.

Daniel Bär, Chris Biemann, Iryna Gurevych, and Torsten Zesch. 2012. UKP: Computing semantic textual similarity by combining multiple content similarity measures. In *Proceedings of the Sixth International Workshop on Semantic Evaluation*, SemEval '12, pages 435–440, Montréal, Canada.

Alberto Barrón-Cedeño, Simone Filice, Giovanni Da San Martino, Shafiq Joty, Lluís Màrquez, Preslav Nakov, and Alessandro Moschitti. 2015. Thread-level information for comment classification in community question answering. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, ACL-IJCNLP '15, pages 687–693, Beijing, China.

Yuri Boykov and Vladimir Kolmogorov. 2004. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(9):1124–1137, September.

Thomas H. Cormen, Clifford Stein, Ronald L. Rivest, and Charles E. Leiserson. 2001. *Introduction to Algorithms*. McGraw-Hill Higher Education.

Yongshuai Hou, Cong Tan, Xiaolong Wang, Yaoyun Zhang, Jun Xu, and Qingcai Chen. 2015. HITSZ-ICRC: Exploiting classification approach for answer selection in community question answering. In *Proceedings of the 9th International Workshop on Semantic Evaluation*, SemEval '15, pages 196–202, Denver, CO.

Paul Jaccard. 1901. Étude comparative de la distribution florale dans une portion des Alpes et des Jura. *Bulletin del la Société Vaudoise des Sciences Naturelles*, 37:547–579.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA.

Caroline Lyon, James Malcolm, and Bob Dickerson. 2001. Detecting short passages of similar text in large document collections. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '01, pages 118–125, Pittsburgh, PA.

Alessandro Moschitti. 2006. Efficient convolution kernels for dependency and constituent syntactic trees. In *Proceedings of the 17th European Conference on Machine Learning*, ECML '06, pages 318–329, Berlin, Germany.

Kevin Murphy. 2012. *Machine Learning A Probabilistic Perspective*. The MIT Press.

Preslav Nakov, Lluís Màrquez, Walid Magdy, Alessandro Moschitti, Jim Glass, and Bilal Randeree. 2015. SemEval-2015 task 3: Answer selection in community question answering. In *Proceedings of the 9th International Workshop on Semantic Evaluation*, SemEval '15, pages 269–281, Denver, CO.

Massimo Nicosia, Simone Filice, Alberto Barrón-Cedeño, Iman Saleh, Hamdy Mubarak, Wei Gao, Preslav Nakov, Giovanni Da San Martino, Alessandro Moschitti, Kareem Darwish, Lluís Màrquez, Shafiq Joty, and Walid Magdy. 2015. QCRI: Answer selection for community question answering - experiments for Arabic and English. In *Proceedings of the 9th International Workshop on Semantic Evaluation*, SemEval '15, pages 203–209, Denver, CO.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, ACL '04, pages 271–278, Barcelona, Spain.

Zhonghua Qu and Yang Liu. 2011. Finding problem solving threads in online forum. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, IJCNLP '11, pages 1413–1417, Chiang Mai, Thailand.

Dan Roth and Wen-tau Yih. 2004. A linear programming formulation for global inference in natural language tasks. In *Proceedings of the Eighth Conference on Computational Natural Language Learning*, CoNLL '04, pages 1–8, Boston, MA.

Libin Shen and Aravind K. Joshi. 2003. An SVM based voting algorithm with application to parse reranking. In *Proceedings of the Seventh Conference on Natural Language Learning*, CONLL '03, pages 9–16, Edmonton, Canada.

Michael Wise. 1996. Yap3: Improved detection of similarities in computer program and other texts. In *Proceedings of the Twenty-seventh SIGCSE Technical Symposium on Computer Science Education*, SIGCSE '96, pages 130–134, New York, NY.

Xiaoqiang Zhou, Baotian Hu, Qingcai Chen, Buzhou Tang, and Xiaolong Wang. 2015a. Answer sequence learning with neural networks for answer selection in community question answering. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*, pages 713–718, Beijing, China.

Xiaoqiang Zhou, Baotian Hu, Jiaxin Lin, Yang xiang, and Xiaolong Wang. 2015b. ICRC-HIT: A deep learning based comment sequence labeling system for answer selection challenge. In *Proceedings of the 9th International Workshop on Semantic Evaluation*, SemEval '15, pages 210–214, Denver, CO.

Key Concept Identification for Medical Information Retrieval

Jiaping Zheng
College of Information
and Computer Sciences
University of Massachusetts
Amherst, MA
jzheng@cs.umass.edu

Hong Yu
Bedford VA Medical Center
Bedford, MA
Department of Quantitative Health Sciences
University of Massachusetts
Worcester, MA
hong.yu@umassmed.edu

Abstract

The difficult language in Electronic Health Records (EHRs) presents a challenge to patients' understanding of their own conditions. One approach to lowering the barrier is to provide tailored patient education based on their own EHR notes. We are developing a system to retrieve EHR note-tailored online consumer oriented health education materials. We explored topic model and key concept identification methods to construct queries from the EHR notes. Our experiments show that queries using identified key concepts with pseudo-relevance feedback significantly outperform (over 10-fold improvement) the baseline system of using the full text note.

1 Introduction

Allowing patients access to their own electronic health records (EHRs) can enhance medical understanding and provide clinical relevant benefits (Wiljer et al., 2006), including increased medication adherence (Delbanco et al., 2012). However, EHR notes present unique challenges to the average patients. Since these notes are not usually targeted at the patients (Mossanen et al., 2014), languages that may be difficult for non-medical professionals to comprehend are prevalent, including medical terms, abbreviations, and medical domain-specific language patterns. The valuable and authoritative information contained in the EHR is thus less accessible to the patients, who ultimately stand to benefit the most from the information.

To address the challenges, we are developing a system to link medical notes to targeted education materials from trusted resources. The textual narratives in the EHR notes are not conducive to effective and efficient query. We therefore explored topic model and key concept identification methods to construct short queries from the EHR notes.

2 Related Work

Patent retrieval (Fujii et al., 2007) is similar to this work, as the queries are usually long and complex patent documents. Several methods have been proposed to construct shorter queries from the documents. For example, words in the summary section of a patent document can be ranked by TFIDF scores and extracted to form a query (Xue and Croft, 2009). Sentences that are similar to pseudo-relevant documents according to a language model are also used to reduce query length (Ganguly et al., 2011). Other similarity measures such as Kullback-Leibler divergence are used to extract terms, which are expanded to generate queries in the patent retrieval domain (Mahdabi et al., 2012). However, the patent retrieval domain is recall-driven, while in our scenario, patients are generally not expected to read relevant education documents exhaustively.

Various methods have also been proposed to retrieve documents relevant to passages of text or web documents. A model extended from CRF is proposed to identify noun phrases and named entities from a user-selected passage as queries (Lee and Croft, 2012). Similarly, noun phrases in a verbose query are also used as candidates for key concepts (Bendersky and Croft, 2008). Other related work that reduces long queries includes ranking all subsets of the original query (Kumaran and Carvalho, 2009). However, typical EHR notes are longer than the passages and verbose queries in these systems, which makes the graphical model and other learning based models less efficient. Moreover, parsers as required by Bendersky et al. (2009) and Named Entity Recognizers for the medical domain are less effective than the general domain.

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 579–584,
Lisbon, Portugal, 17-21 September 2015.

TREC Clinical Decision Support Track[1] is an IR challenge to link medical cases to information relevant for patient care. Unlike our system, this task is designed to address the physicians' information needs of diagnosis, testing and treatment for the patients.

3 Materials and Methods

3.1 Data

MedlinePlus[2] provides current and reliable information about over 900 health topics pages and 1000 medication pages to users in consumer-oriented lay language. Additionally, the medical encyclopedia section includes over 7000 articles about diseases, tests, symptoms, injuries, and surgeries. We include in this study the textual narratives in the "health topics", "drugs, supplements, and herbal information", and "medical encyclopedia" sections of the MedlinePlus as the collection of educational materials. There are a total of approximately 9400 articles in this collection, which we designate as *MedlinePlus*. Table 3.1 summarizes the characteristics of the collection.

We index the *MedlinePlus* documents with Galago, an advanced open source search engine. Galago implements the inference network retrieval model (Turtle and Croft, 1991). This model calculates the probability of the user's information needs being satisfied given a document in a directed acyclic graph. This framework is applied in many information retrieval tasks, and shown to be successful (Metzler et al., 2004).

Twenty progress notes are randomly selected from a corpus of de-identified EHR notes as the EHR document collection. Each note contains on average 261 tokens, with a standard deviation of 133. A physician read each note, and manually identified relevant education materials from the *MedlinePlus* documents. Each EHR note is linked to 22 education material documents on average. For example, Table 3.1 shows the summary of one EHR note and some of its relevant *MedlinePlus* documents. There are approximately 30 sentences or 360 tokens in the actual document.

We adopted Mean Average Precision (MAP) and Precision at 10 to evaluate the IR systems.

[1] http://www.trec-cds.org/

[2] http://www.nlm.nih.gov/medlineplus/

Summary of EHR Note
Patient remains in ICU with the following problems: respiratory failure, hemodynamics, renal failure, status post liver transplant, atrial fib, infectious disease, nutrition.
Select Relevant Documents
Respiratory Failure
Deep Vein Thrombosis
Aspiration pneumonia
Pulmonary Hypertension
Kidney Failure
Atrial Fibrillation or Flutter
Liver Transplantation
Dialysis - Hemodialysis

Table 2: Example EHR Note and its relevant documents

3.2 IR Systems

We investigate several query generation approaches, whereby short queries are built from an EHR note. In our queries, sequential dependence model (Metzler and Croft, 2005) was used to capture the dependencies in a multi-word query term. In this model, given a query, documents are ranked based on features of documents containing a single query term, two query terms sequentially appearing in the query, and two query terms in any order. This model has been shown to be effective in many applications (Balasubramanian et al., 2007; Cartright et al., 2011; Bendersky et al., 2009).

3.2.1 Baseline Approach

Our baseline approach issues each of the 20 test EHR notes as the query to the *MedlinePlus* document index and retrieves top 500 relevant documents. Although queries are not generally as long as EHR notes, an average patient without adequate medical knowledge may have difficulties constructing effective queries. Thus, this baseline can be considered as a proxy to how a patient actually conducts his own search in the real word.

3.2.2 LDA Models

We trained LDA topic models from over 6000 de-identified EHR notes to identify prominent topics from the test notes. The number of topics are selected based on retrieval performance. LDA models extract distributions over individual word tokens for each topic. However, medical con-

Document Type	Documents (Tokens)	Average Tokens (StdDev)
Health Topics	956 (141,185)	147.7 (37.6)
Medical Encyclopedia	7078 (5,126,101)	724.2 (363.7)
Drugs, Supplements, and Herbal Information	1332 (1,726,570)	1296.2 (992.8)
Total	9366 (6,993,856)	749.1 (565.9)

Table 1: MedlinePlus Collection

cepts often contain more than one token. We employ turbo topics (Blei and Lafferty, 2009) to find phrases from these topics. This method builds significant n-grams based on a language model of arbitrary length expressions. Queries are then generated from the n-grams. We take the top 5 phrases as queries from the topics that has a combined probability of over 80%.

3.2.3 Learning-Based Key Concept Identification

We also developed learning-based key concept identification to build queries from EHR notes. We employed Conditional Random Fields (CRF) model (Lafferty et al., 2001) to identify key concepts, which are most in need of explanation by external education materials. These key concepts can be considered in a broad sense topics, as they also capture various aspects of the EHR note content. We explored lexical, morphological, UMLS (Bodenreider, 2004) semantic type, and word embeddings as features. The word embeddings were induced from a combination of Wikipedia articles in the Medicine category and de-identified EHR progress notes.

Three CRF models are learned using different training data. One uses Wikipedia articles. Wikipedia, especially the Medicine category, is an appealing resource for such information as the human curated links in them are naturally concepts that are important. However, the number of articles in the Medicine category outnumbers our EHR notes substantially, we therefore restricted the Wikipedia articles to the Diabetes category. The paragraphs before the table of contents box are used. Anchor texts in these paragraphs are treated as key concepts. The second model uses EHR notes. Training data for the model was generated from the retrieval gold standard. A phrase in an EHR note would be annotated as a key concept if it matches the title of one of the note's relevant *MedlinePlus* documents. Lastly, we augmented the EHR corpus with Wikipedia articles, and compared its performance to the other CRF-based models. Leave-one-out cross validation is used in the last two models.

	System	P@10	MAP	Increase
1	Baseline	0%	0.0091	-
2	CHV	5%	0.0240	2.6
3	LDA	10%	0.0489	5.4
4	Key (Wiki)	16%	0.0851	9.4
5	Key (EHR)	16.5%	0.0879	9.7
6	Key (Wiki+EHR)	18%	0.1030	11.3

Table 3: System Performance

3.2.4 Query Expansion

We explored query expansion by incorporating relevance feedback from pseudo-relevant documents. The initial queries are generated using methods described previously. Among the top 20 retrieval results, those with a title that matches one of the identified key concepts are considered pseudo-relevant documents. This additional requirement is to ensure that the expanded concepts do not drift from the main topic of the medical notes. From these documents, medical concepts are extracted using MetaMap (Aronson and Lang, 2010). These concepts, with their synonyms provided by UMLS, are used as expansions.

4 Experiment Results

4.1 LDA Models

100 topics are learned from the de-identified EHR note collection. This level of topic granularity shows the best performance in our experiments. The retrieval result is shown in Table 4, row 3. The improvement over the baseline is 5.4 folds, and is statistically significant using a paired Student's t-test ($p < 0.05$).

4.2 Key Concept Identification

The key concept identification performance of the three CRF models is shown in Table 4.2. Retrieval

	Training Data		
	Wiki	EHR	Wiki+EHR
Precision	16.27%	35.92%	33.79%
Recall	26.88%	34.09%	33.18%
F1	18.74%	33.70%	32.54%

Table 4: Key concept identification performance

Training Data	P@10	MAP
Wiki	20% (16%)	0.1067 (0.0851)
EHR	16% (16.5%)	0.0951 (0.0879)
Wiki+EHR	21% (18%)	0.1169 (0.1030)

Table 5: System performance with pseudo-relevance feedback. Numbers in parentheses are without pseudo-relevance feedback.

performance of these models are shown in Table 4, rows 4 to 6. All systems showed a statistical significant improvement over the baseline. The last model's improvement is also statistically significant over the LDA approach. Query expansion methods further improved system performance, as shown in Table 4.2.

5 Discussions

From the LDA model, Table 5 shows the top 10 n-grams from 7 topics trained on the medical text. It is clear that while topics like the first one capture medical concepts, others like the second one do not. The LDA results also highlight the noisy nature of the EHR notes. Queries formed by including the generic or noisy terms such as "continue on" will not benefit retrieval results. Examining the retrieval results, we found that when the prominent topics include medical concepts, the top 10 results usually contain at least one relevant document. When only generic topics are identified, relevant documents are absent in the top 10 results.

The CRF models, on the other hand, are better at capturing the concepts in the EHR notes. All three models outperformed the LDA model. One drawback of using models trained from title-matching phrases is that the identified key concepts fail to cover the scope of the medical concepts contained in an EHR note. On average only 23.6% of the concepts are annotated as key phrases in each note. In the evaluation of the CRF models for their key concept identification performance, inclusion of Wikipedia data slightly decreased the F1 score when they were tested on EHR notes. This can be attributed to the fact that the Wikipedia articles outnumbered the EHR data by 7 times. However, this data helped improve the coverage of the key concepts. In one document, the augmented model identified two more out of the total eight key concepts.

	Phrases with the highest probability
1	dialysis, hemodialysis, catheter, renal failure, renal, coumadin, line, picc line, dialysis catheter, failure
2	job id, today, point, continue on, reasonable, try to, continue, yesterday, left, right
3	continue, patient, job id, pain, patient has, normal, patient s, white count, secondary to, culture
4	liver, ascites, normal, tenderness, fluid, stable, elevated, today, edema, chest
5	preliminary, patient, patient s, time, blood, mmoll, patient has, routine, high, vial
6	diarrhea, abdominal, flagyl, stool, abdominal pain, colitis, abdomen, difficile, fluid, distended
7	bipap, pneumonia, year old, respiratory failure, failure, minutes, requiring, encephalopathy, ards, patient

Table 6: Top 10 n-grams from 7 topics using the LDA model

Finally, to investigate the gap between medical language and lay language, we substituted the medical concepts recognized by MetaMap with their consumer-oriented counterparts created by the Consumer Health Vocabulary (CHV) Initiative (Zeng and Tse, 2006). Performance using the substituted EHR notes as shown in Table 4, row 2 more than doubled. The gap highlights the issue that patients may have difficulty finding relevant health information without assistance due to vocabulary mismatch.

6 Conclusions, Limitations and Future Plan

We have shown that using full EHR notes is ineffective at retrieving relevant education materials. Identifying key concepts of an EHR note and then using the key concepts as query terms re-

sult in significantly improved performance (over 10-fold). Furthermore, a query expansion approach in which key concepts are complemented by other medical concepts from pseudo-relevant documents further improves the performance.

One limitation of our design is that only one physician provided relevancy judgments. Additional annotators would provide a more rigorous set of gold standard, allowing us to measure inter-annotator agreement.

There are several directions we can explore in our future research. Firstly, our key concept identification methods are not optimized for the retrieval results, but for the identification subtask only. We hypothesize that directly optimizing the key concept identifier for retrieval would lead to better performance. We would also investigate domain adaptation techniques to learn key concept identification models from other data sources.

Acknowledgments

This work was supported in part by the Award 1I01HX001457 from the United States Department of Veterans Affairs Health Services Research and Development Program Investigator Initiated Research. The contents do not represent the views of the U.S. Department of Veterans Affairs or the United States Government.

References

Alan R Aronson and Franois-Michel Lang. 2010. An overview of MetaMap: historical perspective and recent advances. *J Am Med Inform Assoc*, 17(3):229–236, June.

Niranjan Balasubramanian, James Allan, and W. Bruce Croft. 2007. A comparison of sentence retrieval techniques. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '07, pages 813–814, New York, NY, USA. ACM.

Michael Bendersky and W. Bruce Croft. 2008. Discovering key concepts in verbose queries. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '08, pages 491–498, New York, NY, USA. ACM.

Michael Bendersky, W. Bruce Croft, and David A. Smith. 2009. Two-stage query segmentation for information retrieval. In *Proceedings of the 32Nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '09, pages 810–811, New York, NY, USA. ACM.

David M. Blei and John D. Lafferty. 2009. Visualizing topics with multi-word expressions. *arXiv:0907.1013 [stat]*, July. arXiv: 0907.1013.

Olivier Bodenreider. 2004. The unified medical language system (UMLS): integrating biomedical terminology. *Nucleic Acids Res.*, 32(Database issue):D267–270, January.

Marc-Allen Cartright, Henry A. Feild, and James Allan. 2011. Evidence finding using a collection of books. In *Proceedings of the 4th ACM Workshop on Online Books, Complementary Social Media and Crowdsourcing*, BooksOnline '11, pages 11–18, New York, NY, USA. ACM.

Tom Delbanco, Jan Walker, Sigall K. Bell, Jonathan D. Darer, Joann G. Elmore, Nadine Farag, Henry J. Feldman, Roanne Mejilla, Long Ngo, James D. Ralston, Stephen E. Ross, Neha Trivedi, Elisabeth Vodicka, and Suzanne G. Leveille. 2012. Inviting patients to read their doctors' notes: a quasi-experimental study and a look ahead. *Ann. Intern. Med.*, 157(7):461–470, October.

Atsushi Fujii, Makoto Iwayama, and Noriko Kando. 2007. Introduction to the special issue on patent processing. *Information Processing & Management*, 43(5):1149–1153, September.

Debasis Ganguly, Johannes Leveling, Walid Magdy, and Gareth J.F. Jones. 2011. Patent query reduction using pseudo relevance feedback. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, CIKM '11, pages 1953–1956, New York, NY, USA. ACM.

Giridhar Kumaran and Vitor R. Carvalho. 2009. Reducing long queries using query quality predictors. In *Proceedings of the 32Nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '09, pages 564–571, New York, NY, USA. ACM.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. 18th International Conf. on Machine Learning*, pages 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Chia-Jung Lee and W. Bruce Croft. 2012. Generating queries from user-selected text. In *Proceedings of the 4th Information Interaction in Context Symposium*, IIIX '12, pages 100–109, New York, NY, USA. ACM.

Parvaz Mahdabi, Linda Andersson, Mostafa Keikha, and Fabio Crestani. 2012. Automatic refinement of patent queries using concept importance predictors. In *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '12, pages 505–514, New York, NY, USA. ACM.

Donald Metzler and W. Bruce Croft. 2005. A markov random field model for term dependencies. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '05, pages 472–479, New York, NY, USA. ACM.

Donald Metzler, Trevor Strohman, Yun Zhou, and W. B. Croft. 2004. Indri at TREC 2005: Terabyte track.

Matthew Mossanen, Lawrence D. True, Jonathan L. Wright, Funda Vakar-Lopez, Danielle Lavallee, and John L. Gore. 2014. Surgical pathology and the patient: a systematic review evaluating the primary audience of pathology reports. *Hum. Pathol.*, July.

Howard Turtle and W. Bruce Croft. 1991. Evaluation of an inference network-based retrieval model. *ACM Trans. Inf. Syst.*, 9(3):187–222, July.

David Wiljer, Sima Bogomilsky, Pamela Catton, Cindy Murray, Janice Stewart, and Mark Minden. 2006. Getting results for hematology patients through access to the electronic health record. *Can Oncol Nurs J*, 16(3):154–164.

Xiaobing Xue and W. Bruce Croft. 2009. Transforming patents into prior-art queries. In *Proceedings of the 32Nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '09, pages 808–809, New York, NY, USA. ACM.

Qing T. Zeng and Tony Tse. 2006. Exploring and developing consumer health vocabularies. *J Am Med Inform Assoc*, 13(1):24–29, February.

Image-Mediated Learning for Zero-Shot Cross-Lingual Document Retrieval

Ruka Funaki, Hideki Nakayama
Machine Perception Group
Graduate School of Information Science and Technology
The University of Tokyo
{funaki,nakayama}@nlab.ci.i.u-tokyo.ac.jp

Abstract

We propose an image-mediated learning approach for cross-lingual document retrieval where no or only a few parallel corpora are available. Using the images in image-text documents of each language as the hub, we derive a common semantic subspace bridging two languages by means of generalized canonical correlation analysis. For the purpose of evaluation, we create and release a new document dataset consisting of three types of data (English text, Japanese text, and images). Our approach substantially enhances retrieval accuracy in zero-shot and few-shot scenarios where text-to-text examples are scarce.

1 Introduction

Cross-lingual document retrieval (CLDR) is the task of finding relevant documents in one language given a query document in another language. While sufficiently large-scale corpora are critical for parallel corpus-based learning methods, manually creating corpora requires huge human effort and is unrealistic in many cases.

A straightforward approach is to crawl bilingual documents from the Web for use as training data. However, because most documents on the Web are written in one language, it is not always easy to collect a sufficient number of multilingual documents, especially those involving minor languages. Let us consider the multimedia information in documents. We can, for example, find abundant pairings of text and images, e.g., text with the ALT property of <IMG> tags in HTML, text with photos posted to social networking sites, and articles on Web news posted with images. Unlike text, an image is a universal representation; we can easily understand the semantic content of images regardless of our mother tongue. Motivated by this observation, we expect that we can learn the relation of two languages indirectly through images, even if we do not have sufficient bilingual text pairs (Figure 1).

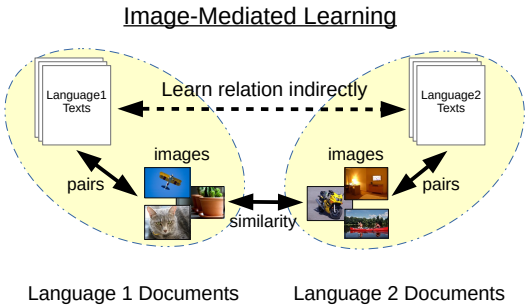


Figure 1: Concept of image-mediated learning. Our idea is to learn the relation between two languages indirectly by using images attached to text. If two documents written in different languages include images with similar image features, it is likely that the texts contained in the two documents are similar. Based on this idea, we seek the relation of texts written in different languages mediated by the similarity between images.

Generally, traditional image recognition techniques (or image features) are very poor compared with those in the natural language processing field. In recent years, however, deep learning has resulted in a breakthrough in visual recognition and dramatically improved image recognition accuracy in generic domains, which is rapidly approaching human recognition levels (Fang et al., 2015). We expect that these state-of-the-art image recognition technologies can effectively assist CLDR tasks.

We show that hub images enable zero-shot training of CLDR systems and improve retrieval accuracy given only a few parallel text samples.

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 585–590,
Lisbon, Portugal, 17-21 September 2015.

2 Related Work

2.1 Multimodal Learning for CLDR

Multimodal learning, defined as a framework for machine learning using inputs from multiple media or sensors, has played a key role in various cross-modal applications. The most widely used standard method for multimodal learning is canonical correlation analysis (CCA) (Hotelling, 1936), which projects multimodal data into a shared representation. For example, CCA has been successfully used in image retrieval (tag to images) and image annotation (image to tags) (Hardoon et al., 2004; Rasiwasia et al., 2010; Gong et al., 2014). In the context of CLDR, each language's texts constitute one modality. CCA has also commonly been used for cross-lingual information retrieval (Vinokourov et al., 2002; Li and Shawe-Taylor, 2004; Udupa and Khapra, 2010). Whereas CCA can handle only two modalities, we need to consider relations between three modalities because we use images in addition to the two languages. Therefore, we focus on an extension of CCA, generalized canonical correlation analysis (GCCA), to handle more than two inputs (Kettenring, 1971).

2.2 Zero-Shot Learning for CLDR

Our core idea is to use another modality (image) as a hub to indirectly learn the relevance between two different languages. The work by Rupnik et al. is probably the closest to ours (Rupnik et al., 2012). In their study, they used a popular language (e.g., English) with enough bilingual documents shared with other languages as a hub to enhance CLDR for minor languages with few direct bilingual texts available. Nevertheless, this method assumes that parallel corpora of the hub and target languages exist and therefore, its application is limited to specific domains where manual translations are readily available, such as Wikipedia and news sites. Contrarily, because we use images as the hub, we can use documents closed with respect to each language for training. Considering that current generic Web documents are mostly closed with respect to one language, yet equipped with rich multimedia data, our setup is assumed to be more reasonable.

	Division	English	Images	Japanese
1	[train-E/I]	E_1	I_1	-
2	[train-I/J]	-	I_2	J_2
3	[train-E/J]	E_3	-	J_3
4	[test-E/J]	E_4	-	J_4

Table 1: Division of training and test data. Each division of training dataset is missing one of the three modalities.

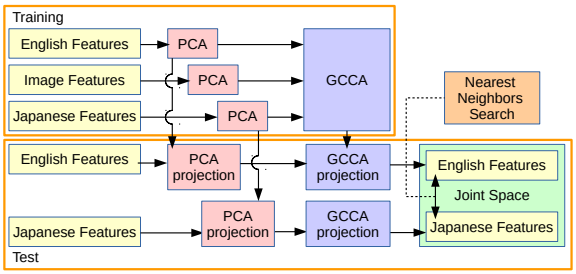


Figure 2: System overview

3 Our Approach

3.1 Overview of Image-Mediated Learning

We use the following notations for specifying each non-overlapping data division.

1. [train-E/I]: Training documents consisting of English text and images.
2. [train-I/J]: Training documents consisting of images and Japanese text.
3. [train-E/J]: Training documents consisting of English and Japanese text.
4. [test-E/J]: Test documents consisting of English and Japanese text.

We define IDs for each modality in each division as given in Table 1. For example, E_1 represents features of English text in the [train-E/I] division. Typical CLDR based on parallel corpora uses only [train-E/J] for training and [test-E/J] for evaluation. In the zero-shot learning scenario without any [train-E/J] data, we use only [train-E/I] and [train-I/J] for training. In the few-shot learning scenario, we also use a small number of [train-E/J] samples. We call this approach image-mediated learning.

An overview of our system is depicted in Figure 2. We compress features by principal component analysis (PCA) and train them by GCCA. For testing, we compress features by PCA, project features by GCCA, then, search the nearest neighbors from Japanese to English in the joint space.

3.2 Feature Extraction

A convolutional neural network (CNN) is one of the most successful deep learning methods for visual recognition. It is known that we can obtain very good image features by taking activation of hidden neurons in a network pre-trained by a sufficiently large dataset (Donahue et al., 2013). We apply the CNN model pre-trained using the ILSVRC2012 dataset (Russakovsky et al., 2015) provided by Caffe (Jia et al., 2014), a standard deep learning software package in the field of visual recognition.

As the text feature for both English and Japanese, we use the bag-of-words (BoW) representation and term frequency-inverse document frequency (TF-IDF) weighting. The MeCab (Kudo et al., 2004) library is used to divide Japanese text into words by morphological analysis. No preprocessing approaches like eliminating stop words and stemming, are used.

3.3 GCCA

GCCA is a generalization of CCA for any m modalities ($m = 3$ in our case). Although several slightly different versions of GCCA have been proposed (Carroll, 1968; Rupnik et al., 2012; Velden and Takane, 2012), we implement the simplest one (Kettenring, 1971) because GCCA itself is not the main focus of this study.

Let E, I, and J denote English, images, and Japanese, respectively. For feature vector $\mathbf{x}_k, \forall k \in \{E, I, J\}$, let $\mathbf{z}_k = (\mathbf{x}_k - \overline{\mathbf{x}}_k)\mathbf{h}_k$ denote its canonical variables. Σ_{ij} denotes a covariance matrix of modalities i and j where $i, j \in \{E, I, J\}$. Projection vectors $\mathbf{h}_k$ are computed such that they maximize the sum of correlations between each pair of modalities obtained by solving the following generalized eigenvalue problem:

$$\frac{1}{2}\begin{pmatrix} \mathbf{0} & \Sigma_{EI} & \Sigma_{EJ} \\ \Sigma_{IE} & \mathbf{0} & \Sigma_{IJ} \\ \Sigma_{JE} & \Sigma_{JI} & \mathbf{0} \end{pmatrix}\mathbf{h} = \rho\begin{pmatrix} \Sigma_{EE} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \Sigma_{II} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \Sigma_{JJ} \end{pmatrix}\mathbf{h} \quad (1)$$

where $\mathbf{h} = (\mathbf{h}_E^T, \mathbf{h}_I^T, \mathbf{h}_J^T)^T$. The canonical axises $\mathbf{h}$ are normalized such that $\frac{1}{3}\sum_{k\in\{E,I,J\}}^{3} \mathbf{h}_k^T \Sigma_{kk} \mathbf{h}_k = 1$. Additionally, we add a regularization term to the self covariance matrices to prevent over-fitting; that is, we set $\Sigma_{kk} \rightarrow \Sigma_{kk} + \alpha I$, where α is a parameter to avoid the singularity issue.

Despite our training datasets having only two of the three modalities as given in Table 1, we can handle this situation naturally by computing covariance matrices from the available data only. For example, in the few-shot learning scenario, we compute Σ_{EI} using E_1 and I_1, and Σ_{EE} using E_1 and E_3. In the zero-shot learning scenario, because [train-E/J] is not available, we compute Σ_{EE} using E_1 only and use a zero matrix for Σ_{EJ}.

3.4 Nearest Neighbor Search in Joint Space

We can find relevant documents in another language by computing the distances from the query documents using the coupled canonical subspaces. Having set Japanese as the query language, we retrieve documents written in English. Nearest neighbors are obtained as follows:

$$\hat{j} := \arg\min_j d(\mathbf{z}_J^i, \mathbf{z}_E^j), \quad (2)$$

where $\mathbf{z}_J^i$, $\mathbf{z}_E^j$ are projected feature vectors of the query and target documents, respectively, and $d(\cdot)$ is a distance function, which in our case, is the Euclidean distance.

4 Experiment

4.1 Pascal Sentence Dataset with Japanese Translation

The UIUC Pascal Sentence Dataset (Rashtchian et al., 2010) contains 1000 images, each of which is annotated with five English sentences describing its content. This dataset was originally created for the study of sentence generation from images, which is one of the current hot topics in computer vision. To establish a new benchmark dataset for image-mediated CLDR, we included a Japanese translation for each English sentence provided by professional translators[1], as shown in Figure 3. In this experiment, we bundled the five sentences attached to each image for use as one text document. Therefore, in our setup, each of the 1000 documents in the dataset consists of three items: an image, and the corresponding English and Japanese text.

[1]Dataset is available at:
http://www.nlab.ci.i.u-tokyo.ac.jp/dataset/pascal_sentence_jp/

English Texts	Images	Japanese Texts
- A family on a boat with a cross on a river - A happy couple with a young child wearing a life preserver sitting on a boat. - A man, a woman, and a child sit on boat with a large cross on it. - A man, women and small child sitting on top of a boat moving along the river. - Family of three sitting on deck, child wearing red vest, brush and shoes are seen in the foreground.		- 川で十字架のあるボートに乗っている家族。 - ボートに座っている、救命具を着た幼い子どもと幸せなカップル。 - 男性、女性と子どもが大きな十字架のあるボートに座っています。 - 川に沿って動いているボートの上部に座っている男性、女性と小さな子ども。 - ブラシと靴が前景に写されている、子どもが赤いベストを着て、デッキに座っている三人の家族。
- A black and white cow in a grassy field stares at the camera. - A black and white cow standing in a grassy field. - A black and white cow stands on grass against a partly cloudy blue sky. - a cow is gazing over the grass he is about to graze - Black and white cow standing in grassy field.		- 草地の黒と白の雌牛がカメラをじっと見ています。 - 草地に立っている黒と白の雌牛。 - 一部曇った青空を背に黒と白の雌牛が草地に立っています。 - 雌牛が食べようとしている草をじっと眺めています。 - 草地に立っている黒と白の雌牛。
⋮	⋮	⋮

Figure 3: Examples from the Pascal Sentence Dataset with Japanese translations: each image has about five sentences describing it from different perspectives.

4.2 Evaluation

We randomly sampled data from the dataset for each division in Table 1 without any overlap; we ignored the modality of each document that was not available in each data division (e.g., Japanese text in [train-E/I]). We ran experiments with varying sample sizes for [train-E/I] and [train-I/J], that is, 100, 200, 300, and 400. Furthermore, we gradually increased the number of [train-E/J] samples from 0 to 100 to emulate the few-shot learning scenario. The size of the test data [test-E/J] was fixed at 100. Following this setup, we performed image-mediated CLDR based on GCCA, and compared the results with those obtained by standard CLDR using only [train-E/J] data with CCA. We evaluated the performance with respect to the top-1 Japanese to English retrieval accuracy in the test data. Given that we used 100 test samples, the chance rate was 1%. For each run, we conducted 50 trials randomly replacing data and used the average score. All features were compressed into 100 dimensions via PCA and α was set to 0.01.

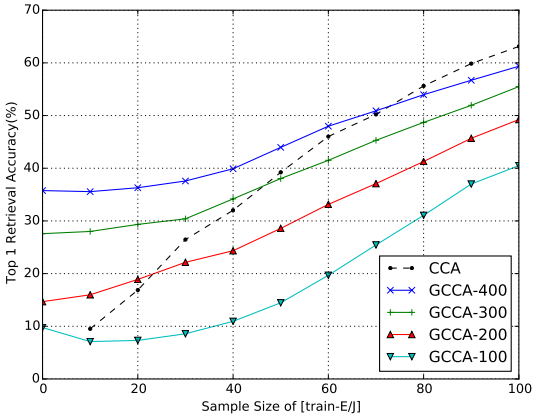


Figure 4: Retrieval accuracy varying the number of [train-E/J] data. Each colored line shows the performance of our method with a different sample size of [train-E/I] and [train-I/J] data (e.g., GCCA-400 denotes respective 400 samples of [train-E/I] and [train-I/J] for GCCA). We used image features extracted from GoogLeNet and text features represented as bags-of-words.

The experimental results, illustrated in Figure 4, clearly show that better accuracy is obtained with a greater number of text-image data in both zero-shot and few-shot scenarios. We can expect even better zero-shot accuracy with more text-image data, although, we cannot increase [train-E/I] and [train-I/J] more than 400 each in the current setup because of the restricted dataset size. We summarized results in zero-shot scenario in Table 2 in several cases. Although both GCCA and CCA show improved performance as the sample size of [train-E/J] increases, not surprisingly, GCCA is gradually overtaken by CCA when we have enough samples to learn the relevance between English and Japanese texts directly. However, accuracies of image-mediated learning in the cases when [train-E/J] is scarce are higher than CCA baseline. Hence, we confirmed that the image-mediated model is also effective in the few-shot learning scenario.

Model	Accuracy(%)
GCCA-400(BoW)	37.4 ± 3.8
GCCA-300(BoW)	27.6 ± 3.4
GCCA-200(BoW)	14.7 ± 3.2
GCCA-100(BoW)	9.8 ± 2.3
GCCA-400(TF-IDF)	42.0 ± 4.6
GCCA-300(TF-IDF)	31.6 ± 4.3
GCCA-200(TF-IDF)	17.2 ± 2.6
GCCA-100(TF-IDF)	11.8 ± 2.7

Table 2: Accuracy of zero-shot learning. Image features are extracted from GoogLeNet and both the bag-of-words (BoW) and the TF-IDF model are used as text features.

Feature	Accuracy(%)
CNN(GoogLeNet), BoW	37.4 ± 3.8
CNN(VGG), BoW	31.3 ± 3.5
CNN(CaffeNet), BoW	25.1 ± 3.4
FisherVector, BoW	10.8 ± 2.7
CNN(GoogLeNet), TF-IDF	42.0 ± 4.6
CNN(VGG), TF-IDF	37.8 ± 2.9
CNN(CaffeNet), TF-IDF	29.7 ± 4.4
FisherVector, TF-IDF	12.6 ± 2.7

Table 3: Accuracy of zero-shot learning in multiple image features. The sample size of both [train-E/I] and [train-I/J] is 400. Both the bag-of-words (BoW) and the TF-IDF model are used as text features.

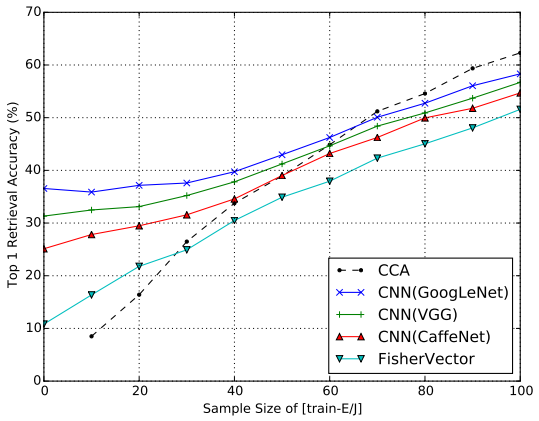


Figure 5: Retrieval accuracy using different image features in image-mediated CLDR. The sample size of both [train-E/I] and [train-I/J] is 400. Text features are based on the bag-of-words model.

4.3 Effect of Image Features

We also verified the effect of the performance of image features in our framework (see Figure 5 and Table 3). CNN has improved dramatically over the last few years, and many new powerful pre-trained networks are currently available. We compared three different features extracted from GoogLeNet (Szegedy et al., 2014), VGG 16 layers (Chatfield et al., 2014), and CaffeNet (Jia et al., 2014; Krizhevsky et al., 2012). Additionally, we tested the Fisher Vector (Perronnin et al., 2010), which was the standard hand-crafted image feature before deep learning. We extracted features from the pool5/7x7_s1 layer in GoogLeNet, fc6 layer in VGG, and fc6 layer in CaffeNet. For the Fisher Vector, following the standard implementation, we compressed SIFT descriptors (Lowe, 1999) into 64 dimensions by PCA, and used a Gaussian mixture model with 64 components. We used four spatial grids for the final feature extraction. Overall, the order of performance of features corresponds to that known in the image classification domain (Russakovsky et al., 2015). This result indicates that when more powerful image features are used, better performance can be achieved in image-mediated CLDR.

5 Conclusion

We proposed an image-mediated learning approach to realize zero-shot or few-shot CLDR. For evaluation, we created and released a new dataset consisting of Japanese, English, and image triplets, based on the widely used Pascal Sentence Dataset. We showed that state-of-the-art CNN-based image features can substantially improve zero-shot CLDR performance. Considering that image features have continued to improve rapidly since the deep learning breakthrough and the universality of images in Web documents, this approach could become even more important in the future.

Acknowledgments

This work was supported by JST CREST, JSPS KAKENHI Grant Number 26730085. We thank the three anonymous reviewers for their helpful comments.

References

Douglas J Carroll. 1968. Generalization of canonical correlation analysis to three or more sets of vari-

ables. In *Proceedings of the 76th Annual Convention of the American Psychological Association*, volume 3, pages 227–228.

Ken Chatfield, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2014. Return of the Devil in the Details: Delving Deep into Convolutional Nets. In *Proceedings of the British Machine Vision Conference*, pages 1–11.

Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. 2013. DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition. In *Proceedings of the International Conference on Machine Learning*, pages 647–655.

Hao Fang, Saurabh Gupta, Forrest Iandola, K. Rupesh Srivastava, Li Deng, Piotr Dollár, Jianfeng Gao, Xiaodong He, Margaret Mitchell, John C. Platt, C. Lawrence Zitnick, and Geoffrey Zweig. 2015. From Captions to Visual Concepts and Back. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Yunchao Gong, Qifa Ke, Michael Isard, and Svetlana Lazebnik. 2014. A Multiview Embedding Space for Modeling Internet Images, Tags, and their Semantics. *International Journal of Computer Vision*, 106(2):210–233.

David R Hardoon, Sandor Szedmak, and John Shawe-Taylor. 2004. Canonical Correlation Analysis: an Overview with Application to Learning Methods. *Neural Computation*, 16(12):2639–2664.

Harold Hotelling. 1936. Relations between Two Sets of Variants. *Biometrika*, 28:321–377.

Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. 2014. Caffe : Convolutional Architecture for Fast Feature Embedding. In *Proceedings of the ACM International Conference on Multimedia*, pages 675–678.

Jon Robers Kettenring. 1971. Canonical Analysis of Several Sets of Variables. *Biometrika*, 58(3):433–451.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105.

Taku Kudo, Kaoru Yamamoto, and Yuji Matsumoto. 2004. Applying Conditional Random Fields to Japanese Morphological Analysis. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 230–237.

Yaoyong Li and John Shawe-Taylor. 2004. Using KCCA for Japanese-English Cross-language Information Retrieval and Classification. In *Learning Methods for Text Understanding and Mining Workshop*, pages 117–133.

David G Lowe. 1999. Object recognition from local scale-invariant features. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 2, pages 1150–1157.

Florent Perronnin, Jorge Sánchez, and Thomas Mensink. 2010. Improving the Fisher kernel for large-scale image classification. In *Proceedings of the European Conference on Computer Vision*, pages 143–156.

Cyrus Rashtchian, Peter Young, Micah Hodosh, Julia Hockenmaier, and North Goodwin Ave. 2010. Collecting Image Annotations Using Amazon ' s Mechanical Turk. *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, pages 139–147.

Nikhil Rasiwasia, Jose Costa Pereira, Emanuele Coviello, Gabriel Doyle, Gert R.G. Lanckriet, Roger Levy, and Nuno Vasconcelos. 2010. A New Approach to Cross-modal Multimedia Retrieval. *Proceedings of the International Conference on Multimedia*, pages 251–260.

Jan Rupnik, Andrej Muhič, and Primo Škraba. 2012. Cross-Lingual Document Retrieval through Hub Languages. In *Neural Information Processing Systems Workshop*.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. 2015. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*.

Christian Szegedy, Scott Reed, Pierre Sermanet, Vincent Vanhoucke, and Andrew Rabinovich. 2014. Going deeper with convolutions. *CoRR abs/1409.4842*.

Raghavendra Udupa and Mitesh M Khapra. 2010. Improving the Multilingual User Experience of Wikipedia Using Cross-Language Name Search. *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 492–500.

Michel Velden and Yoshio Takane. 2012. Generalized Canonical Correlation Analysis with Missing Values. *Computational Statistics*, 27(3):551–571.

Alexei Vinokourov, John Shawe-Taylor, and Nello Cristianini. 2002. Inferring a Semantic Representation of Text via Cross-Language Correlation Analysis. *Advances in Neural Information Processing Systems*, pages 1473–1480.

Detecting Risks in the Banking System by Sentiment Analysis

Clemens Nopp
TU Wien
clemens.nopp@alumni.tuwien.ac.at

Allan Hanbury
Institute of Software Technology
and Interactive Systems
TU Wien
hanbury@ifs.tuwien.ac.at

Abstract

In November 2014, the European Central Bank (ECB) started to directly supervise the largest banks in the Eurozone via the Single Supervisory Mechanism (SSM). While supervisory risk assessments are usually based on quantitative data and surveys, this work explores whether *sentiment analysis* is capable of measuring a bank's attitude and opinions towards risk by analyzing text data. For realizing this study, a collection consisting of more than 500 CEO letters and outlook sections extracted from bank annual reports is built up. Based on these data, two distinct experiments are conducted. The evaluations find promising opportunities, but also limitations for risk sentiment analysis in banking supervision. At the level of individual banks, predictions are relatively inaccurate. In contrast, the analysis of aggregated figures revealed strong and significant correlations between uncertainty or negativity in textual disclosures and the quantitative risk indicator's future evolution. Risk sentiment analysis should therefore rather be used for macroprudential analyses than for assessments of individual banks.

1 Introduction

From 2007 on, a global crisis struck the financial markets and led to a severe slow-down of the real economy. It was triggered by the collapsing US subprime mortgage sector, where loans had been issued to borrowers with poor credit ratings. Due to the tight interconnectedness of the financial system, problems quickly propagated in the global banking system. Governments had to bail out important institutions like *Northern Rock*, but such solutions could not be provided for every troubled bank. In September 2008, the large investment bank *Lehman Brothers* had to file bankruptcy. In the aftermath of this event, further banks had to be rescued in order to stabilize the financial system. This deep financial crisis highlighted the necessity of better financial regulation as well as more effective financial supervision in the future (Hodson and Quaglia, 2009).

As a reaction to the crisis and its severe economic consequences, EU institutions decided to build up an *European Banking Union* (EBU). The EBU consists of three pillars, one of them being a new system of financial supervision, the *Single Supervisory Mechanism* (SSM). Its goal is to "promote long-term safety and soundness of credit institutions and the stability of the financial system within the Union and each Member State [...]" (Council of the EU, 2013, p. 72).

For supervising over 120 of the largest banks in the Eurozone, the SSM utilizes a range of information sources in order to detect vulnerabilities and risks. The sources include mainly backward-looking quantitative *Key Risk Indicators* (KRIs), which are complemented with surveys in order to include forward-looking information as well (European Banking Authority, 2014). However, another source of information seems to be largely untapped, namely textual data published by the banks. Publications like periodic reports, press releases, and news published for investors also contain forward-looking information. Analyzing this readily available data would be more cost-efficient in comparison to traditional approaches like surveys. It could provide answers to questions like: what does official communication by banks reveal about their expectations and attitudes towards risk?

In this paper, we present a novel application of *sentiment analysis* for exploring attitudes and opinions about risk in textual disclosures by

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 591–600,
Lisbon, Portugal, 17-21 September 2015.

banks. In particular, this work (1) finds suitable data sources, (2) identifies appropriate techniques for risk sentiment analysis, and (3) analyzes risk sentiment within the last decade in order to cover the financial crisis of 2007-08 adequately. The derived sentiment scores quantify uncertainty, negativity, and positivity in the analyzed documents. All of them are interesting with regards to risk sentiment analysis: uncertainty relates to risk in a direct way since the latter are "uncertainties resulting in adverse variations of profitability or in losses" (Bessis, 2002, p. 11). Highly negative sentiment refers to current or future problems, and too positive sentiment could represent overconfidence. We find that sentiment scores reflect not only the financial crisis, but also other major economic events within the last decade.

In addition, we test for correlations between the sentiment scores and a popular quantitative risk indicator. It turns out that aggregated risk sentiment in forward-looking documents is a leading indicator for the actual risk figures, so it can be used within predictive models.

The remainder of this paper, which is based on the Master's thesis of one of the authors (Nopp, 2015), is organized as follows: first, we give an overview on related work in the field of risk sentiment analysis. The following section introduces the chosen sources for text data and quantitative figures. Afterwards, we give an overview on the chosen methodologies and evaluate the experimental results. The last section concludes.

2 Related Work

Sentiment analysis in general and its application in the financial domain in particular gained a lot of interest within the last decade. There is a number of studies which aim to identify risks by means of text mining. A common question tackled by researchers is whether corporate disclosures drive stock price volatilities or future earnings of the respective firm (Groth and Muntermann, 2011; Kogan et al., 2009; Tsai and Wang, 2013). Hence, they focus on the risk an *investor* takes if he or she buys stocks of a company. Generally spoken, these studies find significant correlations between sentiment extracted from corporate disclosures and future volatilities. Other papers deal with *financial distress prediction*, for example Hajek and Olej (2013). As a baseline, they classify companies based on financial indicators. It turned out that the inclusion of sentiment indicators improved financial distress prediction.

Among the text data sources for these studies are mainly annual reports, but also news stories or earning calls transcripts[1]. Kogan et al. (2009) exclude irrelevant information from the annual reports by focusing on a section which contains important forward-looking content.

In the related studies, authors work with similar approaches for extracting sentiment from texts. Linguistic preprocessing generally involves tokenization, lemmatization, and removing non-essential items like tables, exhibits, or digit sequences. In almost every study, the authors also make use of term weighting schemes. With the selected features and additional quantitative data, the studies either employ machine learning algorithms, or use the data for regression analyses.

Although none of the mentioned papers focuses on risk sentiment analysis in the banking industry, parts of their processing pipelines and approaches can be reused for this work. With regards to the selection of appropriate data sources, it can be concluded that analyzing annual reports is very popular in this field of research. Hence, these data should also be considered for the experiments of this study. In contrast to the majority of the related papers, we only use specific sections of the annual reports, namely CEO letters and outlook sections (see Section 3).

Regarding the machine learning algorithms and the incorporation of quantitative indicators, the approaches of Groth and Muntermann (2011), Kogan et al. (2009), and Hajek and Olej (2013) are a good basis for the experiments of this study. All of them define the document labels based on suitable quantitative indicators. For labeling, the related studies consider the fact that text data are forward-looking, but quantitative indicators reflect the past. Hence, the indicators are taken from one period after publication of the text data. The labeled data are then used for training machine learning algorithms. Since the focus of our work lies on banks, we make use of a specific quantitative risk indicator which is not employed by related studies. In the following section, we introduce this indicator and the selected text data sources.

[1] Earning calls are regular events where managers report about the company's current situation and answer questions from business analysts.

3 Data Sources

Among this work's aims is to test for relations between textual risk sentiment and quantitative risk indicators. A careful selection of sources for both types of data is crucial since irrelevant ones would lead to biased conclusions.

Quantitative Risk Indicator. The selected quantitative risk indicator has to represent financial health and the general risk exposure of a bank within a specific period or at a specific point in time. Furthermore, the data have to be (1) publicly accessible, (2) available for each analyzed bank, (3) published at least annually, and (4) comparable among the different banks.

A comparison of several quantitative risk indicators based on expert interviews revealed that only the *Tier 1 Capital Ratio* (T1) fulfills all criteria. The T1 is one of the most important ratios based on risk-weighted amount of the bank's assets. In particular, it refers to the bank's Tier 1 capital as a percentage of its risk-weighted assets:

$$\text{Tier 1 Capital Ratio} = \frac{\text{Tier 1 Capital}}{\text{Risk-Weighted Assets}} \quad (1)$$

Tier 1 capital is considered as the best form of bank capital and has to fulfill several criteria making it relatively secure. As Cannata et al. (2012, p. 12) put it, this ratio "measures the ability of the bank to absorb losses". If the T1 is high, the bank acts conservatively and with a high risk buffer. A high ratio can be achieved by either increasing the Tier 1 capital or by reducing the amount of risk-weighted assets, i.e. reducing the amount of total assets or replacing them with safer ones.

The T1 also played a major role during the 2014 EU-wide banking stress test, which was an important part of the preparation phase for the Single Supervisory Mechanism. The stress test had the purpose to assess the resilience of large EU banks in different macroeconomic scenarios, measured by the impact on the T1.

Text Data Sources. In order to minimize noise and to enhance the sentiment analysis validity, it is crucial to work with the documents well adapted to the task of risk sentiment analysis. Like the quantitative risk indicators, they need to be (1) publicly accessible, (2) available for every analyzed bank, and (3) published at least annually. In addition, for this study, the documents need to be (4) written in the English language, (5) directly published by the bank, and (6) contain forward-looking and subjective information about the bank's attitude and expectations towards risk.

These criteria are best fulfilled by two types of document published in the banks' annual reports, namely *CEO letters* and *outlook sections*. The former are carefully crafted documents which contain valuable information about the management's opinions about risk. Amernic et al. (2010) recognize in their study from 2010 that the word choice of managers strongly influences companies, and CEO letters are a way for communicating their attitudes and values.

Outlook sections are usually a part of the *management report*, which is a textual summary of the bank's results, its business environment, and regulatory as well as internal developments. In their outlook on the next year, banks write about the expected macroeconomic environment, management guidelines, and priorities for the next period. These documents might be less subjective compared to CEO letters, but they are usually more comprehensive and contain interesting forward-looking information.

Collection of Data. The annual reports for this work were collected via a *Bloomberg Terminal*, supplemented by direct downloads from bank websites. In total, over 500 documents from 27 banks which published them between 2001 and 2013 were collected. The sample contains banks from all 12 countries which have belonged to the Eurozone at least since 2002. This promotes comparability of the data because the banks operated in similar economic circumstances and with the same currency.

Further data were retrieved from the online database *Bankscope*: the bank's country of residence, its full name, its size measured by total assets, and its Tier 1 capital ratio at the end of each year between 2001 and 2013. 31 % of the Tier 1 capital ratios could not be directly retrieved from the database, so they had to be manually extracted from the respective annual reports.

4 Methodologies

Two independent approaches are employed for the risk sentiment analysis. First, a lexicon-based approach derives and analyzes negativity, positivity, and uncertainty in publications by banks. The second approach aims to predict the evolution of

quantitative risk indicators by means of supervised classification. The aim of both approaches is to assess the potential of risk sentiment analysis in banking supervision.

Creation of the Document Collection. The original documents are provided as PDF files. For building up the collection, they have to be parsed in order to acquire plain text files containing the required sections. One method for extracting the relevant sections is to split the original PDF files according to their bookmarks and to convert them into plain text files afterwards. Another way is to convert the PDF files already in the first step and to extract the relevant sections by making use of specific *tokens*. For example, a typical CEO letter is delimited by the tokens *Dear Shareholders* and *Sincerely*. If neither of these semi-automated approaches is applicable, the extraction has to be done manually[2].

Table 4 gives an overview of the number of documents in the created collection. It shows that the number of published outlook sections constantly increased between 2002 and 2008. From 2009 on, the number was quite stable. The number of CEO letters also increased over time, but only until 2008, when some CEOs stopped writing letters in the course of the financial crisis.

Year	# of CEO letters	# of outlooks
2002	15	14
2003	19	19
2004	17	20
2005	19	20
2006	19	21
2007	23	22
2008	25	23
2009	21	23
2010	20	23
2011	21	23
2012	20	23
2013	22	24
2014	22	23
Total	**263**	**278**

Table 1: An overview of the document collection.

4.1 Lexicon-based Approach

The first experiment is about analyzing sentiment scores derived from the documents by incorporating finance-specific word lists. The objective of this experiment is to show how the language of forward-looking disclosures by European banks evolved within the last decade. The workflow consists of the following steps: (1) pre-processing the collected data, (2) the actual sentiment analysis which derives the scores, (3) data consolidation, and (4) data evaluation.

Sentiment Tagging. In the first step of the analysis, sentiment words in the textual data are tagged. In particular, this study works with negative words (*Fin-Neg*), positive words (*Fin-Pos*), and words related to uncertainty (*Fin-Unc*). All of these word lists are provided by Loughran and McDonald (2011). Such topic-specific word lists are necessary because many words bear a different sentiment if used in a financial context: according to Loughran and McDonald (2011), almost three quarters (73.8 %) of typically negative words cannot be considered as negative when they appear in financial texts. Kearney and Liu (2014) give the examples *tax* and *liability*. These words appear in the *Harvard IV Negative Word List* (H4N), but are neutral when used in a financial context, e.g. in an annual report. Table 2 lists some examples for sentiment words in the financial context.

Positive	efficient, stabilized, vibrant
Negative	closure, postpone, threat
Uncertainty	approximately, might, volatility

Table 2: Examples for opinion words in the financial context (Loughran and McDonald, 2011).

Term Weighting. All terms in a document are normalized by

$$N_j = \frac{1}{\sqrt{\sum_{i=0}^{m}(G_i L_{i,j})^2}}. \qquad (2)$$

This equation is based on Salton and Buckley (1988) and accounts for documents of different lengths. G_i is the *global weight* of term i and $L_{i,j}$ the *local weight* of term i in document j. An established method for the latter is given by the following formula (Manning and Schütze, 1999, p. 543):

$$L_{i,j} = \begin{cases} 1 + \log(tf_{i,j}) & \text{if } tf_{i,j} \geq 1 \\ 0 & \text{otherwise} \end{cases} \qquad (3)$$

[2]This was the case for around 20 % of the documents.

The term frequency is denoted as $tf_{i,j}$. The most popular global weight is the *inverse document frequency* (IDF). In

$$G_i = \log\left(\frac{N}{df_i}\right), \tag{4}$$

the total number of documents is denoted by N, and df_i is the number of documents where term i occurs at least once (Salton and Buckley, 1988).

Valence Shifting. In order to account for negated sentiment words, the simple negation handling algorithm proposed by Polanyi and Zaenen (2006) is implemented. If one of the three direct predecessors of a sentiment word is a negation word[3], its sentiment score will be negated. This is done by assigning -1 to the valence shifter variable v_i of term i. If there is no negation word among the predecessors, v_i is set to 1.

Calculating Sentiment Scores. The document-level sentiment scores are calculated for three sentiment classes, namely uncertainty, positivity, and negativity. In

$$s_{c,j} = \sum_{i \in c} L_{i,j} G_i N_j v_i, \tag{5}$$

the term-level sentiment score is represented by the product of the term weights $L_{i,j}$ and G_i, the normalization factor N_j, and the valence shifter v_i. The document sentiment score $s_{c,j}$ is the sum of the term sentiment scores which belong to the document j and the sentiment class c.

Data Consolidation and Evaluation. After calculating the sentiment scores, the data are filtered and grouped in order to prepare them for the evaluations. In particular, the data are filtered according to specific countries and grouped by year respectively by bank.

4.2 Supervised Classification

For the second experiment, the documents are labeled based on a quantitative risk measure, namely the T1 dating to the end of the period referred to in the CEO letters and outlook sections. These data are then used for training supervised classification algorithms which aim to predict the indicator's evolution.

[3]The considered negation words are *no*, *not*, *don't*, *never*, *none*, and *neither*.

The experiment consists of three steps: (1) reading and parsing the collected data as well as assigning the class labels, (2) linguistic preprocessing and feature selection, and (3) classifying the data with *Naïve Bayes* (NB) and *Support Vector Machine* (SVM).

Assigning the Class Labels. The Tier 1 capital ratio is published by banks at least once a year. Since it is actually a continuous measure, it always strongly depends on the previous year's ratio. Banking supervisors like the ECB are interested in the future evolution of the ratio: if it increases, the bank acts in a less risky way, and vice versa. Hence, appropriate labels for the supervised classification task are *UP* for an increasing T1, and *DOWN* for a decreasing one. We assume that the T1 did not change notably if the difference to the previous year is less than 0.2 percent points[4]. In this case, no class label is assigned.

Preprocessing and Feature Selection. Linguistic preprocessing comprises the removal of punctuation, numbers, single characters, and stop words. The remaining words are converted to lower case. Furthermore, the terms are weighted according to the term weighting strategy presented in Section 4.1.

For feature selection, two approaches are followed. The first one assumes that the sentiment words used in the lexicon-based analysis are the relevant features for this experiment. Hence, all words which do not appear in the first experiment's dictionaries are removed. The second approach utilizes a *Snowball Stemmer* to ensure that different versions of the same word are treated as equal. Its feature selection strategy is based on the concepts of *document frequency* (DF) and *information gain* (IG). For DF, tests showed that a lower bound of 20 documents yields the best results. The objective of the IG measure is to identify those features which have the highest discriminatory power in a classification problem. It measures the impurity of a dataset, i.e. its *entropy*. If a feature is able to reduce the entropy in a data set by a large amount, its *information gain* is high. Such features have a relatively high ability to predict the corresponding class. For calculating the information gain, one has to compute the entropies given the presence or absence of a feature in a data set and subtract the results from the entropy

[4]This affected 17 % of the data points in the sample.

of the original data set (Aggarwal and Zhai, 2012, p. 169).

Classification. The outcome of the previous steps is a set of document vectors with associated class labels. With these data, the classification algorithms *Naïve Bayes* (NB) and *Support Vector Machine* (SVM) are trained. The latter is used in its basic version, i.e. with a linear kernel. The performance measures are determined by employing *10-fold cross validation*, which helps to avoid problems like overfitting.

While Naïve Bayes works without parameters, the linear SVM depends on the parameter C. Its optimal value of 111 was determined by conducting an automated *grid search*.

5 Evaluation of the Experiments

Both experiments aim to capture attitudes and opinions about risk by analyzing CEO letters and outlook sections of Eurozone banks. In this section, conclusions are drawn from the results of the experiments.

5.1 Evaluation of the Lexicon-based Approach

The outcome of the lexicon-based approach consists of sentiment scores for each document representing the degrees of uncertainty, negativity, and positivity.

Evolution of Sentiment Over Time. Figure 1 shows how sentiment in CEO letters has been evolving since 2002. The evolution of sentiment in outlook sections is not depicted, but is very similar to that of CEO letters. The individual data points represent the arithmetic mean of the document-level sentiment scores for each year. In 2002 and 2003, CEO letters contained more negative sentiment than in the following years. Banks might have emphasized that the recession following the burst of the *dot-com bubble* was still not over and that recovery had not yet arrived. Between 2003 and 2006, the letters became more positive and less negative from year to year. The turning point was in 2006—from that time on, negativity in CEO letters rose and quadrupled within three years. During the same period, positive sentiment scores decreased continuously. The summit of these evolutions was in 2009, in the midst of the financial crisis. The letters in 2010 had been already much more optimistic, but negativity increased in 2011 and 2012 again when CEOs recognized that the crisis was still not over.

The evolution of the uncertainty scores is similar to the negative sentiment scores. This observation is supported by a high correlation coefficient of 0.93 between uncertainty and negativity scores. Since 2012, uncertainty has been decreasing quite sharply. This can potentially be attributed to an important and often-cited speech by ECB president Mario Draghi, who calmed the financial markets with the announcement to do "whatever it takes to preserve the Euro. And believe me, it will be enough"[5].

Another observation is that the average uncertainty scores are much lower than the average positivity and negativity scores. A plausible interpretation thereof is that CEOs rather use clear statements than uncertain language.

Do Sentiment Scores Predict Quantitative Risk Measures? A comparison of the T1 average evolution and the corresponding sentiment scores reveals interesting relations, see Figure 2. The correlation coefficients in Table 3 indicate that a higher degree of uncertainty or negativity in the documents is commonly followed by a higher increase of the T1, and vice versa.

It is interesting to analyze the data by a regression model for predicting the T1 evolution. Table 4 shows such a model with *negativity* as the only explaining variable. The coefficients can be interpreted as follows: if the average negativity score rises by one unit, the T1 evolution increases by 0.9963 pp. If negativity is zero, the Tier 1 capital ratio would decrease by the computed intercept, which is -0.502 pp. Both coefficients are statistically significant if a 95 % confidence level is assumed.

About 76 % of the average T1 evolution's *variation* can be explained by the negativity score. A model of similar quality could be constructed by analyzing uncertainty in outlook sections. Hence, sentiment scores can be considered as an additional leading indicator for the future evolution of the Tier 1 capital ratio.

Limitations. A drawback of this regression model is that it cannot model *external shocks* which influence the T1 evolution, but are not ad-

[5] A transcript of this speech is available at `https://www.ecb.europa.eu/press/key/date/2012/html/sp120726.en.html`, accessed April 20th, 2015.

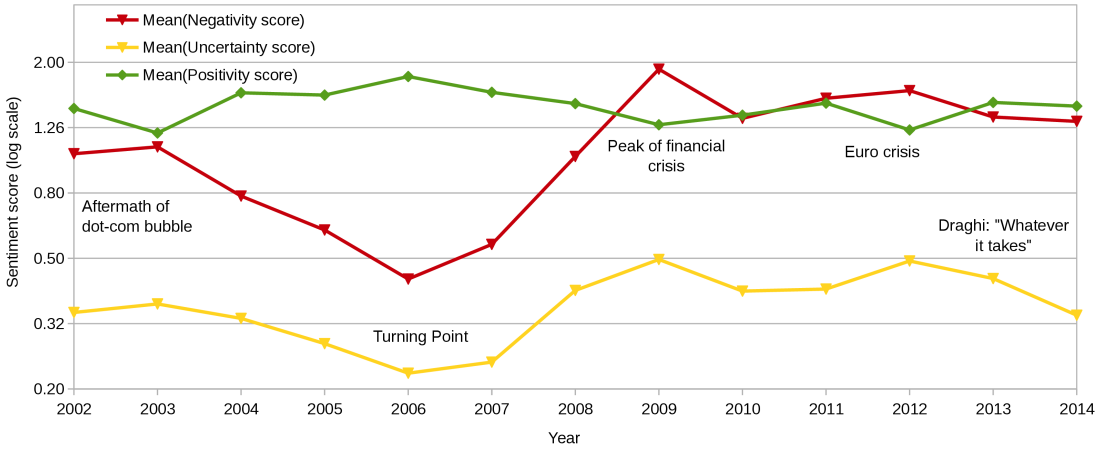


Figure 1: Evolution of positivity, negativity, and uncertainty in CEO letters over time.

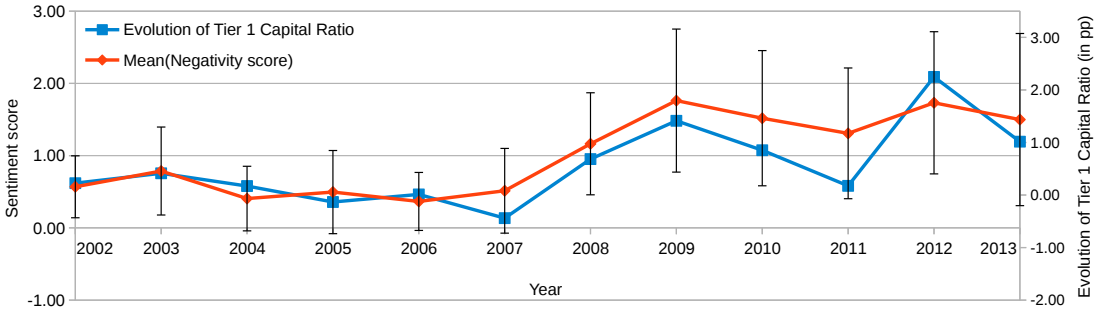


Figure 2: Evolution of the Tier 1 capital ratio compared to negativity in outlook sections. The error bars represent the standard deviation of the negativity scores.

Correlation coefficient	Uncertainty	Negativity	Positivity
T1 evolution (CEO letters)	0.86	0.79	-0.69
T1 evolution (Outlooks)	0.85	0.89	0.12

Table 3: Correlation coefficients between T1 evolution and sentiment scores.

Variable	Coeff.	Std. Err.	t-value	P>t
Mean(Negativity score)	0.9963	0.1647	6.0478	0.0001
Intercept	-0.5020	0.1883	-2.6651	0.0237

Table 4: Regression model based on negativity in outlook sections.

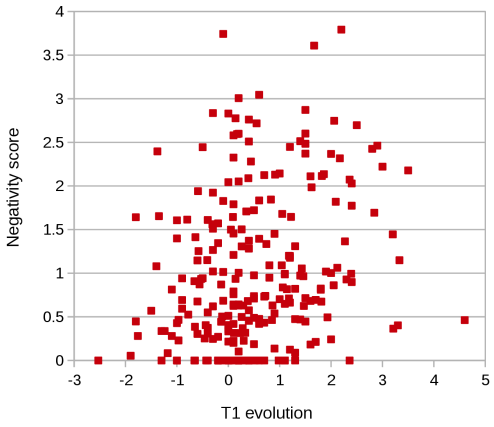


Figure 3: Individual negativity scores in outlook sections compared to the T1 evolution.

equately covered in the text data. Examples for such shocks would be new regulations concerning the minimum capital ratio or monetary policy actions by the ECB. A further limitation is induced by the fact that our methodology makes use of the *bag of words* (BoW) model, which ignores the documents internal structure. Hence, it is not possible to utilize information like word order and grammar, although this definitely plays a role in carefully crafted documents like CEO letters.

It should also be emphasized that the model is based on figures aggregated by year. Applying it on the data of *individual* banks could lead to incorrect conclusions. This assumption is supported by Figure 3, which compares negativity scores of individual outlook sections with the associated T1 evolutions. Although it is still possible to identify a positive relationship between the variables, the variance is too big for satisfactory representation by a regression model[6]. This observation is in line with the relatively high standard deviations if the figures are aggregated by year, see Figure 2.

5.2 Evaluation of the Supervised Classification Approach

The supervised classification experiment aims to assess whether this approach works better than the lexicon-based approach in terms of predicting the T1 evolution for *individual* banks based on their CEO letters or outlook sections. The class labels *UP* and *DOWN* have been assigned according to the direction of the T1 evolution. Table 5 gives an overview of the experiments and lists the respective performance measures. An analysis of the data in the table reveals interesting results. First, feature selection based on document frequency and information gain works better than the approach based on word lists. Second, the classifiers trained with CEO letters yield better results than the ones trained with outlook sections. Finally, three out of the four SVM results are not meaningful due to the following reason: the parameter optimization of C suggests to choose a very low value, which indeed maximizes the classifier accuracy—but these SVMs simply assign the class *UP* to every instance. These classifiers can be seen as a baseline for comparisons. However, the remaining SVM clearly yields the best results among the employed algorithms.

None of the classifiers based on the feature selection method (1) is able to outperform the baseline (assigning every instance to the UP class). Both SVMs simply classify every instance as *UP*, and the Naïve Bayes classifiers also deliver unsatisfactory results. Feature selection based on document frequency and information gain achieves better results than the first one, but only when the classifiers are trained with the CEO letter collection. Most likely, this can be explained with the fact that outlook sections provide less terms with discriminatory power than CEO letters. Naïve Bayes correctly classifies 75 % of the instances, and the optimized SVM yields 79.2 %. The other SVM performance measures can be interpreted as follows: 81 % of the instances classified as *UP* were indeed instances where the Tier 1 capital ratio increased (= precision U). Furthermore, the SVM correctly identified almost 92 % of the instances which belong to the class *UP* (= recall U).

These results are better than the baseline and demonstrate a noticeable potential for supervised classification even at the level of individual bank disclosures. Nevertheless, they are not good enough for reliable predictions. However, the aggregated classification data accurately predict whether the majority of banks will increase or decrease their Tier 1 capital ratio in the following year: for 12 out of 13 years, the algorithm correctly predicts the direction of the T1 evolution. This finding is in line with the lexicon-based approach, where the aggregated data yielded much better results than the individual ones.

[6] If the regression model is built with non-aggregated data, it explains only 6.6 % of the variation.

Feature selection	Document type	Classifier	Accuracy	Precision U	Recall U	Precision D	Recall D
(1) based on topic-specific sentiment words	CEO letters	NB	0.703	0.741	0.889	0.500	0.263
		SVM	0.703	0.703	**1.000**	n.a.	0.000
	Outlook sections	NB	0.563	0.685	0.703	0.246	0.230
		SVM	0.703	0.703	**1.000**	n.a.	0.000
(2) based on document frequency and information gain	CEO letters	NB	0.750	0.774	0.911	0.636	0.368
		SVM	**0.792**	**0.810**	0.919	**0.718**	**0.491**
	Outlook sections	NB	0.704	0.704	**1.000**	n.a.	0.000
		SVM	0.704	0.704	**1.000**	n.a.	0.000

Table 5: Overview of the results of the supervised classification experiment. Bold numbers indicate the best results, U class *UP*, and D class *DOWN*.

6 Conclusion

This study explored how banking supervisors could utilize *sentiment analysis* for risk assessments. The analysis of potential document types revealed that two sections in a bank's annual report are particularly well suited for this work, namely *CEO letters* and *outlook sections*. The former represent the *tone from the top* and provide subjective information about the bank's current and future situation. Outlook sections are exclusively forward-looking and reveal opinions about the near future. Furthermore, the *Tier 1 capital ratio* (T1) is the best suited quantitative risk indicator. The T1 sets the most secure forms of bank capital in relation to its risk-weighted assets and is widely used in banking supervision, e.g. as a key ratio for the ECB's stress test in fall 2014.

The lexicon-based analysis showed that sentiment scores reflect major economic events between 2002 and 2014 very well. In addition, there is a strong correlation between uncertainty, negativity, and the Tier 1 capital ratio evolution over time. Hence, the sentiment scores could be used in regression models for predicting the T1 evolution. However, the results are only meaningful if the figures are aggregated by year. Applying the model on data of individual banks leads to inaccurate results. It should also be noted that this method is not meant to be used as a stand-alone estimator for the T1 evolution. Instead, it should be combined with other estimation methods.

The supervised risk classification approach correctly classifies 79.2 % of the CEO letters. This is not good if one considers that it is possible to yield an accuracy of 70 % simply by assigning the class *UP* to every instance. However, if the results of the best SVM classifier are aggregated by year, the data correctly predict for 12 out of 13 years whether the majority of banks will increase or decrease their Tier 1 capital ratio.

The described systems have the potential to provide valuable insights for banking supervisors, in particular because of the strong correlation between sentiment scores derived from textual data and the T1. Because of the mentioned limitations, these techniques should only be used for macroprudential analyses, i.e. the promotion of stability in the whole financial system. Examples are predictions for the average Tier 1 capital ratio's evolution in the whole Eurozone or in groups of countries. Another option is to improve existing risk prediction frameworks.

For future research, it would be interesting to validate the results by conducting the study on a larger scale. One could incorporate data from all European banks, or from other regions. The approach could also be used for other document types, for example analyst reports or internal memos, or in other industries. Regarding the methodology, it would be interesting to see how alternative algorithms or word lists would affect the results.

References

Charu C. Aggarwal and Cheng Xiang Zhai. 2012. A Survey of Text Classification Algorithms. In Charu C. Aggarwal and Cheng Xiang Zhai, editors, *Mining Text Data*, pages 163–222. Springer Science+Business Media.

Joel Amernic, Russel Craig, and Dennis Tourish. 2010. *Measuring and Assessing Tone at the Top Using Annual Report CEO Letters*. Institute of Chartered Accountants of Scotland, Edinburgh. Retrieved March 28th, 2014, from `http://eprints.port.ac.uk/12648/1/CRAIG_2010_pub_Bk_Measuring_and_assessing_tone_at_the_top_using_annual_report_CEO_letters.pdf`.

Joël Bessis. 2002. *Risk Management in Banking*. John Wiley & Sons Ltd, Chichester, United Kingdom, 2nd edition.

Francesco Cannata, Simone Casellina, and Gregorio Guidi. 2012. *Inside the labyrinth of Basel risk-weighted assets: how not to get lost*. Number 132 in Questioni di Economia e Finanza. Banca d'Italia. Retrieved September 20th, 2014, from http://www.bancaditalia.it/pubblicazioni/qef/2012-0132/QEF_132.pdf.

Council of the EU. 2013. Council Regulation (EU) 1024/2013 of 15 October 2013 conferring specific tasks on the European Central Bank concerning policies relating to the prudential supervision of credit institutions. *Official Journal of the European Union*, L287:63–89.

European Banking Authority. 2014. Risk Assessment of the European Banking System - June 2014, 6. Retrieved July 27th, 2014, from https://www.eba.europa.eu/documents/10180/556730/EBA+Risk+Assessment+Report+June+2014.pdf/.

Sven S. Groth and Jan Muntermann. 2011. An intraday market risk management approach based on textual analysis. *Decision Support Systems*, 50(4):680–691.

Petr Hájek and Vladimír Olej. 2013. Evaluating Sentiment in Annual Reports for Financial Distress Prediction Using Neural Networks and Support Vector Machines. In Lazaros Iliadis, Harris Papadopoulos, and Chrisina Jayne, editors, *Engineering Applications of Neural Networks*, volume 384 of *Communications in Computer and Information Science*, pages 1–10. Springer Berlin Heidelberg.

Dermot Hodson and Lucia Quaglia. 2009. European Perspectives on the Global Financial Crisis: Introduction. *Journal of Common Market Studies*, 47(5):939–953.

Colm Kearney and Sha Liu. 2014. Textual sentiment in finance: A survey of methods and models. *International Review of Financial Analysis*, 33(2014):171–185. Retrieved May 26th, 2014, from http://papers.ssrn.com/sol3/papers.cfm?abstract_id=2213801.

Shimon Kogan, Dimitry Levin, Bryan R. Routledge, Jacob S. Sagi, and Noah A. Smith. 2009. Predicting Risk from Financial Reports with Regression. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 272–280. Association for Computational Linguistics.

Tim Loughran and Bill McDonald. 2011. When Is a Liability Not a Liability? Textual Analysis, Dictionaries, and 10-Ks. *Journal of Finance*, 66(1):35–65, 02.

Christopher D. Manning and Hinrich Schütze. 1999. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, MA, USA.

Clemens Nopp. 2015. Risk Sentiment Analysis in Banking Supervision. Master's thesis, Vienna University of Technology.

Livia Polanyi and Annie Zaenen. 2006. Contextual valence shifters. In JamesG. Shanahan, Yan Qu, and Janyce Wiebe, editors, *Computing Attitude and Affect in Text: Theory and Applications*, volume 20 of *The Information Retrieval Series*, pages 1–10. Springer Netherlands.

Gerard Salton and Christopher Buckley. 1988. Term-weighting Approaches in Automatic Text Retrieval. *Information Processing and Management*, 24(5):513–523, August.

Ming-Feng Tsai and Chuan-Ju Wang. 2013. Risk Ranking from Financial Reports. In *Proceedings of the 35th European Conference on Advances in Information Retrieval*, pages 804–807. Springer-Verlag.

Sentiment Flow – A General Model of Web Review Argumentation

Henning Wachsmuth and **Johannes Kiesel** and **Benno Stein**
Faculty of Media, Bauhaus-Universität Weimar, Germany
{henning.wachsmuth,johannes.kiesel,benno.stein}@uni-weimar.de

Abstract

Web reviews have been intensively studied in argumentation-related tasks such as sentiment analysis. However, due to their focus on content-based features, many sentiment analysis approaches are effective only for reviews from those domains they have been specifically modeled for. This paper puts its focus on domain independence and asks whether a general model can be found for how people argue in web reviews. Our hypothesis is that people express their global sentiment on a topic with similar sequences of local sentiment independent of the domain. We model such *sentiment flow* robustly under uncertainty through abstraction. To test our hypothesis, we predict global sentiment based on sentiment flow. In systematic experiments, we improve over the domain independence of strong baselines. Our findings suggest that sentiment flow qualifies as a general model of web review argumentation.

1 Introduction

The web is full of user-generated reviews on products, services, and works of art, like those from Amazon, TripAdvisor, and Rotten Tomatoes. Such web reviews provide facts, positive opinions, and negative opinions on different aspects. By that, the reviews express, implicitly or explicitly, an overall opinion on the topic in question. From an abstract viewpoint, the argumentation of a web review can thus be seen as a composition of local sentiments used to justify some global sentiment.

Both local and global sentiment of reviews are in the focus of numerous sentiment analysis approaches (cf. Section 2 for details). Many of these approaches model reviews primarily with content-based features, derived from the words in the reviews. The use of words, however, varies strongly across domains, as illustrated in Figure 1 for a product, a hotel, and a movie review. As a consequence, sentiment analysis suffers from domain dependence (Wu et al., 2010), i.e., high effectiveness is often achieved only in the domain an approach has been specifically modeled for. To adapt to other domains, prior knowledge about these domains or about domain-independent features is needed (Prettenhofer and Stein, 2010).

This paper considers the question as to whether the overall argumentation of web reviews can be modeled in a general way in order to increase domain independence in sentiment analysis. We observe that people structure web reviews largely sequentially—in contrast to the complex structures of many other argumentative texts. While the reviewed aspects differ between domains, our assumption is that the overall argumentation of a web review is generally represented by a sequence of local sentiments, called the review's *sentiment flow* (Mao and Lebanon, 2007). In particular, we hypothesize that, under an adequate model, similar sentiment flows express similar global sentiments, also across domains. All reviews in Figure 1, for instance, express neutral global sentiment starting with positive, continuing with negative, and ending with positive local sentiment.

Unlike in our previous approach (Wachsmuth et al., 2014a), we analyze the major abstraction steps when modeling sentiment flow to represent global sentiment. A general model should abstract from both content and other domain differences, such as a review's length or the density of local sentiment in it. Based on web review corpora with known sentiment flows, we empirically analyze several model variants across three domains. Our results offer clear evidence for the truth of our hypothesis, indicating the generality of sentiment flow as a model of web review argumentation.

The abstract nature of sentiment flow, however, does not directly achieve domain independence, as

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 601–611,
Lisbon, Portugal, 17-21 September 2015.

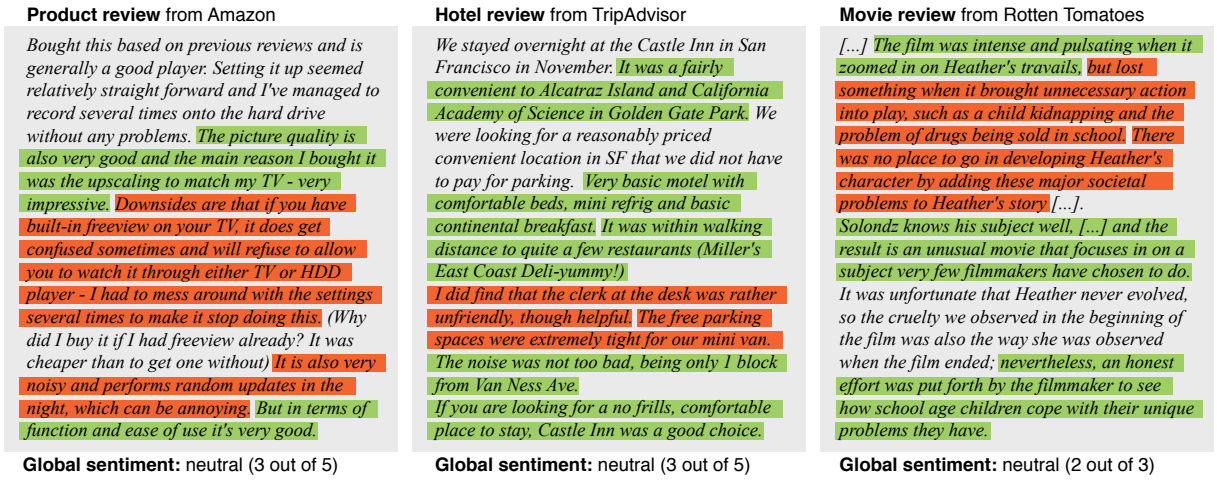


Figure 1. Example web reviews with neutral global sentiment from three domains, taken from the corpora described in Section 5. Corpus annotations of positive and negative local sentiment are marked in light green and medium red, respectively.

the recognition of local sentiment in unknown reviews may still be domain-dependent. We therefore also present a novel edit distance approach to robustly compare flows, when local sentiment is obtained using state-of-the-art techniques (Socher et al., 2013). In systematic cross-domain experiments with the given corpora, we classify global sentiment based on sentiment flow without any domain adaptation. While not being perfectly effective, our approach improves over the domain robustness of strong baselines.

Altogether, the paper's main contributions are:

1. Evidence that sentiment flow qualifies as a general model of the overall argumentation of web reviews across domains.
2. A domain-robust approach for the classification of the global sentiment of web reviews.

2 Related Work

As surveyed by Pang and Lee (2008) and by Liu (2012), numerous sentiment analysis approaches have been proposed for different text types, levels of granularity, sentiment scales, and domains. We target at global text-level sentiment of web reviews. While we distinguish three sentiment classes here, our approach can be adapted to other scales. Our goal is not to optimize sentiment analysis in a specific domain, but to find a model that supports sentiment analysis across domains.

As common in text classification (Manning et al., 2008), sentiment analysis often relies on words and other content features, which tends to be prone to domain dependence (Wu et al., 2010). Existing domain adaptation techniques for sentiment analysis require a few training texts from each target domain or a few domain-independent pivot features to align domain-specific features (Prettenhofer and Stein, 2010). Our model complements these techniques and could be leveraged for pivot features. In tasks like authorship attribution and argumentative zoning, non-topical words benefit domain independence (Menon and Choi, 2011; Ó Séaghdha and Teufel, 2014). Instead, we focus on the local sentiment on different aspects in a review here.

Aspect-based sentiment analysis extracts fine-grained opinions from a review (Popescu and Etzioni, 2005). These aspects in turn impact the review's global sentiment (Wang et al., 2010). However, relevant aspects naturally tend to be domain-specific, like the picture quality of HDD players or the beds of hotels (cf. Figure 1). While weakly-supervised approaches to extract aspects and local sentiment exist (Brody and Elhadad, 2010; Lazaridou et al., 2013), it is not clear how to align aspects from different domains. We ignore aspects here, only preserving the local sentiment itself.

State-of-the-art approaches for classifying local sentiment within a domain model the composition of words, e.g., relying on deep learning (Socher et al., 2013). We do not compete with such an approach, but we use it to then predict global sentiment. Täckström and McDonald (2011) observe that local and global sentiment correlate, aiming for the opposite direction, though. In (Wachsmuth et al., 2014b), we already compute frequent flows of local sentiment, but we neither analyze their generality, nor do we use them for prediction.

The idea of modeling sentiment flow was introduced by Mao and Lebanon (2007) who classify local sentiment based on neighboring local sentiment in a review. When inferring global sentiment from a flow, however, the authors model only

single flow positions, not their ordering. In contrast, we capture the overall structure of reviews in (Wachsmuth et al., 2014a) by measuring the similarity of a given flow to known sentiment flow patterns. We point out the domain robustness of sentiment flow there, but we still use domain-specific local sentiment classifiers and we do not handle some major domain differences of web reviews. Both limitations are addressed in this paper, where we align flows similar to how Persing et al. (2010) align essay organizations.

We claim that sentiment flow models a review's argumentation, such that local sentiments ressemble rhetorical moves. Comparable simplifications are common for scientific argumentation (Teufel, 2014). Usually, argumentative texts are studied more deeply, considering different types of argument components and their relations (Mochales and Moens, 2011). Mining such structure is getting increasing attention recently (Habernal et al., 2014), also in the analysis of reviews (Villalba and Saint-Dizier, 2012). To express global sentiment, however, web reviews argue in simpler ways.

3 Web Review Argumentation

Argumentation refers to the exchange of opinions, to defending positions, and to convincing others of certain stances (van Eemeren et al., 2014). A review is a written form of monological argumentation, where an author structures a selection of arguments in order to justify his or her conclusion on a topic of discussion (Besnard and Hunter, 2008). Reviews, in particular, discuss products, services, works of art, or similar. The arguments in a review correspond to objective facts, positive and negative opinions, and mixtures of these on the topic as a whole or on specific aspects of the topic.

In this paper, we are interested in the *overall argumentation* of reviews. Our assumption is that the conclusion of a review's overall argumentation consists in its global sentiment. Global sentiment is often explicitly reflected by an assigned overall rating, at least for *web reviews*.

Many web reviews are written by people in an ad-hoc fashion to quickly share opinions. As a result, unlike other argumentative texts, web reviews often remain with a sequential structure (Villalba and Saint-Dizier, 2012) and miss explicit relations between the shared opinions. E.g., while the product review and the hotel review in Figure 1 cover opinions on several aspects, no deliberate structure is found in their argumentation. However, the excerpt of the more professional movie review shows that this is not always the case.

3.1 Domain Differences

In Figure 1, we categorize domains by source and topical theme (e.g., Amazon products). Other granularities would be possible (e.g., consumer electronics) or other categorization schemes (e.g., user vs. pro reviews). That being said, we speak of *domains* only to roughly distinguish web reviews that vary in how they argue for a conclusion.[1] We observe major differences in three broad respects:

Content Especially topical web review domains differ widely regarding the terms and phrases that play a role in their argumentation. This includes the aspects being discussed (e.g., "beds" of hotels vs. the "subject" of movies) as well as the words used to express sentiment and their explicitness (e.g., "yummy" vs. "an unusual movie").

Form As sketched above, further differences refer to the structure and style of web reviews. Some are rather subtle, like a careful use of paragraph breaks, whereas others are obvious, like a review's length. The movie review in Figure 1, for instance, is actually over twice as long as the shown excerpt, starting with an objective synopsis of the plot and including "sub-reviews" of different aspects.[2]

Subjectivity Finally, the use of subjectivity varies across web review domains: First, the density of sentiment tends to be high in some cases, like hotel reviews (cf. Figure 1), but low in others, like movie reviews, where objective plot descriptions and subjective opinions often alternate (the shortened excerpt in Figure 1 hides this to some extent). Second, sentiment is sometimes very intense (as in the product review in Figure 1), sometimes subtle. And third, in some domains even single sentences often contain mixed sentiment, whereas in others opinions tend to be laid out across sentences.

We will empirically underpin most observations in Section 5, where we analyze the domain independence of the model described next.

4 Sentiment Flow as a General Model

In the following, we introduce our model of web review argumentation. We discuss how to abstract for generality and how to deal with uncertainty.

[1] In the end, this paper seeks for findings that generalize from domains, making an exact distinction unnecessary.

[2] Also, many web reviews have explicit structure elements like a title. To obtain a common ground, however, we consider only the plain text of a web review's body in this paper.

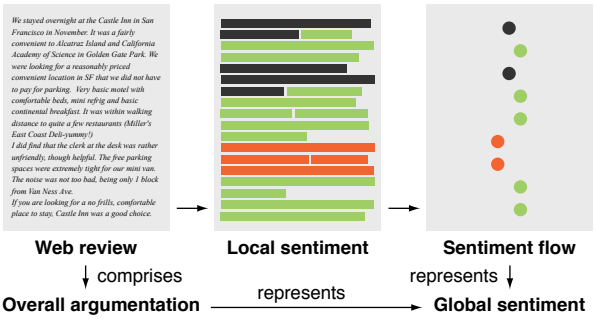


Figure 2. Modeling the overall argumentation of a web review as a flow of positive (light green), neutral (dark gray), and negative (medium red) local sentiments.

Figure 3. The original sentiment flow from Figure 2 and the resulting flow for each of the three proposed transformations.

4.1 Modeling a Review by its Sentiment Flow

We propose a fairly simple argumentation model based on the observation that many web reviews are organized sequentially (cf. Section 3). As we assume that the overall argumentation of a web review represents global sentiment in the first place, we fully abstract from the content of the facts and opinions that serve as arguments. In particular, we model the argumentation of a web review solely by its *sentiment flow*, i.e., the sequence of local sentiments comprised in the review's text. We do not presume the granularity of local sentiment, but we propose to distinguish positive, neutral, and negative local sentiment, which is the common ground of related works (cf. Section 2).

Figure 2 illustrates how we model web review argumentation. Our hypothesis is that similar sentiment flows are used across domains of web reviews to express the same global sentiment. However, because of the domain differences described in Section 3, we do not expect that the *original* sentiment flows of web reviews generalize well.

4.2 Abstracting Flows for Generality

By concept, sentiment flow avoids to capture content and some facets of form like paragraph usage. To abstract from the length of reviews, Mao and Lebanon (2007) and our approach in (Wachsmuth et al., 2014a) length-normalize sentiment flow via interpolation. While this may preserve all information, it does not account for sub-reviews and the density of subjectivity. Here, we investigate more informed ways of abstracting flows. In particular, we consider three transformations of flows:

Change Deletion of repeating local sentiments. The rationale is to reduce subjectivity differences by focusing on changes of local sentiment.

NoLoops Deletion of repeating sequences of two or more local sentiments. The rationale is to reduce length differences by merging similar sub-reviews.

2Class Deletion of neutral local sentiments. The rationale is to reduce length and subjectivity differences emanating from objective descriptions.

Figure 3 exemplifies the three transformations. They are not commutative, as can partly be seen for the example. In Section 5, we test what combinations of transformations lead to an adequate sentiment flow model. While more transformations will benefit generality, the lost specificity may decrease the correlation with global sentiment.

4.3 Analyzing Flows under Uncertainty

Given an adequate sentiment flow model, we seek to find out to what extent it enables domain-robust sentiment analysis. This brings up two challenges related to uncertainty: (1) The classification of local sentiment in unknown reviews will not be free of errors, and, (2) reviews may comprise flows for which the global sentiment is unknown.

Classification errors are naturally problematic for modeling sentiment flow. At least, some errors are bypassed by the three transformations. E.g., if one negative local sentiment in the original flow in Figure 3 is misclassified as positive, the *Change* transformation fixes this. If it is classified as neutral, *Change* and *2Class* together eliminate the effect. Moreover, errors can be countered by limiting the impact of single positions in a flow.

In (Wachsmuth et al., 2014a), we learn to infer global sentiment from the Manhattan distances between a sentiment flow and a set of common flows, thereby analyzing the flow as a whole. The common flows are found in a preceding clustering step. While we adopt the learning approach here, the Manhattan distances imply that flows are similar only if their changes are at similar positions.

Instead, we compare sentiment flows (modified with zero to three transformations) based on their normalized minimum *edit distance* (Cormen et al., 2009). Analog to Persing et al. (2010), we incrementally compute the edit distance using sequence alignment. To this end, we specify costs for pos-

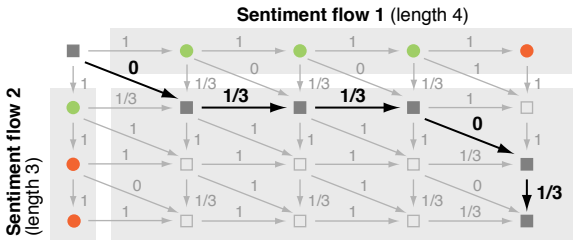


Figure 4. Computation of the normalized edit distance of two sentiment flows, resulting in $(2 \cdot 0 + 3 \cdot 1/3) \,/\, 4 = 1/4$.

sible edit operations, i.e., substitutions, insertions, and deletions of single local sentiments. We map positive local sentiment to the value 1.0, neutral to 0.5, and negative to 0.0. The cost is then provided by a function d for any two values s and s':

$$d(s, s') = \begin{cases} |s - s'| & \text{If } s' \text{ substitutes } s. \\ \alpha + (1-\alpha) \cdot |s - s'| & \text{If } s' \text{ is inserted or deleted after } s. \end{cases}$$

Here, $\alpha \in [0, 1]$ specifies some fixed cost (we set α to $1/3$ in Section 6). The intuition behind d is to have a higher cost the more s and s' differ. Still, insertions and deletions are never free, as they affect differences that remain after applying transformations to abstract from irrelevant differences.

Figure 4 illustrates the alignment of two flows as a shortest-path search. We normalize the flows' minimum edit distance by their maximum length. Before we evaluate if the edit distance captures flow similarity more robustly than the Manhattan distance, we analyze what representation of sentiment flows proves most general. This will also reveal that the proposed abstractions reduce the need to perform clustering for finding common flows.

5 Analysis of the Generality of the Model

We now report on experiments on corpora from three domains that empirically analyze to what extent different sentiment flow variants qualify as general models of web review argumentation.[3]

5.1 Ground-Truth Data with Sentiment Flow

We process three existing corpora with local sentiment annotations of complete texts. While the first two are available online, we obtained the last from the authors. Each corpus comprises English web reviews from one broad topical domain. Table 1 lists some statistics of the three corpora, which indicate clear domain differences.

Product Domain The *Finegrained Sentiment Data Set, Release 1* (Täckström and McDonald, 2011) contains 294 Amazon reviews, nearly balanced among five categories: books (59 reviews), DVD (59), electronics (57), music (59), and videogames (60). We use the first three for training and the others for testing. Under the authors' mapping from Amazon star ratings to global sentiment, all categories subsume 19 to 20 positive, neutral, and negative reviews each. In each review, every sentence is classified as positive, negative, neutral, mixed, or irrelevant. To match the other corpora, we merge the three latter into one neutral class.

Corpus domain	Sentences per text	Tokens per sent.	Local sentiment positive	neutral	negative
Product	14.0	22.7	24.1%	41.5%	34.4%
Hotel	11.5	18.3	38.0%	20.3%	41.7%
Movie	28.8	30.3	17.6%	61.2%	21.2%

Table 1. Sentences, tokens, and annotated local sentiments for the domains represented by the given web review corpora.

Hotel Domain Our *ArguAna TripAdvisor corpus* (Wachsmuth et al., 2014b) consists of 2 100 TripAdvisor reviews, 300 for seven hotel locations each. Three locations belong to a predefined training set and two to a validation and a test set each. For all locations, the reviews are evenly distributed over the five TripAdvisor overall scores. In accordance with the product corpus, we see score 4–5 as positive global sentiment, 3 as neutral, and 1–2 as negative. In each review, all main clauses together with their subordinate clauses have been classified as being positive, negative, or neutral.

Movie Domain Finally, the third corpus (Mao and Lebanon, 2007) compiles 450 Rotten Tomatoes reviews from the *Cornell Movie Review Data scale dataset v1.0* (Pang and Lee, 2005) that refer to two authors. We use the 201 reviews of Scott Renshaw for training and the 249 of Dennis Schwartz for testing. The reviews lack punctuation, capitalization, and their overall ratings. We recovered the overall ratings from the original dataset based on the rating scale 0–2, resulting in 178 positive, 139 neutral, and 133 negative reviews. In each review, Mao and Lebanon (2007) classified all sentences to be very positive, positive, neutral, negative, or very negative, which we reduce to three classes.

5.2 Experimental Set-up

To find the most general model of web review argumentation across domains, we compare 16 sentiment flow variants using three measures:

Model Variants The original sentiment flow of all corpus reviews can be directly derived from the ground-truth data. In each model variant, the flow is modified by a combination of zero to three of the

[3] The source code that can be used to reproduce the experiments is provided at http://www.arguana.com/software.

Training domain	Model variant of sentiment flows	# Flows		Aggregate recall			Weighted precision			W'd Hellinger distance		
		all	1%	Product	Hotel	Movie	Product	Hotel	Movie	Product	Hotel	Movie
Product	2class-change-noloops	7	7	100.0	100.0	100.0	75.6	69.9	62.4	0.17	0.21	0.23
175 reviews	2class-noloops-change	11	7	90.8	96.0	98.0	74.1	71.0	64.6	0.17	0.26	0.28
	change-noloops-2class	22	14	89.9	81.5	80.7	74.8	74.6	68.3	0.19	0.26	0.26
	noloops-2class-change	11	8	89.9	86.1	86.9	73.8	72.3	66.2	0.16	0.24	0.27
	2class-change	12	8	89.9	85.6	87.1	74.8	73.2	66.3	0.17	0.25	0.27
	change-2class-noloops	37	20	**85.7**	**85.8**	**76.9**	78.4	74.1	72.5	0.19	0.26	0.33
	noloops-change-2class	35	19	81.5	72.8	62.2	76.3	76.3	73.6	**0.17**	0.26	**0.28**
	2class-noloops	49	24	77.3	62.9	60.0	81.5	**80.8**	**76.7**	0.27	0.27	0.31
	change-2class	47	22	77.3	61.6	49.6	79.3	79.5	71.7	0.20	**0.24**	**0.28**
	change-noloops	55	29	70.6	64.6	59.1	**84.5**	77.2	72.2	0.25	0.30	0.32
	noloops-2class	67	24	62.2	48.2	36.7	85.1	82.7	78.8	0.18	0.25	0.22
	noloops-change	78	29	55.5	52.0	30.9	89.4	79.7	77.0	0.16	0.28	0.23
	change	93	30	49.6	43.5	27.3	89.8	82.6	75.6	0.18	0.25	0.22
	2class	91	27	45.4	36.8	28.7	83.3	85.1	79.8	0.21	0.22	0.22
	noloops	153	17	12.6	14.0	6.0	100.0	91.5	85.2	0.07	0.11	0.04
	original	173	2	0.8	3.6	0.0	100.0	94.7	0.0	0.02	0.05	0.00
Hotel	2class-change-noloops	7	7	100.0	100.0	100.0	71.1	68.5	62.4	0.19	0.06	0.13
900 reviews	2class-noloops-change	20	14	99.7	98.8	99.8	72.0	69.3	64.8	0.25	0.08	0.17
	change-noloops-2class	85	21	99.0	91.2	91.3	74.6	72.0	67.9	0.26	0.14	0.19
	noloops-2class-change	27	15	100.0	98.3	99.6	72.4	69.8	65.2	0.26	0.10	0.17
	2class-change	31	17	99.7	98.2	98.9	72.7	70.6	65.6	0.26	0.11	0.19
	change-2class-noloops	91	22	**92.9**	**91.7**	**85.3**	75.5	72.5	72.7	**0.23**	**0.14**	0.26
	noloops-change-2class	145	21	90.5	85.3	76.7	74.4	74.6	73.0	0.28	0.15	0.29
	2class-noloops	246	19	85.0	77.2	75.6	78.0	**81.0**	**75.6**	0.27	0.20	0.27
	change-2class	231	19	88.1	74.0	56.2	75.7	76.4	73.5	0.26	0.17	0.28
	change-noloops	212	24	69.7	77.7	66.7	**80.0**	75.5	73.7	0.20	0.18	0.**24**
	noloops-2class	398	14	64.6	55.3	44.0	81.6	82.5	77.8	0.19	0.15	0.26
	noloops-change	343	17	48.0	64.0	38.0	81.6	77.9	77.8	0.19	0.15	0.21
	change	426	14	36.1	52.3	24.7	83.0	77.7	77.5	0.18	0.14	0.16
	2class	549	17	54.4	32.5	29.8	83.1	86.2	77.6	0.14	0.08	0.17
	noloops	626	9	9.2	27.7	1.8	92.6	83.7	100.0	0.04	0.08	0.01
	original	743	4	1.4	16.5	0.0	75.0	78.8	0.0	0.01	0.04	0.00
Movie	2class-change-noloops	6	6	97.3	94.5	99.6	71.7	70.3	58.9	0.26	0.10	0.19
201 reviews	2class-noloops-change	14	10	96.9	93.7	99.2	72.6	70.7	59.9	0.32	0.15	0.24
	change-noloops-2class	44	17	96.6	85.4	91.6	75.4	73.9	62.7	0.34	0.22	0.32
	noloops-2class-change	16	14	96.9	92.2	98.0	73.0	71.0	59.8	0.33	0.16	0.26
	2class-change	19	15	96.9	90.8	98.0	73.3	72.1	61.1	0.33	0.18	0.28
	change-2class-noloops	57	19	**85.7**	**73.0**	**81.1**	76.2	73.4	68.8	0.36	0.25	0.30
	noloops-change-2class	84	19	82.0	63.0	72.7	77.6	72.3	71.8	0.34	0.25	0.27
	2class-noloops	103	20	78.2	58.0	62.7	78.7	**85.1**	**72.4**	0.34	0.24	0.27
	change-2class	107	20	57.5	34.4	40.6	72.8	76.2	72.3	**0.33**	0.22	**0.22**
	change-noloops	94	17	66.7	41.6	66.7	**81.6**	81.0	70.5	0.35	**0.21**	0.33
	noloops-2class	154	8	38.8	22.1	28.5	86.8	89.7	85.9	0.19	0.14	0.14
	noloops-change	146	9	30.6	19.7	30.5	87.8	83.3	77.6	0.16	0.14	0.18
	change	161	8	16.0	9.9	13.7	85.1	87.5	82.4	0.15	0.10	0.12
	2class	182	5	21.8	11.1	8.0	90.6	96.6	95.0	0.08	0.06	0.03
	noloops	200	5	0.3	0.6	0.4	100.0	91.7	100.0	0.01	0.01	0.00
	original	200	0	0.0	0.0	0.0	0.0	0.0	0.0	0.00	0.00	0.00

Table 2. Results on the generality of sentiment flow for all evaluated model variants on ground-truth data for each combination of training and test domain. The most general variants in terms of both aggregated recall and weighted precision are marked in bold. For illustration, *# Flows* lists the numbers of *all* flows in the training reviews and of those with a recall of at least *1%*.

transformations from Section 4. All variants are named according to the applied transformations.

Measures In (Wachsmuth et al., 2014b), we propose specific notions of the recall and precision of a sentiment flow f in a given collection of reviews: The recall R_f denotes the relative frequency of reviews with flow f, while the precision $P_f(\mathrm{s})$ with respect to some global sentiment s denotes the relative co-occurrence of f with s. Here, we extend these measures for complete models as follows.

We define the *aggregate recall* of a model on a collection of reviews as the sum of the recall of the set F of all its known sentiment flows:

$$\textit{Aggregate Recall}(F) = \sum_{f \in F} R_f$$

With *weighted precision*, we denote the sum of the maximum precision of each such flow in F, weighted with the recall of the flow:

$$\textit{Weighted Precision}(F) = \sum_{f \in F} \max_{\mathrm{s}} \{P_f(\mathrm{s})\} \cdot R_f$$

In addition, we assess how much two domains differ under a given model variant. To this end, we measure the *Hellinger distance* H_f (in the range $[0, 1]$) between the global sentiment distributions of each flow f known for both domains:

Model variant	Domain	Most common flow	Rank	Recall	Positive	Neutral	Negative
change-2class-noloops	**Product**	(negative, negative)	**1.**	13.6	0.0 %	25.0 %	75.0 %
	Hotel		9.	4.2	0.0 %	8.9 %	91.1 %
	Movie		4.	6.7	0.0 %	0.0 %	100.0 %
	Product	(positive, negative, positive)	8.	3.4	34.1 %	65.9 %	0.0 %
	Hotel		**1.**	10.5	45.1 %	49.6 %	5.3 %
	Movie		14.	2.3	0.0 %	73.3 %	26.7 %
	Product	(negative, negative, positive, negative)	10.	3.4	0.0 %	33.3 %	66.7 %
	Hotel		15.	2.2	0.0 %	16.7 %	83.3 %
	Movie		**1.**	8.6	0.0 %	57.9 %	42.1 %
2class-noloops	**Product**	(negative, negative)	**1.**	15.3	7.6 %	29.6 %	62.8 %
	Hotel		5.	4.4	0.0 %	8.5 %	91.5 %
	Movie		**1.**	6.7	0.0 %	0.0 %	100.0 %
	Product	(positive, positive)	3.	12.8	86.8 %	8.8 %	4.4 %
	Hotel		**1.**	7.6	87.8 %	12.2 %	0.0 %
	Movie		10.	2.1	61.1 %	38.9 %	0.0 %
change-noloops	**Product**	(positive, neutral, positive)	**1.**	10.5	94.6 %	0.0 %	5.4 %
	Hotel		6.	3.3	88.9 %	11.1 %	0.0 %
	Movie		23.	1.3	100.0 %	0.0 %	0.0 %
	Product	(positive)	22.	1.1	50.9 %	49.1 %	0.0 %
	Hotel		**1.**	6.6	83.1 %	14.1 %	2.8 %
	Movie		–	–	–	–	–
	Product	(neutral, negative)	2.	9.1	6.5 %	24.9 %	68.6 %
	Hotel		5.	3.5	0.0 %	10.5 %	89.5 %
	Movie		**1.**	7.6	8.6 %	0.0 %	91.4 %

Table 3. The most common flow in the training set of each evaluated domain for three of the 16 evaluated model variants. For each flow, the recall rank, the recall, and the distribution over the three global sentiments within each domain are given.

$$H_f(\mathbf{p}_1, \mathbf{p}_2) = \frac{1}{\sqrt{2}} \cdot \sqrt{\sum_{\mathrm{s}} (\sqrt{\mathbf{p}_1(\mathrm{s})} - \sqrt{\mathbf{p}_2(\mathrm{s})})^2}$$

Here, $\mathbf{p}_1$ and $\mathbf{p}_2$ denote the global sentiment distributions of f. For *weighted Hellinger distances*, we multiply the distance of each flow in F with the sum of its recall in both domains.[4]

Experiments Given all 16 possible model variants for all reviews, we analyze the generality of each variant for every combination of domains. I.e., we first determine the known sentiment flows on the training set of one domain. Then, we compute the aggregate recall, weighted precision, and weighted Hellinger distance once for the in-domain test set and once for both full out-of-domain corpora.[5]

5.3 Results on the Generality across Domains

Table 2 contains the number of known flows and the experiment results for each domain combination. Model variants whose benefit seems limited are not marked in bold: The bottom six have a low aggregate recall in all domains, suggesting that they do not generalize well. Most significantly, the *original* flows from the movie training set are not found in any test domain. The top five achieve almost total recall, but much less precision than the others, indicating that they abstract too much.

Among the five robust model variants (marked in bold), *change-2class-noloops* has the highest aggregate recall throughout, ranging from 73.0 to 92.9. Consistently, global sentiment is represented best by *change-noloops* in the product domain and by *2class-noloops* in the other domains (with up to 85.1 weighted precision). Also, *2class-noloops* is third-best in terms of recall. While no clear "winner" exists, this variant seems most promising for modeling web review argumentation.

The weighted Hellinger distances show that the domain differences of many variants are small. On average, *change-2class* has the most stable global sentiment distribution. Most distances are only slightly higher out-of-domain than in-domain or even lower. Hence, sentiment flows hardly vary stronger across domains than within a domain.

5.4 The Most Common Sentiment Flows

To investigate what sentiment flows actually occur in web reviews, we determined the flow with highest recall for the training set of each corpus. For comparability, we balanced the flows in the training set before by weighting their occurrences according to the distribution of global sentiment.[6]

[4] We chose the Hellinger distance, as it applies to distributions with zero-probabilities (unlike alternatives like the KL-divergence). Also, it is a true metric (Lebret and Collobert, 2014), allowing for relative comparisons. On the flipside, the meaning of concrete distances is not clear by itself.

[5] Here, we use *all* occurring sentiment flows to evaluate a model variant in its overall manifestation. In Section 6, we consider only frequent flows in order to refrain from outliers.

[6] E.g., if 40% of all reviews are positive, 30% neutral, and 30% negative, then the occurrences of flows with positive global sentiment are weighted by 0.75 and the others by 1.0.

Table 3 shows the recall and the sentiment distribution of each such flow in all evaluated domains exemplarily for three of the model variants discussed above. While high-recall flows naturally tend to be simple, we also observe more complex flows, such as *(negative, negative, positive, negative)* in case of *change-2class-noloops*. Except for the *change-noloops* flow *(positive)*, which does not occur at all in the movie training set, all shown flows are common across domains, achieving a recall of over 2% in most cases. For *2class-noloops*, only two flows are listed, because *(negative, negative)* is the most common flow in both the product and the movie domain. Regarding the distribution of global sentiments, nearly all flows behave similar across domains. The only exception is *(positive, negative, positive)* in *change-2class-noloops*, which never turns out negative in product reviews but never positive in movie reviews.

Altogether, we conclude that sentiment flow is not a fully precise model of web review argumentation, but it proves general with respect to global sentiment. What remains to be checked is the benefit of modeling sentiment flow under uncertainty.

6 Analysis of the Robustness of the Model

Finally, we evaluate how effectively and domain-robustly sentiment flow predicts global sentiment, when local sentiment is not given but classified. To analyze the domain independence of our model, no knowledge about target domains is used.

6.1 Sentiment Analysis Approaches

To classify sentiment flows, we build on the edit distance approach presented in Section 4:

Model Variants As in Section 5, we look at all 16 possible variants of the proposed model. For each variant, we determine all sentiment flows that represent at least 1% of all reviews in a given training set. We learn a mapping from the edit distance between a review's sentiment flow and each of these flows to global sentiment. For robustness, we also combine different model variants.

We compare the accuracy of the model variants to previous approaches evaluated on the given corpora. In addition, we analyze domain robustness based on three baselines, which relate to the three abstraction levels in Figure 2 (cf. Section 4):

Bag-of-Words (b1) The frequencies of all tokens that occur in at least 5% of all training reviews.

Local Sentiment (b2) The frequencies of positive, neutral, and negative local sentiment in a review as well as the first and last local sentiment (analog to Section 4, local sentiment is mapped to $[0, 1]$).

Sentiment Flow Patterns (b3) The Manhattan distances to those sentiment flows obtained by our clustering approach (Wachsmuth et al., 2014a).

All approaches are used as feature types in machine learning (with values normalized to $[0, 1]$).

6.2 Experimental Set-up

We tackle three-class sentiment analysis, which is supposed to be particularly hard due to the fuzzy nature of neutral sentiment (Täckström and McDonald, 2011). Given the three review corpora described in Section 5, we proceed as follows:[7]

Local Sentiment For feature computations, we split all reviews into tokens and sentences. Then, we classify local sentiment with the algorithm of Socher et al. (2013) from Stanford CoreNLP.[8] The algorithm was trained on subjective movie review sentences. We found that its accuracy is limited to around 50% on the given corpora, partly as it tends to misclassify objective sentences. Still, we use it to avoid any adaptation to the domains at hand.

Global Sentiment To determine global sentiment, we perform supervised learning based on the feature types outlined above. In particular, we use the default configuration of the random forest classifier from Weka (Breiman, 2001; Hall et al., 2009) without any parameter optimization.

Experiments Having classified local sentiment in all reviews, we learn a random forest classifier on each feature type and different feature sets for all combinations of training and test domain. To prevent class bias, the training sets are balanced with duplicate oversampling. Since the size of the corpora is limited, we evaluate in-domain accuracy on the whole corpora using 10-fold cross-validation, averaged over five runs. Afterwards, we test the out-of-domain accuracy by applying the learned classifier to the other complete corpora.

6.3 Results on the Domain Robustness

Table 4 lists accuracy results for all domain combinations. In the movie domain, we obtain an overall accuracy of 71.8. On average, we thus succeed over Pang and Lee (2005) who report about 75 on the reviews of Scott Renshaw and 63 on those of Dennis Schwartz. Similarly, we beat all our three-class sentiment analysis results from (Wachsmuth,

[7] Again, see http://www.arguana.com/software for code.
[8] Stanford CoreNLP, http://nlp.stanford.edu/software

Training	Feature types	Product	Hotel	Movie
Product	b1 Bag-of-words	49.0	45.9	32.4
	b2 Local sentiment	51.7	50.4	39.3
	b3 Sentiment flow patterns	46.8	57.5	47.8
	All baseline features b1-3	51.9	58.8	49.8
	v1 change-2class	46.0	46.6	41.3
	v2 change-2class-noloops	48.7	46.9	38.4
	v3 noloops-2class	48.7	50.1	43.6
	v4 2class	52.0	53.8	44.9
	v5 2class-noloops	48.0	50.4	42.4
	All model variants v1-5	50.5	51.3	42.4
	All flows (v1-5 + b3)	50.9	58.2	**51.1**
	All sentiment (v1-5 + b2-3)	50.8	59.7	50.2
	All features (v1-5 + b1-3)	**54.2**	**60.0**	48.7
Hotel	b1 Bag-of-words	37.8	**79.6**	39.8
	b2 Local sentiment	51.4	64.2	51.1
	b3 Sentiment flow patterns	50.7	74.2	51.1
	All baseline features b1-3	54.8	78.9	48.7
	v1 change-2class	43.2	54.3	43.3
	v2 change-2class-noloops	46.6	49.4	45.3
	v3 noloops-2class	49.3	57.4	46.7
	v4 2class	52.4	58.6	51.6
	v5 2class-noloops	46.9	54.0	48.2
	All model variants v1-5	53.4	69.0	**54.7**
	All flows (v1-5 + b3)	53.8	75.5	53.6
	All sentiment (v1-5 + b2-3)	**57.1**	75.6	51.8
	All features (v1-5 + b1-3)	56.4	79.0	53.3
Movie	b1 Bag-of-words	35.0	41.2	64.8
	b2 Local sentiment	43.2	44.2	59.0
	b3 Sentiment flow patterns	42.2	39.5	67.2
	All baseline features b1-3	48.0	50.4	70.5
	v1 change-2class	42.9	44.0	44.8
	v2 change-2class-noloops	40.8	48.1	44.2
	v3 noloops-2class	44.9	46.5	50.7
	v4 2class	–	–	–
	v5 2class-noloops	44.2	44.7	55.9
	All model variants v1-5	44.6	49.7	60.9
	All flows (v1-5 + b3)	47.6	51.9	65.2
	All sentiment (v1-5 + b2-3)	**49.7**	**54.1**	65.9
	All features (v1-5 + b1-3)	48.0	52.3	**71.8**

Table 4. Accuracy of predicting 3-class global sentiment for each combination of training and test domain using the baselines and/or a selection of the 16 evaluated model variants.

2015) in the hotel domain. In the product domain, our approach fails to compete with Täckström and McDonald (2011) who classify the global sentiment of 66.6% of all reviews correctly after training on large-scale product corpora. The small size of the given corpus explains the limited in-domain accuracy in Table 4; even some out-of-domain classifiers perform better on the product reviews. Still, the value 54.2 significantly improves over all baselines under a paired t-test ($p < 5\%$).

As expected, bag-of-words (b1) proves strong in some in-domain tasks—achieving even the best overall accuracy in the hotel domain (79.6)—but it consistently fails out-of-domain. Although less clear, similar observations can be made for b2. This shows that a restriction to the distribution of local sentiment is insufficient to tackle domain dependence. The sentiment flow patterns are comparably effective out-of-domain, but still suffer from the domain change on the evaluated corpora.

For space reasons, we compare the baselines b1 to b3 only to a selection of five of the most effective model variants, v1 to v5. Alone, these variants only occasionally do better than the sentiment flow patterns (b3). However, their combination (v1–5) clearly outperforms b3 in 4 out of 6 out-of-domain experiments. A strong variant is *2class-noloops*, which already proved general in the results from Section 5. In contrast, *2class* (v4) appears controversial. While it turns out being both effective and domain-robust when training in the product and hotel domain, no *2class* sentiment flow represents at least 1% of the movie corpus, emphasizing that more abstraction is required for robustness.

Altogether, the bottom lines of each domain in Table 4 provide clear evidence that our approach improves domain robustness in sentiment analysis: In all cases, the out-of-domain accuracy is best when using our sentiment flow features v1–5. At the same time, our results suggest that very high effectiveness might require more adaptation to the target domain. In this regard, modeling sentiment flow serves as a promising basis to align more effective but domain-dependent features.

7 Conclusion

This paper puts the goal of domain independence in the sentiment analysis of web reviews into the focus. In particular, we hypothesize that an abstract model of the local sentiment flow in a review generally captures the review's overall argumentation regarding global sentiment. In ground-truth data from three domains, we have found clear evidence for our hypothesis, indicating that people write reviews in similar ways across domains.

On this basis, we have presented a novel learning approach, which predicts the global sentiment of a review from the edit distance between the review's sentiment flow and a set of common flows. While we determined common flows with clustering in previous work (Wachsmuth et al., 2014a), instead here we rely on different flow abstractions at the same time. Systematic experiments emphasize that, in this manner, our approach achieves domain robustness without any domain adaptation even when the accuracy of the local sentiment in the flows is limited.

However, our experiments also show that sentiment flow alone does not always suffice to predict global sentiment. In future sentiment analysis approaches, sentiment flows may therefore rather serve as pivot features for domain adaptation.

References

Philippe Besnard and Anthony Hunter. 2008. *Elements of Argumentation*. The MIT Press.

Leo Breiman. 2001. Random Forests. *Machine Learning*, 45(1):5–32.

Samuel Brody and Noemie Elhadad. 2010. An Unsupervised Aspect-Sentiment Model for Online Reviews. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 804–812.

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. 2009. *Introduction to Algorithms*. MIT Press, third edition.

Ivan Habernal, Judith Eckle-Kohler, and Iryna Gurevych. 2014. Argumentation Mining on the Web from Information Seeking Perspective. In *Proceedings of the Workshop on Frontiers and Connections between Argumentation Theory and Natural Language Processing*, pages 26–39.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. 2009. The WEKA Data Mining Software: An Update. *SIGKDD Explorations*, 11(1):10–18.

Minqing Hu and Bing Liu. 2004. Mining and Summarizing Customer Reviews. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 168–177.

Angeliki Lazaridou, Ivan Titov, and Caroline Sporleder. 2013. A Bayesian Model for Joint Unsupervised Induction of Sentiment, Aspect and Discourse Representations. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1630–1639.

Rémi Lebret and Ronan Collobert. 2014. Word Embeddings through Hellinger PCA. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 482–490.

Bing Liu. 2012. *Sentiment Analysis and Opinion Mining*. Morgan & Claypool.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press.

Yi Mao and Guy Lebanon. 2007. Isotonic Conditional Random Fields and Local Sentiment Flow. *Advances in Neural Information Processing Systems*, 19:961–968.

Rohith Menon and Yejin Choi. 2011. Domain Independent Authorship Attribution without Domain Adaptation. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing 2011*, pages 309–315.

Raquel Mochales and Marie-Francine Moens. 2011. Argumentation Mining. *Artificial Intelligence and Law*, 19(1):1–22.

Diarmuid Ó Séaghdha and Simone Teufel. 2014. Unsupervised Learning of Rhetorical Structure with Un-topic Models. In *Proceedings of he 25th International Conference on Computational Linguistics: Technical Papers*, pages 2–13.

Bo Pang and Lillian Lee. 2005. Seeing Stars: Exploiting Class Relationships for Sentiment Categorization with Respect to Rating Scales. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 115–124.

Bo Pang and Lillian Lee. 2008. Opinion Mining and Sentiment Analysis. *Foundations and Trends in Informal Retrieval*, 2(1–2):1–135.

Isaac Persing, Alan Davis, and Vincent Ng. 2010. Modeling Essay Organization in Student Essays. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 229–239.

Ana-Maria Popescu and Oren Etzioni. 2005. Extracting Product Features and Opinions from Reviews. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 339–346.

Peter Prettenhofer and Benno Stein. 2010. Cross-Language Text Classification using Structural Correspondence Learning. In *Proceedings of the 48th Annual Meeting of the Association of Computational Linguistics*, pages 1118–1127.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. 2013. Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642.

Oscar Täckström and Ryan McDonald. 2011. Discovering Fine-grained Sentiment with Latent Variable Structured Prediction Models. In *Proceedings of the 33rd European Conference on Advances in Information Retrieval*, pages 368–374.

Simone Teufel. 2014. Scientific Argumentation Detection as Limited-Domain Intention Recognition. In *Proceedings of the Workshop on Frontiers and Connections between Argumentation Theory and Natural Language Processing*, pages 101–109.

Frans H. van Eemeren, Bart Garssen, Erik C. W. Krabbe, A. Francisca Snoeck Henkemans, Bart Verheij, and Jean H. M. Wagemans. 2014. *Handbook of Argumentation Theory*. Springer.

Maria Paz Garcia Villalba and Patrick Saint-Dizier. 2012. Some Facets of Argument Mining for Opinion Analysis. In *Proceedings of the 2012 Conference on Computational Models of Argument*, pages 23–34.

Henning Wachsmuth, Martin Trenkmann, Benno Stein, and Gregor Engels. 2014a. Modeling Review Argumentation for Robust Sentiment Analysis. In *Proceedings of the 25th International Conference on Computational Linguistics: Technical Papers*, pages 553–564.

Henning Wachsmuth, Martin Trenkmann, Benno Stein, Gregor Engels, and Tsvetomira Palakarska. 2014b. A Review Corpus for Argumentation Analysis. In *Proceedings of the 15th International Conference on Intelligent Text Processing and Computational Linguistics*, pages 115–127.

Henning Wachsmuth. 2015. *Pipelines for Ad-hoc Large-Scale Text Mining*. To appear in Lecture Notes in Computer Science. Springer, available at http://is.upb.de/?id=wachsmuth.

Hongning Wang, Yue Lu, and Chengxiang Zhai. 2010. Latent Aspect Rating Analysis on Review Text Data: A Rating Regression Approach. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 783–792.

Qiong Wu, Songbo Tan, Miyi Duan, and Xueqi Cheng. 2010. A Two-Stage Algorithm for Domain Adaptation with Application to Sentiment Transfer Problems. In *Information Retrieval Technology*, volume 6458 of *Lecture Notes in Computer Science*, pages 443–453. Springer.

Neural Networks for Open Domain Targeted Sentiment

Meishan Zhang and **Yue Zhang** and **Duy-Tin Vo**
Singapore University of Technology and Design
{meishan_zhang, yue_zhang}@sutd.edu.sg,
duytin_vo@mymail.sutd.edu.sg

Abstract

Open domain targeted sentiment is the joint information extraction task that finds target mentions together with the sentiment towards each mention from a text corpus. The task is typically modeled as a sequence labeling problem, and solved using state-of-the-art labelers such as CRF. We empirically study the effect of word embeddings and automatic feature combinations on the task by extending a CRF baseline using neural networks, which have demonstrated large potentials for sentiment analysis. Results show that the neural model can give better results by significantly increasing the recall. In addition, we propose a novel integration of neural and discrete features, which combines their relative advantages, leading to significantly higher results compared to both baselines.

1 Introduction

Targeted sentiment analysis has drawn growing research interests over the past few years. Compared with traditional sentiment analysis tasks, which extract the overall sentiment of a document, a sentence or a tweet, targeted sentiment analysis extracts the sentiment over given targeted entities from a text, and therefore is practically more informative. An example is shown in Figure 1. There are at least two practical scenarios:

(1) Certain entities of concern are specified, and the requirement is to extract the sentiment towards their mentions in a text. For example, one can be interested in the sentiment towards *Google Inc.*, *Microsoft* and *Facebook* in financial news texts, or the sentiment towards *Manchester United*, *Liverpool* and *Chelsea* in tweets.

So excited to meet my [**baby Farah**]$_+$!!!
[**Baseball Warehouse**]$_+$: easy to understand information.
The [**#Afghan #Parlaiment Speaker**]$_-$ should Resign .
Saw [**Erykah Badu**]$_-$ last night , vile venue unfortunately .
[**AW service**]$_0$ will be back at work .

Figure 1: *Targeted sentiment analysis*.

(2) No specified target is given, and the requirement is to find sentiments towards entities in the open domain. For example, one might be interested extracting the mentions to all persons and organizations, together with the sentiments towards each mention, from a news archive or a collection of novels.

There are two sub tasks in targeted sentiment analysis, namely entity recognition and sentiment classification for each entity mention which apply to both scenarios above. In scenario (1), *entity recognition* is relatively trivial, and can typically be achieved by pattern matching. Partly due to this reason, most previous work has addressed targeted sentiment analysis as a pure classification task, assuming that target mentions have been given (Jiang et al., 2011; Chen et al., 2012; Dong et al., 2014; Vo and Zhang, 2015). For scenario (2), a named entity recognition (NER) system can be used to extract targets, before the same targeted sentiment classification algorithms are applied. There has also been work that concentrates on extracting opinion targets (Jin et al., 2009; Jakob and Gurevych, 2010). In both cases, the data in Figure 1 can be used for training sentiment classifiers.

Mitchell et al. (2013) took a different approach, extracting named entities and their sentiment classes jointly. They model the joint task

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 612–621,
Lisbon, Portugal, 17-21 September 2015.

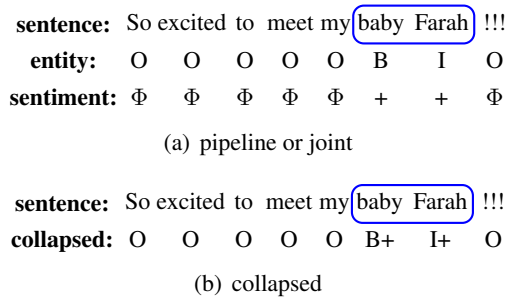


Figure 2: Pipeline, joint and collapsed models for *open targeted sentiment analysis*.

as an extension to the NER task, where an extra sentiment label is assigned to each named entity, in addition to the entity label. As a result, the task can be solved using sequence labeling methods. As claimed by Mitchell et al. (2013), the joint task is particularly suitable when no extra resources are available for training separate syntactic analyzers or name entity recognizers. Such situations can include tweets and low-resource languages/domains. Interestingly, because of containing entity information, the annotation in Figure 1 suffices for training joint entity and sentiment labels even if it is the only resource available.

The annotations in Figure 1 can be transformed into label sequences, as shown in Figure 2. Figure 2 consists of two types of labels, where the B/I/O labels indicate span boundaries, and the +/-/0 labels indicate sentiment classes. The two types of labels can be assigned in a span→sentiment pipeline, or jointly as a multi-label task. Alternatively, as shown in Figure 2(b), the two types of labels can be collapsed into a joint label, such as B+ and I-, indicating the beginning of a positive entity and the middle of a negative entity, respectively. The collapsed labels allow joint entity recognition and sentiment classification to be achieved using a standard sequence labeler.

Mitchell et al. (2013) compare a pipeline model, a joint model and a collapsed model under the same conditional random field (CRF) framework, finding that the pipeline method outperforms the joint model on a tweet dataset. Intuitively, the interaction between entity boundaries and sentiment classes might not be as strong as that between more closely-coupled sources of information, such as word boundaries and POS (Zhang and Clark, 2008), or named entities and constituents (Finkel and Manning, 2009), for which joint models significantly outperform pipeline models. On the other hand, there do exist cases where entity boundaries and sentiment classes reinforce each other. For example, in a tweet such as 'I like *X*.', the contextual pattern indicate both a positive sentiment and an entity in the place of *X*.

Recently, neural network models have been increasingly used for sentiment analysis (Socher et al., 2013; Kalchbrenner et al., 2014; dos Santos and Gatti, 2014), achieving highly competitive results, which show large potentials of neural network models for this task. The main advantages of neural networks are two-fold. First, neural models use real-valued hidden layers to automatically learn feature combinations, which can capture complex semantic information that are difficult to express using traditional discrete manual features. Second, neural networks take distributed word embeddings as inputs, which can be trained from large-scale raw text, thus alleviating the scarcity of annotated data to some extent. In this paper, we exploit structured neural models for open targeted sentiment.

We take the CRF model of Mitchell et al. (2013) as the baseline, and explore two research questions. First, we make an empirical comparison between discrete and neural CRF models, and further combine the strengths of each model via feature integration. Second, we compare the effects of the pipeline, joint and collapsed models for open targeted sentiment analysis under the neural model settings. Our experiments show that the neural model gives competitive results compared with the discrete baseline, with relatively higher recalls. In addition, the integrated model significantly improves over both the discrete and the neural models.

2 Related Work

Targeted sentiment analysis is closely related prior work on aspect-oriented (Hu and Liu, 2004), feature-oriented (Popescu and Etzioni, 2007) and topic-oriented (Yi et al., 2003) sentiment analysis. These related tasks are typically concentrated on product review settings. In contrast, targeted sentiment analysis has a more general setting.

Recently, Wang et al. (2011) proposed a topic-oriented model, which extracts sentiments towards certain topics from tweets. Topics in their model resemble targets in our work, although topics are represented by hashtags, which exists in 14.6% tweets and 27.5% subjective tweets (Wang et al.,

2011). In contrast, targeted sentiment analysis can identify all the mentions to target entities in tweets, thereby having a larger coverage. The drawback is that the identification of mentions is subject to errors, and thus suffers a lower precision compared to hashtag matching.

Sequence labeling models have been used for extracting opinions and target entities as a joint task. Jin et al. (2009) use HMM to extract opinion-baring expressions and opinion targets. Li et al. (2010) improve the results by using CRF to identify the opinion expressions and targets jointly. The task is sometimes referred to as fine-grained sentiment analysis (Wiebe et al., 2005). It is different from our setting in that the predicate-argument relation between opinion-baring expressions and target entities are not explicitly modeled.

Recently, Yang and Cardie (2013) use CRF to extract opinion-baring expressions, opinion holders and opinion targets simultaneously. Their method is also centralized on opinion-baring expressions and therefore in line with Jin et al. (2009) and Li et al. (2010). In contrast, targeted sentiment analysis directly studies entity mentions and the sentiment on each mention, without explicitly modeling the way in which the opinion is expressed. As a result, our task is more useful for applications such as broad-stroke reputation management, but offer less fine-grained operational insight. It requires less fine-grained manual annotation.

As discussed in the introduction, targeted sentiment analysis falls into two main settings. The first is targeted sentiment classification, assuming that entity mentions are given. Most previous work fall under this category (Jiang et al., 2011; Chen et al., 2012; Dong et al., 2014). The second is open domain targeted sentiment, which has been discussed by Mitchell et al. (2013). The task jointly extracts entities and sentiment classes, and is analogous to joint entity and relation extraction (Li and Ji, 2014) in that both are information extraction tasks with multi-label outputs.

Our work is related to the line of work on using neural networks for sentiment analysis. Socher et al. (2011) use recursive auto-encoders for sentiment analysis on the sentence level. They further extend the method to a syntactic treebank annotated with sentiment labels (Socher et al., 2013). More recently, Kalchbrenner et al. (2014) use a dynamic pooling network to include the structure

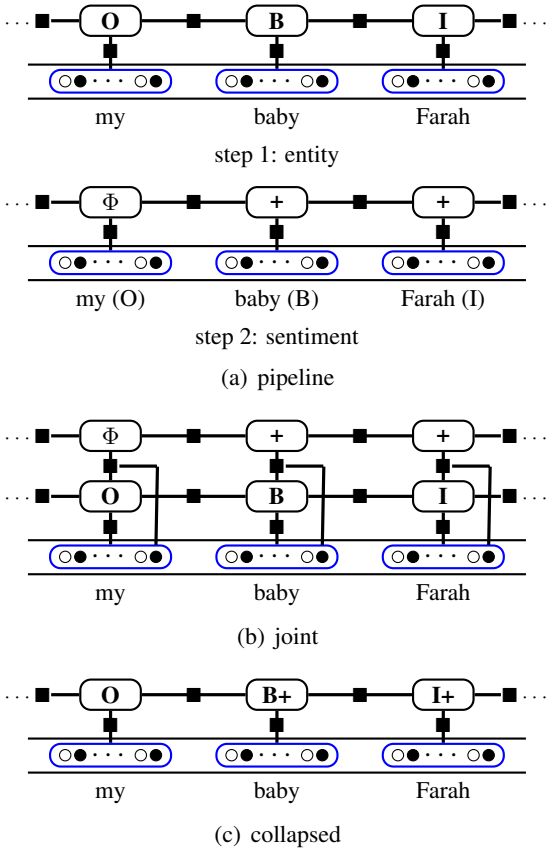


Figure 3: Discrete CRF models for pipeline, joint and collapsed targeted sentiment labeling.

of a sentence automatically, before classifying its sentiment. Zhou et al. (2014) apply deep belief networks for semi-supervised sentiment classification. dos Santos and Gatti (2014) use deep convolution neural networks with rich features to classify sentiments over tweets and movie reviews. These methods use different models to represent sentence structures, performing sentiment analysis on the sentence level, without modeling targets.

Dong et al. (2014) perform targeted sentiment classification by using a recursive neural network to model the transmission of sentiment signal from opinion baring expressions to a target. They assume that the target mention is given, and perform three-way sentiment classification. In contrast, we apply a structural neural model for open domain targeted sentiment analysis, identifying and classifying all targets in a sentence simultaneously.

3 Discrete CRF Baselines

As shown in Figure 2, the input $\vec{x}$ to our tasks is a word sequence. Assuming no external resources, there is no POS given to each input word x_i. For

the pipeline and collapsed tasks, there is a single output label sequence $\vec{y}$. For the joint task, there are two label sequences $\vec{y}$ and $\vec{z}$, for entity and sentiment labels, respectively. We take the models of Mitchell et al. (2013) as our baseline, which are standard CRFs with discrete manual features. To facilitate comparison between the discrete baseline and our neural models, we give a unified formulation to all the models in this paper, introducing the neural and integrated models as extensions to the discrete models.

The baseline CRF structures for pipeline, joint and collapsed targeted sentiment analysis are shown in Figure 3(a), 3(b) and 3(c), respectively. In the figures, the input features are represented as black and white circles, indicating that they take $0/1$ binary values. The labels O, B and I indicate a non-target, the beginning of a target, and part of a target, respectively. The labels $+$, $-$, 0 and Φ indicate positive, negative, neutral and *NULL* sentiments, respectively. The *NULL* sentiment is assigned to O entities automatically, and modeled as a hidden variable in the pipeline and joint CRFs.[1] The collapsed labels take combined meanings from their components.

The links between labels and inputs represent output clique potentials:

$$\Psi(\vec{x}, y_i) = exp\Big\{\vec{\theta} \cdot \vec{f}(\vec{x}, y_i)\Big\},$$

where $\vec{f}(\vec{x}, y_i)$, is a discrete manual feature vector, and $\vec{\theta}$ is the model parameter vector.

The links between labels represent edge clique potentials:

$$\Phi(\vec{x}, y_i, y_{i-1}) = exp\Big\{\tau(y_i, y_{i-1})\Big\},$$

where $\tau(y_i, y_{i-1})$ is the transition weight, which is also a model parameter.

For both the pipeline and collapsed models, the conditional probability of a label sequence given an input sequence is:

$$P(\vec{y}|\vec{x}) = \frac{\prod_{i=1}^{|x|} \Psi(\vec{x}, y_i) \prod_{j=1}^{|x|} \Phi(\vec{x}, y_i, y_{i-1})}{Z(\vec{x})},$$

[1]Note the difference between neural and *NULL* sentiments. The former indicates that a target does not bare any sentiment, and the latter simply means that the term is not a part of a target.

surface features
word identity; word length; message length;
punctuation characters; has digit; has dash; is lower case;
is 3 or 4 letters; first letter capitalized; sentence position;
more than one letter capitalized; Jerboa features;
linguistic features
function words; can syllabify; curse words;
laugh words; words for good, bad, no, my;
intensifiers; slang words; abbreviations;
common verb endings; common noun endings;
subjective suffixes and prefixes;
cluster features
Brown cluster at length 3; Brown cluster at length 5;
sentiment features
is sentiment-bearing word; prior sentiment polarity;

Table 1: Discrete features.

where $Z(\vec{x})$ is the partition function:

$$Z(\vec{x}) = \sum_{\vec{y}'} \Big(\prod_{i=1}^{|x|} \Psi(\vec{x}, y'_i) \prod_{j=1}^{|x|} \Phi(\vec{x}, y'_i, y'_{i-1}) \Big),$$

For the joint model, we apply a multi-label CRF structure, where there are two separate sets of output clique potentials $\Psi_1(\vec{x}, y_i)$ and $\Psi_2(\vec{x}, z_i)$ and two separate sets of edge clique potentials $\Phi_1(\vec{x}, y_i, y_{i-1})$ and $\Phi_2(\vec{x}, z_i, z_{i-1})$ for the label sets $\{B, I, O\}$ and $\{+, -, 0\}$, respectively. In the Figure 3(b), there are also links between the span label y_i and the sentiment label z_i for each word x_i. These links indicate label dependencies, which are constraints for decoding. For example, if $y_i = O$, then z_i must be ϕ.

We apply Viterbi decoding for all tasks, and training is performed using a max-margin objective, which is discussed in Section 6. Our training algorithm is different from that of Mitchell et al. (2013), but gives similar discrete CRF accuracies in our experiments. Wang and Mori (2009) also applied a max-margin trainig strategy to train CRF models. The set of features is taken from Mitchell et al. (2013) without changes, as shown in Table 1. Here the cluster features refer to Brown word clusters (Brown et al., 1992).

4 Neural Models

We extend the discrete baseline system with two salient changes, which are illustrated in Figure 4. First, the input discrete features are replaced with continuous word embeddings. Each node in the input takes a real value between 0 and 1, as represented by grey nodes in Figure 4. Second, a hidden

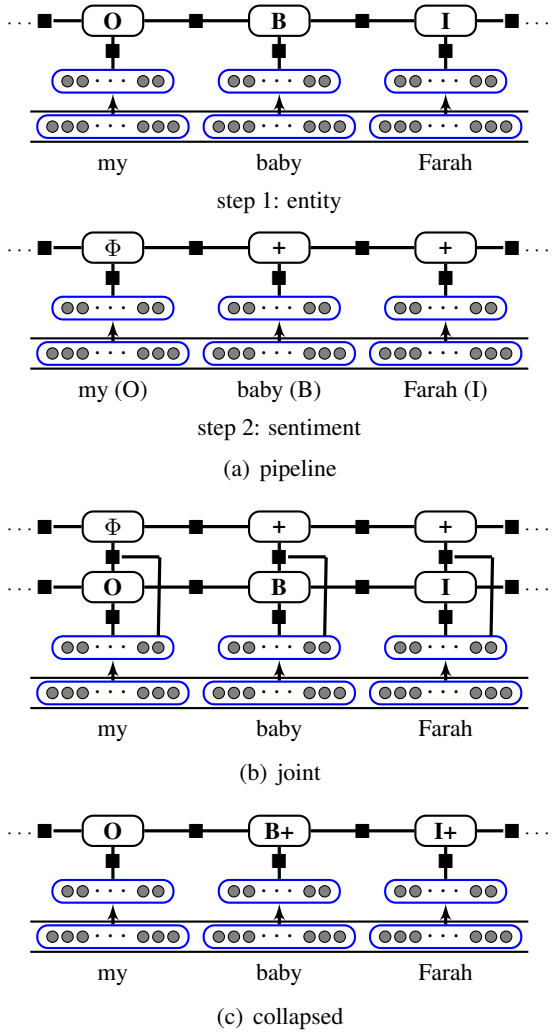


Figure 4: Neural networks for pipeline, joint and collapsed targeted sentiment labeling.

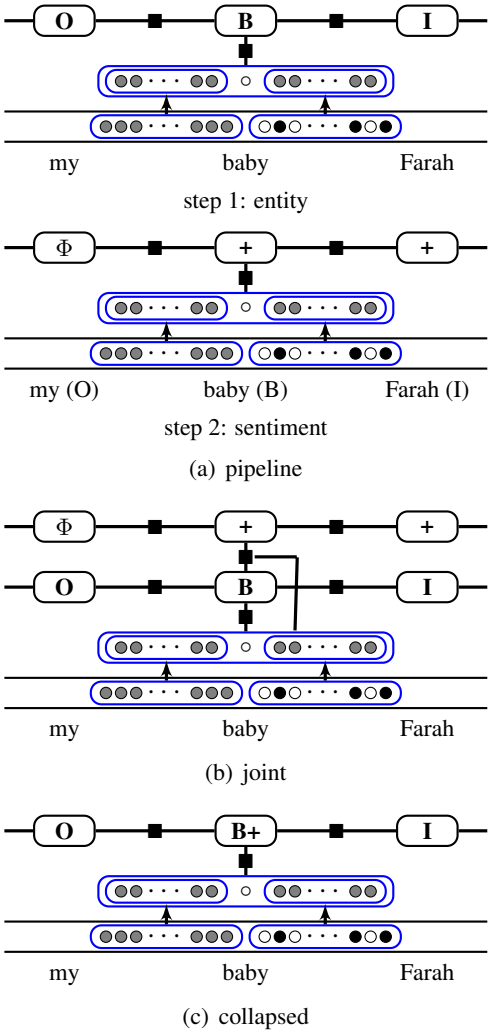


Figure 5: Integrated models for pipeline, joint and collapsed targeted sentiment labeling.

neural layer $\vec{h}$ is added between the input nodes $\vec{x}$ and the label nodes y_i.

Formally, the links between the input nodes $\vec{x}$ and the hidden nodes $\vec{h}_i$ for the node y_i in Figure 4 represent a feature combination function:

$$\vec{h}_i = tanh\Big(\mathbf{W}.(e(\vec{x}_{i-2}) \oplus e(\vec{x}_{i-1}) \oplus e(\vec{x}_i) \oplus e(\vec{x}_{i+1}) \oplus e(\vec{x}_{i+2})) + \vec{b}\Big)$$

where e is the embedding lookup function, $\oplus$ is the vector concatenation function, the matrix $\mathbf{W}$ and vector $\vec{b}$ are model parameters and *tanh* is the activation function.

The output clique potential of y_i becomes:

$$\Psi(\vec{x}, y_i) = exp\Big\{\vec{\sigma} \cdot \vec{h}_i\Big\}$$

where $\vec{\sigma}$ is a model parameter, and the edge clique potentials remain the same as the baseline. By using a hidden layer for automatic feature combinations, the neural model is free of manual features, and can benefit from unsupervised embeddings. Decoding and training are performed using the same algorithms as the baseline.

The major neural architectures in Figure 4 have been explored as *conditional neural fields* by Peng et al. (2009) and *neural conditional random fields* by Do et al. (2010), and is connected to the sentence-level likelihood neural networks of Collobert et al. (2011), as pointed out by Wang and Manning (2013b). The main differences between our model and the prior work are in the multi-label settings and training details.

5 Integrated Models

Gleaning different sources of information, neural features and discrete linear features comple-

ments each other. As a result, a model that integrates both features can potentially achieve performance improvements. Most work attempts to add neural word embeddings into a discrete linear model (Turian et al., 2010; Yu et al., 2013; Guo et al., 2014), or add discreted features into a neural model (Ma et al., 2014). We make a novel combination of the discrete models and the neural models by integrating both types of inputs into a same CRF framework.[2]

The architectures of the integrated models are shown in Figure 5. The main difference between Figure 5 and Figure 3 is the input layer. The integrated model takes both continuous word embeddings, which are shown in grey nodes, and discrete manual features, which are shown in black or white nodes, as the input.

A separate hidden layer is given to each type of input nodes, with the hidden layer for the embeddings being the same as the neural baseline:

$$\vec{h}_i = tanh\Big(\mathbf{W} \cdot (e(\vec{x}_{i-2}) \oplus e(\vec{x}_{i-1}) \oplus e(\vec{x}_i) \oplus e(\vec{x}_{i+1}) \oplus e(\vec{x}_{i+2})) + \vec{b}\Big)$$

The hidden nodes $\vec{g}_i$ between the discrete features and the node y_i are:

$$\vec{g}_i = tanh\Big(\vec{\theta} \cdot \vec{f}(\vec{x}, y_i)\Big)$$

Finally, the output clique potential of y_i becomes:

$$\vec{\Psi}(\vec{x}, y_i) = exp\Big\{\vec{\sigma} \cdot (\vec{h}_i \oplus \vec{g}_i)\Big\}$$

The edge clique potentials remain the same as the baseline models; the same training and decoding algorithms are used.

6 Training

We use a max-margin objective to train our model parameters Θ, which consist of $\vec{\theta}$, τ, $\mathbf{W}$, $\vec{b}$ and $\vec{\sigma}$ for each model. The objective function is defined as:

$$L(\Theta) = \frac{1}{N}\sum_{n=1}^{N} l(\vec{x}_n, \vec{y}_n, \Theta) + \frac{\lambda}{2} \parallel \Theta \parallel^2,$$

where $(\vec{x}_n, \vec{y}_n)|_{n=1}^{N}$ are the set of training examples, λ is a regularization parameter, and $l(\vec{x}_n, \vec{y}_n, \Theta)$ is the loss function towards one example $(\vec{x}_n, \vec{y}_n)$.

The loss function is defined as:

$$l(\vec{x}_n, \vec{y}_n, \Theta) = \max_{\vec{y}}(s(\vec{x}_n, \vec{y}, \Theta) + \delta(\vec{y}, \vec{y}_n)) - s(\vec{x}_n, \vec{y}_n, \Theta),$$

where $s(\vec{x}, \vec{y}, \Theta) = logP(\vec{y}|\vec{x})$ is the log probability of $\vec{y}$, and $\delta(\vec{y}, \vec{y}_n)$ is the Hamming distance between $\vec{y}$ and $\vec{y}_n$.

We use online learning to train model parameters, updating the parameters using the AdaGrad algorithm (Duchi et al., 2011). One thing to note is that, our objective function is not differentiable because of the loss function $l(\vec{x}_n, \vec{y}_n, \Theta)$. Thus we use sub-gradients for $l(\vec{x}_n, \vec{y}_n, \Theta)$ instead, which can be computed by the formula:

$$\frac{\partial l(\vec{x}_n, \vec{y}_n, \Theta)}{\partial \Theta} = \frac{\partial s(\vec{x}_n, \hat{\vec{y}}, \Theta)}{\partial \Theta} - \frac{\partial s(\vec{x}_n, \vec{y}_n, \Theta)}{\partial \Theta},$$

where $\hat{\vec{y}}$ is the predicted label sequence which corresponds to $l(\vec{x}_n, \vec{y}_n, \Theta)$.

Maximum-likelihood training is a commonly used alternative to max-margin training for neural networks. It has been applied to the models of Do et al. (2010) and Collobert et al. (2011), for example. However, our experiments show that maximum-likelihood training cannot be applied to open-domain targeted sentiment tasks. Although giving comparable overall accuracies in both entity and sentiment labels, it suffers from unbalanced sentiment labels, assigning the neutral sentiment to most entities. This problem can be addressed by imposing a polarity-sensitive cost to the training, such as the sentence-level averaged F1-score between positive, negative and neutral labels. We skip these results due to space limitations. In contrast, max-margin training does not suffer from the label skew issue, thanks to the use of Hamming loss in the objective function.

7 Experiments

7.1 Experimental Settings

Data: We use the data of Mitchell et al. (2013)[3] to conduct all the experiments, which consist of entity and sentiment annotations on both English and Spanish tweets. Simple normalizations are

[2] Wang and Manning (2013a) also investigated the integration of discrete and neural features in CRF models. They compared the effect of integration without hidden layers (i.e. Turian et al. (2010)) and with hidden layers (i.e. our methods) for NER and chunking, finding that the formal outperforms the latter. Our results are different from theirs, and a hidden layer gives significant improvements to the targeted sentiment analysis task.

[3] http://www.m-mitchell.com/code/index.html

Domain	#Sent	#Entities	#+	#-	#0
English	2,350	3,288	707	275	2,306
Spanish	5,145	6,658	1,555	1,007	4,096

Table 2: Experimental corpus statistics.

conducted to replace all usernames and URLs into the special tokens $\langle username \rangle$ and $\langle url \rangle$, respectively. Following Mitchell et al. (2013), we report ten-fold cross-validation results. During training, we split 10% of the training corpus as the development corpus to tune hyper-parameters. Table 2 shows the corpus statistics.

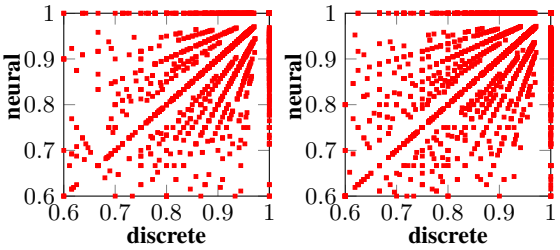
Parameters: For all the neural models, we set the hidden layer size $|\vec{h}|$ for neural features to 200, the hidden layer size $|\vec{g}|$ for discrete features to 30, the initial learning rate for adagrad to 0.01 and the regularization parameter λ to 10^{-8}. English and Spanish word embeddings are trained using the *word2vec* tool[4], with respective corpora of 20 minion random tweets crawled by tweet API[5]. The size of word embeddings is 100. For English, there are 8,061 unique words, for which 25% are out of *word embedding* vocabulary (OOE) words, while for Spanish, there are 14,648 unique words, for which 15% are OOE words.

Metrics: We take full-span metrics for evaluation, which is different from Mitchell et al. (2013), who evaluate mainly the beginning of spans. We measure the precision, recall and F-score of entity recognition (**Entity**), targeted sentiment analysis (**SA**) (both entity and sentiment), and targeted subjectivity detection (**Subjectivity**) (both entity and subjectivity, namely merging the + and - labels as "1" label, and performing two-way 0/1 subjectivity classification on entities). For **SA**, an entity is taken as correct only when the span and the sentiment are both correctly recognized. Similarly, for **Subjectivity**, an entity is taken as correct only when both the span and the subjectivity are correctly recognized.

Code: We make the C++ implementations of the discrete, neural and combined models available and GPL, at `https://github.com/SUTDNLP/OpenTargetedSentiment`.

7.2 Comparing Neural and Discrete Models

The main results on both the English and Spanish dataset are shown in Table 3, which are measured on the pipeline, the joint and the collapsed tasks, respectively. As can be seen from the table, the neural models give higher F-scores than the discrete CRF models on the English dataset, while comparable overall F-scores on the Spanish dataset. The gains on English are mostly attributed to improved recalls, while the precision of the neural CRF models are relatively lower. A likely reason for this observation is that the neural model takes embedding inputs, which allow semantically similar words to be represented with similar vectors. As a result, the neural model can better capture patterns that do not occur in the training data. In contrast, the discrete model is based on manually defined binary features, which do not fire if not contained in the training data. Because discrete feature instantiation is based on exact matching, the discrete model gives a relatively higher precision.



(a) English (b) Spanish

Figure 6: Labeling accuracy comparisons.

To further contrast the discrete and neural models, we draw the per-word accuracies of sentiment labels according to both models in Figure 6. In the figure, each dot represents the accuracy of a sentence, measured in the pipeline task. The dots for both English and Spanish are scattered from the reverse diagonal, showing that the two models make very different errors, which suggests that model integration can lead to better accuracies.

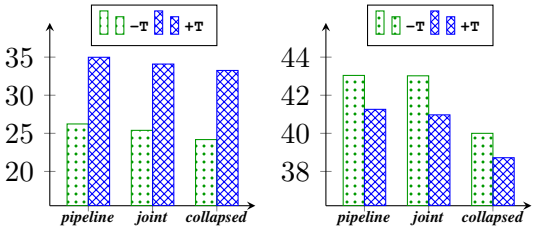
7.3 The Integrated Model

As shown in Table 3, the integrated model combines the relative advantages of both pure models, improving the recall over the discrete model and the precision over the neural model. In most cases, it gives the best results in terms of both precision and recall. For the English pipeline model, the integrated model improves the entity recognition F-score from 43.84% to 55.67% (significant with $p < 10^{-5}$ by pair-wise t-test) as compared to the discrete baseline, namely Mitchell et al. (2013).

[4]https://code.google.com/p/word2vec/
[5]https://dev.twitter.com/

Model	English						Spanish					
	Entity			SA			Entity			SA		
	P	R	F	P	R	F	P	R	F	P	R	F
Pipeline												
discrete	59.37	34.83	43.84	42.97	25.21	31.73	**70.77**	47.75	57.00	**46.55**	31.38	37.47
neural	53.64	44.87	48.67	37.53	31.38	34.04	65.59	47.82	55.27	41.50	30.27	34.98
integrated	**60.69**	**51.63**	**55.67**	**43.71**	**37.12**	**40.06**	70.23	**62.00**	**65.76**	45.99	**40.57**	**43.04**
Joint												
discrete	59.55	34.06	43.30	43.09	24.67	31.35	71.08	47.56	56.96	46.36	31.02	37.15
neural	54.45	42.12	47.17	37.55	28.95	32.45	65.05	47.79	55.07	40.28	29.58	34.09
integrated	**61.47**	**49.28**	**54.59**	**44.62**	**35.84**	**39.67**	**71.32**	**61.11**	**65.74**	**46.67**	**39.99**	**43.02**
Collapsed												
discrete	**64.16**	26.03	36.95	**48.35**	19.64	27.86	73.18	35.11	47.42	**49.85**	23.91	32.30
neural	58.53	37.25	45.30	43.12	27.44	33.36	67.43	43.2	52.64	42.61	27.27	33.25
integrated	63.55	**44.98**	**52.58**	46.32	**32.84**	**38.36**	**73.51**	**53.3**	**61.71**	47.69	**34.53**	**40.00**

Table 3: Main results.



(a) Neural (b) Integrated

Figure 7: Effect of fine-tuning (+T — with fine-tuning; -T — without fine-tuning).

The overall SA score is improved from 31.73% to 40.06% ($p < 10^{-5}$). Similar improvements are achieved to the other test datasets.

7.4 Fine-tuning Word Embeddings

In the experiments above, word embeddings are fine-tuned for the neural models, but not for the integrated models. By fine-tuning, embeddings of in-vocabulary words are treated as model parameters, and updated with other parameters in supervised training. This can improve the accuracy of the model by significantly enlarging the parameter space. However, it can make the embeddings of OOV words less useful to the model, because the hidden layers are tuned with adjusted embeddings.

Figure 7 shows the effectiveness of fine-tuning on the neural and integrated models using the Spanish data. Similar findings apply to the English data. The neural model heavily relies on fine-tuning of embeddings, and a likely reason is that manual discrete features offer sufficient parameters for capturing in-vocabulary patterns. On the other hand, thanks to the rich discrete features in parameter space, the integrated model does not rely on fine-tuning of word embeddings, which even caused slight overfitting and reduced the performances. This makes the non-fine-tuned integrated model potentially advantageous in handling test data with many OOV words.

7.5 Comparing pipeline, joint and collapsed models

Mitchell et al. (2013) find that for discrete CRF, the pipeline task gives competitive overall performances compared with the joint task. This suggests a relatively weak connection between entity boundary information and sentiment classes. We re-examine the comparisons under the neural network setting, where automatic feature combinations can be useful in capturing more subtle correlations between two sources of information.

As shown in Table 3, the overall results are similar to those of Mitchell et al. (2013), with both the neural and the integrated models demonstrating the same trends as the discrete baselines. A more detail analysis, however, shows some relative strengths of the joint task. Table 4 give the precision, recall and F-scores of subjectivity, and those of SA excluding neutral sentiment labels on the Spanish data. Findings on the English dataset are consistent.

The latter metrics highlight sentiment polarities, which can be relatively more useful. The joint task gives better F-scores on both metrics, which suggest that is a considerable choice for open targeted sentiment. When there is external resource for en-

Model	Subjectivity			SA/0		
	P	R	F	P	R	F
pipeline	47.92	**42.26**	44.84	**42.93**	18.02	25.14
joint	49.17	42.13	**45.32**	40.93	**21.62**	**27.93**
collapsed	**49.63**	35.94	41.63	42.10	15.62	22.49

Table 4: Results on subjectivity and polarity.

tity recognition, the pipeline can be a favorable choice. On the other hand, although useful for some joint sequence labeling task (Ng and Low, 2004), the collapsed task does not seem to address the joint sentiment task as effectively. We find this result empirical, but consistent across our datasets.

8 Conclusion

We explored open domain targeted sentiment analysis using neural network models, which gave competitive results when evaluated against a strong discrete CRF baseline, with relatively higher recalls. Given complementary error distributions by the discrete and neural CRFs, we proposed a novel combination which significantly outperformed both models. Under the neural setting, we find that it is preferable to solve open targeted sentiment as a pipeline or joint multi-label task, but not as a joint task with collapsed labels.

Acknowledgments

We thank the anonymous reviewers for their constructive comments, which helped to improve the paper. This work is supported by the Singapore Ministry of Education (MOE) AcRF Tier 2 grant T2MOE201301 and SRG ISTD 2012 038 from Singapore University of Technology and Design.

References

Peter F Brown, Peter V Desouza, Robert L Mercer, Vincent J Della Pietra, and Jenifer C Lai. 1992. Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467–479.

Lu Chen, Wenbo Wang, Meenakshi Nagarajan, Shaojun Wang, and Amit P Sheth. 2012. Extracting diverse sentiment expressions with target-dependent polarity from twitter. In *ICWSM*.

R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537.

Trinh Do, Thierry Arti, et al. 2010. Neural conditional random fields. In *International Conference on Artificial Intelligence and Statistics*, pages 177–184.

Li Dong, Furu Wei, Chuanqi Tan, Duyu Tang, Ming Zhou, and Ke Xu. 2014. Adaptive recursive neural network for target-dependent twitter sentiment classification. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 49–54.

Cicero dos Santos and Maira Gatti. 2014. Deep convolutional neural networks for sentiment analysis of short texts. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 69–78.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159.

Jenny Rose Finkel and Christopher D Manning. 2009. Joint parsing and named entity recognition. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 326–334.

Jiang Guo, Wanxiang Che, Haifeng Wang, and Ting Liu. 2014. Revisiting embedding features for simple semi-supervised learning. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 110–120.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177.

Niklas Jakob and Iryna Gurevych. 2010. Extracting opinion targets in a single-and cross-domain setting with conditional random fields. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1035–1045.

Long Jiang, Mo Yu, Ming Zhou, Xiaohua Liu, and Tiejun Zhao. 2011. Target-dependent twitter sentiment classification. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 151–160.

Wei Jin, Hung Hay Ho, and Rohini K Srihari. 2009. A novel lexicalized hmm-based learning framework for web opinion mining. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 465–472.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 655–665.

Qi Li and Heng Ji. 2014. Incremental joint extraction of entity mentions and relations. In *Proceedings of the Association for Computational Linguistics*.

Fangtao Li, Chao Han, Minlie Huang, Xiaoyan Zhu, Ying-Ju Xia, Shu Zhang, and Hao Yu. 2010. Structure-aware review mining and summarization. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 653–661.

Ji Ma, Yue Zhang, and Jingbo Zhu. 2014. Tagging the web: Building a robust web tagger with neural network. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 144–154.

Margaret Mitchell, Jacqui Aguilar, Theresa Wilson, and Benjamin Van Durme. 2013. Open domain targeted sentiment. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1643–1654.

Hwee Tou Ng and Jin Kiat Low. 2004. Chinese part-of-speech tagging: One-at-a-time or all-at-once? word-based or character-based? In *EMNLP*, pages 277–284.

Jian Peng, Liefeng Bo, and Jinbo Xu. 2009. Conditional neural fields. In *Advances in neural information processing systems*, pages 1419–1427.

Ana-Maria Popescu and Orena Etzioni. 2007. Extracting product features and opinions from reviews. In *Natural language processing and text mining*, pages 9–28. Springer.

Richard Socher, Jeffrey Pennington, Eric H Huang, Andrew Y Ng, and Christopher D Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 151–161.

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, volume 1631, page 1642.

Joseph Turian, Lev-Arie Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 384–394.

Duy-Tin Vo and Yue Zhang. 2015. Target-dependent twitter sentiment classification with rich automatic features. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI 2015)*, pages 1347–1353.

Mengqiu Wang and Christopher D. Manning. 2013a. Effect of non-linear deep architecture in sequence labeling. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 1285–1291.

Sida Wang and Christopher Manning. 2013b. Fast dropout training. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 118–126.

Yang Wang and Greg Mori. 2009. Max-margin hidden conditional random fields for human action recognition. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 872–879.

Xiaolong Wang, Furu Wei, Xiaohua Liu, Ming Zhou, and Ming Zhang. 2011. Topic sentiment analysis in twitter: a graph-based hashtag sentiment classification approach. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 1031–1040.

Janyce Wiebe, Theresa Wilson, and Claire Cardie. 2005. Annotating expressions of opinions and emotions in language. *Language resources and evaluation*, 39(2-3):165–210.

Bishan Yang and Claire Cardie. 2013. Joint inference for fine-grained opinion extraction. In *ACL (1)*, pages 1640–1649.

Jeonghee Yi, Tetsuya Nasukawa, Razvan Bunescu, and Wayne Niblack. 2003. Sentiment analyzer: Extracting sentiments about a given topic using natural language processing techniques. In *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*, pages 427–434.

Mo Yu, Tiejun Zhao, Daxiang Dong, Hao Tian, and Dianhai Yu. 2013. Compound embedding features for semi-supervised learning. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 563–568.

Yue Zhang and Stephen Clark. 2008. Joint word segmentation and POS tagging using a single perceptron. In *Proceedings of ACL-08: HLT*, pages 888–896.

Shusen Zhou, Qingcai Chen, Xiaolong Wang, and Xiaoling Li. 2014. Hybrid deep belief networks for semi-supervised sentiment classification. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 1341–1349.

Extracting Condition-Opinion Relations Toward Fine-grained Opinion Mining

Yuki Nakayama **Atsushi Fujii**
Department of Computer Science
Graduate School of Information Science and Engineering
Tokyo Institute of Technology
{nakayama.y.aj@m,fujii@cs}.titech.ac.jp

Abstract

A fundamental issue in opinion mining is to search a corpus for opinion units, each of which typically comprises the evaluation by an author for a target object from an aspect, such as "This hotel is in a good location". However, few attempts have been made to address cases where the validity of an evaluation is restricted on a condition in the source text, such as "for traveling with small kids". In this paper, we propose a method to extract condition-opinion relations from online reviews, which enables fine-grained analysis for the utility of target objects depending the user attribute, purpose, and situation. Our method uses supervised machine learning to identify sequences of words or phrases that comprise conditions for opinions. We propose several features associated with lexical and syntactic information, and show their effectiveness experimentally.

1 Introduction

Reflecting the rapid growth in the use of opinionated texts on the Web, such as comments on news articles and customer reviews, opinion mining has been explored to facilitate utilizing opinions mainly for improving products and decision-making purposes. While in a broad sense opinion mining refers to a process to discover useful knowledge latent in a corpus of opinionated texts, fundamental issues involve modeling an unit of opinions and searching the corpus for those units, each of which typically comprises the evaluation by an author for a target object from an aspect. Other elements, such as when the opinion was submitted, can optionally be included in an opinion unit. We take the following review sentence as an example opinionated description.

(1) I think hotel A offers a reasonable price if you take a family trip with small kids.

From the above example, existing methods (Pang and Lee, 2008; Seki et al., 2009; Jin et al., 2009; Zhao et al., 2010; He et al., 2011; Liu and Zhang, 2012; Liu et al., 2013; Yang and Cardie, 2013; Liu et al., 2014) are intended to extract the following quintuple as an opinion unit.

> Target = "hotel A", Aspect = "price", Evaluation (Polarity) = "reasonable" (positive), Holder = "I (author)", Time = N/A

Depending on the application, "Evaluation" can be any of a literal opinion word (e.g., "reasonable"), a polarity (positive/negative), or a value for multi-point scale rating.

Given those standardized units extracted from a corpus, it is feasible to overview the distribution of values for each element or a combination of elements. For example, those who intend to improve the quality of hotel A may investigate representative values for "Aspect" in the units satisfying "Target=hotel A & Polarity=negative", while those who look for accommodation may collect the opinion units for one or more candidate hotels and investigate the distribution of values for "Polarity" on an aspect-by-aspect basis.

However, in the above example (1), the evaluation for hotel A ("a reasonable price") is valid for "if you take a family trip with small kids", and thus it is not clear whether this evaluation is valid irrespective of the condition. For example, the price may not be reasonable for a single customer intending for business purposes. In this paper, we shall call such a condition "condition for opinion (CFO)". We define CFO as a condition for which an opinion unit has a polarity.

The existing methods for opinion mining, which do not consider whether a target opinion is conditional, potentially overestimate or underestimate

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 622–631,
Lisbon, Portugal, 17-21 September 2015.

the utility of hotel A and consequently decrease the quality of opinion mining. We manually analyzed the first 7 000 sentences in the Rakuten Travel data, which consists of 348 564 Japanese reviews for hotels in Japan (see Section 4 for details of this data) and found that 2 272 sentences are opinions, of which 630 opinions are conditional and thus the result for an existing method includes up to 28% (630/2272) errors.

Motivated by the above discussion, in this paper we propose a method to extract pairs of a CFO and its corresponding opinion unit from online reviews. This method provides two solutions to the above problem. First, a passive solution is detecting whether an opinion includes a CFO and, if any, isolating that opinion from the target of opinion mining. As a result, we can avoid potential errors as much as possible but the coverage is decreased.

Second, an active solution is identifying the span of each CFO in conditional opinions and classify them according to semantic categories, such as purpose, situation, and user attribute so that finer-grained opinion mining can be realized. For example, the distribution of positive and negative opinions can be available on a category-by-category basis. However, in this paper we focus only on the identification for CFOs and leave the semantic classification future work.

To produce a practical model for CFOs, it is important to investigate them from a grammar point of view. It can easily be predicted that a typical grammatical unit for CFOs is a conditional clause as in example (1). Additionally, restrictive modifiers in general can potentially be CFOs because they restrict the validity of an opinion unit from a specific perspective. A restrictive modifier comprises a word, phrase, or clause. The CFO in example (1), which is a dependent clause functioning as a condition, is also a restrictive modifier.

Example (2), which has the same meaning as example (1), includes a CFO as a prepositional phrase.

(2) Hotel A offers a *reasonable* price **for taking a family trip with small kids**.

We denote CFOs and opinion words in bold and italic faces, respectively. Examples (3) and (4) also include a CFO as a prepositional phrase. Unlike example (2), the validity of "reasonable" is restricted from time and comparison points of view, respectively.

(3) Hotel A offers a *reasonable* price **during this holiday season**.

(4) Hotel A offers a *reasonable* price **for a four star hotel**.

In example (5), which has a similar meaning to example (1), the CFO is a dependent clause functioning as a reason.

(5) Hotel A offers a *reasonable* price **because we take a family trip with small kids**.

Finally, as in example (6), an opinion holder can also be a CFO because the evaluation is restricted from a perspective of that specific person.

(6) **My mother** regarded hotel A as a *reasonable* choice.

If the restriction by a CFO is associated with a user-related perspective, we call such CFOs "user-restrictive CFOs (U-CFOs)". In other words, target users to whom an opinion unit is relevant are restricted by its corresponding U-CFO, although those users may agree or disagree with the opinion. The CFOs in examples (1), (2), and (5) are U-CFOs because the target users are mainly those who intend to travel with their children.

The CFO in example (3) is also U-CFO because the target users are those who intend to travel during a specific holiday season. The CFO in example (6) is also U-CFO because the opinion holder ("my mother") implies the opinion is relevant mainly to adult females. However, opinion holders who do not represent user-related perspectives, such as "I" without any profile, are not U-CFOs.

The CFO in example (4) is not a U-CFO because the relevance of the opinion is not restricted to specific customers. It may be argued that in example (4) the target users are restricted to those who are interested in the price. However, in example (4) the price restricts the aspect of the opinion unit, and should not be confused with U-CFOs and even CFOs, which restrict the validity of the opinion unit.

If we fully utilize U-CFOs, as discussed for the active solution above, we need to classify U-CFOs into semantic categories so that users can selectively read relevant opinions. In other words, the identification for U-CFOs facilitates predicting the review helpfulness (O'Mahony and Smyth, 2010; Moghaddam et al., 2012). Candidate categories

include demographic and psychographic attributes for target users (e.g., age and hobby) and situations of target users (e.g., purpose, time, and place). However, we leave the classification for U-CFOs future work.

2 Related work

As described in Section 1, the fundamental methods for opinion mining include opinion extraction, which identifies elements for opinion units (i.e., target, aspect, evaluation, holder, and time) (He et al., 2011; Jin et al., 2009; Liu et al., 2013; Seki et al., 2009; Yang and Cardie, 2013; Zhao et al., 2010), and opinion classification, which determines the non-literal evaluation of each opinion unit based on bipolar categories (i.e., positive and negative) (He et al., 2011; Meng et al., 2012) or multipoint scale categories (Fu and Wang, 2010; Moghaddam and Ester, 2013). However, none of these methods intends to determine whether or not an opinion is conditional and to extract their condition.

Narayanan et al. (2009) proposed a method for sentiment classification targeting conditional sentences. Although a conditional opinion is a kind of conditional sentence, their research is fundamentally different from our research. Narayanan et al. (2009) targeted such a conditional sentence that comprises a single opinion as a whole, and intended to categorize its polarity into any of positive, negative, or neutral. Examples (7) and (8) are such conditional sentences associated with neutral and positive categories, respectively.

(7) Hotel A would not have survived if the price was not reasonable.

(8) If you are looking for a hotel with a reasonable price, stay at hotel A.

In example (7), although the subordinate clause includes the opinion word "reasonable", none of the subordinate clause, main clause, or entire sentence is an opinion. In example (8), the entire sentence is an unconditional opinion about the price for hotel A, but the main and subordinate clauses are not opinions independently. In contrast, the purpose of our research is to identify conditional opinions, in which the main and subordinate clauses are an opinion and its condition, respectively.

Kim and Hovy (2006) proposed a method to identify a reason for the evaluation in an opinion, such as "the service was terrible because the staff was rude". Although as discussed in Section 1 reasons can be CFOs, their purpose is to identify grounds that justify the evaluation and thus is different from our purpose.

As discussed in Section 1, our research is related to predicting the review helpfulness (O'Mahony and Smyth, 2010; Moghaddam et al., 2012). The method proposed by O'Mahony and Smyth (2010) determines the helpfulness of a product review independent of the user profile and thus cannot recommend reviews based on user-related attributes.

Moghaddam et al. (2012) used collaborative filtering to predict the review helpfulness. The evaluation by a target user for past reviews is used to model the user and predict the helpfulness for unread reviews, which results in different predictions depending on the user. An advantage of collaborative filtering is its applicability to items whose content is usually difficult to analyze, such as videos. However, this advantage is diluted in recommending review text, from which effective features for user modeling, such as U-CFOs, can be obtained by opinion mining.

3 Proposed method

The task in this paper is to extract condition-opinion relations from reviews in Japanese. Currently, we assume that an opinion unit and its corresponding CFO are in the same sentence, and thus perform the extraction on a sentence-by-sentence basis. Given a sentence in reviews, we first search for an opinion unit, and if found, we also search for its corresponding CFO. Because in the first process we rely on an existing method for the opinion extraction, in this paper we focus only on the extraction for CFOs.

As discussed in Section 1, because CFOs can be different grammatical units, their length and structure are not standardized. We model the extraction for CFOs as the BIO chunking, which labels each token in a sentence as being the beginning (B), inside (I), or outside (O) of a span of interest. We use "Other" to refer to "O" to avoid confusion between "O" and "0" (zero). To subdivide "B" and "I" into U-CFOs and other CFOs, we use suffixes "U" and "C", respectively, such as "BU" denoting the beginning of a U-CFO. We use "Cond" to refer to any of BU, IU, BC, or IC.

Because we use the same method for both U-

CFOs and other CFOs, the above distinction only increases the number of categories to which each token is classified. If the distinction of U-CFOs is not important, the above suffixes can be omitted.

We regard Japanese *bunsetsu* phrases, which consist of a content word and one or more postpositional particles, as tokens, and extract a sequence of a BU-phrase and one or more IU-phrases, or an independent BU-phrase as a condition. The same method is used for BC/IC-phrases. However, words and phrases in an opinion unit are classified into its corresponding element. For example, an aspect phrase is classified into the aspect category.

Given an input sequence of *bunsetsu* phrases, $\boldsymbol{x} = x_1 \dots x_n$, our task is to predict a sequence of labels, $\boldsymbol{y} = y_1 \dots y_n$, where $y_i \in \{BU, IU, BC, IC, Other, Target, Aspect, OpinionWord\}$. However, because an opinion unit in an input sentence has been identified in advance, the task is a quinary classification with respect to $y_i \in \{BU, IU, BC, IC, Other\}$. We use Conditional Random Fields (CRF) (Lafferty et al., 2001) to train a classifier for categorizing each *bunsetsu* phrase into any of the aforementioned five categories. We use a combination of unigram and bigram models and calculate the conditional probability, $p(\boldsymbol{y}|\boldsymbol{x})$, for linear-chain CRF by Equation (1).

$$p(\boldsymbol{y}|\boldsymbol{x}) = \frac{1}{Z_{\boldsymbol{x}}} \exp\Big(\sum_{i,k} \lambda_k \cdot f_k(y_i, \boldsymbol{x}) + \sum_{i,k} \mu_k \cdot g_k(y_{i-1}, y_i, \boldsymbol{x})\Big) \quad (1)$$

Here, $Z_{\boldsymbol{x}}$ denotes a normalization factor, and f_k and g_k denote feature functions for unigram and bigram models, respectively. Let $x_{i,v}$ denote a feature value for x_i. While in the unigram model y_i depends on either $x_{i-1,v}$ or $x_{i,v}$, in the bigram model y_i depends on either a combination of $x_{i,v}$ and y_{i-1} or that of $x_{i-1,v}$ and y_{i-1}. Feature functions are produced for any possible combinations of the values for the variables used ($x_{i,v}$, y_{i-1}, and y_i in f_k), and take 1 if the corresponding combination appears and 0 otherwise. We use the four combinations "unigram $x_{i,v}$", "unigram $x_{i-1,v}$", "bigram y_{i-1} $x_{i,v}$", and "bigram y_{i-1} $x_{i-1,v}$" for feature functions.

The question here is how CFOs and U-CFOs can be modeled and what kind of features are needed. We assume characteristics of CFOs, and U-CFOs and partially exemplify their validity using Figure 1, which depicts an example input sentence and information related to its constituent *bunsetsu* phrases. In the upper part of Figure 1, a rectangle and an arrow denote a bunsetstu phrase and a syntactic dependency between two phrases, respectively, and in each phrase we show Japanese words based on the Hepburn system and their English translations in parentheses.

CFOs are associated with the following characteristics.

(a) By definition, CFOs determine the validity of the evaluation in an opinion unit, and thus syntactically modify an opinion word. Consequently, CFOs usually do not modify other elements in an opinion unit, such as an aspect.

(b) Like a conjunction in a conditional clause in English, such as "if", a CFO in Japanese also includes a clue expression, which is usually a functional expression (Matsuyoshi et al., 2006) in the tail phrase, such as "ni wa ("for" in English)".

(c) The distribution for parts of speech as the head of CFOs is skewed and heads of CFOs are usually a noun or verb.

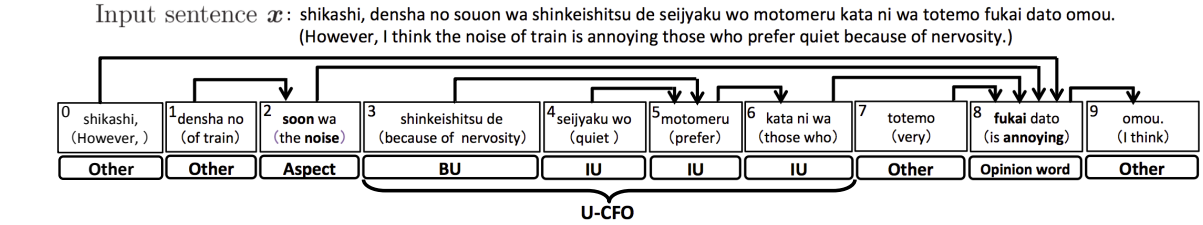
Additionally, U-CFOs are associated with the following characteristics.

(d) If a CFO is an opinion holder as in example (6) in Section 1, it is usually a U-CFO, which is the subject appearing at the beginning of a target sentence.

(e) By definition, U-CFOs include expressions related to user attributes, such as "nervosity" in Figure 1.

In this paper, we propose thirteen features to model CFOs and U-CFOs. In the bottom part of Figure 1, for each phrase we show the values of the thirteen features F1–F13 described below. These features were developed for the above five characteristics. F1–F5, F7–F10 and F13 are associated with (a), (b) and (c), respectively, while F6 and F11–F12 are associated with (d) and (e).

F1: Dependency distance to opinion word CFOs, which affect the evaluation in that opinion, usually syntactically modifies the opinion word. Thus, there should be a pass of dependencies between a Cond-phrase and the opinion word, and a



F1	1	2	aspect	3	3	2	1	1	opinion word	-1
F2	8	7	aspect	5	4	1	2	1	opinion word	-1
F3	1	0	aspect	1	1	1	1	1	opinion word	1
F4	2	1	aspect	-1	-1	-1	-1	-1	opinion word	-1
F5	7	5	aspect	2	1	1	1	0	opinion word	0
F6	0	0	aspect	0	0	0	0	0	opinion word	0
F7	nothing	nothing	aspect	nothing	nothing	nothing	ni wa	nothing	opinion word	nothing
F8	nothing	nothing	aspect	nothing	nothing	nothing	topic	nothing	opinion word	nothing
F9	nothing	nothing	aspect	ni wa	ni wa	ni wa	ni wa	nothing	opinion word	nothing
F10	nothing	nothing	aspect	topic	topic	topic	topic	nothing	opinion word	nothing
F11	nothing	densha	aspect	shinkeishitsu	nothing	nothing	kata	nothing	opinion word	nothing
F12	0	1	aspect	1	0	0	1	0	opinion word	0
F13	conjunction	noun	aspect	noun	noun	verb	noun	adverb	opinion word	verb

Figure 1: Example of Japanese sentence and the feature value for each constituent *bunsetsu* phrase

phrase that leads to the opinion word via a smaller number of dependency arrows is more likely to be a Cond-phrase. We use the dependency distance (i.e., the number of dependencies) between a phrase in question and the opinion word as the value for feature F1. The value for a phrase is -1 if there is no pass between that phrase and the opinion word. We use "CaboCha" (Kudo and Matsumoto, 2002) for dependency analysis purposes.

F2: Phrase distance to opinion word F1 is not robust against errors of the dependency analysis. To alleviate this problem, we approximate the dependency distance by a phrase distance. In practice, we subtract the ID for a phrase in question from that for the opinion word as the value for feature F2. If the opinion word consists of more than one phrase, we take the minimum difference. Because in Japanese a modifier is usually followed by its modifying object, a phrase with a negative value for feature F2 is usually an Other-phrase. For example, in the last phrase in Figure 1, which cannot be a modifier for the opinion word, is an Other-phrase.

F3: Dependency pass to aspect Because a CFO rarely modifies an aspect, for the value of feature F3 we take 0 if there is a pass of dependencies between a phrase in question and an aspect and 1 otherwise.

F4: Phrase distance to aspect Similar to F1, F3 is not robust against errors of the dependency analysis. As in F2, we approximate the value of F4 by a phrase distance between a phrase including an aspect and a phrase in question.

F5: Difference between values for F2 and F1 A CFO usually consists of a sequence of Cond-phrases where each phrase modifies the next phrase, as in Figure 1. There is a tendency that as the difference of values of F1 and F2 for a phrase becomes smaller, that phrase is more likely to be a Cond-phrase. In Figure 1, the values for Cond-phrases #3–#6 are smaller than those for Other-phrases #0–#1.

F6: Beginning of sentence The subject of an opinion sentence is often its U-CFO because the evaluation is valid only from the perspective of that specific subject. For example, in "my daughter was pleased with toys in the room" the positive evaluation is restricted by the daughter's perspective. Thus, the value of feature F6 takes 1 for the first phrase in a sentence excluding a conjunction, and 0 otherwise.

F7: Clue expression Because a CFO often ends with one or more specific particles or auxiliary verbs, we use the existence of those clue expressions in a phrase as the value for feature F7. We use words in a dictionary of Japanese functional expressions "Tsutsuji" (Matsuyoshi et al., 2006)

as the clue expressions. Table 1 shows examples of entries for Tsutsuji. Each entry is represented in a hierarchy structure with nine abstraction levels. We firstly collected "Head words" in the nineteen categories (e.g., resultative condition and purpose in L2) associated with our purpose, consulting "Meaning categories". Then we collected "Surface forms" corresponding to the collected head words and identified their corresponding surface forms to standardize different forms. For example, for ID 1 and ID 3 in Table 1, "to sure ba" and "nde" are regarded as identical to "to suru to" and "node", respectively. As a result, we collected 388 words, such as "ba (if)" and "ni (for)" and used their existence in a phrase in question as value for F7.

F8: Semantic categories for clue expression Because the data sparseness is a crucial problem for F7, we use the existence of semantic categories in Tsutsuji as the values of F8 for smoothing purposes. For example, in Table 1, "to suru to" and "ba" have the same feature values "resultative condition". If a clue expression belongs to more than one semantic category as in "ni" of Table 1, the feature value is a set of these categories.

F9: Dependency pass to phrase including clue expression (Surface form) As described in F7 above, the last phrase in a CFO often includes one or more clue expressions. In addition, a CFO often consists of more than one phrase. Given those conditions, a phrase that modifies a phrase containing a clue expression is also likely to be a Cond-phrase. We use the existence of a dependency pass between a phrase in question and a phrase containing a clue expression as the values of feature F9.

F10: Dependency pass to phrase including clue expression (Category) As with F8, we use the existence of semantic categories of Tsutsuji as the values of feature F10.

F11: Restrictive words We use the existence of words that are strongly associated with U-CFO as the value for F11. We call such words restrictive words. We automatically produced a dictionary of restrictive words from advertising slogans for hotels, which often include descriptions for target users, such as "★joshikai ya kappuru ni osusume!!★ (Recommended to girls get-together and couples)". First, we extracted words in the advertising slogan based on the following steps.

Entry ID	Abstraction levels			
	L1: Head word	L2: Meaning categories	...	L9: Surface forms
1	to suru to	resultative condition	...	to sure ba
2	ba		...	ba
3	node	reason	...	nde
4	ni	purpose	...	ni
5		target	...	

Table 1: Example entries for Tsutsuji

Step 1: Extracting sentences that match to a regular expression "(| hito | mono | kata) ni (| wa | mo) osusume" (i.e., "recommended to" or "recommend to those who").

Step 2: Collecting a sequence of content words for each bunsetsu-phrase in the extracted sentences.

For the above advertising slogan, we can collect two restrictive words "joshikai (girls get-together)" and "kappuru (couple)" by performing those 2 steps.

Second, we collected a sequence of independent words for bunsetsu phrases which comprises U-CFO in an annotated corpus. We combined the extracted words from the advertising slogans an annotated corpora, discarded redundancy, and standardized similar words, such as "kanko suru (do sightseeing) and "kanko (sightseeing)". As a result, we collected 934 words.

Finally, we calculated a mutual information like score, $Score(r, u)$, between a restrictive word r and labels u, Cond-phrases for U-CFOs (i.e., phrases labeled with either of BU or IU), by Equation 2.

$$Score(r, u) = P(r, u) \log \frac{P(r, u)}{P(r)P(u)} \quad (2)$$

$P(r, u)$ denotes the probability that a phrase including r is labeled with BU or IU in the annotated corpus. $P(r)$ denotes the probability that a phrase including r appears in the annotated corpus while $P(u)$ denotes the probability that a phrase labeled with BU or IU in the annotated corpus. If a phrase includes a restrictive word r and $Score(r, u)$ is greater than threshold θ, the feature value is r, and "nothing" otherwise.

F12: Existence of restrictive word Because the data sparseness is a crucial problem for F11, we integrate all the restrictive words for F11 into a single category for smoothing purposes. The value for F12 is the existence (1/0) of restrictive words.

F13: Part of speech for head The likelihood that a phrase in question is a Cond-phrases partially depends on the part of speech for the head in that phrase. For example, in Figure 1, a phrase whose head is a noun or verb tends to be a Cond-phrase

4 Experiments

To evaluate the effectiveness of our method, we used the Rakuten Travel data[1], which consists of 348 564 Japanese reviews for hotels in Japan. From this dataset, we selected 580 reviews and manually identified elements for opinion units. We removed sentences consisting only of opinion unit such as "The location is good" from the evaluation. As a result, 3 155 sentences remained, which comprise our corpus. To evaluate the effectiveness for identifying CFOs, we used the manually annotated opinion elements as output of a pseudo automatic method.

Given the above corpus, two annotators independently identified U-CFOs or CFOs, if any, for each opinion unit. For both annotations of CFOs and U-CFOs, the Kappa value for the inter-annotator agreement was 0.87, indicating strong agreement. We show the details of our corpus in Table 2. Using this corpus, we performed 10-fold cross-validation and compared different methods from different perspectives. Also, we determined the threshold for Score (see Eq 2) by a development set for each fold.

To evaluate the effectiveness of extracting U-CFOs and CFOs independently, we first classified bunsetsu phrases into any of BU, IU, BC, IC, or Other. Then, for the U-CFO extraction we regarded phrases for BU and IU as the Cond-phrases while for the CFO extraction we regarded phrases for BU, IU, BC, and IC as the Cond-phrases.

We used "Partial match" and "Exact match", which denote different criteria for the correctness of methods under evaluation. While in the partial match each method was requested to only detect whether or not a test sentence includes CFO, in the exact match each method was also requested to identify the span of each CFO. Also, we used different evaluation measures, namely precision (P), recall (R), F-measure (F), and accuracy (A).

Rule-based method and SVM-based method are used for comparison purposes. Rule-based method first identifies a *bunsetsu* phrase whose dependency distance to the opinion word is 1 and including a clue expression (see Section 3), and also identifies a sequence of the phrases from which there is a dependency path to the above phrase as a CFO. For example, in Figure 1 because phrase #6 includes a clue expression, a sequence of phrases #3–#6 is extracted as a CFO. These rules are based on features F1, F7 and F9. For the U-CFO extraction task, we regarded a sequence of Cond-phrases extracted by the above method as U-CFO if that sequence includes a restrictive word. For SVM, the thirteen features F1–F13 proposed in Section 3 was used. We used LIBSVM (Chang and Lin, 2011) to train a classifier. Our method used CRF to train a classifier with the thirteen features and four patterns for feature functions. We used CRF++[2] to train a classifier for each phrase and regularized the parameters using L2-norm.

Figure 2 shows the relationship between values of regularization parameter and F-measure for exact match. In Figure 2, "Rule", "SVM", and "CRF" denote a rule-based method, SVM-based method, and our method, respectively. The F-measure for Rule, independent of the regularization parameter, is a constant. While the F-measure for SVM substantially varied depending on the parameter value, that for CRF did not vary that much. Additionally, the F-measure for CRF was larger than that for SVM irrespective of the parameter value and matching criterion.

Table 3 shows results obtained with the optimal value for the regularization parameter. Looking at Table 3, one can see that CRF outperformed the other methods in terms of F-measure and accuracy for both partial and exact matches. We used the two-tailed paired t-test for statistical testing and found that the differences of CRF and each of the other methods in F-measure and accuracy were statistically significant at the 1% level irrespective of the configuration.

Figure 3 shows the effectiveness of the proposed features for exact match. The horizontal axis "w/o X" denotes a method without feature X. The vertical axis denotes a ratio of each method to our method. If a method without feature X takes less than 1 for value of vertical axis, the feature X is effective for extracting CFOs. Looking at Figure 3, one can see that our complete method outperformed any variation of our method in terms of

[1] https://alaginrc.nict.go.jp/resources/rakuten-dataset/rakuten-outline.html

[2] http://crfpp.googlecode.com/svn/trunk/doc/index.html

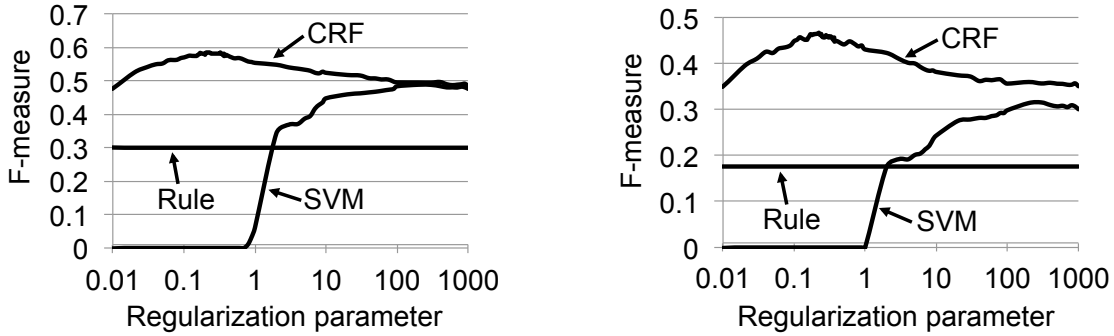
Opinion sentence	# w/ CFO	799
	# w/ U-CFO	526
	# w/o CFO	1 257
# Non opinion sentence		1 099
# Total		3 155

(a) Sentence unit

# BU-phrase		571
# IU-phrase		741
# BC-phrase		307
# IC-phrase		632
# Other-phrase		16 584
Opinion unit	# Opinion word	3 764
	# Aspect	3 406
	# Target	132
# Total		26 137

(b) Phrase unit

Table 2: Details of our corpus



(a) CFO extraction (b) U-CFO extraction

Figure 2: Relationship between values for regularization parameter and F-measure in exact match

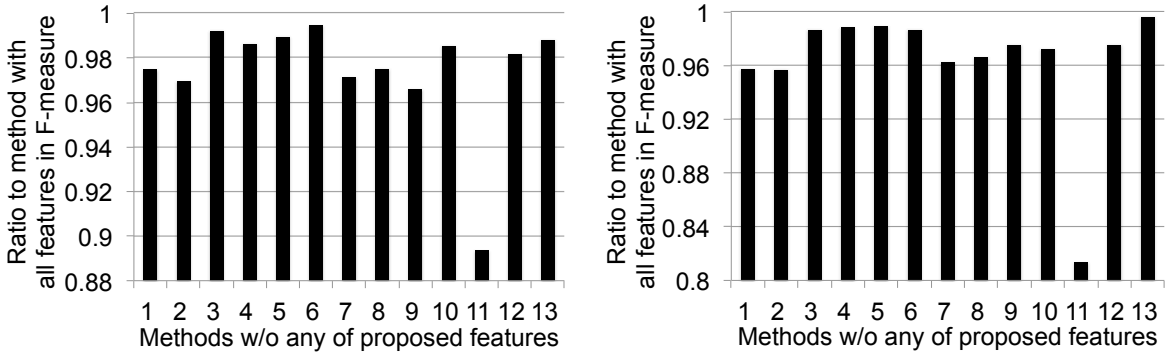
	Partial match				Exact match			
	P	R	F	A	P	R	F	A
Rule	.576	.790	.664	.771	.294	.319	.300	.649
SVM	.779	.842	.809	.885	.490	.500	.488	.789
CRF	.889	.758	.818	.915	.592	.580	.583	.865

(a) CFO extraction

	Partial match				Exact match			
	P	R	F	A	P	R	F	A
Rule	.395	.692	.502	.740	.174	.187	.176	.663
SVM	.622	.703	.659	.863	.319	.323	.315	.800
CRF	.818	.643	.720	.917	.472	.462	.464	.892

(b) U-CFO extraction

Table 3: Results for different configurations (P: precision, R: recall, F: F-measure, A: accuracy)



(a) CFO extraction (b) U-CFO extraction

Figure 3: Effectiveness of proposed features for exact match

F-measure. Thus, we conclude that each of our thirteen features was independently effective for extracting CFO and U-CFO in review sentences and that when used together the improvement was even greater.

For the U-CFO extraction, we analyzed the errors by our method. The total number of errors was 363 by condition unit. We describe causes of the errors with example sentences, translated into English by the authors. In those examples, double and single underlines denote false positive and false negative, respectively. For each cause, we show the number of errors in parentheses.

E1 (124) Errors were due to F11 and F12 with insufficient dictionary for restrictive words. Typically, low frequency words (e.g., pilgrimage) and words related to miscellaneous activities during a travel (e.g., charging a battery of a mobile phone) were not included in our dictionary. While it is important to increase the vocabulary size of our dictionary, identifying synonymous expressions with partial matching (e.g., go to sleep / go to bed) is also important.

E2 (53) Errors were due to dependency analysis, which often mistakenly recognizes sentence boundaries in an informal writing style and dependency relations in a sentence comprising a phrase, such as "the best location for fully enjoying Asakusa". In this example, CaboCha mistakenly associated the adnominal modifier "for fully enjoying Asakusa" with "location (aspect)" instead of "best (opinion word)". As a result, F1 and F3 did not regard this modifier as a U-CFO.

E3 (40) Restrictive modifiers that modify a non-opinion segment were mistakenly extracted as U-CFOs. For example, in "I used this hotel for business and the meal was good", "for business" includes the clue expression "for" but does not modifies the opinion unit.

E4 (39) Similar to E3 but errors were due to restrictive words instead of clue expressions. In the example for E3, the restrict word "business" caused the error.

E5 (26) U-CFOs that consist of a large number of phrases were often not extracted due to F5, such as "This hotel is acceptable for one night to take the train at the Chuo station next morning".

E6 (25) Errors were due to irrelevant entries in our restrictive word dictionary.

E7 (11) Due to the sparseness problem for restrictive words in the training data, U-CFOs and CFOs were not correctly distinguished.

E8 (9) Errors were due to part-of-speech tagging.

E9 (6) Errors were due to extracting modifiers consisting of a personal pronoun without additional user-related attributes, such as "enough for me" , as U-CFOs. We need to identify whether an expression for a person is associated with user-related attributes, such as "the bed is small for a person who is tall", which indicates a physical attribute of a user.

Additionally, there are 65 errors for which we have not found a reason.

5 Conclusion

Although a number of methods have been proposed to search an opinionated corpus for opinion units, few attempts have so far been made at addressing cases where the validity of an evaluation is restricted on a condition in the source text. We proposed a method to identify such conditions from sentences including opinion units. Our method performs sequence labeling to determine whether each phrase is a constituent of an condition for opinion. We proposed thirteen features associated with lexical and syntactic information of Japanese, and showed their effectiveness using reviews for hotels. The contributions of this paper are introducing the notion of conditions for opinions, which is language-independent, proposing a method to extract condition-opinion relations from opinionated corpora, and giving an insight into its potential applications in opinion mining.

Acknowledgments

We would like to thank Professor Takenobu Tokunaga (Tokyo Institute of Technology) for his valuable comments. This research was supported in part by Grant-in-Aid for Scientific Research (Grant No. 15H02747).

References

Chih-Chung Chang and Chih-Jen Lin. 2011. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2(3):27.

Guohong Fu and Xin Wang. 2010. Chinese sentence-level sentiment classification based on fuzzy sets. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 312–319.

Yulan He, Chenghua Lin, and Harith Alani. 2011. Automatically extracting polarity-bearing topics for cross-domain sentiment classification. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, pages 123–131.

Wei Jin, Hung Hay Ho, and Rohini K. Srihari. 2009. Opinionminer: A novel machine learning system for web opinion mining and extraction. In *Proceedings of the 15th ACM SIGKDD*, pages 1195–1204.

Soo-Min Kim and Eduard Hovy. 2006. Automatic identification of pro and con reasons in online reviews. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, pages 483–490.

Taku Kudo and Yuji Matsumoto. 2002. Japanese dependency analysis using cascaded chunking. In *Proceedings of the 6th Conference on Natural Language Learning*, pages 1–7.

John Lafferty, Andrew McCallum, and Fernando C.N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning*, pages 282–289.

Bing Liu and Lei Zhang. 2012. A survey of opinion mining and sentiment analysis. In C.C. Aggarwal and C.X.Zhai, editors, *Mining Text Data*, pages 415–463. Springer.

Kang Liu, Liheng Xu, and Jun Zhao. 2013. Syntactic patterns versus word alignment: Extracting opinion targets from online reviews. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 1754–1763.

Kang Liu, Liheng Xu, and Jun Zhao. 2014. Extracting opinion targets and opinion words from online reviews with graph co-ranking. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 314–324.

Suguru Matsuyoshi, Satoshi Sato, and Takehito Utsuro. 2006. Compilation of a dictionary of Japanese functional expressions with hierarchical organization. In Yuji Matsumoto, Richard Sproat, Kam-Fai Wong, and Min Zhang, editors, *Computer Processing of Oriental Languages. Beyond the Orient: The Research Challenges Ahead*, volume 4285 of *Lecture Notes in Computer Science*, pages 395–402. Springer Berlin Heidelberg.

Xinfan Meng, Furu Wei, Xiaohua Liu, Ming Zhou, Ge Xu, and Houfeng Wang. 2012. Cross-lingual mixture model for sentiment classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, pages 572–581.

Samaneh Moghaddam and Martin Ester. 2013. The FLDA model for aspect-based opinion mining: Addressing the cold start problem. In *Proceedings of the 22nd International Conference on World Wide Web*, pages 909–918.

Samaneh Moghaddam, Mohsen Jamali, and Martin Ester. 2012. ETF: Extended tensor factorization model for personalizing prediction of review helpfulness. In *Proceedings of the Fifth ACM International Conference on Web Search and Data Mining*, pages 163–172.

Ramanathan Narayanan, Bing Liu, and Alok Choudhary. 2009. Sentiment analysis of conditional sentences. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 180–189.

M.P. O'Mahony and B. Smyth. 2010. A classification-based review recommender. *Knowledge-Based Systems*, 23(4):323–329.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 2(1–2):1–135.

Yohei Seki, Noriko Kando, and Masaki Aono. 2009. Multilingual opinion holder identification using author and authority viewpoints. *Information Processing and Management*, 45(2):189–199.

Bishan Yang and Claire Cardie. 2013. Joint inference for fine-grained opinion extraction. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 1640–1649.

Wayne Xin Zhao, Jing Jiang, Hongfei Yan, and Xiaoming Li. 2010. Jointly modeling aspects and opinions with a MaxEnt-LDA hybrid. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 56–65.

A large annotated corpus for learning natural language inference

Samuel R. Bowman[*†]
sbowman@stanford.edu

Gabor Angeli[†‡]
angeli@stanford.edu

Christopher Potts[*]
cgpotts@stanford.edu

Christopher D. Manning[*†‡]
manning@stanford.edu

[*]Stanford Linguistics [†]Stanford NLP Group [‡]Stanford Computer Science

Abstract

Understanding entailment and contradiction is fundamental to understanding natural language, and inference about entailment and contradiction is a valuable testing ground for the development of semantic representations. However, machine learning research in this area has been dramatically limited by the lack of large-scale resources. To address this, we introduce the Stanford Natural Language Inference corpus, a new, freely available collection of labeled sentence pairs, written by humans doing a novel grounded task based on image captioning. At 570K pairs, it is two orders of magnitude larger than all other resources of its type. This increase in scale allows lexicalized classifiers to outperform some sophisticated existing entailment models, and it allows a neural network-based model to perform competitively on natural language inference benchmarks for the first time.

1 Introduction

The semantic concepts of entailment and contradiction are central to all aspects of natural language meaning (Katz, 1972; van Benthem, 2008), from the lexicon to the content of entire texts. Thus, *natural language inference* (NLI) — characterizing and using these relations in computational systems (Fyodorov et al., 2000; Condoravdi et al., 2003; Bos and Markert, 2005; Dagan et al., 2006; MacCartney and Manning, 2009) — is essential in tasks ranging from information retrieval to semantic parsing to commonsense reasoning.

NLI has been addressed using a variety of techniques, including those based on symbolic logic, knowledge bases, and neural networks. In recent years, it has become an important testing ground for approaches employing *distributed* word and phrase representations. Distributed representations excel at capturing relations based in similarity, and have proven effective at modeling simple dimensions of meaning like evaluative sentiment (e.g., Socher et al. 2013), but it is less clear that they can be trained to support the full range of logical and commonsense inferences required for NLI (Bowman et al., 2015; Weston et al., 2015b; Weston et al., 2015a). In a SemEval 2014 task aimed at evaluating distributed representations for NLI, the best-performing systems relied heavily on additional features and reasoning capabilities (Marelli et al., 2014a).

Our ultimate objective is to provide an empirical evaluation of learning-centered approaches to NLI, advancing the case for NLI as a tool for the evaluation of domain-general approaches to semantic representation. However, in our view, existing NLI corpora do not permit such an assessment. They are generally too small for training modern data-intensive, wide-coverage models, many contain sentences that were algorithmically generated, and they are often beset with indeterminacies of event and entity coreference that significantly impact annotation quality.

To address this, this paper introduces the Stanford Natural Language Inference (SNLI) corpus, a collection of sentence pairs labeled for entailment, contradiction, and semantic independence. At 570,152 sentence pairs, SNLI is two orders of magnitude larger than all other resources of its type. And, in contrast to many such resources, all of its sentences and labels were written by humans in a grounded, naturalistic context. In a separate validation phase, we collected four additional judgments for each label for 56,941 of the examples. Of these, 98% of cases emerge with a three-annotator consensus, and 58% see a unanimous consensus from all five annotators.

In this paper, we use this corpus to evaluate

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 632–642,
Lisbon, Portugal, 17-21 September 2015.

A man inspects the uniform of a figure in some East Asian country.	**contradiction** C C C C C	The man is sleeping
An older and younger man smiling.	**neutral** N N E N N	Two men are smiling and laughing at the cats playing on the floor.
A black race car starts up in front of a crowd of people.	**contradiction** C C C C C	A man is driving down a lonely road.
A soccer game with multiple males playing.	**entailment** E E E E E	Some men are playing a sport.
A smiling costumed woman is holding an umbrella.	**neutral** N N E C N	A happy woman in a fairy costume holds an umbrella.

Table 1: Randomly chosen examples from the development section of our new corpus, shown with both the selected gold labels and the full set of labels (abbreviated) from the individual annotators, including (in the first position) the label used by the initial author of the pair.

a variety of models for natural language inference, including rule-based systems, simple linear classifiers, and neural network-based models. We find that two models achieve comparable performance: a feature-rich classifier model and a neural network model centered around a Long Short-Term Memory network (LSTM; Hochreiter and Schmidhuber 1997). We further evaluate the LSTM model by taking advantage of its ready support for transfer learning, and show that it can be adapted to an existing NLI challenge task, yielding the best reported performance by a neural network model and approaching the overall state of the art.

2 A new corpus for NLI

To date, the primary sources of annotated NLI corpora have been the Recognizing Textual Entailment (RTE) challenge tasks.[1] These are generally high-quality, hand-labeled data sets, and they have stimulated innovative logical and statistical models of natural language reasoning, but their small size (fewer than a thousand examples each) limits their utility as a testbed for learned distributed representations. The data for the SemEval 2014 task called Sentences Involving Compositional Knowledge (SICK) is a step up in terms of size, but only to 4,500 training examples, and its partly automatic construction introduced some spurious patterns into the data (Marelli et al. 2014a, §6). The Denotation Graph entailment set (Young et al., 2014) contains millions of examples of entailments between sentences and artificially constructed short phrases, but it was labeled using fully automatic methods, and is noisy enough that it is probably suitable only as a source of supplementary training data. Outside the domain of sentence-level entailment, Levy et al. (2014) introduce a large corpus of semi-automatically annotated entailment examples between subject–verb–object relation triples, and the second release of the Paraphrase Database (Pavlick et al., 2015) includes automatically generated entailment annotations over a large corpus of pairs of words and short phrases.

Existing resources suffer from a subtler issue that impacts even projects using only human-provided annotations: indeterminacies of event and entity coreference lead to insurmountable indeterminacy concerning the correct semantic label (de Marneffe et al. 2008 §4.3; Marelli et al. 2014b). For an example of the pitfalls surrounding entity coreference, consider the sentence pair *A boat sank in the Pacific Ocean* and *A boat sank in the Atlantic Ocean*. The pair could be labeled as a contradiction if one assumes that the two sentences refer to the same single event, but could also be reasonably labeled as neutral if that assumption is not made. In order to ensure that our labeling scheme assigns a single correct label to every pair, we must select one of these approaches across the board, but both choices present problems. If we opt not to assume that events are coreferent, then we will only ever find contradictions between sentences that make broad universal assertions, but if we opt to assume coreference, new counterintuitive predictions emerge. For example, *Ruth Bader Ginsburg was appointed to the US Supreme Court* and *I had a sandwich for lunch today* would unintuitively be labeled as a contradiction, rather than neutral, under this assumption.

Entity coreference presents a similar kind of indeterminacy, as in the pair *A tourist visited New*

[1] http://aclweb.org/aclwiki/index.php?title=Textual_Entailment_Resource_Pool

York and *A tourist visited the city*. Assuming coreference between *New York* and *the city* justifies labeling the pair as an entailment, but without that assumption *the city* could be taken to refer to a specific unknown city, leaving the pair neutral. This kind of indeterminacy of label can be resolved only once the questions of coreference are resolved.

With SNLI, we sought to address the issues of size, quality, and indeterminacy. To do this, we employed a crowdsourcing framework with the following crucial innovations. First, the examples were grounded in specific scenarios, and the premise and hypothesis sentences in each example were constrained to describe that scenario from the same perspective, which helps greatly in controlling event and entity coreference.[2] Second, the prompt gave participants the freedom to produce entirely novel sentences within the task setting, which led to richer examples than we see with the more proscribed string-editing techniques of earlier approaches, without sacrificing consistency. Third, a subset of the resulting sentences were sent to a validation task aimed at providing a highly reliable set of annotations over the same data, and at identifying areas of inferential uncertainty.

2.1 Data collection

We used Amazon Mechanical Turk for data collection. In each individual task (each HIT), a worker was presented with premise scene descriptions from a pre-existing corpus, and asked to supply hypotheses for each of our three labels—*entailment*, *neutral*, and *contradiction*—forcing the data to be balanced among these classes.

The instructions that we provided to the workers are shown in Figure 1. Below the instructions were three fields for each of three requested sentences, corresponding to our *entailment*, *neutral*, and *contradiction* labels, a fourth field (marked optional) for reporting problems, and a link to an FAQ page. That FAQ grew over the course of data collection. It warned about disallowed techniques (e.g., reusing the same sentence for many different prompts, which we saw in a few cases), provided guidance concerning sentence length and

> We will show you the caption for a photo. We will not show you the photo. Using only the caption and what you know about the world:
>
> - Write one alternate caption that is **definitely** a **true** description of the photo. *Example: For the caption "Two dogs are running through a field." you could write "There are animals outdoors."*
> - Write one alternate caption that **might be** a **true** description of the photo. *Example: For the caption "Two dogs are running through a field." you could write "Some puppies are running to catch a stick."*
> - Write one alternate caption that is **definitely** a **false** description of the photo. *Example: For the caption "Two dogs are running through a field." you could write "The pets are sitting on a couch." This is different from the* maybe correct *category because it's impossible for the dogs to be both running and sitting.*

Figure 1: The instructions used on Mechanical Turk for data collection.

complexity (we did not enforce a minimum length, and we allowed bare NPs as well as full sentences), and reviewed logistical issues around payment timing. About 2,500 workers contributed.

For the premises, we used captions from the Flickr30k corpus (Young et al., 2014), a collection of approximately 160k captions (corresponding to about 30k images) collected in an earlier crowdsourced effort.[3] The captions were not authored by the photographers who took the source images, and they tend to contain relatively literal scene descriptions that are suited to our approach, rather than those typically associated with personal photographs (as in their example: *Our trip to the Olympic Peninsula*). In order to ensure that the label for each sentence pair can be recovered solely based on the available text, we did not use the images at all during corpus collection.

Table 2 reports some key statistics about the collected corpus, and Figure 2 shows the distributions of sentence lengths for both our source hypotheses and our newly collected premises. We observed that while premise sentences varied considerably in length, hypothesis sentences tended to be as

[2] Issues of coreference are not completely solved, but greatly mitigated. For example, with the premise sentence *A dog is lying in the grass*, a worker could safely assume that the dog is the most prominent thing in the photo, and very likely the only dog, and build contradicting sentences assuming reference to the same dog.

[3] We additionally include about 4k sentence pairs from a pilot study in which the premise sentences were instead drawn from the VisualGenome corpus (under construction; `visualgenome.org`). These examples appear only in the training set, and have pair identifiers prefixed with `vg` in our corpus.

Data set sizes:	
Training pairs	550,152
Development pairs	10,000
Test pairs	10,000
Sentence length:	
Premise mean token count	14.1
Hypothesis mean token count	8.3
Parser output:	
Premise 'S'-rooted parses	74.0%
Hypothesis 'S'-rooted parses	88.9%
Distinct words (ignoring case)	37,026

Table 2: Key statistics for the raw sentence pairs in SNLI. Since the two halves of each pair were collected separately, we report some statistics for both.

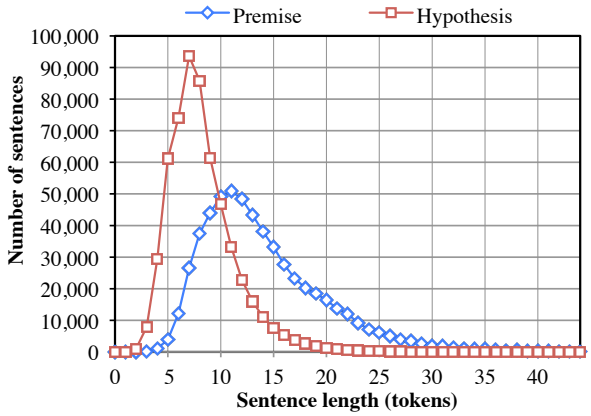


Figure 2: The distribution of sentence length.

short as possible while still providing enough information to yield a clear judgment, clustering at around seven words. We also observed that the bulk of the sentences from both sources were syntactically complete rather than fragments, and the frequency with which the parser produces a parse rooted with an 'S' (sentence) node attests to this.

2.2 Data validation

In order to measure the quality of our corpus, and in order to construct maximally useful testing and development sets, we performed an additional round of validation for about 10% of our data. This validation phase followed the same basic form as the Mechanical Turk labeling task used to label the SICK entailment data: we presented workers with pairs of sentences in batches of five, and asked them to choose a single label for each pair. We supplied each pair to four annotators, yielding five labels per pair including the label used by the original author. The instructions were similar to the instructions for initial data collection shown in Figure 1, and linked to a similar FAQ. Though we initially used a very restrictive qualification (based on past approval rate) to select workers for the validation task, we nonetheless discovered (and deleted) some instances of random guessing in an early batch of work, and subsequently instituted a fully closed qualification restricted to about 30 trusted workers.

For each pair that we validated, we assigned a gold label. If any one of the three labels was chosen by at least three of the five annotators, it was chosen as the gold label. If there was no such consensus, which occurred in about 2% of cases, we assigned the placeholder label '-'. While these unlabeled examples are included in the corpus distribution, they are unlikely to be helpful for the standard NLI classification task, and we do not include them in either training or evaluation in the experiments that we discuss in this paper.

The results of this validation process are summarized in Table 3. Nearly all of the examples received a majority label, indicating broad consensus about the nature of the data and categories. The gold-labeled examples are very nearly evenly distributed across the three labels. The Fleiss κ scores (computed over every example with a full five annotations) are likely to be conservative given our large and unevenly distributed pool of annotators, but they still provide insights about the levels of disagreement across the three semantic classes. This disagreement likely reflects not just the limitations of large crowdsourcing efforts but also the uncertainty inherent in naturalistic NLI. Regardless, the overall rate of agreement is extremely high, suggesting that the corpus is sufficiently high quality to pose a challenging but realistic machine learning task.

2.3 The distributed corpus

Table 1 shows a set of randomly chosen validated examples from the development set with their labels. Qualitatively, we find the data that we collected draws fairly extensively on commonsense knowledge, and that hypothesis and premise sentences often differ structurally in significant ways, suggesting that there is room for improvement beyond superficial word alignment models. We also find the sentences that we collected to be largely

General:	
Validated pairs	56,951
Pairs w/ unanimous gold label	58.3%
Individual annotator label agreement:	
Individual label = gold label	89.0%
Individual label = author's label	85.8%
Gold label/author's label agreement:	
Gold label = author's label	91.2%
Gold label ≠ author's label	6.8%
No gold label (no 3 labels match)	2.0%
Fleiss κ:	
contradiction	0.77
entailment	0.72
neutral	0.60
Overall	0.70

Table 3: Statistics for the validated pairs. The *author's label* is the label used by the worker who wrote the premise to create the sentence pair. A *gold label* reflects a consensus of three votes from among the author and the four annotators.

fluent, correctly spelled English, with a mix of full sentences and caption-style noun phrase fragments, though punctuation and capitalization are often omitted.

The corpus is available under a CreativeCommons Attribution-ShareAlike license, the same license used for the Flickr30k source captions. It can be downloaded at:
`nlp.stanford.edu/projects/snli/`

Partition We distribute the corpus with a prespecified train/test/development split. The test and development sets contain 10k examples each. Each original ImageFlickr caption occurs in only one of the three sets, and all of the examples in the test and development sets have been validated.

Parses The distributed corpus includes parses produced by the Stanford PCFG Parser 3.5.2 (Klein and Manning, 2003), trained on the standard training set as well as on the Brown Corpus (Francis and Kucera 1979), which we found to improve the parse quality of the descriptive sentences and noun phrases found in the descriptions.

3 Our data as a platform for evaluation

The most immediate application for our corpus is in developing models for the task of NLI. In particular, since it is dramatically larger than any existing corpus of comparable quality, we expect it to be suitable for training parameter-rich models like neural networks, which have not previously been competitive at this task. Our ability to evaluate standard classifier-base NLI models, however, was limited to those which were designed to scale to SNLI's size without modification, so a more complete comparison of approaches will have to wait for future work. In this section, we explore the performance of three classes of models which could scale readily: (i) models from a well-known NLI system, the Excitement Open Platform; (ii) variants of a strong but simple feature-based classifier model, which makes use of both unlexicalized and lexicalized features, and (iii) distributed representation models, including a baseline model and neural network sequence models.

System	SNLI	SICK	RTE-3
Edit Distance Based	71.9	65.4	61.9
Classifier Based	72.2	71.4	61.5
+ Lexical Resources	**75.0**	**78.8**	**63.6**

Table 4: 2-class test accuracy for two simple baseline systems included in the Excitement Open Platform, as well as SICK and RTE results for a model making use of more sophisticated lexical resources.

3.1 Excitement Open Platform models

The first class of models is from the Excitement Open Platform (EOP, Padó et al. 2014; Magnini et al. 2014)—an open source platform for RTE research. EOP is a tool for quickly developing NLI systems while sharing components such as common lexical resources and evaluation sets. We evaluate on two algorithms included in the distribution: a simple edit-distance based algorithm and a classifier-based algorithm, the latter both in a bare form and augmented with EOP's full suite of lexical resources.

Our initial goal was to better understand the difficulty of the task of classifying SNLI corpus inferences, rather than necessarily the performance of a state-of-the-art RTE system. We approached this by running the same system on several data sets: our own test set, the SICK test data, and the standard RTE-3 test set (Giampiccolo et al., 2007). We report results in Table 4. Each of the models

was separately trained on the training set of each corpus. All models are evaluated only on 2-class entailment. To convert 3-class problems like SICK and SNLI to this setting, all instances of *contradiction* and *unknown* are converted to nonentailment. This yields a most-frequent-class baseline accuracy of 66% on SNLI, and 71% on SICK. This is intended primarily to demonstrate the difficulty of the task, rather than necessarily the performance of a state-of-the-art RTE system. The edit distance algorithm tunes the weight of the three case-insensitive edit distance operations on the training set, after removing stop words. In addition to the base classifier-based system distributed with the platform, we train a variant which includes information from WordNet (Miller, 1995) and VerbOcean (Chklovski and Pantel, 2004), and makes use of features based on tree patterns and dependency tree skeletons (Wang and Neumann, 2007).

3.2 Lexicalized Classifier

Unlike the RTE datasets, SNLI's size supports approaches which make use of rich lexicalized features. We evaluate a simple lexicalized classifier to explore the ability of non-specialized models to exploit these features in lieu of more involved language understanding. Our classifier implements 6 feature types; 3 unlexicalized and 3 lexicalized:

1. The BLEU score of the hypothesis with respect to the premise, using an n-gram length between 1 and 4.
2. The length difference between the hypothesis and the premise, as a real-valued feature.
3. The overlap between words in the premise and hypothesis, both as an absolute count and a percentage of possible overlap, and both over all words and over just nouns, verbs, adjectives, and adverbs.
4. An indicator for every unigram and bigram in the hypothesis.
5. Cross-unigrams: for every pair of words across the premise and hypothesis which share a POS tag, an indicator feature over the two words.
6. Cross-bigrams: for every pair of bigrams across the premise and hypothesis which share a POS tag on the second word, an indicator feature over the two bigrams.

We report results in Table 5, along with ablation studies for removing the cross-bigram features (leaving only the cross-unigram feature) and for removing all lexicalized features. On our large corpus in particular, there is a substantial jump in accuracy from using lexicalized features, and another from using the very sparse cross-bigram features. The latter result suggests that there is value in letting the classifier automatically learn to recognize structures like explicit negations and adjective modification. A similar result was shown in Wang and Manning (2012) for bigram features in sentiment analysis.

System	SNLI		SICK	
	Train	Test	Train	Test
Lexicalized	99.7	**78.2**	90.4	**77.8**
Unigrams Only	93.1	71.6	88.1	77.0
Unlexicalized	49.4	50.4	69.9	69.6

Table 5: 3-class accuracy, training on either our data or SICK, including models lacking cross-bigram features (Feature 6), and lacking all lexical features (Features 4–6). We report results both on the test set and the training set to judge overfitting.

It is surprising that the classifier performs as well as it does without any notion of alignment or tree transformations. Although we expect that richer models would perform better, the results suggest that given enough data, cross bigrams with the noisy part-of-speech overlap constraint can produce an effective model.

3.3 Sentence embeddings and NLI

SNLI is suitably large and diverse to make it possible to train neural network models that produce distributed representations of sentence meaning. In this section, we compare the performance of three such models on the corpus. To focus specifically on the strengths of these models at producing informative sentence representations, we use sentence embedding as an intermediate step in the NLI classification task: each model must produce a vector representation of each of the two sentences without using any context from the other sentence, and the two resulting vectors are then passed to a neural network classifier which predicts the label for the pair. This choice allows us to focus on existing models for sentence embedding, and it allows us to evaluate the ability of those models to learn useful representations of meaning (which may be independently useful for subsequent tasks), at the cost of excluding from con-

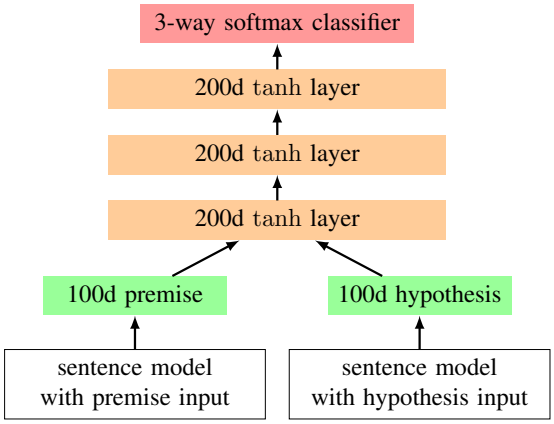


Figure 3: The neural network classification architecture: for each sentence embedding model evaluated in Tables 6 and 7, two identical copies of the model are run with the two sentences as input, and their outputs are used as the two 100d inputs shown here.

sideration possible strong neural models for NLI that directly compare the two inputs at the word or phrase level.

Our neural network classifier, depicted in Figure 3 (and based on a one-layer model in Bowman et al. 2015), is simply a stack of three 200d tanh layers, with the bottom layer taking the concatenated sentence representations as input and the top layer feeding a softmax classifier, all trained jointly with the sentence embedding model itself.

We test three sentence embedding models, each set to use 100d phrase and sentence embeddings. Our baseline sentence embedding model simply sums the embeddings of the words in each sentence. In addition, we experiment with two simple sequence embedding models: a plain RNN and an LSTM RNN (Hochreiter and Schmidhuber, 1997).

The word embeddings for all of the models are initialized with the 300d reference GloVe vectors (840B token version, Pennington et al. 2014) and fine-tuned as part of training. In addition, all of the models use an additional tanh neural network layer to map these 300d embeddings into the lower-dimensional phrase and sentence embedding space. All of the models are randomly initialized using standard techniques and trained using AdaDelta (Zeiler, 2012) minibatch SGD until performance on the development set stops improving. We applied L2 regularization to all models, manually tuning the strength coefficient λ for each, and additionally applied dropout (Srivastava et al., 2014) to the inputs and outputs of the sentence embedding models (though not to its internal connections) with a fixed dropout rate. All models were implemented in a common framework for this paper, and the implementations will be made available at publication time.

Sentence model	Train	Test
100d Sum of words	79.3	75.3
100d RNN	73.1	72.2
100d LSTM RNN	84.8	**77.6**

Table 6: Accuracy in 3-class classification on our training and test sets for each model.

The results are shown in Table 6. The sum of words model performed slightly worse than the fundamentally similar lexicalized classifier—while the sum of words model can use pretrained word embeddings to better handle rare words, it lacks even the rudimentary sensitivity to word order that the lexicalized model's bigram features provide. Of the two RNN models, the LSTM's more robust ability to learn long-term dependencies serves it well, giving it a substantial advantage over the plain RNN, and resulting in performance that is essentially equivalent to the lexicalized classifier on the test set (LSTM performance near the stopping iteration varies by up to 0.5% between evaluation steps). While the lexicalized model fits the training set almost perfectly, the gap between train and test set accuracy is relatively small for all three neural network models, suggesting that research into significantly higher capacity versions of these models would be productive.

3.4 Analysis and discussion

Figure 4 shows a learning curve for the LSTM and the lexicalized and unlexicalized feature-based models. It shows that the large size of the corpus is crucial to both the LSTM and the lexicalized model, and suggests that additional data would yield still better performance for both. In addition, though the LSTM and the lexicalized model show similar performance when trained on the current full corpus, the somewhat steeper slope for the LSTM hints that its ability to learn arbitrarily structured representations of sentence meaning may give it an advantage over the more constrained lexicalized model on still larger datasets.

We were struck by the speed with which the lexicalized classifier outperforms its unlexicalized

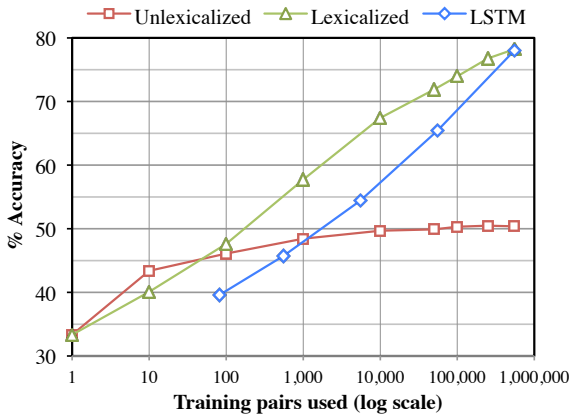


Figure 4: A learning curve showing how the baseline classifiers and the LSTM perform when trained to convergence on varied amounts of training data. The y-axis starts near a random-chance accuracy of 33%. The minibatch size of 64 that we used to tune the LSTM sets a lower bound on data for that model.

counterpart. With only 100 training examples, the cross-bigram classifier is already performing better. Empirically, we find that the top weighted features for the classifier trained on 100 examples tend to be high precision entailments; e.g., *playing* → *outside* (most scenes are outdoors), *a banana* → *person eating*. If relatively few spurious entailments get high weight—as it appears is the case—then it makes sense that, when these do fire, they boost accuracy in identifying entailments.

There are revealing patterns in the errors common to all the models considered here. Despite the large size of the training corpus and the distributional information captured by GloVe initialization, many lexical relationships are still misanalyzed, leading to incorrect predictions of *independent*, even for pairs that are common in the training corpus like *beach/surf* and *sprinter/runner*. Semantic mistakes at the phrasal level (e.g., predicting contradiction for *A male is placing an order in a deli/A man buying a sandwich at a deli*) indicate that additional attention to compositional semantics would pay off. However, many of the persistent problems run deeper, to inferences that depend on world knowledge and context-specific inferences, as in the entailment pair *A race car driver leaps from a burning car/A race car driver escaping danger*, for which both the lexicalized classifier and the LSTM predict *neutral*. In other cases, the models' attempts to shortcut this kind of inference through lexical cues can lead them astray. Some of these examples have qualities reminiscent of Winograd schemas (Winograd, 1972; Levesque, 2013). For example, all the models wrongly predict entailment for *A young girl throws sand toward the ocean/A girl can't stand the ocean*, presumably because of distributional associations between *throws* and *can't stand*.

Analysis of the models' predictions also yields insights into the extent to which they grapple with event and entity coreference. For the most part, the original image prompts contained a focal element that the caption writer identified with a syntactic subject, following information structuring conventions associating subjects and topics in English (Ward and Birner, 2004). Our annotators generally followed suit, writing sentences that, while structurally diverse, share topic/focus (theme/rheme) structure with their premises. This promotes a coherent, situation-specific construal of each sentence pair. This is information that our models can easily take advantage of, but it can lead them astray. For instance, all of them stumble with the amusingly simple case *A woman prepares ingredients for a bowl of soup/A soup bowl prepares a woman*, in which prior expectations about parallelism are not met. Another headline example of this type is *A man wearing padded arm protection is being bitten by a German shepherd dog/A man bit a dog*, which all the models wrongly diagnose as *entailment*, though the sentences report two very different stories. A model with access to explicit information about syntactic or semantic structure should perform better on cases like these.

4 Transfer learning with SICK

To the extent that successfully training a neural network model like our LSTM on SNLI forces that model to encode broadly accurate representations of English scene descriptions and to build an entailment classifier over those relations, we should expect it to be readily possible to adapt the trained model for use on other NLI tasks. In this section, we evaluate on the SICK entailment task using a simple transfer learning method (Pratt et al., 1991) and achieve competitive results.

To perform transfer, we take the parameters of the LSTM RNN model trained on SNLI and use them to initialize a new model, which is trained from that point only on the training portion of SICK. The only newly initialized parameters are

Training sets	Train	Test
Our data only	42.0	46.7
SICK only	100.0	71.3
Our data and SICK (transfer)	99.9	**80.8**

Table 7: LSTM 3-class accuracy on the SICK train and test sets under three training regimes.

softmax layer parameters and the embeddings for words that appear in SICK, but not in SNLI (which are populated with GloVe embeddings as above). We use the same model hyperparameters that were used to train the original model, with the exception of the L2 regularization strength, which is re-tuned. We additionally transfer the accumulators that are used by AdaDelta to set the learning rates. This lowers the starting learning rates, and is intended to ensure that the model does not learn too quickly in its first few epochs after transfer and destroy the knowledge accumulated in the pre-transfer phase of training.

The results are shown in Table 7. Training on SICK alone yields poor performance, and the model trained on SNLI fails when tested on SICK data, labeling more *neutral* examples as *contradiction*s than correctly, possibly as a result of subtle differences in how the labeling task was presented. In contrast, transferring SNLI representations to SICK yields the best performance yet reported for an unaugmented neural network model, surpasses the available EOP models, and approaches both the overall state of the art at 84.6% (Lai and Hockenmaier, 2014) and the 84% level of interannotator agreement, which likely represents an approximate performance ceiling. This suggests that the introduction of a large high-quality corpus makes it possible to train representation-learning models for sentence meaning that are competitive with the best hand-engineered models on inference tasks.

We attempted to apply this same transfer evaluation technique to the RTE-3 challenge, but found that the small training set (800 examples) did not allow the model to adapt to the unfamiliar genre of text used in that corpus, such that no training configuration yielded competitive performance. Further research on effective transfer learning on small data sets with neural models might facilitate improvements here.

5 Conclusion

Natural languages are powerful vehicles for reasoning, and nearly all questions about meaningfulness in language can be reduced to questions of entailment and contradiction in context. This suggests that NLI is an ideal testing ground for theories of semantic representation, and that training for NLI tasks can provide rich domain-general semantic representations. To date, however, it has not been possible to fully realize this potential due to the limited nature of existing NLI resources. This paper sought to remedy this with a new, large-scale, naturalistic corpus of sentence pairs labeled for entailment, contradiction, and independence. We used this corpus to evaluate a range of models, and found that both simple lexicalized models and neural network models perform well, and that the representations learned by a neural network model on our corpus can be used to dramatically improve performance on a standard challenge dataset. We hope that SNLI presents valuable training data and a challenging testbed for the continued application of machine learning to semantic representation.

Acknowledgments

We gratefully acknowledge support from a Google Faculty Research Award, a gift from Bloomberg L.P., the Defense Advanced Research Projects Agency (DARPA) Deep Exploration and Filtering of Text (DEFT) Program under Air Force Research Laboratory (AFRL) contract no. FA8750-13-2-0040, the National Science Foundation under grant no. IIS 1159679, and the Department of the Navy, Office of Naval Research, under grant no. N00014-10-1-0109. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of Google, Bloomberg L.P., DARPA, AFRL NSF, ONR, or the US government. We also thank our many excellent Mechanical Turk contributors.

References

Johan Bos and Katja Markert. 2005. Recognising textual entailment with logical inference. In *Proc. EMNLP*.

Samuel R. Bowman, Christopher Potts, and Christopher D. Manning. 2015. Recursive neural networks can learn logical semantics. In *Proc. of the 3rd Workshop on Continuous Vector Space Models and their Compositionality*.

Timothy Chklovski and Patrick Pantel. 2004. VerbOcean: Mining the web for fine-grained semantic verb relations. In *Proc. EMNLP*.

Cleo Condoravdi, Dick Crouch, Valeria de Paiva, Reinhard Stolle, and Daniel G. Bobrow. 2003. Entailment, intensionality and text understanding. In *Proc. of the HLT-NAACL 2003 Workshop on Text Meaning*.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The PASCAL recognising textual entailment challenge. In *Machine learning challenges. Evaluating predictive uncertainty, visual object classification, and recognising tectual entailment*, pages 177–190. Springer.

Marie-Catherine de Marneffe, Anna N. Rafferty, and Christopher D. Manning. 2008. Finding contradictions in text. In *Proc. ACL*.

W. Nelson Francis and Henry Kucera. 1979. Brown corpus manual. Brown University.

Yaroslav Fyodorov, Yoad Winter, and Nissim Francez. 2000. A natural logic inference system. In *Proc. of the 2nd Workshop on Inference in Computational Semantics*.

Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and Bill Dolan. 2007. The third PASCAL recognizing textual entailment challenge. In *Proc. of the ACL-PASCAL workshop on textual entailment and paraphrasing*.

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

Jerrold J. Katz. 1972. *Semantic Theory*. Harper & Row, New York.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proc. ACL*.

Alice Lai and Julia Hockenmaier. 2014. Illinois-LH: A denotational and distributional approach to semantics. In *Proc. SemEval*.

Hector J. Levesque. 2013. On our best behaviour. In *Proc. AAAI*.

Omer Levy, Ido Dagan, and Jacob Goldberger. 2014. Focused entailment graphs for open IE propositions. In *Proc. CoNLL*.

Bill MacCartney and Christopher D Manning. 2009. An extended model of natural logic. In *Proc. of the Eighth International Conference on Computational Semantics*.

Bernardo Magnini, Roberto Zanoli, Ido Dagan, Kathrin Eichler, Günter Neumann, Tae-Gil Noh, Sebastian Pado, Asher Stern, and Omer Levy. 2014. The Excitement Open Platform for textual inferences. *Proc. ACL*.

Marco Marelli, Luisa Bentivogli, Marco Baroni, Raffaella Bernardi, Stefano Menini, and Roberto Zamparelli. 2014a. SemEval-2014 task 1: Evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. In *Proc. SemEval*.

Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, and Roberto Zamparelli. 2014b. A SICK cure for the evaluation of compositional distributional semantic models. In *Proc. LREC*.

George A Miller. 1995. WordNet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Sebastian Padó, Tae-Gil Noh, Asher Stern, Rui Wang, and Roberto Zanoli. 2014. Design and realization of a modular architecture for textual entailment. *Journal of Natural Language Engineering*.

Ellie Pavlick, Johan Bos, Malvina Nissim, Charley Beller, Ben Van Durme, and Chris Callison-Burch. 2015. PPDB 2.0: Better paraphrase ranking, fine-grained entailment relations, word embeddings, and style classification. In *Proc. ACL*.

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

Lorien Y Pratt, Jack Mostow, Candace A Kamm, and Ace A Kamm. 1991. Direct transfer of learned information among neural networks. In *Proc. AAAI*.

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

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *JMLR*.

Johan van Benthem. 2008. A brief history of natural logic. In M. Chakraborty, B. Löwe, M. Nath Mitra, and S. Sarukki, editors, *Logic, Navya-Nyaya and Applications: Homage to Bimal Matilal*. College Publications.

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

Rui Wang and Günter Neumann. 2007. Recognizing textual entailment using sentence similarity based on dependency tree skeletons. In *ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*.

Gregory Ward and Betty Birner. 2004. Information structure and non-canonical syntax. In Laurence R. Horn and Gregory Ward, editors, *Handbook of Pragmatics*, pages 153–174. Blackwell, Oxford.

Jason Weston, Antoine Bordes, Sumit Chopra, and Tomas Mikolov. 2015a. Towards AI-complete question answering: A set of prerequisite toy tasks. arXiv:1502.05698.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2015b. Memory networks. In *Proc. ICLR*.

Terry Winograd. 1972. Understanding natural language. *Cognitive Psychology*, 3(1):1–191.

Peter Young, Alice Lai, Micah Hodosh, and Julia Hockenmaier. 2014. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *TACL*, 2:67–78.

Matthew D. Zeiler. 2012. ADADELTA: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Question-Answer Driven Semantic Role Labeling: Using Natural Language to Annotate Natural Language

Luheng He **Mike Lewis** **Luke Zettlemoyer**
Computer Science & Engineering
University of Washington
Seattle, WA
{luheng,mlewis,lsz}@cs.washington.edu

Abstract

This paper introduces the task of question-answer driven semantic role labeling (QA-SRL), where question-answer pairs are used to represent predicate-argument structure. For example, the verb "introduce" in the previous sentence would be labeled with the questions "What is introduced?", and "What introduces something?", each paired with the phrase from the sentence that gives the correct answer. Posing the problem this way allows the questions themselves to define the set of possible roles, without the need for predefined frame or thematic role ontologies. It also allows for scalable data collection by annotators with very little training and no linguistic expertise. We gather data in two domains, newswire text and Wikipedia articles, and introduce simple classifier-based models for predicting which questions to ask and what their answers should be. Our results show that non-expert annotators can produce high quality QA-SRL data, and also establish baseline performance levels for future work on this task.

1 Introduction

Semantic role labeling (SRL) is the widely studied challenge of recovering predicate-argument structure for natural language words, typically verbs. The goal is to determine "who does what to whom," "when," and "where," etc. (Palmer et al., 2010; Johansson and Nugues, 2008). However, this intuition is difficult to formalize and fundamental aspects of the task vary across efforts, for example FrameNet (Baker et al., 1998) models a large set of interpretable thematic roles (AGENT, PATIENT, etc.) while PropBank (Palmer et al., 2005) uses a small set of verb-specific roles (ARG0, ARG1, etc.). Existing task definitions can be complex and require significant linguistic expertise to understand,[1] causing challenges for data annotation and use in many target applications.

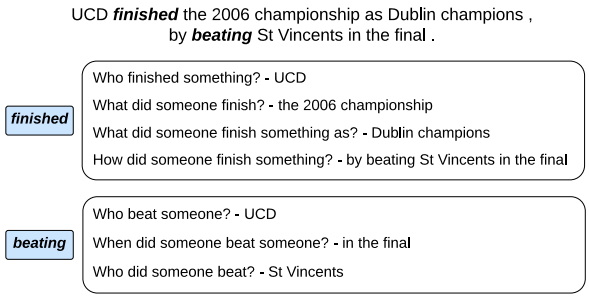


Figure 1: QA-SRL annotations for a Wikipedia sentence.

In this paper, we introduce a new question-answer driven SRL task formulation (QA-SRL), which uses question-answer pairs to label verbal predicate-argument structure. For example, for the sentence in Figure 1, we can ask a short question containing a verb, e.g. *"Who finished something?"*, and whose answer is a phrase from the original sentence, in this case *"UCD."* The answer tells us that *"UCD"* is an argument of "finished," while the question provides an indirect label on the role that *"UCD"* plays. Enumerating all such pairs, as we will see later, provides a relatively complete representation of the original verb's arguments and modifiers.

The QA-SRL task formulation has a number of advantages. It can be easily explained to non-expert annotators with a short tutorial and a few examples. Moreover, the formulation does not depend on any pre-defined inventory of semantic roles or frames, or build on any existing gram-

[1]The PropBank annotation guide is 89 pages (Bonial et al., 2010), and the FrameNet guide is 119 pages (Ruppenhofer et al., 2006). Our QA-driven annotation instructions are 5 pages.

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 643–653,
Lisbon, Portugal, 17-21 September 2015.

mar formalisms. Nonetheless, as we will show, it still represents the argument and modifier attachment decisions that have motivated previous SRL definitions, and which are of crucial importance for semantic understanding in a range of NLP tasks, such as machine translation (Liu and Gildea, 2010) and coreference resolution (Ponzetto and Strube, 2006). The annotations also, perhaps surprisingly, capture other implicit arguments that cannot be read directly off of the syntax, as was required for previous SRL approaches. For example, in *"It was his mother's birthday, so he was going to play her favorite tune"*, annotators created the QA pair *"When would someone play something? His mother's birthday"* which describes an implicit temporal relation. Finally, QA-SRL data can be easily examined, proofread, and improved by anyone who speaks the language and understands the sentence; we use natural language to label the structure of natural language.

We present a scalable approach for QA-SRL annotation and baseline models for predicting QA pairs. Given a sentence and target word (the verb), we ask annotators to provide as many question-answer pairs as possible, where the question comes from a templated space of wh-questions[2] and the answer is a phrase from the original sentence. This approach guides annotators to quickly construct high quality questions within a very large space of possibilities. Given a corpus of QA-SRL annotated sentences, we also train baseline classifiers for both predicting a set of questions to ask, and what their answers should be. The question generation aspect of QA-SRL is unique to our formulation, and corresponds roughly to identifying what semantic role labels are present in previous formulations of the task. For example, the question *"Who finished something"* in Figure 1 corresponds to the AGENT role in FrameNet. Table 1 also shows examples of similar correspondences for PropBank roles. Instead of pre-defining the labels, as done in previous work, the questions themselves define the set of possibilities.

Experiments demonstrate high quality data annotation with very little annotator training and establish baseline performance levels for the task. We hired non-expert, part-time annotators on Upwork (previously oDesk) to label over 3,000 sentences (nearly 8,000 verbs) across two domains (newswire and Wikipedia) at a cost of approximately $0.50 per verb. We show that the data is high quality, rivaling PropBank in many aspects including coverage, and easily gathered in non-newswire domains.[3] The baseline performance levels for question generation and answering reinforce the quality of the data and highlight the potential for future work on this task.

In summary, our contributions are:

- We introduce the task of question-answer driven semantic role labeling (QA-SRL), by using question-answer pairs to specify verbal arguments and the roles they play, without predefining an inventory of frames or semantic roles.
- We present a novel, lightweight template-based scheme (Section 3) that enables the high quality QA-SRL data annotation with very little training and no linguistic expertise.
- We define two new QA-SRL sub-tasks, question generation and answer identification, and present baseline learning approaches for both (Sections 4 and 5). The results demonstrate that our data is high-quality and supports the study of better learning algorithms.

2 Related Work

The success of syntactic annotation projects such as the Penn Treebank (Marcus et al., 1993) has led to numerous efforts to create semantic annotations for large corpora. The major distinguishing features of our approach are that it is not tied to any linguistic theory and that it can be annotated by non-experts with minimal training.

Existing SRL task formulations are closely related to our work. FrameNet (Baker et al., 1998) contains a detailed lexicon of verb senses and thematic roles. However, this complexity increases the difficulty of annotation. While the FrameNet project is decades old, the largest fully annotated corpus contains about 3,000 sentences (Chen et al., 2010). We were able to annotate over 3,000 sentences within weeks. PropBank (Kingsbury and Palmer, 2002), NomBank (Meyers et al., 2004) and OntoNotes (Hovy et al., 2006) circumvent the need for a large lexicon of roles, by defin-

[2] Questions starting with a wh-word, such as *who*, *what*, *when*, *how*, etc.

[3] Our hope is that this approach will generalize not only across different domains in English, as we show in this paper, but also to other languages. We will leave those explorations to future work.

Sentence	CoNLL-2009		QA-SRL	
(1) Stock-fund managers , meantime , went into October with less cash on hand than they **held** earlier this year .	A0	they	Who had held something?	Stock-fund managers / they
	AM-TMP	year	When had someone held something?	earlier this year
			What had someone held?	less cash on hand
			Where had someone held something?	on hand
(2) Mr. Spielvogel **added** pointedly : " The pressure on commissions did n't begin with Al Achenbaum . "	A0	Spielvogel	Who added something?	Mr. Spielvogel
	A1	did	What was added?	" The pressure on commissions did n't begin with Al Achenbaum . "
	AM-MNR	pointedly	How was something added?	pointedly
(3) He claimed losses totaling $ 42,455 – and the IRS **denied** them all .	A0	IRS	Who denied something?	IRS
	A1	them	What was denied?	losses / them
(4) The consumer - products and newsprint company said net **rose** to $ 108.8 million , or $ 1.35 a share , from $ 90.5 million , or $ 1.12 a share , a year ago .	A1	net	What rose?	net
	A3	$/ago	What did something rise from?	$ 90.5 million , or $ 1.12 a share
	A4	to	What did something rise to?	$ 108.8 million , or $ 1.35 a share
			When did something rise?	a year ago
(5) Mr. Agnew was vice president of the U.S. from 1969 until he **resigned** in 1973 .	A0	he	Who resigned from something?	Mr. Agnew
	AM-TMP	in	When did someone resign from something?	1973
			What did someone resign from?	vice president of the U.S.
(6) Mr. Gorbachev badly **needs** a diversion from the serious economic problems and ethnic unrest he faces at home .	A0	Gorbachev	Who needs something?	Gorbachev/he
	A1	diversion	What does someone need?	a diversion from the serious economic problems and ethnic unrest he faces at home
	AM-ADV	badly	How does someone need something?	badly
			What does someone need something from?	the serious economic problems and ethnic unrest he faces at home .
(7) Even a federal measure in June allowing houses to **add** research fees to their commissions did n't stop it .	A0	houses	What added something?	houses
	A1	fees	What was added?	research fees
	A2	to		
			When was something added?	June
(8) This year , Mr. Wathen says the firm will be able to **service** debt and still turn a modest profit .	A0	firm	Who will service something?	the firm
	A1	debt	What will be serviced?	debt
			When will something be serviced?	this year
(9) Clad in his trademark black velvet suit , the soft - spoken clarinetist announced that . . . and that it was his mother 's birthday , so he was going to **play** her favorite tune from the record .	A0	he	Who would play something?	the soft - spoken clarinetist / he
	A1	tune	What would be played?	her favorite tune from the record
			When would someone play something?	his mother 's birthday

Table 1: Comparison between CoNLL-2009 relations and QA-SRL annotations. While closely related to PropBank predicate-argument relations, QA pairs also contain information about within-sentence coreference (Ex 3, 5, 6, 9), implicit or inferred relations (Ex 4, 7, 8, 9) and roles that are not defined in PropBank (Ex 1, 5). Annotation mistakes are rare, but for example include missing pronouns (Ex 5) and prepositional attachment errors (Ex 6).

ing the core semantic roles in a predicate-specific manner. This means that frames need to be created for every verb, and it requires experts to distinguish between different senses and different roles.

Our work is also related to recent, more general semantic annotation efforts. Abstract Meaning Representation (Banarescu et al., 2013) can be viewed as an extension of PropBank with additional semantic information. Sentences take 8-13 minutes to annotate—which is slower than ours, but the annotations are more detailed. Universal Cognitive Conceptual Annotation (UCCA) (Abend and Rappoport, 2013) is an attempt to create a linguistically universal annotation scheme by using general labels such as *argument* or *scene*. The UCCA foundational layer does not distinguish semantic roles, so *Frogs eat herons* and *Herons eat frogs* will receive identical annotation — thereby discarding information which is potentially useful for translation or question answering. They report similar agreement with PropBank to our approach (roughly 90%), but annotator training time was an order-of-magnitude higher (30-40 hours). The Groningen Meaning Bank (Basile et al., 2012) project annotates text by manually correcting the output of existing semantic parsers. They show that some annotation can be crowdsourced using "games with a purpose" — however, this does not include its predicate-argument structure, which requires expert knowledge of their syntactic and semantic formalisms. Finally, Reisinger et al. (2015) study crowdsourcing semantic role labels based on Dowty's proto-roles, given gold predicate and argument mentions. This work directly complements our focus on labeling predicate-argument structure.

The idea of expressing the meaning of natural language in terms of natural language is related to natural logic (MacCartney and Manning, 2007), in which they use natural language for logical inference. Similarly, we model predicate-argument structure of a sentence with a set of question-

Field	Description	Example of Values	No. Values
WH*	Question words (wh-words)	who, what, when, where, why, how, how much	7
AUX	Auxiliary verbs	is, have, could, is n't	36
SBJ	Place-holding words for the subject position	someone, something	2
TRG*	Some form of the target word	built, building, been built	≈ 12
OBJ1	Place-holding words for the object position	someone, something	2
PP	Frequent prepositions (by, to, for, with, about) and prepositions (unigrams or bigrams) that occur in the sentence	to, for, from, by	≈ 10
OBJ2	Similar to OBJ1, but with more options	someone, something, do, do something, doing something	9

Table 2: Fields in our question annotation template, with descriptions, example values, and the total number of possible values for each. **WH*** and **TRG*** are required; all other fields can be left empty.

WH*	AUX	SBJ	TRG*	OBJ1	PP	OBJ2	
Who			built	something			?
What	had	someone	said				?
What	was	someone	expected		to	do	?
Where	might	something	rise		from		?

Table 3: Four example questions written with our question annotation template.

answer pairs. While existing work on natural logic has relied on small entailment datasets for training, our method allows practical large-scale annotation of training data.

Parser evaluation using textual entailment (Yuret et al., 2010) is a method for evaluating syntactic parsers based on entailment examples. In a similar spirit to our work, they abstract away from linguistic formalisms by using natural language inference. We focus on semantic rather than syntactic annotation, and introduce a scalable method for gathering data that allows both training and evaluation. Stern and Dagan (2014) applied textual entailment to recognize implicit predicate-argument structure that are not explicitly expressed in syntactic structure.

3 QA-based Semantic Dataset

This section describes our annotation process in more detail, and discusses agreement between our annotations and PropBank. Table 1 shows examples provided by non-expert annotators.[4]

3.1 Annotation Task Design

We annotate verbs with pairs of questions and answers that provide information about predicate-argument structure. Given a sentence s and a verbal predicate v in the sentence, annotators must produce a set of wh-questions that contain v and whose answers are phrases in s.

To speed annotation and simplify downstream processing, we define a small grammar over possible questions. The questions are constrained with a template with seven fields, $q \in$ **WH** $\times$ **AUX** $\times$ **SBJ** $\times$ **TRG** $\times$ **OBJ1** $\times$ **PP** $\times$ **OBJ2**, each associated with a list of possible options. Descriptions for each field are shown in Table 2. The grammar is sufficiently general to capture a wide-range of questions about predicate-argument structure—some examples are given in Table 3.

The precise form of the question template is a function of the verb v and sentence s, for two of the fields. For the **TRG** field, we generate a list of inflections forms of v using the Wiktionary dictionary. For the **PP** field, the candidates are all the prepositions that occurred in the sentence s, and some frequently-used prepositions - *by*, *to*, *for*, *with*, and *about*. We also include preposition bigrams (e.g., *out for*) from s.

Answers are constrained to be a subset of the words in the sentence but do not necessarily have to be contiguous spans. We also allow questions to have multiple answers, which is useful for annotating graph structured dependencies such as those in examples 3 and 6 in Table 1.

3.2 Data Preparation

We annotated over 3000 sentences (nearly 8,000 verbs) in total across two domains: newswire (PropBank) and Wikipedia. Table 4 shows the full data statistics. In the newswire domain, we sampled sentences from the English training data of CoNLL-2009 shared task (Hajič et al.,

[4] Our dataset is freely available at: `https://dada.cs.washington.edu/qasrl`.

Dataset	Sentences	Verbs	QAs
newswire-train	744	2020	4904
newswire-dev	249	664	1606
newswire-test	248	652	1599
Wikipedia-train	1174	2647	6414
Wikipedia-dev	392	895	2183
Wikipedia-test	393	898	2201

Table 4: Annotated data statistics.

2009), excluding questions and sentences with fewer than 10 words. For the Wikipedia domain, we randomly sampled sentences from the English Wikipedia, excluding questions and sentences with fewer than 10 or more than 60 words.

In each sentence, we need to first identify the candidates for verbal predicates. In principle, a separate stage of annotation could identify verbs—but for simplicity, we instead used POS-tags. We used gold POS-tags for newswire, and predicted POS-tags (using Stanford tagger (Toutanova et al., 2003)) in Wikipedia. Annotators can choose to skip a candidate verb if they are unable to write questions for it. Annotators skipped 136 verbs (3%) in Wikipedia data and 50 verbs (1.5%) in PropBank data.

3.3 Annotation Process

For annotation, we hired 10 part-time, non-expert annotators from Upwork (previously oDesk) and paid $10 per hour for their work. The average cost was $0.58 per verb ($1.57 per sentence) for newswire text and $0.45 per verb ($1.01 per sentence) on the Wikipedia domain. The annotators are given a short tutorial and a small set of sample annotations (about 10 sentences). Annotators were hired if they showed good understanding of English and our task. The entire screening process usually took less than 2 hours.

Writing QA pairs for each sentence takes 6 minutes on average for Wikipedia and 9 minutes on newswire, depending on the length and complexity of the sentence and the domain of the text.

3.4 Agreement with Gold PropBank Data (CoNLL-2009)

PropBank is the most widely used annotation of predicate-argument structure. While our annotation captures different information from PropBank, it is closely related. To investigate the similarity between the annotation schemes, we measured the overlap between the newswire domain (1241 sentences) of our QA-SRL dataset and the PropBank dataset.

	All Roles	Core	Adjuncts
Precision	81.4	85.9	59.9
Recall	86.3	89.8	63.6

Table 5: Agreement with gold PropBank (CoNLL-2009) for all roles, core roles, and adjuncts. Precision is the percentage of QA pairs covering exactly one PropBank relation. Recall is the percentage of PropBank relations covered by exactly one QA pair.

For each PropBank predicate that we have annotated with our scheme, we compute the agreement between the PropBank arguments and the QA-SRL answers. We ignore modality, reference, discourse and negation roles, as they are outside the scope of our current annotation. An annotated answer is judged to match the PropBank argument if either (1) the gold argument head is within the annotated answer span, or (2) the gold argument head is a preposition and at least one of its children is within the answer span.

We measure the macro-averaged precision and recall of our annotation against PropBank, with the proportion of our QA-pairs that are match a PropBank relation, and the proportion of PropBank relations covered by our annotation. The results are shown in Table 5, and demonstrate high overall agreement with PropBank. Agreement for core arguments [5] is especially strong, showing much of the expert linguist annotation in PropBank can be recovered with our simple scheme. Agreement for adjuncts is lower, because the annotated QAs often contain inferred roles, especially for *why*, *when* and *where* questions (See examples 4, 7 and 8 in Table 1). These inferred roles are typically correct, but outside of the scope of PropBank annotations; they point to exciting opportunities for future work with QA-SRL data. On the other hand, the adverbial arguments in PropBank are sometimes neglected by annotators, thus becoming a major source of recall loss.

Table 6 shows the overlap between our annotated question words and PropBank argument labels. There are many unsurprising correlations—*who* questions are strongly associated with Prop-

[5] In PropBank, A0-A5 are the core arguments. In QA-SRL, the core arguments include QA pairs with a question that starts with *Who* or *What*.

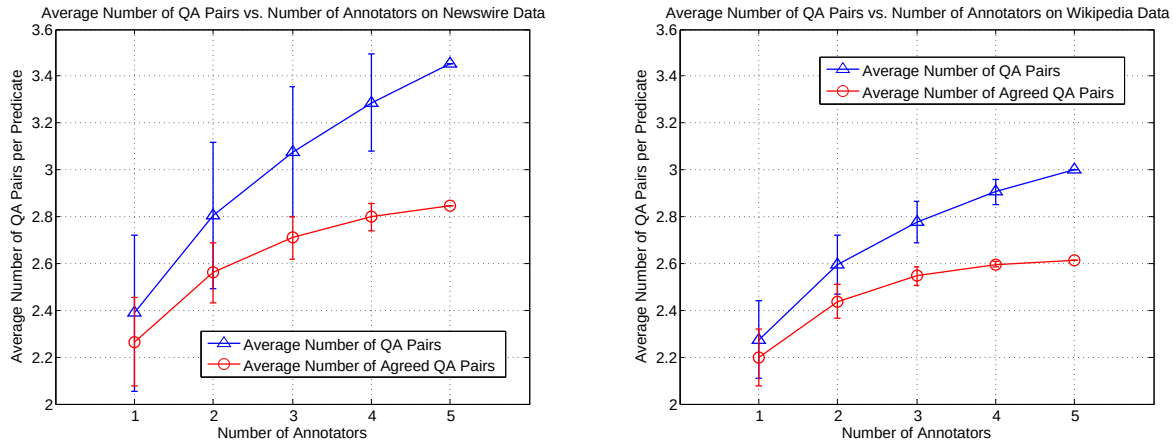


Figure 2: Inter-annotator agreement measured on 100 newswire sentences and 108 Wikipedia sentences, comparing the total number of annotators to the number of unique QA pairs produced and the number of agreed pairs. A pair is considered agreed if two or more annotators produced it.

	WHO	WHAT	WHEN	WHERE	WHY	HOW	HOW MUCH
A0	1575	414	3	5	17	28	2
A1	285	2481	4	25	20	23	95
A2	85	364	2	49	17	51	74
A3	11	62	7	8	4	16	31
A4	2	30	5	11	2	4	30
A5	0	0	0	1	0	2	0
ADV	5	44	9	2	25	27	6
CAU	0	3	1	0	23	1	0
DIR	0	6	1	13	0	4	0
EXT	0	4	0	0	0	5	5
LOC	1	35	10	89	0	13	11
MNR	5	47	2	8	4	108	14
PNC	2	21	0	1	39	7	2
PRD	1	1	0	0	0	1	0
TMP	2	51	341	2	11	20	10

Table 6: Co-occurrence of wh-words in QA-SRL annoations and role labels in PropBank.

Bank agents (A0), and *where* and *when* questions correspond to PropBank temporal and locative roles, respectively. Some types of questions are divided much more evenly among PropBank roles, such as *How much*. These cases show how our questions can produce a more easily interpretable annotation than PropBank labels, which are predicate-specific and can be difficult to understand without reference to the frame files.

Together, these results suggest that non-experts can annotate much of the information contained in PropBank, and produce a more easily interpretable annotation.

3.5 Inter-Annotator Agreement

To judge the reliability of the data, we measured agreement on a portion of the data (100 sentences in the newswire domain and 108 sentences in the Wikipedia domain) annotated by five annotators.

Measuring agreement is complicated by the fact that the same question can be asked in multiple ways—for example *"Who resigned?"* and *"Who resigned from something?"*—and annotators may choose different, although usually highly overlapping, answer spans. We consider two QA pairs to be equivalent if (1) they have the same wh-word and (2) they have overlapping answer spans. In this analysis, *Who* and *What* are considered to be the same wh-word.

Figure 2 shows how the number of different QA pairs (both overall and agreed) increases with number of annotations. A QA pair is considered to be agreed upon if it is proposed by at least two of the five annotators. After five annotators, the number of agreed QA pairs starts to asymptote. A single annotator finds roughly 80% of the agreed QA pairs that are found by five annotators, suggesting that high recall can be achieved with a single stage of annotation. To further improve precision, future work should explore a second stage of annotation where annotators check each other's work, for example by answering each other's questions.

4 Question Generation

Given a sentence s and a target verb v, we want to automatically generate a set of questions containing v that are answerable with phrases from s. This task is important because generating answerable questions requires understanding the predicate-argument structure of the sentence. In

essence, questions play the part of semantic roles in our approach.[6]

We present a baseline that breaks down question generation into two steps: (1) we first use a classifier to predict a set of roles for verb v that are likely present in the sentence, from a small, heuristically defined set of possibilities and then (2) generate one question for each predicted role, using templates extracted from the training set.

Mapping Question Fields to Semantic Roles To generate questions, we first have to decide the primary role we want to target; each question's answer is associated with a specific semantic role. For example, given the sentence *UCD finished the 2006 championship* and target verb *finished*, we could ask either: (Q1) *Who finished something?* or (Q2) *What did someone finish?*. Q1 targets the role associated with the person doing the finishing, while Q2 focuses on the thing being finished. To generate high quality questions, it is also often necessary to refer to roles other than the primary role, with pronouns. For example, Q2 uses "someone" to refer to the finisher.

Although it is difficult to know a priori the ideal set of possible roles, our baseline uses a simple discrete set, and introduces heuristics for identifying the roles a question refers to. The roles $\mathcal{R}$ include:

$$\begin{aligned}\mathcal{R} =& \{\text{R0}, \text{R1}, \text{R2}, \text{R2}[p], w, w[p]\}\\ w \in& \{\textbf{Where}, \textbf{When}, \textbf{Why}, \textbf{How}, \textbf{HowMuch}\}\\ p \in& \text{Prepositions}\end{aligned}$$

We then normalize the annotated questions by mapping its fields **WH**, **SBJ**, **OBJ1** and **OBJ2** to the roles $r \in \mathcal{R}$, using a small set of rules listed in Table 7. In our example, the **WH** field of the Q1 (*Who*) and the **SBJ** of Q2 (*someone*) are both mapped to role R0. The **WH** of Q2 (*What*) and the **OBJ1** of Q1 (*something*) are mapped to role R1. Some roles can be subclassed with prepositions. For example, the **WH** field of the question *What did something rise from?* is mapped to R2[from].

In most cases, R0 is related to the A0/AGENT roles in PropBank/FrameNet, and R1/R2 are related to A1/PATIENT roles. Since our questions are defined in a templated space, we are able to do this mapping heuristically with reasonable accuracy. In the future, we might try to induce the set of possible roles given each target verb, following the semantic role induction work of Titov and Klementiev (2012) and Lang and Lapata (2011), or use crowdsourcing to label proto-roles, following Reisinger et al. (2015).

[6]The task also has applications to semi-automatic annotation of sentences with our scheme, if we could generate questions with high enough recall and only require annotators to provide all the answers. We leave this important direction to future work.

$wh \in$ {Who, What} $\wedge$ voice = active		
WH $\rightarrow$ R0	WH $\rightarrow$ R1	WH $\rightarrow$ R2[p]
SBJ $= \phi$	SBJ $\rightarrow$ R0	SBJ $\rightarrow$ R0
OBJ1 $\rightarrow$ R1	OBJ1 $= \phi$	OBJ1 $\rightarrow$ R1
OBJ2 $\rightarrow$ R2[p]	OBJ2 $\rightarrow$ R2[p]	OBJ2 $= \phi$
$wh \in$ {Who, What} $\wedge$ voice = passive		
WH $\rightarrow$ R1	WH $\rightarrow$ R2	
SBJ $= \phi$	SBJ $\rightarrow$ R1	
OBJ1 $\rightarrow$ R2	OBJ1 $= \phi$	
OBJ2 $\rightarrow$ R2[p]	OBJ2 $\rightarrow$ R2[p]	
$wh \in$ {When, Where, Why, How, HowMuch} $\wedge$ voice = active		
WH $\rightarrow wh[p]$	WH $\rightarrow wh$	
SBJ $\rightarrow$ R0	SBJ $\rightarrow$ R0	
OBJ1 $\rightarrow$ R1	OBJ1 $\rightarrow$ R1	
OBJ2 $= \phi$	OBJ2 $\rightarrow$ R2[p]	
$wh \in$ {When, Where, Why, How, HowMuch} $\wedge$ voice = passive		
WH $\rightarrow wh[p]$	WH $\rightarrow wh$	
SBJ $\rightarrow$ R1	SBJ $\rightarrow$ R1	
OBJ1 $\rightarrow$ R2	OBJ1 $\rightarrow$ R2	
OBJ2 $= \phi$	OBJ2 $\rightarrow$ R2[p]	

Table 7: Mapping question fields to roles in $\mathcal{R}$. The mapping is based on whether certain question fields are empty and the voice of the verb in the question (active or passive). ϕ indicates that a field is either an empty string or equals "do/doing". If a question is in passive voice and contains the preposition "by", then **OBJ2** is tagged with R0 instead, as in *What is built by someone?*

Predicting Question Roles Given this space of possible roles, our first step in generation is to determine which roles are present in a sentence, and select the pronouns that could be used to refer to them in the resulting questions. We formulate this task as a supervised multi-label learning problem. We define the set of possible labels $\mathcal{L}$ by combining the roles in $\mathcal{R}$ with different pronoun values:

$$\begin{aligned}\mathcal{L} =& \{role{:}val \mid role \in \mathcal{R}\}\\ val \in& \{\phi, \text{someone}, \text{something}, \text{do something},\\ & \text{doing something}\}\end{aligned}$$

For example, to support the generation of the questions *Who finished something?* and *What did someone finish?*, we need to first predict the labels R0:someone and R1:something. Adjunct roles, such as *When* and *How*, always take an empty pronoun value.

Question	Abstract Question				
	WH	SBJ	Voice	OBJ1	OBJ2
Who finished something?	R0	/	active	R1	/
What did someone finish?	R1	R0	active	/	/

Table 8: Example surface realization templates from abstract questions.

For each sentence s and verb v, the set of positive training samples corresponds to the set of labels in the annotated questions, and the negative samples are all the other labels in $\mathcal{L}_{train}$, the subset of labels appeared in training data.[7] We train a binary classifier for every label in $\mathcal{L}_{train}$ using L2-regularized logistic regression by Liblinear (Fan et al., 2008), with hyper-parameter $C = 0.1$. Features of the binary classifiers are listed in Table 10. For each sentence s and verb v in the test data, we take the k highest-scoring labels, and generate questions from these.

Question Generation After predicting the set of labels for a verb, we generate a question to query each role. First, we define the concept of an *abstract question*, which provides a template that specifies the role to be queried, other roles to include in the question, and the voice of the verb. Abstract questions can be read directly from our training data.

We can map an abstract question to a surface realization by substituting the slots with the pronoun values of the predicted labels. Table 8 shows the abstract questions we could use to query roles R0 and R1; and the generated questions, based on the set of predicted labels $\{\text{R0:someone}, \text{R1:something}\}$.

Therefore, to generate a question to query a role $r \in \mathcal{R}$, we simply return the most frequent abstract question that occurred in training data that matches the role being queried, and the set of other predicted labels.

Experiments Native English speakers manually evaluated 500 automatically generated questions (5 questions per verb). Annotators judged whether the questions were grammatical [8] and answerable from the sentence.

We evaluated the top k questions produced by our baseline technique. The results in Table 9 show that our system is able to produce questions which are both grammatical and answerable. The average number of QA pairs per verb collected by human annotator is roughly 2.5, demonstrating significant room for improving these results.

[7] We pruned the negative samples that contain prepositions that are not in the sentence or in the set of frequently-used prepositions (by, to, for, with, about).

[8] Some automatically generated questions are ungrammatical because of label prediction errors, such as *Who sneezed someone?*, where the label R1:someone shouldn't be predicted.

	Newswire		Wikipedia	
	Ans.	Gram.	Ans.	Gram.
prec@1	66.0	84.0	72.0	90.0
prec@3	51.3	78.7	53.3	86.0
prec@5	38.4	77.2	40.0	82.0

Table 9: Manual evaluation results for question generation in two domains, including the averaged number of distinct questions that are answerable given the sentence (Ans.) and the averaged number of questions that are grammatical (Gram.).

5 Answer Identification

The goal of the answer identification task is to predict an answer a given sentence s, target verb v and a question q. Our annotated answers can be a series of spans, so the space of all possible answers is $2^{|s|}$. To simplify the problem, we transform our span-based answer annotation to answer head words, thus reducing the answer space to $|s|$. We model whether a word is the head of an answer as a binary classification problem.

Each training sample is a tuple $\langle s, v, q, a, \pm 1 \rangle$. The answer head a is extracted from the k-best dependency parses and the annotated answer span. Given a dependency tree, if any word in the annotated answer span has a parent coming from outside the span, then it is considered an answer head. Therefore, a gold question-answer pair can be transformed into multiple positive training samples. The negative samples come from all the words in the sentence that are not an answer head. For learning, we train a binary classifier for every word in the sentence (except for the verb v).

Experiments We use L2-regularized logistic regression by Liblinear (Fan et al., 2008) for binary classification. Features are listed in Table 10.

The performance of our answer identification approach is measured by accuracy. For evaluation, given each test sentence s, verb v and question q, we output the word with highest predicted score using the binary classifier. If the predicted word is contained inside the annotated answer span, it is considered a correct prediction. We also use the

Feature Class	Question Generation	Answer Identification
Predicate	Token, Predicted POS-tag, Lemma extracted from Wiktionary	
	Dependency parent and edge label, dependency children and edge label	
Question	Question role label, Wh-word, Preposition	
Answer Word	/	Syntactic parent and edge label, Left/Right-most syntactic children,
Predicate-Answer	/	Relative position (left or right), Syntactic relation, Syntactic path

Table 10: Indicator features that are included in our role classifiers for question generation (Section 4) and the answer identification classifier (Section 5). Many come from previous work in SRL (Johansson and Nugues, 2008; Xue and Palmer, 2004). To mitigate syntactic errors, we used 10-best dependency parses from the Stanford parser (Klein and Manning, 2003).

	Newswire	Wikipedia
Classifier	78.7	82.3
Random	26.3	26.9

Table 11: Answer identification accuracy on newswire and Wikipedia text.

baseline method that predicts a random syntactic child from the 1-best parse for each question.

In each of the two domains, we train the binary classifiers on the training set of that domain (See Table 4 for dataset size). Table 11 shows experiment results for answer identification. Our classifier-based method outputs a correct answer head for 80% of the test questions, establishing a useful baseline for future work on this task.

6 Discussion and Future Work

We introduced the task of QA-SRL, where question-answer pairs are used to specify predicate-argument structure. We also presented a scalable annotation approach with high coverage, as compared to existing SRL resources, and introduced baselines for two core QA-SRL subtasks: question generation and answering.

Our annotation scheme has a number of advantages. It is low cost, easily interpretable, and can be performed with very little training and no linguistic expertise. These advantages come, in large part, from the relatively open nature of the QA-SRL task, which does not depend on any linguistic theory of meaning or make use of any frame or role ontologies. We are simply using natural language to annotate natural language.

Although we studied verbal predicate-argument structure, there are significant opportunities for future work to investigate annotating nominal and adjectival predicates. We have also made few language-specific assumptions, and believe the annotation can be generalized to other languages—a major advantage over alternative annotation schemes that require new lexicons to be created for each language.

The biggest challenge in annotating sentences with our scheme is choosing the questions. We introduced a method for generating candidate questions automatically, which has the potential to enable very large-scale annotation by only asking the annotators to provide answers. This will only be possible if performance can be improved to the point where we achieve high recall question with acceptable levels of precision.

Finally, future work will also explore applications of our annotation. Most obviously, the annotation can be used for training question-answering systems, as it directly encodes question-answer pairs. More ambitiously, the annotation has the potential to be used for training parsers. A joint syntactic and semantic parser, such as that of Lewis et al. (2015), could be trained directly on the annotations to improve both the syntactic and semantic models, for example in domain transfer settings. Alternatively, the annotation could be used for active learning: we envisage a scheme where parsers, when faced with ambiguous attachment decisions, can generate a human-readable question whose answer will resolve the attachment.

Acknowledgments

This research was supported in part by the NSF (IIS-1252835), DARPA under the DEFT program through the AFRL (FA8750-13-2-0019), an Allen Distinguished Investigator Award, and a gift from Google. We are grateful to Kenton Lee and Mark Yatskar for evaluating the question generation task, and Eunsol Choi, Yejin Choi, Chloé Kiddon, Victoria Lin, and Swabha Swayamdipta for their helpful comments on the paper. We would also like to thank our freelance workers on oDesk/Upwork for their annotation and the anonymous reviewers for their valuable feedback.

References

Omri Abend and Ari Rappoport. 2013. Universal conceptual cognitive annotation (ucca). In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 228–238.

Collin F Baker, Charles J Fillmore, and John B Lowe. 1998. The berkeley framenet project. In *Proceedings of the 17th International Conference on Computational Linguistics*, volume 1, pages 86–90. Association for Computational Linguistics.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. In *Proceedings of the Linguistic Annotation Workshop*.

Valerio Basile, Johan Bos, Kilian Evang, and Noortje Venhuizen. 2012. Developing a large semantically annotated corpus. In *Proceedings of the 2012 International Conference on Language Resources and Evaluation*, volume 12, pages 3196–3200.

Claire Bonial, Olga Babko-Malaya, Jinho D Choi, Jena Hwang, and Martha Palmer. 2010. Propbank annotation guidelines. *Center for Computational Language and Education Research, CU-Boulder*.

Desai Chen, Nathan Schneider, Dipanjan Das, and Noah A Smith. 2010. Semafor: Frame argument resolution with log-linear models. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 264–267. Association for Computational Linguistics.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. Liblinear: A library for large linear classification. *The Journal of Machine Learning Research*, 9:1871–1874.

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, et al. 2009. The conll-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–18. Association for Computational Linguistics.

Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. Ontonotes: the 90% solution. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 57–60. Association for Computational Linguistics.

Richard Johansson and Pierre Nugues. 2008. Dependency-based semantic role labeling of propbank. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 69–78. Association for Computational Linguistics.

Paul Kingsbury and Martha Palmer. 2002. From treebank to propbank. In *Proceedings of the 2002 International Conference on Language Resources and Evaluation*. Citeseer.

Dan Klein and Christopher D Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*, volume 1, pages 423–430. Association for Computational Linguistics.

Joel Lang and Mirella Lapata. 2011. Unsupervised semantic role induction via split-merge clustering. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, volume 1, pages 1117–1126. Association for Computational Linguistics.

Mike Lewis, Luheng He, and Luke Zettlemoyer. 2015. Joint a* ccg parsing and semantic role labeling. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Ding Liu and Daniel Gildea. 2010. Semantic role features for machine translation. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 716–724. Association for Computational Linguistics.

Bill MacCartney and Christopher D Manning. 2007. Natural logic for textual inference. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pages 193–200. Association for Computational Linguistics.

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

Adam Meyers, Ruth Reeves, Catherine Macleod, Rachel Szekely, Veronika Zielinska, Brian Young, and Ralph Grishman. 2004. The nombank project: An interim report. In *HLT-NAACL 2004 workshop: Frontiers in corpus annotation*, pages 24–31.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106.

Martha Palmer, Daniel Gildea, and Nianwen Xue. 2010. Semantic role labeling. *Synthesis Lectures on Human Language Technologies*, 3(1):1–103.

Simone Paolo Ponzetto and Michael Strube. 2006. Exploiting semantic role labeling, wordnet and wikipedia for coreference resolution. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 192–199. Association for Computational Linguistics.

Drew Reisinger, Rachel Rudinger, Francis Ferraro, Craig Harman, Kyle Rawlins, and Benjamin Van

Durme. 2015. Semantic proto-roles. *Transactions of the Association for Computational Linguistics*, 3:475–488.

Josef Ruppenhofer, Michael Ellsworth, Miriam RL Petruck, Christopher R Johnson, and Jan Scheffczyk. 2006. Framenet ii: Extended theory and practice.

Asher Stern and Ido Dagan. 2014. Recognizing implied predicate-argument relationships in textual inference. *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*.

Ivan Titov and Alexandre Klementiev. 2012. A bayesian approach to unsupervised semantic role induction. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 12–22. Association for Computational Linguistics.

Kristina Toutanova, Dan Klein, Christopher D Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, volume 1, pages 173–180. Association for Computational Linguistics.

Nianwen Xue and Martha Palmer. 2004. Calibrating features for semantic role labeling. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 88–94.

Deniz Yuret, Aydin Han, and Zehra Turgut. 2010. Semeval-2010 task 12: Parser evaluation using textual entailments. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 51–56. Association for Computational Linguistics.

Name List Only? Target Entity Disambiguation in Short Texts

Yixin Cao[1], **Juanzi Li**[1], **Xiaofei Guo**[1], **Shuanhu Bai**[2], **Heng Ji**[3], **Jie Tang**[1]
[1] Tsinghua National Laboratory for Information Science and Technology
Dept. of Computer Science and Technology, Tsinghua University, China 100084
[2] Sina Corporation, China 100084
[3] Dept. of Computer Science, Rensselaer Polytechnic Institute, USA 12180
{caoyixin2011,lijuanzi2008,sophiaguo.thu,jery.tang}@gmail.com
shuanhu@staff.sina.com.cn, jih@rpi.edu

Abstract

Target entity disambiguation (TED), the task of identifying target entities of the same domain, has been recognized as a critical step in various important applications. In this paper, we propose a graph-based model called TremenRank to collectively identify target entities in short texts given a name list only. TremenRank propagates trust within the graph, allowing for an arbitrary number of target entities and texts using inverted index technology. Furthermore, we design a multi-layer directed graph to assign different trust levels to short texts for better performance. The experimental results demonstrate that our model outperforms state-of-the-art methods with an average gain of 24.8% in accuracy and 15.2% in the F1-measure on three datasets in different domains.

1 Introduction

Currently, a growing number of people prefer to express their views and comments online. These messages, which are updated at the rate of millions per day, become a potentially rich source of information. From such a large number of texts, entity disambiguation is a critical step when extracting text information from these messages, and for various applications, such as natural language processing and knowledge acquisition (Dredze et al., 2010). Take the application of customer feedback analysis (CFA) as an example. An enterprise is typically interested in public reviews of its own products as well as those of its competitors'; the identification of these entities is thus critical for further analysis.

The product names comprise a list of entities to be identified. We refer to these entities of the same domain as **target entities**, and the identification process is called **target entity disambiguation** (Wang et al., 2012). All of target entities (e.g., *car brands*) share a common domain, which we refer to as the **target domain**. The target domain is the only constraint to target entities, which implies that the entities are in a specific domain, rather than being general things. In the case of entity recognition from short texts, the disambiguation can be performed on the document level. Given a collection of short documents, our goal is to determine which documents contain the target entities.

Challenge and Related Work

In contrast to traditional entity disambiguation tasks, TED in short texts can require as little information as a name list. There are three types of information that are utilized in Named Entity Disambiguation related tasks: knowledge resources, the context in which a target word occurs and statistical information (Navigli, 2009). However, the lack of the first two types of information makes this problem more challenging.

Knowledge Sparsity A large number of methods focus on using knowledge bases (KBs) like Wikipedia or YAGO to enrich the named entities (e.g., *in-links and out-links*) (Hoffart et al., 2011; Bunescu and Pasca, 2006; Milne and Witten, 2008; Kulkarni et al., 2009; Han et al., 2011; Mihalcea and Csomai, 2007; Shen et al., 2012). These methods compared the context of the entities and their reference pages in the KBs through a similarity measurement. However, we tested 32 different names of General Motors (GM) car brands, and only four of the brands exist in Wikipedia. This circumstance is not unusual. For another larger dataset that included 2468 stock names, we found only 340 of them had their reference pages in Wikipedia. Thus, the methods that rely heavily on KBs might not be appropriate here.

Shortness of the Texts The context in which a

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 654–664,
Lisbon, Portugal, 17-21 September 2015.

target entity occurs plays an important role in disambiguation. (Cassidy et al., 2012; Li et al., 2013) applied the underlying topical coherence information to a set of mentions for entity linking. In (Wang et al., 2012), mentionRank leverages some additional features, such as co-mention; however, people prefer to share their comments in brief or even informal words on social media platforms, such as Twitter, which becomes increasingly important as an information source. We have counted 350,498 microblogs, 37,379 tweets and 34,018 text snippets in various domains in Chinese and English from Twitter, Sina Weibo and Google. After preprocessing, their average length are 16, 5 and 11 words, respectively. To alleviate the shortness issue, additional information has been used to expand the context, such as the user's interest (Shen et al., 2013) and related documents (Guo et al., 2013). Such information is still sparse or unavailable in this case, and thus, the existing methods might not be suitable.

Large Scale Hundreds of millions of tweets are posted daily on Twitter (Liu et al., 2013). When scaling to a large document collection, the disambiguation task becomes increasingly important as the ambiguity increases (Cucerzan, 2007). MentionRank, the only state-of-the-art method for TED, is a graph-based model that focuses on a small set of entities (e.g., 50 or 100 entities) and conducts experiments on thousands of documents (Wang et al., 2012). However, the graph has a quadratic growth as the number of documents increases. In our experiments, a dataset that included 2,468 target entities and 350,498 microblogs generated a directed graph with billions of edges, which required more computer memory than was available even when using a sparse matrix. Therefore, the scalability remains a challenge.

Contributions

To address these challenges, we propose a collective method called TremenRank to disambiguate the target entities simultaneously. The main idea is to propagate trust within a graph based on our observations that true mentions are more similar in context than false mentions in a specific domain. Specifically, inverted indexes is used to construct the graph locally that allows for an arbitrary number of target entities and documents. Furthermore, a multi-layer directed graph is designed to assign different trust levels to documents, which significantly improves the performance. The contributions of our work can be summarized as follows:

- We propose a novel graph-based method, TremenRank, to collectively identify target entities in short texts. This method constructs the graph locally to avoid storing the entire graph in memory, which provides a linear scale up with respect to the number of target entities and documents.
- We design a multi-layer directed (MLD) graph for modeling short texts that possess different trust levels during propagation, which significantly improves the performance.
- We conduct a series of experiments on three practical datasets from different domains. The experimental results demonstrate that TremenRank has a similar performance to the state-of-the-art when addressing the TED problem at a large scale, and the use of MLD graph significantly improves our method.

2 Problem Definition

Example *Suppose that GM wants to collect tweets that talk about its cars. As shown in Figure 1, we take (i) a list of car brands (target entities), and (ii) a collection of tweets (i.e., short texts) as inputs. "Car" is the target domain, and "Sonic" appears in documents d_1 and d_2, which can be characterized as two* ***mentions*** *and the latter is a* ***true mention****. The goal is to find as many true mentions as possible.*

Our method models the collection of documents as a directed graph and outputs a trust score for each document via propagation. The trust score indicates the likelihood of a document containing a true mention. For flexibility, the trust scores lie within the range of 0 to 1 and are globally comparable; thus, we can obtain the top-k documents or choose an appropriate cut-off value to balance the precision and recall for different application requirements. For example, in the application of CFA, a company expects a higher *recall* to achieve a comprehensive understanding of its product, whereas a recommendation system must provide as many *precise* microblogs as possible.

Formally, the problem of TED in short texts is defined as follows:

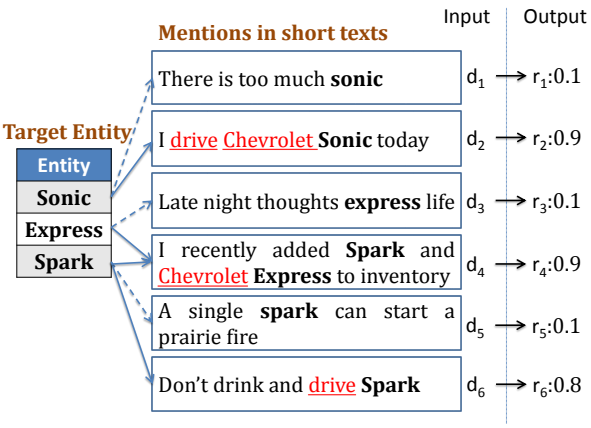


Figure 1: Illustration of TED. True mentions are shown as solid arrows, and the underlined words are their overlapping context.

Target Entity Disambiguation in Short Texts Given a list of target entities $E = \{e_i|i = 1, \ldots, m\}$, and a collection of text documents $D = \{d_j|j = 1, \ldots, n\}$, $E^{d_j} = \{e_1^j, \ldots, e_k^j\}$ is the set of target entities contained in d_j. The goal is to output the trust score $r_j \in [0, 1]$ for each document $d_j \in D$. All the target entities in E^{d_j} share the same trust score r_j.

All of the mentions in one document have the same score because they share a common context. In the graph, the context similarity between documents is computed and used as the edges normally, where the width of the context window that surround the target entity in the document is typically chosen to be 10, 20 or 50 words. However, the documents considered here are limited in length, and a user seldom changes the topic in so few words; thus, we regard the entire document as the context of its target entities. When multiple target entities occur in one document, all of them are more likely to refer to the entities in the target domain. Thus, the trust score for the document is higher, as indicated by d_4 (Figure 1).

The only constraint for the target entities is that they are in the same target domain. This constraint is reasonable in practice. The majority of applications identify a set of entities in one domain at a time. For example, a computer company focuses its attentions on the brands in the computer area, whereas an investment company is mainly interested in stocks. Additionally, even if the target domain is general and contains several small domains, an intuitive solution is to split it into several subproblems, where each subproblem focuses on the target entities in one small domain. Then, the task can be achieved by solving the subproblems individually.

3 Our Approach

TremenRank is a graph-based method that identifies target entities collectively. It propagates trust scores on the graph where each vertex denotes one document and an edge indicates the similarity between them. Considering the large scale of the problem, we obtain the neighbors of one vertex when propagating by searching two indexes instead of storing the entire graph in memory. This approach allows an arbitrary number of target entities and documents to be processed. To further improve the performance, a multi-layer directed graph is designed to treat the documents at different trust levels based on prior estimations.

3.1 Hypotheses

The documents within a domain share the characteristic of unitary similarity. This characteristic implies that all of the true mentions have a similar context due to the target domain constraint and that false mentions are distinct because their meanings belong to diversified domains. We investigated the ambiguity of 2468 stock names (target entities in experiments), and manually labeled 301 of those names. As shown in Figure 2, there are many different meanings for these names outside of the target domain, such as plant, bank, media and animal. The distribution of meanings are long-tailed; thus, we gathered a group of meanings together (the class "others").

Based on the statistical results, we can make the following hypotheses:

- The context of true mentions are similar to one another.
- The context of a false mention is different from any of the true mention.
- False mentions have distinct contexts across different entities.

For example, the true mentions in d_2, d_4 and d_6 (Figure 1) describe car brands of GM and share common pieces of text: "drive" or "Chevrolet". However, "sonic", "express" and "spark" in d_1, d_3 and d_5 are all false mentions; in their contexts, they refer to sound, giving opinions, and a small amount of fire, respectively. These false mentions are different in context from one another and from any true mention.

The assumptions resemble the main insight of MentionRank except the co-mention (multiple entities occur in one document). Co-mention seldom happens in short texts, and can be treated as the same because they share a common context. In conclusion, the assumptions suggest that a collective method could perform better than a method that disambiguates entities separately, because more comprehensive information on the entities from multiple "collaborators" (i.e., the mentions have similar contexts) has been used (Chen and Ji, 2011; Kulkarni et al., 2009; Pennacchiotti and Pantel, 2009; Will et al., 2010; Fernandez et al., 2010; Cucerzan, 2011; Guo et al., 2011; Han and Sun, 2011; Ratinov et al., 2011; Kozareva et al., 2011; Dalton and Dietz, 2013).

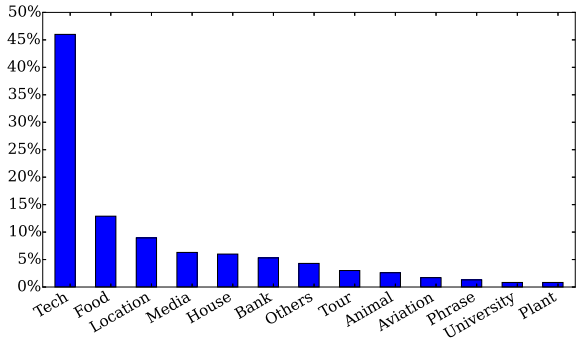


Figure 2: Statistics on the ambiguity in stocks.

3.2 Graph-based Method

Based on these assumptions, we build a graph to represent the documents and their relations, and we perform a TrustRank-like algorithm on the graph. We are given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ that consists of a set $\mathcal{V}$ of N documents (vertices) and a set $\mathcal{E}$ of directed edges, where each edge $(d_i, d_j) \in \mathcal{E}$ denotes that d_i points to d_j, and $o(d_i)$ is the out-neighbors of d_i.

3.2.1 Similarity Measurement

We constructed the edges of the graph according to the similarity relations between documents. Most similarity measurements (Artiles et al., 2010; Miller, 1995) could be used in the proposed method. After some exploration, we found that Jaccard similarity performs better. It can be efficiently calculated through simple set operations:

$$\omega_{ij}^J = J(d_i, d_j) = \frac{\left|\mathcal{W}^{d_i} \bigcap \mathcal{W}^{d_j}\right|}{\left|\mathcal{W}^{d_i} \bigcup \mathcal{W}^{d_j}\right|} \quad (1)$$

where ω_{ij}^J denotes the weight of edge (d_i, d_j) using Jaccard similarity, and $\mathcal{W}^{d_i}$ is the set of words contained in d_i. ω_{ij} varies from 0 to 1, where a value closer to 1 indicates that its two nodes are more similar. We link only similar nodes by choosing an appropriate threshold η. In other words, we have $(d_i, d_j) \in \mathcal{E}$, only if $\omega_{ij}^J > \eta$. This is the foundation to construct the graph locally using inverted indexes.

3.2.2 Inverted Index

When the scale becomes excessively large, such as the 350,000 pieces in our dataset, the number of edges will increase into the billions, producing computational and storage-based difficulties. Considering that the propagation begins with document d_i and that we are required to find all of its out-neighbors $o(d_i)$ through a traversal of the entire dataset, then the complexity is $O(n^2)$. Alternatively, we can represent the entire graph with a matrix; however, its billions of elements would be difficult to store and calculate.

To address the large scale problem, we construct the graph locally via inverted index technology. During propagation, the neighbors of the documents are obtained by searching the indexes in real time. Two types of indexes are used: the document-to-word index and the word-to-document index. The former index records $\mathcal{W}^{d_i}$ of each document $d_i \in D$; the latter index records the occurrence of each word $\mathcal{D}^{w_k} = \{d_j | w_k \in W\}$, where $W = \{w_1, \ldots, w_N\}$ is the word dictionary. Combining these two indexes, the total out-neighbors of a document can be obtained in constant time as follows:

1. Obtain all of the words $\mathcal{W}^{d_i}$ of d_i via searching the document-to-word index.
2. Find the occurrences of each word $w_k \in \mathcal{W}^{d_i}$ in the word-to-document index: $D^{d_i} = \{d_j | \cup_{w_t \in \mathcal{W}^{d_i}} \mathcal{D}^{w_t}\}$. Each document $d_j \in D^{d_i}$ shares at least one common word with d_i.
3. Count the frequency of d_j, which indicates the number of overlapping words between d_i and d_j. Then, the frequency $f_{ij} = |\mathcal{W}^{d_i} \cap \mathcal{W}^{d_j}|, i \neq j$ and Equation 1 becomes:

$$\omega_{ij}^{J} = \frac{f_{ij}}{|\mathcal{W}^{d_i}| + |\mathcal{W}^{d_j}| - f_{ij}} \quad (2)$$

4. Calculate the similarity weight. We obtain the out-neighbors $o(d_i) = \{d_j | \omega_{ij}^{J} \geq \eta\}$.

3.2.3 Trust Propagation

Similar to TrustRank, an individual document propagates the trust score to its neighbors and those who have more neighbors will receive more trust. Intuitively, trust attenuates along the edge. There are several ways for attenuation to take place, such as trust dampening, trust splitting, and a combination of them (Gyöngyi et al., 2004). For example, each document $d_i \in D$ has a trust score r_i, and its out-neighbors obtain $\alpha \cdot r_i$ (or $\frac{r_i}{|o(d_i)|}$) through trust dampening (or splitting). We adopt the third method, in which a document dampens its split trust with the attenuation coefficient α.

The final trust score is determined by two parts: the trust from a document's neighbors and its prior estimation. Then, TremenRank can iteratively propagate via the following function:

$$r_i = \alpha \cdot \sum_{o(d_i)} \frac{r_j}{|o(d_j)|} + (1 - \alpha) \cdot \mathcal{T}(d_i) \quad (3)$$

where $\mathcal{T}(d_i)$ is the prior estimation of the document d_i and is a constant during iterative propagation. Here we simply assign a uniform distribution to all of the documents $\mathcal{T}(d_i) = \frac{1}{|D|}$. A more precise estimation will be discussed later.

At the beginning of the propagation, we initialize the trust scores of the documents with the prior estimation. Then, the trust scores of the documents are updated iteratively until convergence[1]. True mentions receive high scores because they are likely to connect with more trustworthy documents and thus receive more trust through the first term in Equation 3. In contrast, false mentions are dissimilar to most other documents; thus, their scores gradually attenuate (the second term in Equation 3). These scores are globally comparable. One can normalize the final scores by dividing them by the largest score, but the relative ordering of the documents will not change. Thus, one document that is more likely to contain any true mention will receive a higher trust score and be ranked higher.

3.3 Multi-layer Directed Graph

During propagation, all of the documents are initialized with the same trust score and attenuate their trust at the same level. However, different documents should be treated differently. For example, the tweet "*I drive a big car* ***suburban***" is more trustworthy than "*doctors use* ***GMC*** *report system to process harmful patients*", because the former tweet contains the credible context feature "car", which is the name of the target domain. The latter clearly irrelevant text should not be trusted or even be regarded as noise. In this subsection, we first discuss how to make more precise prior estimation of documents based on some of the characteristics of the target domain. Additionally, based on the prior estimation, an MLD graph is built to assign different trust levels to the documents.

3.3.1 Prior Estimation

Ideally, a true mention is similar to true mentions only. To take advantage of the approximate isolation of true mentions, we first extract a set of true mentions to start the propagation. The documents that contain these true mentions are called seeds.

Formally, the entire set of documents D is divided into two groups: (i) the seed set $D^s = \{d_1, \ldots, d_s\}$, and (ii) a subset $D^* = \{d_{s+1}, \ldots, d_n\}$. We estimate a prior trust score for each document via the function $\mathcal{T}$:

$$\mathcal{T}(d_i) = \begin{cases} \frac{1-\epsilon}{|D^*|} & d_i \in D^*, \\ \frac{\epsilon}{|D^s|} & d_i \in D^s. \end{cases} \quad (4)$$

where $|D^s|$ and $|D^*|$ represent the size of the two sets for normalization. $\epsilon \in (0, 1]$ is used for smoothing and indicates the likelihood that we can trust the seeds that actually contain true mentions. In the experiments, we set $\epsilon = 0.9$ based on the accuracy of the seeds extraction method.

There are several methods for extracting seeds, such as manual annotation and pattern-based method. Patterns could be easily derived from the characteristics of the target domain, such as the domain name, product type or unique ID[2]. These methods can typically identify entities with a high accuracy, but their low recall limits the portion of

[1] We select the attenuation coefficient $\alpha = 0.85$ and the number of iterations to be 20, which have been regarded as the standards in the PageRank literature(Page et al., 1999; Krishnan and Raj, 2006). Our experiments also show that 20 iterations are sufficient to achieve convergence.

[2] The exact patterns used in our datasets are detailed in Section 4

true mentions that can be found. We use the results of the pattern-based method as our seeds, which have an accuracy higher than 90%. We do not present the experimental results here due to space limitations.

3.3.2 Graph Construction

The similarity measurement does not consider directions, and thus, the documents are mutually connected. In this section, we construct a layered structure in the graph, where each layer denotes a document trust level that contains any true mention. Thus, we define that the propagation direction from the high trust level to the low trust level.

Figure 3 shows an example of an MLD graph. The blue nodes of the seeds in the top layer are the most trustworthy, and the other white nodes in the higher layer are less similar to the seeds which implies that they are at lower trust levels. Thus, as we move farther away from the seeds, trust attenuates at a constant speed α along with the layers. The nodes in the same layer are also connected.

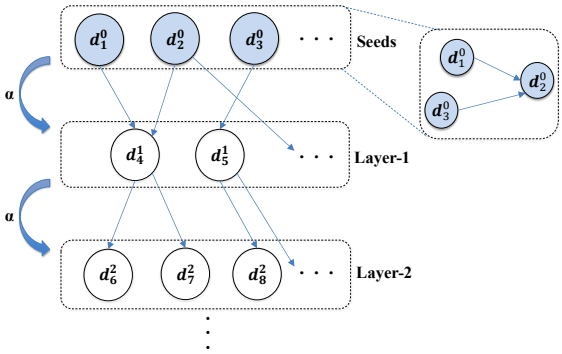


Figure 3: Example of an MLD graph.

The construction algorithm[3] is presented in Algorithm 1, where $D^l = \{d_i^l | l \geq 0\}$ is the set of nodes in layer l, and the nodes in $\bar{D} = \{\bar{d}_j | \bar{d}_j \notin D^l, l \geq 0\}$ are not connected. To simplify our notation, the seeds are set to be in layer 0. Note that $(\cap_{l \geq 0} D^l) \cap \bar{D} = D$.

TremenRank is different from the standard TrustRank and MentionRank in several respects. First, TremenRank is designed to process short texts at a large scale. Second, through a well-designed MLD graph, we consider documents to consist of different trust levels rather than be represented by a unified distribution. Third, TrustRank randomly samples a set of seeds and checks them manually, which limits the number of seeds available; however, the proposed method extracts the seeds automatically and uses large amounts of seeds to produce better prior estimations. Finally, disambiguation occurs at the document level in the proposed method; we assign the same scores to the entities that occur in one document, because they typically share common context features in short texts.

Algorithm 1: Construction of an MLD graph.

Input: D^s, D^*, η, indexes $\mathcal{W}^D, \mathcal{D}^W$
Output: $\mathcal{G}$
Initialize seeds in layer $l = 0$, $\bar{D} = D^*$;
foreach *layer* $l \geq 0$ **do**
  Find the out-neighbors of D^l from $\bar{D}$;
  foreach *document* $d_i^l \in D^l$ **do**
    put $o(d_i^l)$ in the next layer D^{l+1};
    update $\bar{D} = \bar{D} - D^{l+1}$;
  end
  if $|\bar{D}| = 0$ *or* $|D^{l+1}| = 0$ **then**
    break;
  else
    $l = l + 1$;
  end
end

4 Empirical Evaluation

4.1 Data Preparation

Because there is no publicly available benchmark dataset for TED, we constructed three datasets of different domains: Stock, Car and Herb[4]. All of these datasets came from the needs of real-life projects.

1. **Stock** - We collected 2,468 stock names from a stock exchange website, and identified their candidate mentions by string matching from Sina Weibo (Counterpart of Twitter in China). Each stock has a unique ID, which has little ambiguity. Thus, we used this regular expression pattern to extract seeds.

2. **Car** - We collected 32 car brands of General Motors and a group of tweets that contain at least one mention of these brands via the Twitter API. In the seeds extraction step, we use the domain name "car" as the patterns.

[3] A key parameter for the structure of MLD Graph is η, which will be discussed in Section 4.2.

[4] All of the dataset related sources used in this study are listed at http://newsminer.org/TEDs.

3. **Herb** - We randomly selected 1,119 Chinese herb names and collected 40 pieces of text descriptions in the search results per entity from Google. To extract the seeds, we simply added the domain name to the key words (e.g., *"天虫(Worms)* 中草药*(Chinese Medical Herb)"*).

Table 1: Statistics on the datasets.

	E	D	#edges	Positive(%)
Stock	2,468	350,498	2.049B	43
Car	32	37,379	4.91M	7
Herb	1,119	34,018	9.66M	68

Table 1 shows some of the statistics of the datasets created. We chose these datasets because (i) the identification of entities in these domains meets the practical requirements of many applications, (ii) the target entities are ambiguous and (iii) there is little information in the existing KBs. Before applying TremenRank, we preprocessed the plain texts through word segmentation, low frequency words filtering and stop words filtering.

4.2 Experiment

Baseline Methods TED in short texts is a relatively new problem, and there are few specific methods for solving it. To validate the performance of TremenRank and the improvement produced by the MLD graph, we selected the baseline from three different perspectives: (i) a context-based method that identifies target entities separately; (ii) a classic supervised method SVM to classify a document by whether it contains any true mentions; and (iii) the only state-of-the-art MentionRank for TED, which is a collective ranking method.

- **The Context-based method** mines a frequent item set of true mentions and identifies the documents that contain more frequent items.

- **SVM** classifies the documents into two classes: documents that are in the target domain and those that are not. Using context words as features, we train and test SVM on the labeled set with a 10-fold cross validation.

- **MentionRank** is a graph-based method that disambiguates at the entity level. Difficult to apply to the entire large datasets directly, it has been applied to the labeled set.

Evaluation Metrics The performance of the disambiguation task is typically evaluated by *accuracy*, but in TEDs we are also interested in *precision*, *recall* and the *F1-measure* because different applications focus on different aspects. For example, in the application of CFA, a company expects a higher *recall* to collect as many reviews as possible, while in financial news recommendations, users prefer to read more *accurate* microblogs. Because the entire dataset is too large to evaluate directly, we randomly sampled 800 mentions for each dataset and labelled them manually to calculate the performance metrics[5].

Results and Analysis

The overall performances of TremenRank and those of the baseline methods on all of the datasets are shown in Table 2. The following is indicated in the results:

- TremenRank+MLD outperforms all of the baselines with all of the datasets, because it collectively identifies target entities and treats the documents differently based on a precise prior estimation.

- The collective methods tend to perform better. Although, on the Car dataset, SVM achieves the second best performance, the use of MLD graph in TremenRank outperforms all of the methods tested. This is because false mentions that occupy a large proportion of the dataset produce too much noise, whose negative impacts can be reduced through the train set in the supervised method or a precise prior estimations.

- Compared with MentionRank, TremenRank processes a larger number of entities and documents while achieving a similar performance. Combined with a MLD graph, TremenRank shows significant improvements including an average gain of 24.8% in accuracy and 15.2% in the F1-measure on the three datasets.

We also investigate the influence of the main elements in TremenRank below.

Similarity Threshold Different similarity thresholds result in various structures of the MLD graph, and have a great impact on the performance of our

[5] A detailed explanation about these metrics can be found in http://en.wikipedia.org/wiki/Precision_and_Recall.

Table 2: Overall results on the three datasets

Method	Stock				Car				Herb			
	Accu	Prec	Recall	F-score	Accu	Prec	Recall	F-score	Accu	Prec	Recall	F-score
Context	0.704	0.332	0.545	0.410	0.733	0.333	0.256	0.476	0.670	0.200	0.074	0.108
SVM	0.746	0.377	0.998	0.547	0.839	0.371	0.923	0.529	0.567	0.988	0.564	0.718
MentionRank	0.458	0.424	0.828	0.561	0.464	0.301	0.715	0.423	0.644	0.691	0.824	0.752
TremenRank	0.477	0.424	0.803	0.555	0.542	0.310	0.720	0.433	0.737	0.736	0.827	0.779
TremenRank+MLD	0.683	0.575	0.844	**0.684**	0.827	0.624	0.731	**0.673**	0.800	0.774	0.908	**0.836**

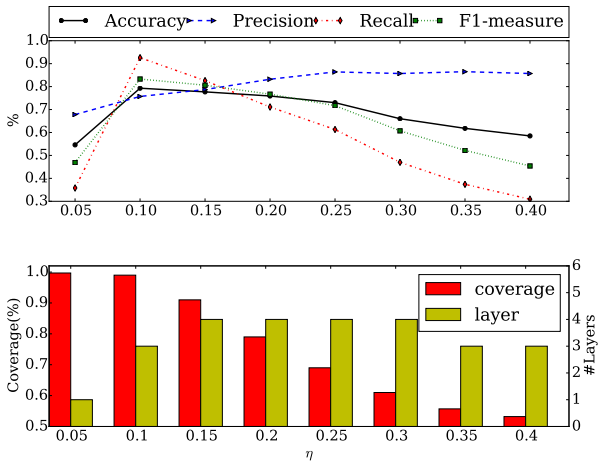


Figure 4: Selection of the Similarity Threshold

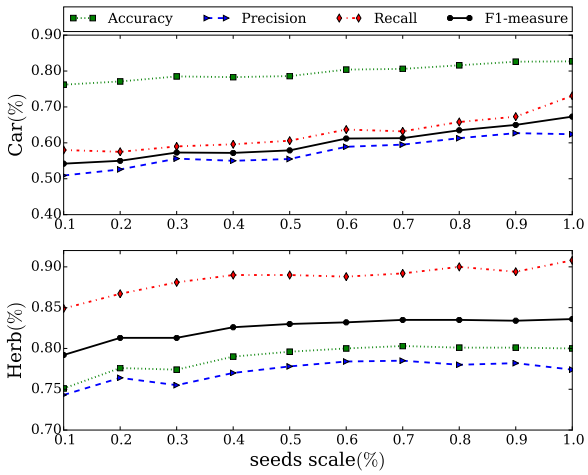


Figure 5: Influence of Seed Scale

method. In this subsection, we study a heuristical method that help choose the best eta according two factors (Both of them can be obtained before propagation). Figure 4 presents the experimental results for different values of η on the dataset of Herb. The performance of TremenRank is showed on the top, and the bottom is the corresponding graph structure represented by two factors: the graph coverage and the number of layers. We can see that precision increases until it becomes steady with the growth of η, and other measurements reach their peaks when $\eta = 0.1$.

This is in accordance with the change of the graph structure. On one hand, the number of layers l declines sharply when η is less than 0.1, this indicates little difference in the trust level of documents in the propagation. On the other hand, our method has no effect on the vertices outside of the graph, so the performance is directly proportional to the graph coverage rate. Therefore, a proper η should ensure a high coverage as well as adequate layers. In experiments, we choose η as 0.1, 0.15, 0.1 for the datasets of Stock, Car and Herb respectively.

Influence of the Seed Scale As the basis of the prior estimation, the seeds have significant influence on the performance of the proposed method via the MLD graph. Intuitively, a larger set of seeds should lead to a more precise estimation and thus better performance. In the experiments on the Car and Herb datasets, we split their seed set into ten parts, and add one part each time. As Figure 5 shows, when increasing the percentage of the seeds gradually, the performance has an overall upward trend (e.g., a 14.6% and 4.4% gain in the F1-measure for the Car and Herb datasets, respectively). This trend occurs because the potential context features of the seeds utilized for propagation increase as the absolute number of seed documents rises. For example, the tweet *"Cadillac Uber airport is classy"* obtains a low score of 0.013 with 50% of the seeds, whereas it is identified successfully with the score of 0.588 when all of the seeds are used for propagation because more context features are introduced, such as *"airport" and "Uber"*. Thus, the proposed method achieves better performance as the seed set grows. This arrangement is helpful for a company that seldom changes its business area and then accumulates seeds continuously to improve performance.

Robustness to Noise Because seeds play an important role in the MLD graph, we further test the

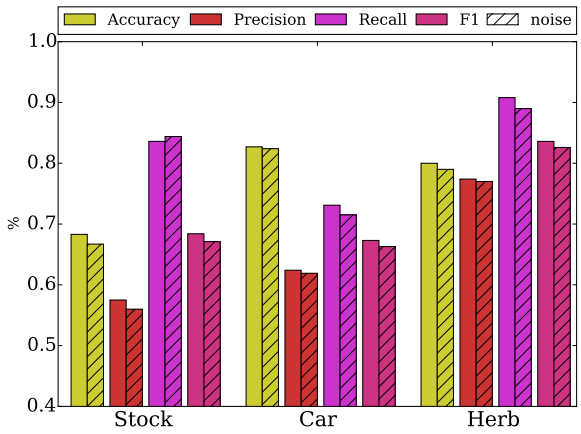


Figure 6: Robustness to Noise

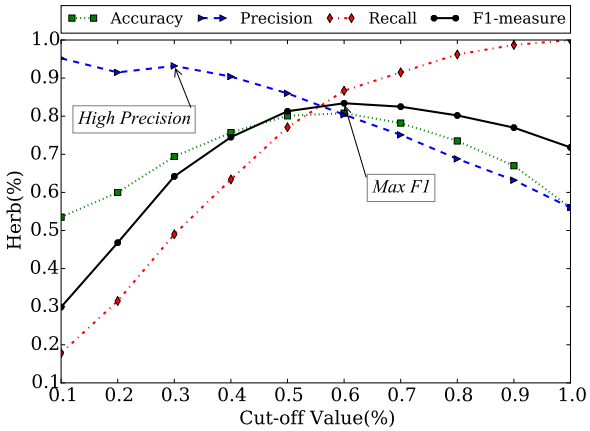


Figure 7: Cut-off Value

robustness of our method with respect to the quality of the seeds. We randomly sample documents outside of the seed set (considered as noise) to replace 20% of the seeds on all the datasets; these results are shown in Figure 6. The introduced noise only leads to a limited decrease in performance (average 1.3%, 1% and 1% in the F1), which is much better than when only using TremenRank (Table 2). More specifically, the experiments that use artificial noise occasionally achieve a higher recall (e.g., on the Stock dataset); this result could occur because the unknown true mentions add some useful edges to the graph, which is helpful when finding more true mentions of the target entities.

Cut-off Value According to the globally comparable trust scores, we can (i) rank all of the documents, and trade off the performance metrics by choosing an appropriate cut-off value γ for various applications, or (ii) rank the documents of an individual entity separately and obtain its top-k mentions.

In the experiments, we set γ as different percentage of the ranked documents. As Figure 7 shows, the recommendation system could use $\gamma = 30\%$ to achieve a 93.2% precision and a 64.2% recall, or a company could use $\gamma = 60\%$ for more reviews that have a relatively high precision.

Efficiency We implemented TremenRank in Java and ran it on a single PC with the Windows 8.1 64-bit operation system. With an Intel(R) Core(TM) i3-3240 (3.40GHz) CPU and 4GB memory, our program converges within 5 iterations, and only consumes approximately 700MB of memory when running steadily. The overall identification times of the Stock, Car and Herb datasets are 12h, 5min and 18min, respectively. The computation time increases exponentially with the increase of the amount of data due to the excessive computations required to search the indexes, which can be optimized in future work.

5 Conclusions and Future Work

In this paper, we addressed a new and increasingly important problem in social content analysis in a challenging scenario: disambiguation of a list of homogenous entities in short texts using names only. We proposed a graph-based method called TremenRank to identify target entities collectively; this method can also hold an arbitrary number of target entities and documents. The performance of this method can be further improved via a well-designed MLD graph. The experimental results show that the proposed method has a significant improvement compared to other approaches.

In the future, we are interested in refining the prior estimation by using the ontology and extending this work to detect the target entities that are not in a list while performing the disambiguation task.

Acknowledgments We'd like to acknowledge Lei Hou, Chi Wang and Hongzhao Huang for their impressive discussions on this paper. The work is supported by 973 Program (No. 2014CB340504), NSFC-ANR (No. 61261130588), Tsinghua University Initiative Scientific Research Program (No. 20131089256), Science and Technology Support Program (No. 2014BAK04B00), and THU-NUS NExT Co-Lab.

References

Javier Artiles, Andrew Borthwick, Julio Gonzalo, Satoshi Sekine, and Enrique Amigó. 2010. Weps-3 evaluation campaign: Overview of the web people search clustering and attribute extraction tasks. In *CLEF (Notebook Papers/LABs/Workshops)*.

Razvan C Bunescu and Marius Pasca. 2006. Using encyclopedic knowledge for named entity disambiguation. In *EACL*, volume 6, pages 9–16.

Taylor Cassidy, Heng Ji, Lev-Arie Ratinov, Arkaitz Zubiaga, and Hongzhao Huang. 2012. Analysis and enhancement of wikification for microblogs with context expansion. In *COLING*, volume 12, pages 441–456. Citeseer.

Zheng Chen and Heng Ji. 2011. Collaborative ranking: A case study on entity linking. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 771–781. Association for Computational Linguistics.

Silviu Cucerzan. 2007. Large-scale named entity disambiguation based on wikipedia data. In *EMNLP-CoNLL*, pages 708–716.

Silviu Cucerzan. 2011. Tac entity linking by performing full-document entity extraction and disambiguation. In *Proceedings of the Text Analysis Conference*, volume 2011.

Jeffrey Dalton and Laura Dietz. 2013. A neighborhood relevance model for entity linking. In *Proceedings of the 10th Conference on Open Research Areas in Information Retrieval*, pages 149–156. LE CENTRE DE HAUTES ETUDES INTERNATIONALES D'INFORMATIQUE DOCUMENTAIRE.

Mark Dredze, Paul McNamee, Delip Rao, Adam Gerber, and Tim Finin. 2010. Entity disambiguation for knowledge base population. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 277–285. Association for Computational Linguistics.

Norberto Fernandez, Jesus A Fisteus, Luis Sanchez, and Eduardo Martin. 2010. Webtlab: A cooccurencebased approach to kbp 2010 entity-linking task. In *Proc. TAC 2010 Workshop*.

Yuhang Guo, Wanxiang Che, Ting Liu, and Sheng Li. 2011. A graph-based method for entity linking. In *IJCNLP*, pages 1010–1018. Citeseer.

Weiwei Guo, Hao Li, Heng Ji, and Mona T Diab. 2013. Linking tweets to news: A framework to enrich short text data in social media. In *ACL (1)*, pages 239–249. Citeseer.

Zoltán Gyöngyi, Hector Garcia-Molina, and Jan Pedersen. 2004. Combating web spam with trustrank. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, pages 576–587.

Xianpei Han and Le Sun. 2011. A generative entity-mention model for linking entities with knowledge base. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 945–954. Association for Computational Linguistics.

Xianpei Han, Le Sun, and Jun Zhao. 2011. Collective entity linking in web text: a graph-based method. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 765–774. ACM.

Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. 2011. Robust disambiguation of named entities in text. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 782–792. Association for Computational Linguistics.

Zornitsa Kozareva, Konstantin Voevodski, and Shang-Hua Teng. 2011. Class label enhancement via related instances. In *Proceedings of the conference on empirical methods in natural language processing*, pages 118–128. Association for Computational Linguistics.

Vijay Krishnan and Rashmi Raj. 2006. Web spam detection with anti-trust rank. In *AIRWeb*, volume 6, pages 37–40.

Sayali Kulkarni, Amit Singh, Ganesh Ramakrishnan, and Soumen Chakrabarti. 2009. Collective annotation of wikipedia entities in web text. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 457–466. ACM.

Yang Li, Chi Wang, Fangqiu Han, Jiawei Han, Dan Roth, and Xifeng Yan. 2013. Mining evidences for named entity disambiguation. In *KDD'13*, pages 1070–1078.

Xiaohua Liu, Yitong Li, Haocheng Wu, Ming Zhou, Furu Wei, and Yi Lu. 2013. Entity linking for tweets. In *ACL (1)*, pages 1304–1311.

Rada Mihalcea and Andras Csomai. 2007. Wikify!: linking documents to encyclopedic knowledge. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 233–242. ACM.

George A. Miller. 1995. Wordnet: A lexical database for english. *Commun. ACM*, 38(11):39–41.

David Milne and Ian H Witten. 2008. Learning to link with wikipedia. In *Proceedings of the 17th ACM conference on Information and knowledge management*, pages 509–518. ACM.

Roberto Navigli. 2009. Word sense disambiguation: A survey. *ACM Computing Surveys (CSUR)*, 41(2):10.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. The pagerank citation ranking: Bringing order to the web.

Marco Pennacchiotti and Patrick Pantel. 2009. Entity extraction via ensemble semantics. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*, pages 238–247. Association for Computational Linguistics.

Lev Ratinov, Dan Roth, Doug Downey, and Mike Anderson. 2011. Local and global algorithms for disambiguation to wikipedia. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 1375–1384. Association for Computational Linguistics.

Wei Shen, Jianyong Wang, Ping Luo, and Min Wang. 2012. Linden: linking named entities with knowledge base via semantic knowledge. In *Proceedings of the 21st international conference on World Wide Web*, pages 449–458. ACM.

Wei Shen, Jianyong Wang, Ping Luo, and Min Wang. 2013. Linking named entities in tweets with knowledge base via user interest modeling. In *KDD*, pages 68–76.

Chi Wang, Kaushik Chakrabarti, Tao Cheng, and Surajit Chaudhuri. 2012. Targeted disambiguation of adhoc, homogeneous sets of named entities. In *WWW*, pages 719–728.

Radford Will, Hachey Ben, Nothman Joel, Honnibal Matthew, and R.Curran James. 2010. Cmcrc at tac10: Document-level entity linking with graph-based re-ranking. In *In Proc. Text Analysis Conference (TAC 2010)*.

Biography-Dependent Collaborative Entity Archiving for Slot Filling

Yu Hong[†]
Soochow University
& Rensselaer Polytechnic Institute
tianxianer@gmail.com

Xiaobin Wang[†]**, Yadong Chen, Jian Wang**
Soochow University
Suzhou, CHN (215006)
czwangxiaobin@gmail.com

Tongtao Zhang, Heng Ji[†]
Rensselaer Polytechnic Institute
NY, USA (12180)
jih@rpi.edu

Abstract

Knowledge Base Population (KBP) tasks, such as slot filling, show the particular importance of entity-oriented automatic relevant document acquisition. Rich, diverse and reliable relevant documents satisfy the fundamental requirement that a KBP system explores the nature of an entity. Towards the bottleneck problem between comprehensiveness and definiteness of acquisition, we propose a collaborative archiving method. In particular we introduce topic modeling methodologies into entity biography profiling, so as to build a bridge between fuzzy and exact matching. On one side, we employ the topics in a small-scale high-quality relevant documents (i.e., exact matching results) to summarize the life slices of a target entity (i.e., biography), and on the other side, we use the biography as a reliable reference material to detect new truly relevant documents from a large-scale partially complete pseudo-feedback (i.e., fuzzy matching results). We leverage the archiving method to enhance slot filling systems. Experiments on KBP corpus show significant improvement over state-of-the-art.

1 Introduction

Entity archiving is an entity-oriented document retrieval task. Towards a target entity of a specific type, such as the ones discussed in this paper, a person or an organization, the goal of entity archiving is to search and collect all relevant documents from large-scale data sets under limited prior knowledge of the entity. We limit our study to the regular English entity archiving, in which the prior knowledge contains the commonly used full name (formatted by English entity naming criteria) along with a gold-standard reference document, such as a news story on President "*George W. Bush*".

Entity archiving plays a fundamental role in KBP tasks. It narrows down the range of the data source for knowledge discovery to small-scale closely related documents. Such documents, on one hand, contain informative content on a target entity, which is extremely favorable for background knowledge extraction. On the other hand, the documents provide definitive evidence for verifying the claimed identity of the entity.

As for KBP slot filling and verification tasks (Surdeanu and Ji, 2014), the archived relevant documents to an entity provide sufficient contexts (provenances) of the concrete instances (fillers) of the entity attributes (slots). See Figure 1, in which both the *provenances* support filler extraction, meanwhile the *provenance 1* additionally provides the evidence to verify the fillers (e.g., is *episcopalism* the true religion of *Bush*?).

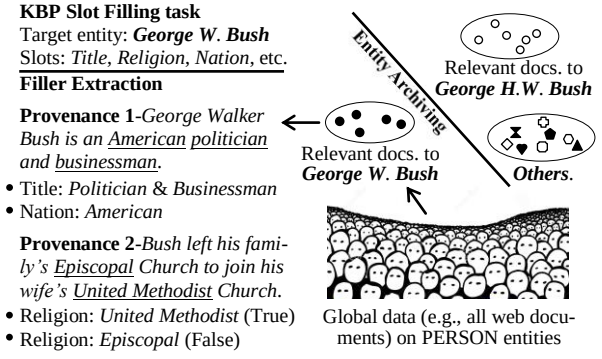


Figure 1**:** Use of entity archiving in slot filling

The main challenges of entity archiving are as

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 665–675,
Lisbon, Portugal, 17-21 September 2015.

follows: 1) it is difficult to retrieve all relevant documents through exact matching at the level of entity name, because an entity can be mentioned in various forms, such as alternate names and abbreviations; 2) in contrast, fuzzy matching introduces a large amount of noise into retrieval results (see the examples in Figure 1), although it is capable of recalling an overwhelming majority of relevant documents; 3) inadequate prior knowledge makes it difficult to generate a full profile of an entity; 4) although pseudo-feedback is helpful to enrich the prior knowledge, traditional entity profiling (e.g., bag-of-words) methods establish vague boundaries among different life slices of an entity. For example, they are incapable of distinguishing the slice of the "*Church Scientologist in Sea Organization*" of *Mark Fisher*[1] from the freelance career as the "*Corporate liaison to Miscavige*". As a result it is difficult to enhance the independent effects of different slices on the entity-document relevance determination.

To solve these problems, we propose a collaborative entity archiving (CEA) method. It employs the exact-matching based document retrieval to obtain a few high-quality reference documents, and leverages fuzzy matching for high-speed acquisition of adequate candidate documents (section 3). In addition, CEA uses the reference documents as prior knowledge to model the topic-level biography of an entity, and identifies the truly relevant documents from the candidates based on biography-document relevance (section 4). Experiments show that CEA has substantial advantages over traditional retrieval methods (section 5.1). We apply CEA to state-of-the-art slot filling systems. Experimental results show that CEA provides consistent gains (section 5.2).

2 Related Work

- **Entity Search**

One research topic similar to entity archiving is entity search. Entity search aims to seek, collect, and rank entities associated with specific information needs (Balog et al, 2011). The TREC Enterprise track featured an *expert finding* task (Balog et al, 2008a): given a topic, return a ranked list of experts on the topic. In response to this problem, there have been considerable efforts on content based retrieval models, as well as feature selection, such as proximity (Petkova and Croft, 2007), document priors (Hu et al, 2006; Zhu et al, 2010), expert-document associations (Balog and De-Rijke, 2008) and external evidence (Serdyukov and Hiemstra, 2008).

Since INEX was launched in 2002, which is an entity ranking task specific to structured data and multimedia (Demartini et al, 2010), structured features have been widely used in entity search, such as the most recent studies on Wikipedia links and categories (Vercoustre et al, 2008; Zhu et al, 2008; Jiang et al, 2009; Weerkamp et al, 2009; Kaptein and Kamps, 2009; Balog et al, 2011) and web link structure (Balog et al, 2008b; You et al, 2011; Blanco et al, 2011; Neumayer et al, 2012; Bron et al, 2013).

- **Slot Filling**

The goal of slot filling is to seek and extract the concrete instances (fillers) specific to multiple entity attributes (slots) from a large-scale textual data set (Ji et al., 2010 and 2011; Surdeanu, 2013; Surdeanu and Ji, 2014). The quality of the fillers largely depends on the performance of entity archiving and information extraction.

Related studies on archiving mainly employed traditional retrieval techniques, including query expansion and string matching (Ji and Grishman, 2011). A few studies involved document ranking and prioritizing by using probability model (Byrne and Dunnion, 2010; Roth et al, 2014) and statistical language model (Chrupala et al, 2010).

For filler extraction, great efforts were made to generate effective patterns and structure perceptrons by supervised learning and reasoning (Chen et al, 2010; Grishman and Min, 2010; Gao et al, 2010; Surdeanu et al, 2011; Louis et al, 2011; Kisiel et al, 2013). And effective feature selection and distant-supervision based classifiers have been explored (Lehmann et al, 2010; Artiles et al, 2011; Sun et al, 2011; Roth and Klakow, 2013; Roth et al, 2014). Recently active learning (Angeli et al, 2014), truth-finding (Yu et al, 2014) as well, scanning (Yu et al., 2015) and ensemble learning (Viswanathan et al., 2015) were introduced to this field.

- **Brief Summary**

In all, entity search concentrates on the analysis of a single specific aspect of an entity, which is of interest or related to a domain. In the *expert finding* task, only *academic career*s of person entities (potential experts) are of concern in entity-document relevance determination. By contrast, for the sake of comprehensive understanding of an entity, entity archiving necessarily

1 Mark Fisher: a PERSON query name in Slot Filing evaluation of 2014, ID=SF14_ENG_031

takes multiple and diverse aspects into account, such as a person's career, family, religion, sociality, academics, etc. Due to the difference in goals, entity search techniques cannot be used directly to solve entity archiving problems.

The performance of conventional retrieval techniques was generally limited due to the lack of precise modeling of the characteristics of an entity. Sparse prior knowledge and absence of effective profiling methods cause difficulties in characterizing the entity. The rest of the paper will be about knowledge acquisition and partition, as well as the collaborative method, along with a topic-level biographical profiling method.

3 Prior Knowledge Acquisition

We use string matching based retrieval methods to acquire relevant documents. It is worth considering that the acquired documents are not straightforwardly defined as the final entity archiving results. As we will show in this section, some of them are reliable, while others are full of noise. Instead, we regard them as the prerequisite knowledge for a coarse-to-fine processing.

In the retrieval phase, a query *Q* is formulated as the full name of the target entity, while a document *D* is represented as a string of words. *D* is determined to be relevant only if it contains some words that match *Q*. Accordingly we name such words as entity mentions. Both *Q* and *D* are preprocessed by tokenization and stop-word filtering. Other commonly used preprocessing steps (stemming and lemmatization) are disabled because they may cause confusion between entity mentions and common words. Table 1 shows the examples where the underlined words in <1> denotes an entity mention but <2> does not.

Mark Fisher (PER) <ID: SF14_ENG_031 >
<1> *Mark Fisher*, Sea Org member After stemming: ***mark fish***
<2>How to *mark fishing* landmarks? After stemming: ***mark fish***
3rd Guard Division (ORG) <ID: SF14_ENG_085 >
<1>The 3rd *Guard Division* of People's Liberation Army of China. After lemmatization: ***guard divide***
<2>The 24 *guards divide* up into 2 groups. After lemmatization: ***guard divide***

Table 1: Inappropriate use of preprocessing

We employ two matching methods for the relevance determination: exact and fuzzy matching.

Exact matching (EM) requires that a sequence of successive words in *D* exactly matches *Q*. By EM, entity archiving regards a full entity name as an indivisible word-order-fixed unit. Accordingly it only acquires the documents which contain the entity mentions in the form of completely-preserved full name.

Fuzzy matching (FM) relaxes the conditions to a large extent, allowing *Q* to be split into nonadjacent words. In particular, it supports the change in word order as well as partial match. By FM, entity archiving is able to retrieve documents that contain the entity mentions in the form of separated, pruned and/or reordered names. Table 2 shows some examples of using these matching methods, where the mark "•" denotes the available methods for a certain form of entity mention.

Mark Fisher (PER) <ID: SF14_ENG_031 >
Mark Fisher, Sea Org member. (**exact**) Availability: EM (•) FM (•)
Mark, husband of Julie *Fisher*. (**separation**) *Fisher* had been Miscavige's aide for 7 years.(**pruning**) *Fisher*'s first name, *Mark*, is impressive due to his inconceivable career change. (**reordering**) Availability: EM ( ) FM (•)

Table 2: Examples of string matching results

EM and FM have substantially different advantages and disadvantages in entity archiving. Table 3 shows the performance of EM and FM based entity archiving on KBP corpus (Surdeanu, 2013). We will introduce the corpus in details in Section 5. EM yields precise archiving results because the constraint conditions are helpful to reduce uncertainty in string matching. In contrast FM-based archiving is able to match entity name mentions with various forms, and thus it achieves higher recall.

	Precision	Recall	F-measure
EM	**72.5**	52.6	61.0
FM	10.8	**86.8**	19.2

Table 3: Effects of EM and FM on archiving

However FM generally introduces much noise, namely those mistakenly retrieved irrelevant documents. The documents are recalled because some pseudo entity mentions they contain can easily satisfy the constraints of fuzzy matching. See examples of the pseudo mentions in Table 4. As a result, FM yields a very low precision score.

Mark Fisher (PER) <ID: SF14_ENG_031 >
PlantWeb is a *mark* of the *Fisher*-Rosemount group of companies. (**separation**)
Deutsche *Mark* was the currency in Germany (**pruning**)
Iconic *Fisher*-Price *mark*. (**reordering**)

Table 4: Examples of pseudo entity mentions obtained by using FM

Undoubtedly it is helpful for global optimization of entity archiving to make full use of the advantages of EM and FM. In view of the above-mentioned investigation, we partition the string matching results into two parts, exact and fuzzy ones, which are used as reliable prior knowledge (named *reference source*) and unrefined prior knowledge (*candidate source*) respectively. Most documents in the *reference* are truly related to the target entity but the scale is not big (see Recall of EM in Table 3), while the *candidate* is full of both true answers and noise (see Precision and Recall of FM in Table 3), respectively. As we will show in the next section, the final archiving results are generated by synthesizing the sources in a collaborative coarse-to-fine way.

4 Collaborative Entity Archiving (CEA)

We propose a Collaborative Entity Archiving approach (CEA for short). CEA synthesizes the reference source and candidate source in a collaborative manner (section 4.1) through a biography-document relevance determination method (section 4.2 and 4.3). In addition, CEA involves mention disambiguation and query expansion in pre-processing to optimize the quality of both sources (section 4.4)

4.1 Overall Framework of CEA

CEA models the biography of an entity by using the topics in the reference source, in which, each topic serves as the description of a slice of life of the target entity (*life slice* for short), as shown in Figure 2. The *Life slice* means an episode in the whole story of the entity, which may represent an event, state or scenario at a certain moment, such as a person's *birth* or an organization's *establishment*.

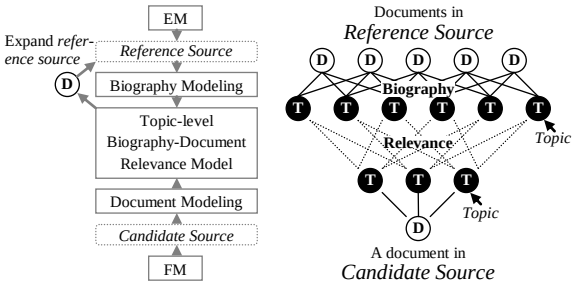


Figure 2**:** Framework of collaborative archiving

CEA pulls out a document from the candidate source, one by one, and measures the biography-document relevance at the topic level. By using a relevance threshold as the discrimination factor, CEA either preserves the document if it is relevant, or filters otherwise. Meanwhile, CEA adds the newly found relevant documents to the reference source, and updates the biography by reshaping life slices (i.e., topics). CEA iteratively goes through the process of biography formulation, biography-document relevance measurement and determination until a condition is satisfied. Finally CEA selects all the preserved documents in the reference source as the final output. Figure 2 shows the framework.

4.2 Biography-Document Relevance Models

We design a generative approach to estimate the biography-document relevance *r*, which calculates the conditional probability that a candidate document *D* generates the biography *B*:

$$r = P(B|D) \quad (1)$$

In total we leverage three probabilistic models for modeling *B* and *D*, including relevance model, topic model and context-level topic model. Then we introduce Hellinger Distance (Lindsay, 1994) into relevance measurement.

- *Relevance Model* **(*RM*)**

Generally, *Relevance Model* (RM) (Huang and Croft, 2009) refers to the probability distribution over all words conditioned on their occurrences in a set of previously-known relevant documents (or high-quality pseudo-relevant documents), i.e., $\forall w \in V, P(w|R)$, where *V* is the vocabulary, *R* is the document set, and *P*(*w*|*R*) can be estimated by TF-IDF. RM is often used in combination with *Document Model* (DM). Similar to RM, DM refers to the probability distribution over words in a particular document, i.e., $\forall w \in V, P(w|D)$. The relevance between *R* and *D* is normally determined by the agreement of RM and DM. The agreement can be estimated with Hellinger Distance between the models:

$$H(RM|DM) = \sum_{w \in V} (\sqrt{P(w|R)} - \sqrt{P(w|D)})^2 \quad (2)$$

RM is a widely-used probabilistic model for information retrieval. It determines the relevance of a document to an object in accordance with homogeneousness in content between the document and the relevant documents of the object.

For an entity, in our case, we generate RM on the reference source, and regard it as the probabilistic model of a macro-level all-embracing biography *B* over the prior knowledge *R*. For a candidate document *D*, the biography-document relevance *r* is measured with Hellinger Distance between RM and DM: *P*(*B*|*D*)=*H*(*RM*|*DM*). We

will demonstrate the effect of RM heavily relies on the quality of reference source in experiments.

- *Topic Model* **(*TM*)**

Empirically, RM is coarse-grained. It mixes up different, separate and incoherent life slices of an entity. A more serious problem is that RM assigns uneven weights to life slices, giving excessive weights to the words about the popular slices, but low or even zeroth weights to the unpopular ones. A popular slice is defined as the slice of greater concern, which is normally frequently mentioned in the reference source, such as the slice of "*the career of George W. Bush* as *the President*" (high weight) versus "*his childhood*" (low weight). As a result, the RM based biography-document relevance is only helpful to identify and recall the documents relevant to the popular slices but not to the unpopular ones.

As a modification, we employ *Topic Model* (TM) for biography modeling. We define a topic in the reference source as an independent fine-grained representation for a microscopic life slice. Accordingly we treat the biography as a bucket of topics. We leverage Latent Dirichlet Allocation (LDA) (Blei et al, 2003; Wei and Croft, 2006) for topic discovery and modeling in the reference source. A topic is modeled as a probability distribution over all words in lexicon conditioned on the association of the words with the topic, denoted as $\forall w \in V$, $P(w|t_R)$, in which t_R refers to a topic in the reference source, representing a life slice *s*. Table 5 shows partial topic models (slices) in the reference source of *Mark Fisher*, where the highlighted probability values by a box reveal the words that well characterize a topic (slice). In the same way, we survey the topics t_C in the *candidate source*, modeled as $\forall w \in V$, $P(w|t_C)$. It is worth noting that those topics (t_C) may represent anything, namely the slices of the target entity or namesakes, related or unrelated events, etc. It means they are full of noise.

Mark Fisher (Slice Modeling) <ID: SF14_ENG_031>			
Slice 1(***s1***), topic t_{s1}: ***political*** *career*			
$P(w\|t_{s1})=0.003$ "*Parliament*"	$P(w\|t_{s1})=8E-5$ "*screenwriter*"	$P(w\|t_{s1})=8E-5$ "*film*"	$P(w\|t_{s1})=4E-5$ "*Bear Fisher*" …
Slice 2(***s2***), topic t_{s2}: ***artistic*** *career*			
$P(w\|t_{s2})=6E-5$ "*Parliament*"	$P(w\|t_{s2})=0.003$ "*screenwriter*"	$P(w\|t_{s2})=0.001$ "*film*"	$P(w\|t_{s2})=3E-5$ "*Bear Fisher*" …
Slice 3(***s3***), topic t_{s3}: ***family***			
$P(w\|t_{s3})=9E-5$ "*Parliament*"	$P(w\|t_{s3})=1E-5$ "*screenwriter*"	$P(w\|t_{s3})=5E-6$ "*film*"	$P(w\|t_{s3})=0.001$ "*Bear Fisher*" …

Table 5: Example of life slice modeling by TM

In practice, given a target entity, its reference source (exact matching results) is a subset of the candidate source (fuzzy matching results). We picked the *reference source* out of the candidate to parse topics independently, forming the set of t_R. Meanwhile, we parse topics in the candidate source to form the set of t_C. Benefitting from the separate treatment, some of the truly related topics (t_R) to the entity (correct slices) can be collected along with less noise. Using the topics as references, we detect the relevant documents in the *candidate source* in terms of the topic-level biography-document relevance $P(B|D)$.

Given a document D in the candidate source, we transform $P(B|D)$ into the combination of topic-document relevance of all topics in the *reference source*. We measure the topic-document relevance with the conditional probability $P(t_R|D)$ that the topic t_R occurs in the document D. Accordingly, $P(B|D)$ is estimated by:

$$r = P(B \mid D) = \prod_{t_R} P(t_R \mid D)$$
$$\log(r) = \sum_{t_R} \log P(t_R \mid D) \quad (3)$$

where, we incorporate the log likelihood into the numerical calculation for the sake of nonzero joint probability.

Due to the separate topic modeling procedures for the reference and candidate sources, the probability $P(t_R|D)$ − a topic t_R in the *reference source* occurs in a candidate document D − cannot be obtained directly. To solve the problem, we introduce the joint probability of topic-topic relevance between topics (t_R) in reference and topics (t_C) in candidate (see the mode in Figure 2) into the probability calculation:

$$\forall t_R \in T_R,\ P(t_R \mid D) = \prod_{t_C \in T_C} P(t_R \mid t_C) P(t_C \mid D) \quad (4)$$

where T_R is the set of all topics in the reference source while T_C is the candidate. The topic-topic relevance $P(t_R|t_C)$ is approximated by Hellinger distance estimation between the topic models of t_R and t_C. As a whole, we measure the biography-document relevance as:

$$\begin{aligned} \log(r) &= \log \prod_{t_R \in T_R} \prod_{t_C \in T_C} P(t_R \mid t_C) P(t_C \mid D) \\ &= \sum_{t_R \in T_R} \sum_{t_C \in T_C} \log P(t_R \mid t_C) P(t_C \mid D) \\ &= \sum_{t_R \in T_R} \sum_{t_C \in T_C} \log H(t_R \mid t_C) P(t_C \mid D) \\ &= \sum_{t_R \in T_R} \sum_{t_C \in T_C} \sum_{w \in V} \log(\sqrt{P(w \mid t_R)} - \sqrt{P(w \mid t_C)})^2 P(t_C \mid D) \end{aligned} \quad (5)$$

We employ the toolkit GibbsLDA++[2] in topic modeling, which is an implementation of LDA using Gibbs sampling (Porteous et al, 2008). GibbsLDA++ makes it easy to parse the topics in a document set as well as estimate topic models $P(w|t)$. Besides, GibbsLDA++ offers the probability over topics in generating a specific document, facilitating the estimation of $P(t_C|D)$ in equation (5). Table 6 shows the operating parameters what we set in experiments, where the ones $\{\alpha, \beta\}$ were set as the default values while the iterative number *num* is an empirical value.

α= 1	*num*= 200
β= 0.1	N_t←HDP

Table 6: Operating parameters of GibbsLDA++

The necessary precondition for GibbsLDA++ in topic partition is to define the number N_t of potential topics in a set of documents. We execute the Hierarchical Dirichlet Processes (Teh et al, 2005), abbr., HDP, to predict N_t. HDP is similar to current hierarchical information organization methods, such as the hierarchical clustering (Kummamuru et al, 2004), unsupervised and coarse-to-fine grained. Hence HDP is useful in exploring the basic rules of topic partition in an automatic way, such as number and granularity. We employ HDP to estimate the number (N_t) of all possible topics in *reference source* and *candidate* separately, acquiring two N_t for each target entity, one per source.

- *Context-level Topic Model* **(*CTM*)**

In consideration of the reliability of contexts in representing closely-related life slices to the entity, we use the contexts around entity mentions to improve the slice-oriented topic modeling.

SEN: A sentence where an entity mention occurs **NEB**: Left and right neighbor sentences of SEN **DEP**: Words dependent on entity mention **SYN**: Words in maximum syntactic subtree in SEN
Context 1: SEN **Context 2**: DEP + SYN **Context 3**: SEN + left NEB + Right NEB

Table 7: Instructions of various types of contexts

A context consists of the words co-occurring with an entity mention in a radius-fixed text span or syntactic or dependent structure (see instructions in Table 7). Given a target entity, the entity mention in the *reference source* is its full name. The union of all contexts in the source defines the vocabulary V_R that most probably represent the slices of the entity. In the *candidate sourc*e, on the contrary, the entity mention can be a reordered, separated or pruned entity name, as well as abbreviation or alias, such as *GWB* (abbr.) and *Dubya* (alias) of *George W. Bush*. Different from the cases in *reference*, the vocabulary V_C obtained from the contexts in *candidate* are closely related to diverse entities or other objects with the same name (see Table 2&4).

CTM measures the biography-document relevance in the same way with *TM*, estimating the topic-level $P(B|D)$ by equation (5). The only difference lies in the available words in topic model $P(w|t)$. For the ones not included in V_R and V_C, CTM assigns a weight zero in topic model no matter what GibbsLDA++ does.

4.3 Unsupervised Threshold Estimation

For each target entity, CEA measures the biography-document relevance for all documents in the candidate source. In the light of the relevance scores, CEA ranks the documents and sets a clear threshold θ to cut off the long tail in the ranking list, in other words, filtering the documents that have a relevance score lower than θ. The preserved documents will be added to the reference source for both biography reformulation and archiving result output.

We estimate the threshold by learning density distribution of documents over relevance scores (Arampatzis et al, 2009). *Density* means the number of documents that have similar relevance scores. The *distribution* is produced by densities within all interval ranges of relevance score. Our empirical findings show that the density distribution fits a mixture of two Gaussians, where the highly relevant documents and the irrelevant ones distribute in two separate Gaussian peaks respectively. Accordingly we define the threshold as the range of relevance score at the extreme point between the peaks, as shown in Figure 3.

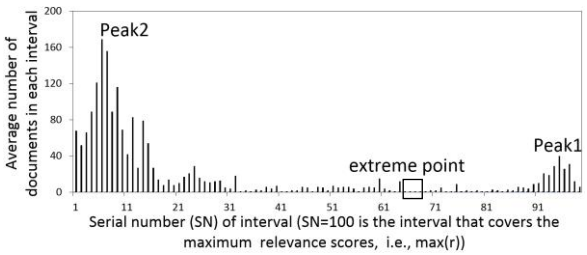


Figure 3: Extremum detection for threshold selection. (Note: Y-axis indicates the density in a specific interval range of relevance score)

In order to detect the extreme point, we firstly use a cubic polynomial function to approximate

2 http://gibbslda.sourceforge.net/

the density distribution. Second, we go through the integral solution of the function in every fine-grained interval range of relevance score (interval is set as max(r)/100). We finally detect the extremum between peaks. The threshold is initialized during runtime exclusively for each target entity, without training. It is re-estimated every time when the biography is reformulated.

4.4 Termination Criterion for Iteration

CEA identifies relevant documents in *candidate source* and moves them to *reference*. Then CEA reshapes statistical models (RM, TM or CTM) over the updated sources. In terms of the reformed models, CEA starts a new round of relevance determination, data movement, and statistical modeling. CEA keeps it going until meeting any of the following termination criterions:

- T1: No more new topic occurs in the *reference source* (N_t doesn't change).
- T2: The number of the documents in Peak1 (Figure 3) begins to increase continuously.

T2 is triggered if T1 loses its efficacy. The invalidation happens when some general slices (i.e., general topics) are mistakenly introduced into the *reference source*, causing large-scale irrelevant document to be recalled and moved to *reference*. It will dramatically increase the number (N_t) of topics in a long term in the iterative procedure, driving CEA to capture more irrelevant documents. Thus Peak1 will be enlarged continuously. However, if as expected, Peak1 should be narrowed with increasing the iteration times because:

- Fewer new related slices appear.
- The number of documents related to the slices is less than that in previous iterations.

4.5 Optimization of EM and FM

In the preprocessing phase, we improve the precision of EM because higher-quality EM results can offer more reliable reference documents for biography modeling. In addition, we expand queries for FM to recall a larger number of relevant documents. It is helpful to minimize the loss of relevant documents before proceeding to CEA.

To improve EM, we focus on identifying the common words that completely match the full name of the target entity. The words normally are elusive and easily treated as a correct entity name, called deceptive name, see that in (1).

(1) *Countrywide Financial* <ORG>
True: *Countrywide Financial Corporation*.
Deceptive: *Bank of America purchased the failing countrywide financial for $4.1 billion*.

To reduce EM errors caused by deceptive names, we use name tagging (Miller et al, 2004) to distinguish deceptive names and true names. Further, we filter the documents that are mistakenly retrieved based on the match between a deceptive name and the full-entity-name based query Q.

We leverage an *Alternate Name Table* (ANT) for query expansion. ANT is a mapping table between entity name and alternate name. An alternate name is either generated according to the naming conventions (Burman et al, 2011), such as abbreviation, suffixation and revivification. Some alternative names were extracted from knowledge base through redirect links (Nia et al, 2014), such as nicknames in Wikipedia dumps. For an entity, we reformulate query Q by the combination of the pre-assigned full entity name and all possible alternate names in ANT, see (2).

(2) **Initial** Q: *Countrywide Financial Corporation.*
Expanded: *Countrywide Financial+Corporation +Corp. +Company +Co. +Ltd. +Co Ltd. +CFC.*

We use the expanded query for FM to increase the number of relevant documents in the candidate source, regardless of whether or not it will introduce a larger scale of new noises.

5 Experiments

We evaluate our methods on KBP 2013 corpus. The corpus contains 2.1M texts collected from web pages, newswires and discussion forums. From this corpus, a slot filling system is required to find fillers for 41 types of slots that represent the attributes of the target entities. There are 25 slot types of person and 16 slot types of organization, such as a Person's *birth date* and an Organization's *founder* (Ji et al., 2010 and 2011).

KBP 2013 includes 100 target entities and ground-truth fillers and provenances, where the ground-truth data was obtained by manual verification and annotation on the pool of system outputs. The provenances contain the IDs of documents relevant to target entities and fine-grained text spans which illustrate the eligibility of fillers.

In total there are 1,851 gold standard relevant documents available for the evaluation of entity archiving. However the data is far from enough because it only covers a small portion of all relevant documents in the pool. Most are excluded since KBP annotators ignore relevant documents in which there isn't any filler for the assigned slots or, although exists, the fillers were inexactly identified by Slot Filling systems. Therefore, we manually went over the pool and extracted 4,405 relevant documents as our ground-truth.

Sources / Archiving	**Before** Optimization of EM & FM (%)						**After** Optimization of EM & FM (%)					
	Micro-Average			Macro-Average			Micro-Average			Macro-Average		
	P	R	F	P	R	F	P	R	F	P	R	F
EM	65.4	40.1	49.7	72.5	52.6	61.0	78.8	24.6	37.5	81.1	46.9	59.4
FM	9.4	74.6	16.7	10.8	86.8	19.2	6.3	92.9	11.8	6.9	94.2	12.9
Baseline	29.1	50.6	36.9	36.1	62.9	45.9	22.6	33.3	26.9	25.1	52.6	34.0
CEA(RM)	49.7	86.5	63.1	62.3	85.8	72.2	59.5	84.3	69.7	64.9	87.4	74.5
CEA(TM)	52.6	87.7	65.8	63.2	86.1	72.9	60.6	82.9	70.0	65.7	85.8	74.4
CEA(CTM1)	63.9	81.7	71.7	69.7	82.7	75.7	65.1	75.6	70.0	69.8	84.4	76.4
CEA(CTM2)	63.9	81.7	71.7	69.7	82.7	75.7	65.1	75.6	70.0	69.8	84.4	76.4
CEA(CTM3)	61.8	84.3	71.4	68.1	83.8	75.1	66.2	71.2	68.6	70.6	77.7	74.0

Table 8: Archiving results (CTM1, 2 &3 are CTM using different types of contexts, context1, 2 &3 in Table 7)

5.1 Archiving Results and Analysis

We evaluate the entity archiving methods by micro and macro Precision (P), Recall (R) and F metrics. Table 8 shows the main results.

- *Overall Archiving Results*

Overall, the proposed CEA methods perform much better than the string matching based entity archiving methods (i.e., EM and FM).

In addition the methods outperform a random-sampling based CEA (baseline), which randomly selects a certain number of documents (candidates) from the candidate source to combine with reference source straightforwardly (for final archiving results generation). The sampling number is set to be the same as the number of candidates eventually archived by RM-based CEA.

Random	RM	TM	CTM1	CTM2	CTM3
5,901	5,901	5,578	3,866	3.866	4,243
5,158	5,158	4,943	4,032	4,032	3,655

Table 9: The number of candidates added to reference source for archiving results generation (the second row indicates the number before optimizing EM and FM, while the third row indicates after optimization)

Table 9 shows the number of candidates archived from the candidate source by all kinds of CEA methods. It demonstrates that the biography-based CEAs yield higher precision (Table 8) after introducing the same or smaller number of candidates in the reference source, revealing the positive effect of biography modeling on entity-oriented document relevance determination.

- *Reliability versus Comprehensiveness*

CEA achieves higher precision by using the optimized EM results as reference source. It demonstrates the importance of reliable prior knowledge for entity understanding as well as detecting relevant documents. However, the reference source causes lower recall scores of all CEA methods. The reason lies in reduction of prior knowledge. As shown in Table 8, the refined reference source (i.e., optimized EM results) covers only 24.6% of all relevant documents, which is far less than the coverage before optimization (nearly 40%).

The reduced prior knowledge provides fewer available life slices of an entity for constructing an informative biography, inevitably resulting in missing some relevant documents. In order to confirm this, we regard the 41 KBP slot types as some readily-made visible life slices, and use the manual annotations of the slot fillers to verify whether a life slice appears in a relevant document. For example, the filler "*Corporate liaison* of the slot *Title* reveals the slice of *freelance career* of *Mark Fisher*. Then we figure out the coverage rate of life slices for both the original reference source and the refined.

Figure 4 exhibits the coverage rates for 5 most frequently occurred life slices. The coverage rate is calculated by the number of reference sources that contain a specific life slice versus 100, i.e., the number of reference sources for the 100 KBP entities (one per entity). It can be found that the refined reference sources miss lots of life slices.

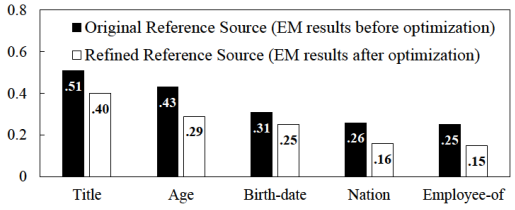


Figure 4: Coverage rates of life slices (for top 5)

- *Comparison among Biography Models*

RM is biased towards the popular life slices in biography modeling. The reasons are as following: 1) RM gives greater weights to the high-frequency words, and 2) popular slices are of much greater public interest and hence frequently mentioned in relevant documents. However, some entities not only share similar names but similar popular slices, such as the *religious vocation* of different *church scientologists*. Therefore

RM is extremely likely to acquire the documents related to the namesakes if they have similar popular background as the target entity, causing a greater loss of precision.

Table 10 shows the top highly-weighted words in RM for the target *Mark Fisher* (a *church scientologist*), along with 2 namesakes who occur most frequently in the incorrect archiving results.

Keywords in RM	The most similar entities
church *committee* *religious* *Sea Org* *policy*	***Miles Mark Fisher*** *Church historian, Educator, Baptist minister and writer* ***Mark Fisher*** *Senior Pastor*

Table 10: Entities of similar background

By contrast, TM independently represents different life slices and combines the effects of the slices on biography-document relevance determination, evenly and exhaustively. Comprehensive and unbiased measurement of every known life slices is helpful in disambiguating entities that have similar backgrounds (definitely not the same in all). As a result, TM improves the precision. And the context-based TM goes further.

5.2 Slot Filling Results and Analysis

We apply our entity archiving methods to two top-ranked slot filling systems in the evaluation of KBP 2013, including LSV (Roth and Klakow, 2013) and Blender (Yu et al 2013).

The LSV incorporates a string matching based entity archiving and a SVM classifier based filler extraction. LSV's archiving model expands queries by using suffixes and Wikipedia anchor texts, and uses mutual information based relevance measure in document ranking and filtering.

Blender employs a hybrid retrieval model for archiving relevant documents. It combines Boolean and VSM models and expands query by an alternate name table similar to ours. For filler extraction, Blender implements truth finding over conflicting claims from multiple rule-based extraction systems.

Methods	P	R	F
LSV's archiving	53.0	88.2	66.2
Blender's archiving	54.6	71.7	62.0

Table 11: Archiving performance of LSV &BLD

Table 11 shows entity archiving performances of LSV and Blender (Macro-Average P, R and F). All CEA methods perform better than the both. With the aim to optimize provenances of fillers, we modify the slot filling systems by substituting their archiving methods with ours. Table 12 exhibits the performance gains after replacement.

Slot Filling / Archiving	LSV (%)			Blender (%)		
	P	R	F	P	R	F
Original system	40.8	30.0	34.6	34.1	22.1	26.8
Mod. (RM)^{CEA}	42.1	30.0	35.0	35.6	23.9	28.6
Mod. (TM)^{CEA}	**42.2**	**30.6**	**35.5**	35.6	23.9	28.6
Mod. (CTM3)^{CEA}	42.7	29.3	34.7	**36.0**	**23.8**	**28.7**

Table 12: Slot filling performance gains

Both LSV and Blender achieve significant gains. The most interesting finding is on the different performance gains. It should reveal the fact that the well-supervised classification based filler extraction of LSV has a better capability of noise resistance, while by contrast, the truth-finding in Blender is capable of identifying valid fillers if the quality of archiving results is high, otherwise easily makes mistake.

6 Conclusion

We doubt that it is easy to maintain the stability of current entity-oriented knowledge acquisition methods, including ours, in dealing with ordinary entities. Most target entities now in use for the evaluation are made to stand as "out of the ordinary", such as well-known enterprises, celebrities or domain experts. As a result, a corpus contains abundant relevant documents of the entities but less about the little-known namesakes. It greatly reduces the interference of namesakes and thus the difficulty of the task.

In future work, we will make the task critical for success by employing the little known namesakes as targets. In addition to verifying the robustness of the CEA method, we will work on the relationship among entities (ACE entity relation types, Doddington et al, 2004) and related events (e.g., causal, temporal and sub-event relations), by which to build graph-based biography.

7 Acknowledgment

This work was supported by the U.S. DARPA DEFT Program No. FA8750-13-2-0041, ARL NS-CTA No. W911NF-09-2-0053, NSF CAREER Award IIS-1523198, AFRL DREAM project, gift awards from IBM, Google, Disney and Bosch. It was also supported by Natural Science Foundation of China (NSFC) No. K111818713, K111818612. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. and CHN Governments. The U.S. and CHN Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

References

Gabor Angeli, Sonal Gupta, Melvin Jose, Christopher D. Manning, Christopher Ré, Julie Tibshirani, Jean Y. Wu, Sen Wu, and Ce Zhang. 2014. Stanford's 2014 slot filling systems. In Proceedings of *the 7th TAC*.

Avi Arampatzis, Jaap Kamps, and Stephen Robertson. 2009. Where to stop reading a ranked list: threshold optimization using truncated score distributions. In Proceedings of *the 32nd SIGIR*, pages 524-531, Boston, Massachusetts, USA, July.

Javier Artiles, Qi Li, Taylor Cassidy, Suzanne Tamang, and Heng Ji. 2011. CUNY BLENDER TAC-KBP2011 temporal slot filling system description. In Proceedings of *the 4th TAC*.

Bruce G. Lindsay. 1994. Efficiency versus robustness: the case for minimum hellinger distance and related methods. *Annals of Statistics*, 22(2): 1081-1114.

Krisztian Balog, Lan Soboroff, Paul Thomas, Peter Bailey, Nick Craswell, and Arjen P. De Vries. 2008a. Overview of the TREC 2008 enterprise track. In Proceedings of *the 17th TREC*, NIST.

Krisztian Balog, Wouter Weerkamp, and Maarten De Rijke. 2008b. A few examples go a long way: constructing query models from elaborate query formulations. In Proceedings of *the 31st SIGIR*, pages 371-378, Singapore, July.

Krisztian Balog and Maarten De Rijke. 2008. Associating people and documents. Advances in Information Retrieval: 296-308

Krisztian Balog, Marc Bron and Maarten De Rijke. 2011. Query modeling for entity search based on terms, categories, and examples. *ACM Transaction on Information System*, 29(4): 22:1-22:31.

Krishna Kummamuru, Rohit Lotlikar, Shourya Roy, Karan Singal, and Raghu Krishnapuram. 2004. A hierarchical monothetic document clustering algorithm for summarization and browsing search results. In proceedings of *the 13rd WWW*, pages 658-665.

Roi Blanco, Peter Mika, Sebastiano Vigna. 2011. Effective and efficient entity search in RDF data. *The Semantic Web-ISWC*: 83-97.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet Allocation. *Journal of Machine Learning Research* 3: 993-1022.

Marc Bron, Krisztian Balog, and Maarten De Rijke. 2013. Example based entity search in the web of data. *Advances in Information Retrieval*: 392-403.

Amev Burman, Arun Jayapal, Sathish Kannan, Madhu Kavilikatta, Ayman Alhelbawy, Leon Derczynski, and Robert Gaizauskas. 2011. USFD at KBP 2011: entity linking, slot filling and temporal bounding. In Proceedings of *the 4th TAC*.

Lorna Byrne and John Dunnion. 2010. UCD IIRG at TAT 2010 KBP slot filling task. In Proceedings of *the 3rd TAC*, Maryland, USA, November.

Zheng Chen, Suzanne Tamang, Adam Lee, Xiang Li, Marissa Passantino, and Heng Ji. 2010. Top-down and bottom-up: a combined approach to slot filling. In Proceedings of *the 6th AIRS*, pages 300-309, Taipei, Taiwan, December.

Grzegorz Chrupala, Saeedeh Momtazi, Michael Wiegand, and Stefan Kazalski. 2010. Saaland university spoken language systems at the slot filling task of TAC KBP 2010. In Proceedings of *the 3rd TAC*.

Gianluca Demartini, Tereza Iofciu, and Arjen P. De Vries. 2010. Overview of the INEX 2009 entity ranking track. In Proceedings of *the 8th International workshop of the Initiative for the Evaluation of XML Retrieval*, pages 254-264.

George Doddington, Alexis Mitchell, Mark Przybocki, Lance Ramshaw, Stephanie Strassel, and Ralph Weischedel. 2004. *The ACE program-task, data, and evaluation*. LREC.

Sanyuan Gao, Yichao Cai, Si Li, Zongyu Zhang, Jingyi Guan, Yan Li, Hao Zhang, Weiran Xu, and Jun Guo. 2010. PRIS at TAC2010 KBP track. In Proceedings of *the 3rd TAC*.

Ralph Grishman and Bonan Min. 2010. New York University KBP 2010 slot filling system. In Proceedings of *the 3rd TAC*.

Guoping Hu, Jingjiang Liu, Hang Li, Yunbo Cao, Jian-Yun Nie, and Jianfeng Gao. 2006. A supervised learning approach to entity search. In Proceedings of *the 3rd AIRS*, pages 54-66, Singapore, October.

Xuanjing Huang and W. Bruce Croft. 2009. A unified relevance model for opinion retrieval. In Proceedings of *the 18th CIKM*, pages 947-956, Hong Kong, China, November.

Jiepu Jiang, Wei Lu, Xianqian Rong, and Yangyan Gao. 2009. Adapting language modeling methods for expert search to rank Wikipedia entities. *Lecture Notes in Computer Science*, 5631: 264-272.

Heng Ji, Ralph Grishman, Hoa Trang Dang, Kira Griffitt, and Joe Ellis. 2010. Overview of the TAC2010 knowledge base population track. In Proceedings of *the 3rd TAC*, Maryland, USA, November.

Heng Ji and Ralph Grishman. 2011. Knowledge base population: successful approaches and challenges. In Proceedings of *the 49th ACL*, pages 1148-1158, Portland, Oregon, USA, June.

Heng Ji, Ralph Grishman, and Hoa Trang Dang. 2011. Overview of the TAC2011 knowledge base population track. In Proceedings of *the 4th TAC*.

Rianne Kaptein and Jaap Kamps. 2009. Finding entities in Wikipedia using links and categories. *Lecture Notes in Computer Science*, 5631: 273-279.

Bryan Kisiel, Justin Betteridge, Matt Gardner, Jayant Krishnamurthy, Ndapa Nakashole, Mehdi Samadi, Partha Talukdar, Drry Wijaya, and Tom Mitchell. 2013. CMUML system for KBP 2013 slot filling. In Proceedings of the *6th TAC*.

John Lehmann, Sean Monahan, Luke Nezda, Arnold Jung, and Ying Shi. 2010. LCC approaches to knowledge base population at TAC 2010. In Proceedings of the *3rd TAC*.

Ludovic Jean-Louis, Romaric Besançon, Olivier Ferret, and Wei Wang. 2011. Using a weakly supervised approach and lexical patterns for the KBP slot filling task. In Proceedings of *the 4th TAC*.

Scott Miller, Jethran Guinness, and Alex Zamanian. 2004. Name tagging with word clusters and discriminative training. In Proceedings of *HLT-NAACL 2004*, pages 337-342, Boston, Massachusetts, USA, May.

Robert Neumayer, Krisztian Balog, and Kjetil Nørvåg. 2012. On the modeling of entities for ad-hoc entity search in the web of data. *Advances in Information Retrieval*: 133-145.

Morteza Shahriari Nia, Christan Grant, Milenko Petrovic, Yang Peng, and Daisy Zhe Wang. 2014. In proceedings of *the 27th FLAIRS*, pages 467-472, Pensacola Beach, Florida, USA, May.

Desislava Petkova and W. Bruce Croft. 2007. Proximity-based document representation for named entity retrieval. In Proceedings of *the 16th CIKM*, pages 731-740, Lisboa, Portugal, November.

Lan Porteous, David Newman, Alexander Ihler, Arthur Asuncion, Padhraic Smyth, and Max Welling. 2008. Fast collapsed gibbs sampling for latent dirichlet allocation. In Proceedings of *the 14th SIGKDD*, pages 24-27, Las Vegas, USA, August.

Benjamin Roth and Dietrich Klakow. 2013. Combining generative and discriminative model scores for distant supervision. In Proceedings of *the EMNLP*, pages 24-29, Seattle, Washington, USA, October.

Benjamin Roth, Tassilo Barth, Michael Wiegand, Mittul Singh and Dietrich Klakow. 2014a. Effective slot filling based on shallow distant supervision methods. *Arxiv preprint arxiv*: 1401-1158.

Pavel Serdyukov and Djoerd Hiemstra. 2008. Being omnipresent to be almighty: the importance of the global web evidence for organizational expert finding. In Proceedings of *the 31s SIGIR*, pages 17-24, Singapore, July.

Ang Sun, Ralph Grishman, Wei Xu, and Bonan Min. 2011. New York University 2011 system for KBP slot filling. In Proceedings of *the 4th TAC*.

Mihai Surdeanu, Sonal Gupta, John Bauer, and David McClosky. 2011. Stanford's distantly-supervised slot filling system. In Proceedings of *the 4th TAC*.

Mihai Surdeanu. 2013. Overview of the TAC2013 knowledge base population evaluation: English slot filling and temporal slot filling. In Proceedings of *the 6th TAC*.

Mihai Surdeanu and Heng Ji. 2014. Overview of the English slot filling track at the TAC2014 knowledge base population evaluation. In Proceedings of *the 7th TAC*.

Yee Whye Teh, Michael I. Jordan, Matthew J. Beal, and David M. Blei. 2005. Hierarchival Dirichlet Processes. *Journal of the American Statistical Association* 101(476): 1-29.

Anne-Marie Vercoustre, James, A. Thom, and Jovan Pehcevski. 2008a. Entity ranking in Wikipedia. In Proceedings of *the 23rd SAC*, pages 1101-1106, Fortaleza, Ceara, Brazil, March.

Vidhoon Viswanathan; Nazneen Fatema Rajani; Yinon Bentor; Raymond Mooney. 2015. Stacked Ensembles of Information Extractors for Knowledge-Base Population. In Proceedings of ACL 2015.

Wouter Weerkamp, Krisztian Balog, and Edgar Meij. 2009. A generative language modeling approach for ranking entiies. *Lecture Notes in Computer Science*, 5631: 292-299.

Xing Wei and W. Bruce Croft. 2006. LDA-based document models for ad-hoc retrieval. In Proceedings of *the 29th SIGIR*, pages 569-577, Seattle, Washington, USA, August.

Gae-won You, Seung-won Hwang, Zaiqing Nie, and Ji-Rong Wen. 2011. Social search: enhancing entity search with social network matching. In Proceedings of *the 14th EDBT*, pages 515-519, Uppsala, Sweden, March.

Dian Yu, Hongzhao Huang, Taylor Cassidy, Heng Ji, Chi Wang, Shi Zhi, Jiawei Han, Clare Voss, and Malik Magdon-Ismail. 2014. The wisdom of minority: unsupervised slot filling validation based on multi-dimensional truth-finding. In Proceedings of *the 25th COLING*, pages 1567-1578, Dublin, Ireland, August.

Dian Yu, Heng Ji, Sujian Li and Chin-Yew Lin. 2015. Why Read if You can Scan: Scoping Strategy for Biographical Fact Extraction. In Proceedings of NAACL-HLT 2015.

Jianhan Zhu, Dawei Song, and Stefan Rűger. 2008. Integrating document features for entity ranking. *Lecture Notes in Computer Science*, 4862: 336-347.

Jianhan Zhu, Xiangji Huang, Dawei Song, and Stefan Rűger. 2010. Integrating multiple document features in language models for expert finding. *Knowledge and Information System*, 23(1):29-54.

Stochastic Top-k ListNet

Tianyi Luo[1], **Dong Wang**[1,2], **Rong Liu**[1,3], **Yiqiao Pan**[1,4]
[1]CSLT, RIIT, Tsinghua University, China
[2]Tsinghua National Lab for Information Science and Technology
[3]Huilan Limited, Beijing, China
[4]Beijing University of Posts and Telecommunications, China
{lty, lr, pyq}@cslt.riit.tsinghua.edu.cn
wangdong99@mails.tsinghua.edu.cn

Abstract

ListNet is a well-known listwise learning to rank model and has gained much attention in recent years. A particular problem of ListNet, however, is the high computation complexity in model training, mainly due to the large number of object permutations involved in computing the gradients. This paper proposes a stochastic ListNet approach which computes the gradient within a bounded permutation subset. It significantly reduces the computation complexity of model training and allows extension to Top-k models, which is impossible with the conventional implementation based on full-set permutations. Meanwhile, the new approach utilizes partial ranking information of human labels, which helps improve model quality. Our experiments demonstrated that the stochastic ListNet method indeed leads to better ranking performance and speeds up the model training remarkably.

1 Introduction

Learning to rank aims to learn a model to rerank a list of objects, e.g., candidate documents in document retrieval. Recent studies show that listwise learning delivers better performance in general than traditional pairwise learning (Liu, 2009), partly attributed to its capability of learning human-labelled scores as a full rank list. A potential disadvantage of listwise learning, however, is the high computation complexity in model training, which is mainly caused by the large number of permutations of the objects to rank.

A typical listwise learning method is the ListNet model proposed by Cao et al. (2007). This model has been utilized to tackle many ranking problems, e.g. modeling the hiring behavior in online labor markets (Kokkodis et al., 2015), ranking sentences in document summarization (Jin et al., 2010), improving detection of musical concepts (Yang et al., 2009) and ranking the results in video search (Yang and Hsu, 2008). Basically, ListNet implements the rank function as a neural network (NN), with the objective function set to be the cross entropy between two probability distributions over the object permutations, one derived from the human-labelled scores and the other derived from the model prediction (network output). In order to deal with the high computation complexity associated with the large number of permutations, Cao et al. (2007) proposed a Top-k approach, which clusters the permutations by the first k objects, so the number of distinct probabilities that need to evaluate in model training reduces from $n!$ to $\frac{n!}{(n-k)!}$, where n is the number of objects in the list.

To ensure efficiency, $k = 1$ was selected in the seminal paper (Cao et al., 2007) and in the open source implementation of RankLib (Dang, 2013). This Top-1 approach is a harsh approximation to the full listwise learning and may constrain the power of the ListNet method. We therefore seek to extend the Top-1 approximation to Top-k (k $>$ 1) models.

The major obstacle for the Top-k extension is the large number of permutations, or more precisely, permutation classes in the Top-k setting. A key idea of this paper is that the rank information involved in the permutation classes is highly redundant and so a small number such permutation classes are sufficient to convey the rank information required to train the model. Meanwhile, the partial rank information associated with the subset of permutation classes may represent more detailed knowledge for model training, leading to better ListNet models.

Based on these two conjectures, we propose a stochastic ListNet method, which samples a subset

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 676–684,
Lisbon, Portugal, 17-21 September 2015.

of the permutation classes (object lists) in model training and based on this subset to train the ListNet model. Three methods are proposed to conduct the sampling. In the uniform distribution method, the candidate objects are selected following a uniform distribution; in the fixed distribution method, the candidate objects are selected following a distribution derived from the human-labeled scores; in the adaptive distribution method, the candidates are selected following a distribution defined by the rank function, i.e., the neural network output. Experimental results demonstrated that the stochastic ListNet method can significantly reduce the computation cost in model training. In fact, if the size of the permutation subset is fixed, the computation complexity is bounded, which allows training Top-k models where k is large. Meanwhile, better performance was obtained with the stochastic ListNet approach, probably due to the learning of partial rank information.

The contributions of the paper are three-fold: (1) proposes a stochastic ListNet method that significantly reduces the training complexity and delivers better ranking performance; (2) investigates Top-k models based on the stochastic ListNet, and studies the impact of a large k; (3) provides an open source implementation based on RankLib.

The rest of the paper is organized as follows. Section 2 introduces some related works, and Section 3 presents the stochastic ListNet method. Section 4 presents the experiments, and the paper is concluded by Section 6.

2 Related Work

This work is an extension of the Top-k ListNet method proposed by Cao et al. (2007). The novelty is that we propose a stochastic learing method which not only speeds up the model training but also produces stronger models. The code is based on the Top-1 ListNet implementation of RankLib (Dang, 2013).

Another related work is the SVM-based pairwise learning to rank model based on stochastic gradient descent (SGD) (Sculley, 2009). In this approach, training instances (queries) are selected randomly and for each query, a number of object pairs are sampled from the object list. These pairs are used to train the SVM model. In the stochastic ListNet method proposed in this paper, the randomly selected training samples are permutation classes (object lists) rather than pairs of objects, and a set of object lists rather than a single pair forms a training sample.

3 Methods

3.1 Review of ListNet

The ListNet approach proposed by Cao et al. (2007) trains a neural network which predicts the scores $z^{(i)}$ of a list of candidate objects $x^{(i)}$ given a query $q^{(i)}$, formulated by $z^{(i)} = f_w(x^{(i)})$, where f_w stands for the scoring function defined by the NN. The objective function is given by:

$$\begin{aligned}\mathcal{L} &= \sum_i \mathcal{L}(y^{(i)}, z^{(i)}) \\ &= \sum_i \sum_{\forall g \in \mathscr{G}_k} P_{y^{(i)}}(g) log(P_{z^{(i)}}(g)) \quad (1)\end{aligned}$$

where $y^{(i)}$ denotes the human-labelled scores, and $\mathscr{G}_k$ is the set of permutation classes defined by:

$$\begin{aligned}\mathscr{G}_k &= \{\mathscr{G}_k(j_1, j_2, ..., j_k) | j_t = 1, 2, ..., n, \\ &\quad s.t.\ j_u \neq j_v \quad for\ \ \forall u \neq v\} \quad (2)\end{aligned}$$

where n is the number of candidate objects, j_t is the object ranked at the t-th position, and $\mathscr{G}_k(j_1, j_2, ..., j_k)$ is a permutation class which involves all the permutations whose first k objects are exactly $(j_1, j_2, ..., j_k)$. Following Cao et al. (2007), the probability of $\mathscr{G}_k(j_1, j_2, ..., j_k)$ can be computed by:

$$P_s(\mathscr{G}(j_1, j_2, ..., j_k)) = \prod_{t=1}^{k} \frac{e^{s_{j_t}}}{\sum_{l=t}^{n} e^{s_{j_l}}}. \quad (3)$$

where s_{j_t} is the score of object at position $j_t (t = 1, 2, , , k)$ at a certain permutation. By this definition of permutation probability, Eq. (1) defines a cross entropy between the distributions over permutations (precisely, permutation classes) derived from the human-labelled scores and the NN-predicted scores. Therefore, optimizing the objective function Eq. (1) with respect to the NN model f_w leads to a scoring function that approximates the human-labelled ranking.

3.2 Stochastic Top-k ListNet

A particular difficulty of the Top-k ListNet method is that it requires very demanding computation in model training. Refer to Eq. (2), the permutation set $\mathscr{G}_k$ involves $\frac{n!}{(n-k)!}$ members, and for each member, computing its probability involves $\frac{(2n-k+1)k}{2}$ summations plus k multiplications and

divisions. To let the algorithm practical, k=1 was selected in (Cao et al., 2007), as well as the public toolkit RankLib (Dang, 2013). Although this is a good solution and reduces computation dramatically, we argue that this approach largely buries the power of ListNet. In fact, setting k=1 effectively marginalizes all the probabilities over the candidate objects of a permutation class except the top one. By this approximation, Eq. (3) reduces to a softmax over the candidate objects, which means that it actually focuses on how the probabilities are distributed over *individual* objects, rather than how the probabilities are distributed over *object lists*. This potentially loses much rank information involved in the human labels.

Another disadvantage of the Top-1 model is that it learns the rank information of the *full list*, but ignores the rank information of *partial sequences*, which may lead to ineffective learning. As an example, considering an object list where the score of the most relevant object is much higher than the scores of others, then the learning is dominated by the highest score, and largely throws away the rank information conveyed by the scores of other objects. It would be quite helpful if the rank information involved in partial sequences of the candidate objects can be learned. Top-k models place distributions over object lists (in length k), and so can learn partial sequences of objects.

We are interested in how to learn Top-k ($k > 1$) models while keeping the computation tractable. To achieve the goal, we propose a stochastic ListNet approach, which samples a small set of the Top-k permutation classes (object lists), and train the Top-k model based on this small set instead of the full set of permutation classes. As a comparison, the full set of permutation classes of the Top-k model is $\frac{n!}{(n-k)!}$, which is computationally prohibitive if $k > 1$. With stochastic ListNet, a subset of the permutation classes that involves only l members are randomly selected. Training the Top-k model based on this subset greatly reduces the computation cost, even with a large k. In fact, the subset approach imposes a bound of the computation cost that is largely determined by the the size of the subset (l), while independent of the total number of objects n and the model order k.

Interestingly, the stochastic approach offers not only quick learning, but also a chance of learning partial ranks. This is obvious because only a subset of the object lists are selected in model training, and so the rank information involved in the subset of the permutation classes can be learned. With the Top-1 model, partial ranks reduces to partial sequences since each object list involves only one object. As we have discussed, learning partial sequences is an advantage of Top-k models with $k > 1$. This means that stochastic Top-1 ListNet possesses some advantages of Top-k ListNet, while the computation cost is much lower.

3.3 Sampling methods for stochastic ListNet

The training process of stochastic ListNet starts from sampling l permutation classes, or object lists. For each object list, k objects are sampled following a particular distribution. As mentioned in Section 1, three distributions are studied in this paper: uniform distribution, fixed distribution and adaptive distribution. They are presented as follows.

Uniform distribution sampling: In this method, all the k objects of a particular object list are sampled with an equal probability. This sampling method is simple but biased towards irrelevant candidates, since there are much more irrelevant objects than relevant ones in the training data. A re-sampling approach is proposed to remedy the bias, as will be discussed in Section 4.

Fixed distribution sampling: In this method, the objects are sampled following a distribution proportional to the human-labelled scores. For instance, in the LETOR dataset that is used in this study, each candidate object (document) is labelled as 2 (very relevant), 1 (relevant) or 0 (irrelevant). These scores are normalized by softmax and are used as the probability distribution when sampling objects. Because the probabilities of relevant objects are larger than those of irrelevant objects, more relevant objects would be selected by this sampling approach in model training.

Adaptive distribution sampling: The fixed distribution sampling mentioned above relies on human-labelled scores, which may be impacted by label errors. Moreover, the absolute values of human labels are not good measures of object relevance. To solve these problems, we choose the outputs of the 'current' neural network as the relevance scores, and sample the objects according to these scores. Note that the network outputs are natural measures of object relevance based on the present ranking model. As the model (the neural network) keeps updated during model training, the

relevance scores are accordingly changed. In each iteration, the relevance scores are re-calculated, and the sampling is based on the new scores in the next iteration.

3.4 Gradients with linear networks

Cao et al. (2007) optimized the ListNet model by gradient descent. For each query, the learn rule is formuated by:

$$w = w - \eta \Delta w$$

where η is the learning rate, and w denotes the parameters of the model f_w. Δw denotes the gradient and it can be computed as follows:

$$\Delta w = \sum_{\forall g \in \mathscr{G}_k} \frac{\partial P_{z^{(i)}(f_w)}(g)}{\partial w} \frac{P_{y^{(i)}}(g)}{P_{z^{(i)}(f_w)}(g)}.$$

For simplicity, a linear NN model was used by Cao et al. (2007). This has been adopted in our study as well, written by $z^{(i)} = f_w(x_j^{(i)}) = w^T x_j^{(i)}$, where $x_j^{(i)}$ denotes the feature vector of the j-th object of the i-th query. In the case of the Top-1 model, it shows that:

$$\Delta w = \sum_j [\sigma(z^{(i)}, j) - \sigma(y^{(i)}, j)] x_j^{(i)}$$

where $\sigma(s, j)$ is the j-th value of the softmax function of the score vector s, given by:

$$\sigma(s^{(i)}, j) = \frac{e^{s_j^{(i)}}}{\sum_{t=1}^{n^{(i)}} e^{s_t^{(i)}}}.$$

In the case of the Top-k model, the gradient(Derivative of cross entropy between $P_{z^{(i)}}$ and $P_{y^{(i)}}$ when $k >= 2$) is a bit complex, but still manageable:

$$\begin{aligned} \Delta w = \sum_{g \in \mathscr{G}_k} [(\prod_{t=1}^{k} \hat{\sigma}(y^{(i)}, t)) \cdot \\ (\sum_{f=1}^{k} \{x_{j_f}^{(i)} - \sum_{v=f}^{n^{(i)}} \hat{\sigma}(z^{(i)}, v) x_{j_v}^{(i)}\})] \end{aligned} \quad (4)$$

where $\hat{\sigma}(\cdot)$ defines a 'partial' softmax(The partial softmax means that the $\sigma(s, f)$ has a similar form as softmax, however when computing the value for each f, the denominator is not the summation from 1 to n, instead a partial sequence from f to n.), given by:

$$\hat{\sigma}(s^{(i)}, f) = \frac{e^{s_{j_f}^{(i)}}}{\sum_{t=f}^{n^{(i)}} e^{s_{j_t}^{(i)}}}.$$

3.5 Stochastic Top-k ListNet algorithm

We present the stochastic Top-k ListNet algorithm, by employing the techniques described above. The gradient descent (GD) approach is adopted. All the training samples are processed sequentially in an iteration. The training runs several iterations until the convergence criterion is reach. Another detail is that the learning rate is multiplied by 0.1 whenever the objective function is worse than the previous iteration. The procedure is illustrated in Algorithm 1, where $\mathcal{L}(t)$ denotes value of the objective function after the t-th iteration.

Algorithm 1 Stochastic Top-k ListNet

Require:

Input:
$\mathscr{D} = \{(q^{(1)}, x^{(1)}, y^{(1)}), ..., (q^{(m)}, x^{(m)}, y^{(m)})\}$: training data
T: number of iterations
η: learning rate

Procedure:

```
Randomly initialize w
for t = 1 to T do
    for i = 1 to m do
        select the i-th training instance (q^(i), x^(i), y^(i)) ∈ D
        Sample the permutation classes G_k
        Compute Δw according to Eq. (4)
        Update f_w: w = w − ηΔw
    end for
    if L(t) < L(t − 1) then
        η = 0.1η
    end if
end for
```

4 Experiments

4.1 Data

The proposed stochastic Top-k ListNet method is tested on the document retrieval task based on the MQ2008 dataset of LETOR 4.0 (Liu et al., 2007). This database was released in early 2007 and has been widely used in learning to rank studies. It contains queries and corresponding candidate documents. The human-labelled scores are among three values $\{0, 1, 2\}$, representing little, medium, and strong relevance between queries and candidate documents, respectively. The training set, validation set and test data all contain 784 queries. The document features used in this study include term frequency, inverse document frequency, B-

M25, and language model scores for IR. Some new features proposed recently are also included, such as HostRank, feature propagation, and topical PageRank.

4.2 Experiment Setup

In our experiments, we consider Top-k models where k = 1, 2, 3, and 4. Although any k is possible with the proposed stochastic ListNet, we will show that simply increasing the model order k does not improve performance. The P@1 and P@10 performance is used as the evaluation metric.

Specially, for all the three distribution sampling methods, the sampling process involves two steps: pre-selection and re-sampling. The pre-selection step samples a list of documents following three distributions mentioned above, and in the re-sampling step, document lists including more relevant documents are retained with a higher probability. For example, denoting the pre-selected document list by (v_1,v_2,...,v_k) where k is the length of the list, and denoting the corresponding human-labelled scores by (s_1, s_2,...,s_k), the probability that the list is retained is given by

$$\frac{\sum_{i=1}^{k} s_i}{kS}$$

where S is the maximum value of the human-labelled scores, which is 2 in our case. The re-sampling approach is designed to encourage document lists containing more relevant documents, which is the most important for the uniform distribution sampling.

In stochastic Top-k ListNet, the learning rate is set as 10^{-3} for k = 1, and 10^{-5} for $k > 1$. These values are set to achieve the best performance on the validation set. Another important parameter of the stochastic Top-k ListNet approach is the number of samples of the document lists (or the size of subset of permutation classes selected), denoted by l. Various settings of l are experimented with in this study. To eliminate randomness in the results, all the experiments are repeated 20 times and the averaged performance is reported.

4.3 Experimental results

The P@1 results on the test dataset with different orders of Top-k ListNet are reported in Figure 1 to Figure 4. In each figure, the number of document lists varies from 5 to 500. For comparison, the results with the conventional ListNet are also presented. Note that the re-sampling approach was not applied to the Top-1 model as we found it caused performance reduction. This is perhaps because the sampling space is small with the Top-1 model, and so re-sampling tends to cause over-emphasis on relevant documents.

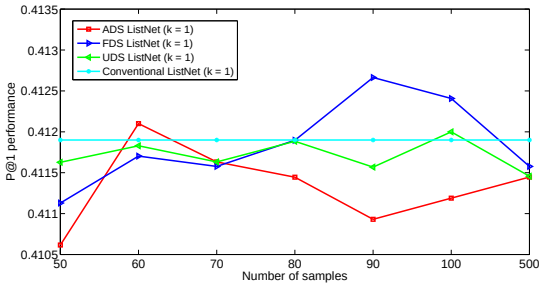


Figure 1: The P@1 performance on the test data with the Top-1 ListNet utilizing the three sampling approaches. The size of the permutation subset varies from 50 to 500.

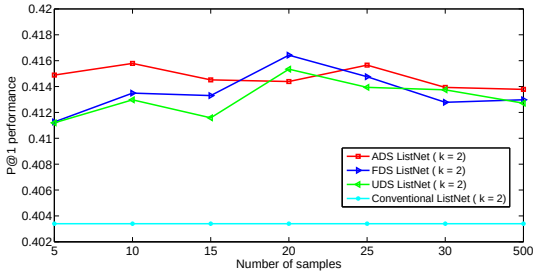


Figure 2: The P@1 performance on the test data with the Top-2 ListNet utilizing the three sampling approaches. The size of the permutation subset varies from 5 to 500.

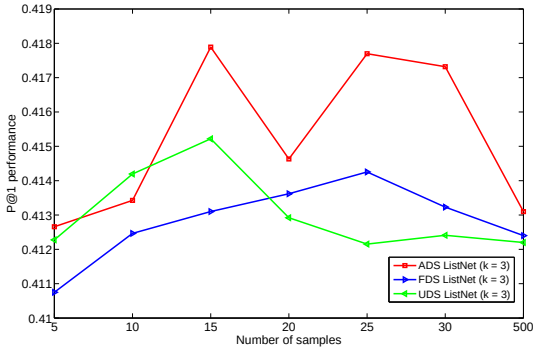


Figure 3: The P@1 performance on the test data with the Top-3 ListNet utilizing the three sampling approaches. The size of the permutation subset varies from 5 to 500.

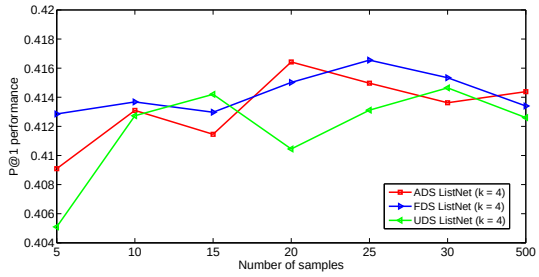


Figure 4: The P@1 performance on the test data with the Top-4 ListNet utilizing the three sampling approaches. The size of the permutation subset varies from 5 to 500.

From these results, we first observe that stochastic ListNet with either fixed or adaptive distribution sampling tends to outperform the conventional ListNet approach, particularly with a large k. This confirms our argument that rank information can be learned from a subset of the permutation classes that are randomly selected, and the partial rank learning can lead to even better performance than the full rank learning, the case of conventional ListNet. This is an interesting result and demonstrates the stochastic ListNet is both faster and better than the conventional ListNet. It is also seen that the adaptive distribution sampling performs slightly better than the fixed distribution sampling. This is not surprising as the adaptive distribution sampling uses a more reasonable relevance score (neural network output) to balance relevant and irrelevant documents. The uniform distribution sampling performs a little worse than the other two sampling methods, probably caused by the less informative uniform distribution.

Another observation is that in all the four figures, the performance of the stochastic ListNet methods increases with more samples of the object lists. However if there are too many samples, the performance starts to decrease. This can be explained by the fact that the sampling prefers relevant documents which are more informative. A larger sample set often includes more informative documents; however if the set is too large, many irrelevant documents will be selected and the performance is reduced. In the case that the number of samples is very large (500 for example for Top-1), the stochastic ListNet falls back to the conventional ListNet, and their performance becomes similar.

Comparing the results with different k, it can be seen that a larger k leads to a better performance with stochastic ListNet. This confirms that high-order Top-k models can learn more ranking information. However, this is not necessarily the case with the conventional ListNet. For example, the Top-2 model does not offer better performance than the Top-1 model. This is perhaps because high-order Top-k models consider a large number of document lists and most of them are not informative, which leads to ineffective learning. Remind that the conventional ListNet is a special case of the stochastic ListNet with a very large sample set, and we have discussed that an over large sample set actually reduces performance.

The averaged training time and the performance in precession are presented in Table 1. For precession, both P@1 and P@10 results are reported, though we focus on P@1 since it is more concerned for applications such as QA. Note that for stochastic ListNet, the optimal number of samples (document lists) has been selected according to the P@1 performance on the validate set.

From these results, it can be seen that the conventional Top-1 ListNet is rather fast, however the Top-2 model is thousands of times slower. With $k > 2$, the training time becomes prohibitive and so they are not listed in the Table. This is expected since the conventional ListNet considers the full set of permutations which is a huge number with a large k. With the stochastic ListNet, the training time is dramatically reduced. Even with a large k, the computation cost is still manageable, because the computation is mostly determined by the number of object lists, rather than the value of k. When comparing the three sampling methods, it can be found the convergence speed of the uniform distribution approach is the slowest, probably due to the ineffective selection for relevant documents. The adaptive distribution sampling is the fastest, probably attributed to the collaborative update of the model and the distribution.

As for the P@1 performance, the stochastic ListNet method generally outperforms its non-stochastic counterpart, particularly with the adaptive distribution sampling. For example, the best P@1 results obtained on the test data with the stochastic Top-1 ListNet is 0.4127, which outperforms the conventional Top-1 ListNet (0.4119). This advantage of stochastic ListNet, as we argued, is largely attributed to its capability of learning partial rank information with samples of partial sequences of the rank list.

Comparing the results with different k values, it can be seen that a larger k tends to offer better P@1 performance on the training set, with either the conventional ListNet or the stochastic ListNet. For example, with the conventional ListNet, the results are 0.4101 vs. 0.4119 with the Top-1 and Top-2 models respectively. However, the performance gap is rather marginal, and the advantage with the large k does not propagate to the results on the test data (as has been seen in Figure 1 and Figure 2). This indicates that for the conventional ListNet, the Top-1 model is not the only choice in the sense of computation complexity, but also the best choice in the sense of P@1 performance.

For stochastic ListNet, the performance improves with k increases. In contrast to the conventional ListNet, this improvement propagates to the results on the test data. For example, with the adaptive distribution sampling, the P@1 results on the training set are 0.4102 vs. 0.4184 with the Top-1 and Top-3 models respectively, and the results on the test data are 0.4121 vs. 0.4177 respectively.

Nevertheless, the P@1 performance improvement with a large k is rather marginal, and an over large k simply reduces the performance. To make it clear, we vary the value of k from 1 to 100 and plot the P@1 results in Figure 5). It can be seen that larger k (> 4) does not offer any merit but causes performance instability, particularly with the adaptive sampling approach. As we have discussed, with the stochastic ListNet, partial rank information can be learned with simple Top-k models, even the Top-1 model. This capability of partial rank learning with simple models reduces the necessity of employing complex Top-k models. This is a highly valuable conclusion, and it suggests that a simple Top-1 or Top-2 model is sufficient for the ListNet method, if the stochastic method is applied. Considering the trade-off between computation cost and model strength, we recommend stochastic Top-2 ListNet which delivers better P@1 performance than the Top-1 model consistently, with sufficiently fast computing. If more computation is affordable, stochastic Top-3 ListNet can be used to obtain better performance.

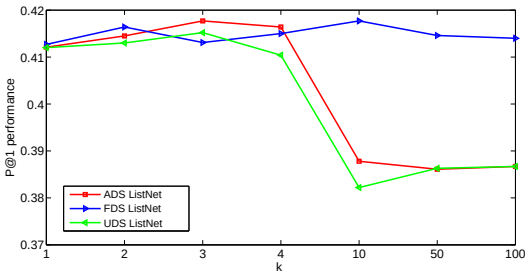


Figure 5: The P@1 performance on the test data with the stochastic Top-k ListNet approach, where k varies from 1 to 100.

Finally, we highlight that the conclusions ob-

				P@1			P@10		
Model	Top-k	Sampling	Time (s)	Train	Val.	Test	Train	Val.	Test
C-ListNet	k=1	-	2.509	0.4101	0.4107	0.4119	**0.2684**	**0.2684**	0.2676
S-ListNet	k=1	UDS	0.753	0.4097	**0.4106**	0.4120	0.2680	0.2683	0.2676
S-ListNet	k=1	FDS	0.391	0.4094	0.4090	**0.4127**	0.2679	0.2681	0.2676
S-ListNet	k=1	ADS	**0.375**	**0.4102**	0.4097	0.4121	0.2680	0.2682	**0.2677**
C-ListNet	k=2	-	2275.5	0.4119	0.4043	0.4043	0.2678	0.2674	0.2674
S-ListNet	k=2	UDS	2.898	0.4140	0.4143	0.4130	0.2682	0.2686	0.2681
S-ListNet	k=2	FDS	2.410	0.4145	0.4144	**0.4164**	0.2684	0.2688	0.2684
S-ListNet	k=2	ADS	**2.013**	**0.4162**	**0.4168**	0.4145	**0.2686**	**0.2689**	**0.2687**
S-ListNet	k=3	UDS	4.358	0.4167	0.4204	0.4152	0.2686	0.2681	0.2680
S-ListNet	k=3	FDS	3.997	0.4137	**0.4205**	0.4131	0.2687	0.2695	0.2685
S-ListNet	k=3	ADS	**3.483**	**0.4184**	0.4196	**0.4177**	**0.2692**	**0.2697**	**0.2689**
S-ListNet	k=4	UDS	6.161	0.4145	0.4226	0.4104	0.2686	0.2694	0.2687
S-ListNet	k=4	FDS	5.773	0.4145	0.4232	0.4150	0.2690	0.2695	0.2686
S-ListNet	k=4	ADS	**4.358**	**0.4149**	**0.4247**	**0.4164**	**0.2692**	**0.2700**	**0.2689**

Table 1: Averaged training time (in seconds), P@1 and P@10 on training, validation (Val.) and test data with different Top-k methods. 'C-ListNet' stands for conventional ListNet, 'S-ListNet' stands for stochastic ListNet.

tained from the P@1 results and the P@10 results perfectly match. In fact, the P@10 results look more consistent between training and test data, and the advantage of the stochastic approach seems more clear, particularly with the adaptive sampling. This is not surprising as the optimization goal of ListNet is essentially to form a good rank that involves multiple candidates, and so P@10 is apt to measure the superiority of a better rank approach.

5 Discussion

An interesting observation with the stochastic ListNet approach is that sampling more relevant documents improves performance. This can be explained by the data imbalance between relevant and irrelevant documents, i.e., there are much more irrelevant documents than relevant documents in the training data. This imbalance leads to biased models that tend to classify all documents as irrelevant. The re-sampling approach can be regarded as a way of balancing the two classes, and the fixed and adaptive distribution sampling can be regarded as another way to achieve the goal. Note that in the fixed distribution sampling, the distribution is solely dependent on the human-labeled scores. These scores are good measures of the *rank* of relevance but not good measures of the relevance itself. A possible way to solve this problem is to learn a scoring function that maps human-labelled scores to more reasonable measures of document relevance, though we took a different way that employs the network outputs as the relevance measures, which is what the adaptive distribution sampling method does. Note that the network output is a natural measure of document relevance, so the adaptive distribution sampling works the best in our experiments.

Another related issue is the harsh labelling of the AM2008 dataset. In this dataset, documents are labelled by only three values $\{0, 1, 2\}$, which is rather imprecise and the rank information is very limited. This harsh labeling is another reason why the uniform distribution sampling does not work: by uniform distribution sampling, there is a large probability that the sampled object lists involve documents that are all labelled by 0. This leads to an inefficient learning. Another consequence of the harsh labeling is that the power of complicated ranking models is largely constrained. For example, with the Top-k ($k > 1$) ListNet model, many of the k documents in a candidate list are labelled as the same score, resulting in limited rank information for the Top-k model to learn. This is why Top-k models did not exhibit much superiority to the Top-1 model in our experiments. We argue that top-k models would provide more contributions with more thorough labels (e.g., scores in real values). This is an ongoing research of our group.

Finally, we highlight that the stochastic approach is not limited to the ListNet model, but any model for listwise learning. It is well known that listwise learning outperforms pairwise learning, due to it is capability of learning full ranks (Liu, 2009). However learning full ranks requires unaffordable computation and so is infeasible in practice, even with the Top-k approximation. Our work demonstrated that learning full ranks can be approximated by learning partial ranks, and a limited number samples of such partial ranks is sufficient to convey the rank information. This stochastic learning is very fast, and even delivers better performance. It can be regarded as a general framework that treats both the pair-wise learning and the full rank learning as two special cases. In fact, if the set of partial ranks involves all the permutation classes, it reduces to the conventional listwise learning, and if the set of partial ranks involves all object pairs, it resembles the pairwise learning. A wide range of listwise learning methods can benefit from the idea of stochastic learning provided in this paper.

6 Conclusion

This paper proposed a stochastic ListNet method to speed up the training of ListNet models and improve the ranking performance. The basic idea is to approximate the full rank learning by learning a small number of partial ranks. Three sampling approaches were proposed to select the partial ranks, and Top-k ListNet models with various complexity (k values) were investigated.

Our preliminary results on the MQ2008 dataset confirmed that the stochastic ListNet approach can dramatically speeds up the model training, and more interestingly, it can produce better ranking performance than the conventional ListNet. Especially, the adaptive distribution sampling method delivered the best P@1 performance. An appealing observation is that the simple Top-2 model is very effective and more complex Top-k models

seem not very necessary, considering the trade-off between training complexity and model strength. This observation, however, is purely based on the MQ2008 dataset. As have been discussed, more detailed human labels may require more complex models, for which the stochastic method proposed in this paper is essential to conduct the model training. For the future work, we plan to study Top-k ListNet models with other databases and apply the stochastic learning approach to other listwise learning to rank methods.

Acknowledgments

This research was supported by the National Science Foundation of China (NSFC) under the project No. 61371136, and the MESTDC PhD Foundation Project No. 20130002120011. It was also supported by Sinovoice and Pachira.

References

Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. 2007. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th international conference on Machine learning*, pages 129–136. ACM.

Van B. Dang. 2013. Ranklib. http://people.cs.umass.edu/vdang/ranklib.html/.

Feng Jin, Minlie Huang, and Xiaoyan Zhu. 2010. A comparative study on ranking and selection strategies for multi-document summarization. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 525–533. Association for Computational Linguistics.

Marios Kokkodis, Panagiotis Papadimitriou, and Panagiotis G Ipeirotis. 2015. Hiring behavior models for online labor markets. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*, pages 223–232. ACM.

Tie-Yan Liu, Jun Xu, Tao Qin, Wenying Xiong, and Hang Li. 2007. Letor: Benchmark dataset for research on learning to rank for information retrieval. In *Proceedings of SIGIR 2007 workshop on learning to rank for information retrieval*, pages 3–10.

Tie-Yan Liu. 2009. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3):225–331.

D Sculley. 2009. Large scale learning to rank. In *NIPS Workshop on Advances in Ranking*, pages 1–6.

Yi-Hsuan Yang and Winston H Hsu. 2008. Video search reranking via online ordinal reranking. In *Multimedia and Expo, 2008 IEEE International Conference on*, pages 285–288. IEEE.

Yi-Hsuan Yang, Yu-Ching Lin, Ann Lee, and Homer H Chen. 2009. Improving musical concept detection by ordinal regression and context fusion. In *ISMIR*, pages 147–152. Citeseer.

Exploring Markov Logic Networks for Question Answering

Tushar Khot[*], **Niranjan Balasubramanian**[+], **Eric Gribkoff**[$],
Ashish Sabharwal[*], **Peter Clark**[*], **Oren Etzioni**[*]
[*]Allen Institute for AI, [+]Stony Brook University, [$]University of Washington
{tushark,ashishs,peterc,orene}@allenai.org, niranjan@cs.stonybrook.edu,
eagribko@cs.washington.edu

Abstract

Elementary-level science exams pose significant knowledge acquisition and reasoning challenges for automatic question answering. We develop a system that reasons with knowledge derived from textbooks, represented in a subset of first-order logic. Automatic extraction, while scalable, often results in knowledge that is incomplete and noisy, motivating use of reasoning mechanisms that handle uncertainty.

Markov Logic Networks (MLNs) seem a natural model for expressing such knowledge, but the exact way of leveraging MLNs is by no means obvious. We investigate three ways of applying MLNs to our task. First, we simply use the extracted science rules directly as MLN clauses and exploit the structure present in hard constraints to improve tractability. Second, we interpret science rules as describing prototypical entities, resulting in a drastically simplified but brittle network. Our third approach, called Praline, uses MLNs to align lexical elements as well as define and control how inference should be performed in this task. Praline demonstrates a 15% accuracy boost and a 10x reduction in runtime as compared to other MLN-based methods, and comparable accuracy to word-based baseline approaches.

1 Introduction

We consider the problem of answering questions in standardized science exams (Clark et al., 2013), which are used as a benchmark for developing knowledge acquisition and reasoning capabilities. The 4th grade science exam dataset from Clark et al. (2013) tests for a wide variety of knowledge and its application to specific scenarios. In particular we focus on a subset that test students' understanding of various kinds of general rules and principles (e.g., *gravity pulls objects towards the Earth*) and their ability to apply these rules to reason about specific situations or scenarios (e.g., *which force is responsible for a ball to drop?*).

Answering these questions can be naturally formulated as a reasoning task given the appropriate form of knowledge. Prior work on reasoning based approaches has largely relied on manually input knowledge (Lenat, 1995). We present an investigation of a reasoning approach that operates over knowledge automatically extracted from text.

In order to effectively reason over knowledge derived from text, a QA system must handle incomplete and potentially noisy knowledge, and allow for reasoning under uncertainty. We cast QA as a reasoning problem in weighted-first order logic. While many probabilistic formalisms exist, we use Markov Logic Networks for the ease of specification of weighted rules. MLNs have been adopted for many NLP tasks (Singla and Domingos, 2006a; Kok and Domingos, 2008; Poon and Domingos, 2009). Recently, Beltagy et al. (2013) and Beltagy and Mooney (2014) have shown that MLNs can be used to reason with rules derived from natural language. While MLNs appear to be a natural fit, it is a priori unclear how to effectively formulate the QA task as an MLN problem. We find that unique characteristics of this domain pose new challenges in efficient inference. Moreover, it is unclear how MLNs might perform on automatically extracted noisy rules and how they would fare against simpler baselines that do not rely as much on structured logical representations.

Our goal is to build a high accuracy reasoning-based QA system that can answer a question in near real time. To this end, we investigate three MLN-based formulations: (1) A natural formulation that is intuitive but suffers from inefficient in-

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 685–694,
Lisbon, Portugal, 17-21 September 2015.

ference (e.g., over 10 minutes on 31% of the questions); (2) an extension that improves efficiency by using prototypical constants, but is brittle to variation in structure; and (3) a formulation with improved flexibility to handle variation in text and structure that is 15% more accurate and 10x faster than the other approaches.

Despite significant improvements over two natural MLN formulations, the best reasoning-based configuration still does not outperform a simpler word-based baseline. We surmise that without effective salience models on text-derived rules, reasoning is unable to leverage the systematic advantages of the MLN-based models. The improved flexibility in the MLN-based models essentially appears to approximate word-based approaches due to the noisy and incomplete nature of the input knowledge. Nevertheless, the reasoning based method shows improved performance when adding multiple rules, which provides a principled way to inject additional knowledge and control inference for further improvements.

2 Background: QA Task

Following Clark et al. (2014), we formulate QA as a reasoning task over knowledge derived from textual sources. A multiple choice question with k answer options is turned into k true-false questions, each of which asserts some known facts (the $setup$) and posits a $query$. The reasoning task is to determine whether the $query$ is true given the $setup$ and the input knowledge.

The input knowledge is derived from 4th-grade science texts and augmented with a web search for terms appearing in the texts. Much of this knowledge is in terms of generalities, expressed naturally as IF-THEN rules. We use the representation and extraction procedures of Clark et al. (2014), recapitulated briefly here for completeness.

Rule Representation: The generalities in text convey information about classes of entities and events. Following the neo-davidsonian reified representation (Curran et al., 2007), we encode information about events (e.g, falling) and entities (e.g., ball or stone) using variables. Predicates such as $agent$, $cause$, $function$, $towards$, and in define semantic relationships between variables. Rather than committing to a type ontology, the variables are associated with their original string representation using an isa predicate.

The "if" or $antecedent$ part of the rule is semantically interpreted as being universally quantified (omitted below for conciseness) whereas every entity or event mentioned only in the "then" or $consequent$ part of the rule is treated as existentially quantified. Both $antecedent$ and $consequent$ are interpreted as conjunctions. For example, "Growing thicker fur in winter helps some animals to stay warm" translates into:

$$\begin{aligned} &isa(g, \text{grow}), isa(a, \text{some animals}), \\ &isa(f, \text{thicker fur}), isa(w, \text{the winter}), \\ &agent(g, a), object(g, f), in(g, w) \\ &\Rightarrow \exists s, r : isa(s, \text{stays}), isa(r, \text{warm}), \\ &\quad enables(g, s), agent(s, a), object(s, r) \end{aligned} \quad (1)$$

Question Representation: The question representation is computed similarly except that we use fixed constants (represented as block letters) rather than variables. For example, consider the question: "A fox grows thick fur as the season changes. This helps the fox to (A) hide from danger (B) attract a mate (C) find food (D) keep warm?" The T/F question corresponding to option (D) translates into:

$$\begin{aligned} setup :& isa(F, \text{fox}), isa(G, \text{grows}), isa(T, \\ & \quad \text{thick fur}), agent(G, F), object(G, T) \\ query :& isa(K, \text{keep warm}), enables(G, K), \\ & \quad agent(K, F) \end{aligned}$$

Lexical Reasoning: Since entity and event variables hold textual values, reasoning must accommodate lexical variability and textual entailment. For example, the surface forms "thick fur" and "thicker fur" are semantically equivalent. Also, the string "fox" entails "some animal". We use a lexical reasoning component based on textual entailment to establish lexical equivalence or entailment between variables.

Most Likely Answer as Inference: Given KB rules and the question as input, we formulate a probabilistic reasoning problem by adding lexical reasoning probabilities and incorporating uncertainties in derived rules. Given setup facts S and k answer options Q_i, we seek the most likely answer option: $\arg\max_{i \in \{1,\ldots,k\}} \Pr[Q_i \mid S, KB]$. This is a Partial MAP computation which is known to be #P-hard (Park, 2002). Hence methods such as Integer Linear Programming are not directly applicable.

2.1 Challenges

Reasoning with text-derived knowledge presents challenges that expose the *brittleness* and *rigidity* inherent in pure logic-based frameworks. Text-derived rules are incomplete and include lexical items as logical elements, making rule application in a pure logical setting extremely brittle: Many relevant rules cannot be applied because their preconditions are not fully satisfied due to poor alignment. For example, naive matching of rule (1) with the facts in the $setup$ would not conclude the $query$ since the rule requires "in the winter" to be true. A robust inference mechanism must allow for rule application with partial evidence. Further, a single text-derived rule may be insufficient to answer a question. For example, "Animals grow thick fur in winter" and "Thick fur helps keep warm" may need to be chained.

3 Probabilistic Formulations

Statistical Relational Learning (SRL) models (Getoor and Taskar, 2007) are a natural fit for QA reasoning. They provide probabilistic semantics over knowledge in first-order logic, thereby handling uncertainty in lexical reasoning and incomplete matching. While there are many SRL formalisms including Stochastic Logic Programs (SLPs) (Muggleton, 1996), ProbLog (Raedt et al., 2007), and PRISM (Sato and Kameya, 2001), we use Markov Logic Networks (MLNs) for their ease of specification and ability to naturally handle potentially cyclic rules.

Markov Logic Networks (MLNs) are relational models represented using weighted first-order logic rules. The rules provide a template for generating a Markov network by grounding the variables to all the constants in the rules. Each rule f_i forms a clique in the ground network and its weight w_i determines the potential for the clique. Since all cliques generated by grounding the same clause have the same weight, the probability of a given assignment is calculated as:

$$P(\mathbf{X} = \mathbf{x}) = \frac{1}{Z} \exp\left(\sum_i w_i n_i(\mathbf{x})\right)$$

where $n_i(\mathbf{x})$ is the number of times the ith formula is satisfied by the world $\mathbf{x}$ and Z is a normalization constant. Intuitively, a rule with a positive weight is more likely to be true than false.Higher the weight of a rule, more likely it is to be true.

We explore three MLN formulations:

a) First-order MLN: Given a question and relevant first-order KB rules, we convert them into an MLN program and let MLN inference automatically handle rule chaining. While a natural first-order formulation of the QA task, this struggles with long conjunctions and existentials in rules, as well as relatively few atoms and little to no symmetries. This results in massive grounding sizes, not remedied easily by existing solutions such as lazy, lifted, or structured inference. We exploit the structure imposed by hard constraints to vastly simplify groundings and bring them to the realm of feasibility, but performance remains poor.

b) Entity Resolution MLN: Instead of reasoning with rules that express generalities over classes of individuals, we replace the variables in the previous formulation with prototypical constants. This reduces the number of groundings, while retaining the crux of the reasoning problem defined over generalities. Combining this idea with existing entity resolution approaches substantially improves scalability. However, this turns out to be too brittle in handling lexical mismatches, especially in the presence of differences in parse structures

c) Praline MLN: Both of the above MLNs rely on exactly matching the relations in the KB and question representation, making them too sensitive to syntactic differences. In response, PRobabilistic ALignment and INferencE (Praline) performs inference using primarily the string constants but guided by the edge or relational structure. We relax the rigidity in rule application by explicitly modeling the desired QA inference behavior instead of relying on MLN's semantics.

3.1 First-Order MLN Formulation

For a set R of first-order KB rules, arguably the most natural way to represent the QA task of computing $\Pr[Q_i \mid S, R]$ as an MLN program M is to simply add R essentially verbatim as first-order rules in M. For all existentially quantified variables, we introduce a new domain constant. Predicates of M are those in R, along with a binary $entails$ predicate representing the lexical entailment blackbox, which allows M to probabilistically connect lexically related constants such as "thick_fur" and "thicker_fur" or "fox" and "some_animals". $entails$ is defined to be closed-world and is not necessarily transitive.

Evidence: Soft evidence for M consists of $entails$ relations between every ordered pair of entity (or event) strings, e.g., $entails(\text{fox}, \text{some\_animals})$. Hard evidence for M comprises grounded atoms in S.

Query: The query atom in M is $result()$, a new zero-arity predicate $result()$ that is made equivalent to the conjunction of the predicates in Q_i that have not been included in the evidence. We are interested in computing $\Pr[result() = \text{true}]$.

Semantic Rules: In addition to KB science rules, we add *semantic rules* that capture the intended meaning of our predicates, such as every event has a unique agent, $cause(x, y) \rightarrow effect(y, x)$, etc. Semantic predicates also enforce natural restrictions such as non-reflexivity, $!r(x, x)$, and anti-symmetry, $r(x, y) \rightarrow !r(y, x)$.

Finally, to help bridge lexical gaps more, we use a simple external lexical alignment algorithm to estimate how much does the $setup$ entail $antecedent_r$ of each KB rule r, and how much does $consequent_r$ entail $query$. These are then added as two additional MLN rules per KB rule.

Our rules have a specific first-order logic form:

$$\forall x_1, .., x_k \bigwedge_i R_i(x_{i_1}, x_{i_2}) \rightarrow \exists x_{k+1}, .., x_{k+m} \bigwedge_j R_j(x_{j_1}, x_{j_2})$$

Existentials spanning conjunctions in the consequent of this rule form can neither be directly fed into existing MLN systems nor naively translated into a standard form without incurring an exponential blowup. We introduce a new "existential" predicate $E_j^{\alpha}(x_1, \ldots, x_k, x_{k+j})$ for each existential variable x_{k+j} in each such rule α. This predicate becomes the consequent of α, and hard MLN rules make it equivant to the original consequent.

3.1.1 Boosting Inference Efficiency.

A bottleneck in using MLN solvers out-of-the-box for this QA formulation is the prohibitively large grounded network size. For example, 34 out of 108 runs timed out during MLN grounding phase after 6 minutes. On average, the ground networks in these runs were of the order of 1.4×10^6 ground clauses. Such behavior has also been observed, perhaps to a lesser degree, in related NLP tasks such as RTE (Beltagy and Mooney, 2014) and STS (Beltagy et al., 2014).

Existing techniques address large grounding size by focusing on relevant atoms (Singla and Domingos, 2006b; Shavlik and Natarajan, 2009) or grouping atoms into large classes of interchangeable atoms (de Salvo Braz et al., 2005; Gogate and Domingos, 2011; Venugopal and Gogate, 2012). Our QA encoding has very few atoms (often under 500) but very long clauses and highly asymmetric structure. This makes existing methods ineffective. For example, lazy inference in Alchemy-1[1] reduced ∼70K ground clauses to ∼56K on a question, while our method, described next, brought it down to only 951 clauses. Further, Lifted Blocked Gibbs and Probabilistic Theorem Proving, as implemented in Alchemy-2, were slower than basic Alchemy-1.

We utilize the combinatorial structure imposed by the set H of hard constraints (e.g., semantic rules, definition style rules, some science rules) present in the MLN, and use it to simplify the grounding of *both* hard and soft constraints. Importantly, this does not alter the first-order MLN semantics. The approach thus embraces hard clauses rather than relaxing them, as is often done in probabilistic inference techniques, especially when avoiding infinite energy barriers in MCMC based methods. Assuming an arbitrary constraint ordering in H, let F_i denote the first i constraints. Starting with $i = 1$, we generate the propositional grounding G_i of F_i, use a propositional satisfiability (SAT) solver to identify the set B_i of *backbone variables* of G_i (i.e., variables that take a fixed value in all solutions to G_i), freeze values of the corresponding atoms in B_i, increment i, and repeat until $G_{|H|}$ has been processed. Although the end result can be described simply as freezing atoms corresponding to the backbone variables in the grounding of H, the incremental process helps us control the intermediate grounding size as a propositional variable is no longer generated for a frozen atom. Once the freezing process is complete, the full grounding of H is further simplified by removing frozen variables. Finally, the soft constraints S are grounded much more efficiently by taking frozen atoms into account. Our approach may also be seen as an extension of a proposal by Papai et al. (2011).

[1] http://alchemy.cs.washington.edu

3.2 Entity Resolution Based MLN

Representing generalities as quantified rules defined over classes of entities or events appears to be a natural formulation, but is also quite inefficient leading to large grounded networks. Despite the drastically reduced number of groundings by our inference approach, the first-order MLN formulation still timed out on 31% of the questions. Hence we consider an alternative formulation that treats generalities as relations expressed over *prototypical* entities and events. This formulation leverages the fact that elementary level science questions can often be answered using relatively simple logical reasoning over exemplar objects and homogeneous classes of objects. The only uncertainty present in our system is what's introduced by lexical variations and extraction errors, which we handle with probabilistic equality.

KB Rules and Question: We define rules over prototypical entity/event *constants*, rather than first-order variables. These constants are tied to their respective string representations, with the understanding that two entities behave similarly if they have lexically similar strings. For example,

$$agent(Grow, Animals), object(Grow, Fur) \Rightarrow enables(Grow, StayWarm)$$

What was a first-order rule in FO-MLN is now already fully grounded! Entities/events in the question are also similarly represented by constants. Note that the efficiency boost using hard constraints (Section 3.1.1) is orthogonal to using prototypical constants and can be applied here as well.

Equivalence or Resolution Rules: Using a simple probabilistic variant of existing Entity/Event Resolution frameworks (Singla and Domingos, 2006a; Kok and Domingos, 2008), we ensure that (a) two entities/events are considered similar when they are tied to lexically similar strings and (b) similar entities/events participate in similar relations w.r.t. other entities/events. This defines soft *clusters* or equivalence classes of entities/events. We use a probabilistic $sameAs$ predicate which is reflexive, symmetric, and transitive, and interacts with the rest of the MLN as follows:

$$isa(x, s), entails(s, s') \rightarrow isa(x, s').$$
$$isa(x, s), isa(y, s) \rightarrow sameAs(x, y).$$
$$w: \ isa(x, s), !isa(y, s) \rightarrow !sameAs(x, y)$$
$$r(x, y), sameAs(y, z) \rightarrow r(x, z).$$

r in the last rule refers to any of the MLN predicates other than $entails$ and isa. The $sameAs$ predicate, as before, is implemented in a typed fashion, separately for entities and events. We will refer to this formulation as ER-MLN.

Partial Match Rules: Due to lexical variability, often not all conjuncts in a rule's $antecedent$ are present in the question's $setup$. To handle incomplete matches, for each KB derived MLN rule of the form $(\wedge_{i=1}^{k} L_i) \rightarrow R$, we also add k soft rules of the form $L_i \rightarrow R$. This adds flexibility, by helping "fire" the rule in a soft manner. This differs from FO-MLN which uses an external alignment system to find parts of the $antecedent$ mentioned in the $setup$, L' and creates one rule $L' \Rightarrow R$.

Comparison with FO-MLN: Long KB rules and question representation now no longer have quantified variables, only the binary or ternary rules above do. These mention at most 3 variables each and thus have relatively manageable groundings. On the other hand, as discussed in the next section, ER-MLN can fail on questions that have distinct entities with similar string representations (e.g. two distinct plants in a question would map to the same entity). Further, it fails to apply valid rules in the presence of syntactic differences such as $agent(\text{Fall}, \text{Things})$ generated by "things fall due to gravity" and $object(\text{Dropped}, \text{Ball})$ for "a student dropped a ball". Although "drop" entails "fall" and "ball" entails "object", ER-MLN cannot reliably bridge the structural difference involving $object$ and $agent$, as these two relationships typically aren't equivalent. Despite these limitations, ER-MLN provides a substantial scalability advantage over FO-MLN on a vast majority of the questions that remain within its scope.

3.3 PRobabilistic ALignment and INferencE

ER-MLN handles some of the word-level lexical variation via *resolution* and soft *partial match* rules that break long antecedents. However, it is still rigid in two respects:

(1) ER-MLN primarily relies on the predicates (also referred to as links or edges) for inference. As a result, even if the words in the $antecedent$ and $setup$ have high entailment scores, the rule will still not "fire" if the edges do not match.

(2) As entities bound to lexically equivalent strings are forced to "behave" identically, ER-

MLN fails on questions that involve two different entities that are bound to equivalent string representations. Consider the question: "A student puts two identical plants in the same type and amount of soil. She puts one of these plants near a sunny window and the other in a dark room. This experiment tests how the plants respond to (A) light (B) air (C) water (D) soil." The entities corresponding to the two plants will be bound to equivalent string representations and hence will be treated as the same entity. To avoid this, we do not force the entailment-based clusters of constants to behave similarly. Instead, as discussed below, we use the clusters to guide inference in a softer manner.

To introduce such flexibility, we define an MLN to directly control how new facts are inferred given the KB rules. The flexibility to control inference helps address two additional QA challenges:

Acyclic inference: While knowledge is extracted from text as a set of directed rules each with an $antecedent$ and a $consequent$, there is no guarantee that the rules taken together are acyclic. For example, a rule stating "Living things → depend on the sun" and "Sun → source of energy for living things" may exist side-by-side. Successful inference for QA must avoid feedback loops.

False unless proven: While MLNs assume atoms not mentioned in any rule to be true with probability 0.5, elementary level science reasoning is better reflected in a system that assumes all atoms to be false unless stated in the question or proven through the application of a rule. This is similar to the semantics of Problog (Raedt et al., 2007) and PRISM (Sato and Kameya, 2001).

While acyclic inference and false unless proven can be handled by setting high negative priors in MLNs, inference behavior is susceptible to variations in these weights. By using hard rules to control the direction of inference, we can explicitly enforce these constraints.

We introduce a unary predicate called $holds$ over string constants to capture the probability of a string constant being true given the $setup$ is true ($\forall x \in setup, holds(x) = true$) and the KB rules hold. Instead of using edges for inference, we use them as factors influencing alignment: similar constants have similar local neighborhoods. With n string constants, this reduces the number of unobserved groundings from $O(n^2)$ edges in the ER-MLN to $O(n)$ existence predicates. For the example rule (1), Praline can be viewed as using the following rule for inference:

$$holds(Grow), holds(Animals), holds(Fur),$$
$$holds(Winter) \Rightarrow holds(Stays), holds(Warm)$$

If we view KB rules and question as a labeled graph G (Figure 1), alignment between string constants corresponds to node alignment in G. The nodes and edges of G are the input to the MLN, and the $holds$ predicate on each node captures the probability of it being true given the $setup$. We now use MLNs (as described below) to define the inference procedure for any such input graph G.

Evidence: We represent the graph structure of G using predicates $node(nodeid)$ and $edge(nodeid, nodeid, label)$. We use $setup(nodeid)$ and $query(nodeid)$ to represent the question's $setup$ and $query$, resp. Similarly, we use $inLhs(nodeid)$ and $inRhs(nodeid)$ to represent rules' $antecedent$ and $consequent$, resp.

Graph Alignment Rules: Similar to the previous approaches, we use entailment scores between words and short phrases to compute the alignment. In addition, we also expect *aligned* nodes to have similar edge structures:

$$aligns(x, y), edge(x, u, r), edge(y, v, s) \Rightarrow aligns(u, v)$$

That is, if node x aligns with y then their children/ancestors should also align. We create copies of these rules for edges with the same label, $r = s$, with a higher weight and for edges with different labels, $r \neq s$, with a lower weight.

Inference Rules: We use MLNs to define the inference procedure to prove the $query$ using the alignments from $aligns$. We assume that any node y that $aligns$ with a node x that $holds$, also $holds$:

$$holds(x), aligns(x, y) \Rightarrow holds(y) \qquad (2)$$

For example, if the setup mentions "fox", all nodes that entail "fox" also hold. As we also use the edge structure during alignment, we would have a lower probability of "fox" in "fox finds food" to align with "animal" in "animal grows fur" as compared to "animal" in "animal finds food".

We use KB rules to further infer new facts that should hold based on the rule structure. We compute $lhsHolds$, the probability of the rule's

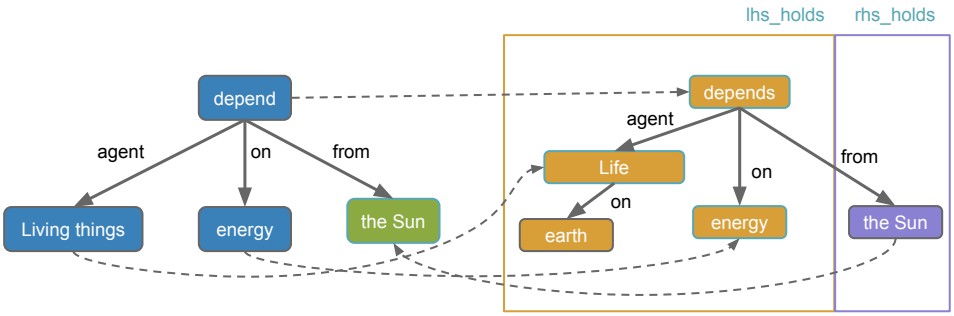


Figure 1: KB rule and question as a graph where blue:$setup$; green:$query$; orange:$antecedent$; purple:$consequent$; dotted lines: alignments. $lhsHolds$ combines individual probabilities of $antecedent$ nodes and $rhsHolds$ captures the probability of the $consequent$.

$antecedent$ holding, and use it to infer $rhsHolds$, the probability of the $consequent$. Similar to ER-MLN, we break the rule into multiple small rules.[2]

$$w : holds(x), inLhs(x, r) \Rightarrow lhsHolds(r)$$
$$w : !holds(x), inLhs(x, r) \Rightarrow !lhsHolds(r)$$
$$lhsHolds(r) \Rightarrow rhsHolds(r).$$
$$rhsHolds(r), inRhs(r, x) \Rightarrow holds(x).$$

Acyclic inference: We use two predicates, $proves(nodeid, nodeid)$ and $ruleProves(rule, rule)$, to capture the inference chain between nodes and rules, resp. To ensure acyclicity in inference, we add transitive clauses over these predicates and disallow reflexivity, i.e., $!proves(x, x)$, and update rule (2):

$$w_p : proves(x, y), holds(x) \Rightarrow holds(y)$$
$$w_a : aligns(x, y) \Rightarrow proves(x, y)$$

We capture inference direction between rules by checking consequent and antecedent alignments:

$$proves(x, y), inrhs(x, r1), inlhs(y, r2) \Rightarrow ruleProves(r1, r2).$$

False unless proven: To ensure that nodes hold only if they can be proven from $setup$, we add bidirectional implications to our rules. An alternative is to introduce a strong negative prior on $holds$ and have a higher positive weight on all other clauses that conclude $holds$. However, the performance of our MLNs was very sensitive to the choice of the weight. We instead model this constraint explicitly. Figure 1 shows a sample inference chain using dotted lines.

[2] An intuitive alternative for the 2nd clause doesn't capture the intending meaning, $-w : !holds(x), inLhs(x, r) \Rightarrow lhsHolds(r)$

Praline defines a meta-inference procedure that is easily modifiable to enforce desired QA inference behavior, e.g. $w : aligns(x, y), setup(x) \Rightarrow !query(y)$ would prevent a term from the $setup$ to align with the $query$. Further, by representing the input KB and question as evidence, we can define a single static first-order MLN for all the questions instead of a compiled MLN for every question. This opens up the possibility of learning weights of this static MLN, which would be challenging for the previous two approaches.[3]

4 Empirical Evaluation

We used Tuffy 0.4[4] (Niu et al., 2011) as the base MLN solver[5] and extended it to incorporate the hard-constraint based grounding reduction technique discussed earlier, implemented using the SAT solver Glucose 3.0[6] (Audemard and Simon, 2009) exploiting its "solving under assumptions" capability for efficiency. We used a 10 minute timelimit, including a max of 6 minutes for grounding. Marginal inference was performed using MC-SAT (Poon and Domingos, 2006), with default parameters and 5000 flips per sample to generate 500 samples for marginal estimation.

We used a 2-core 2.5 GHz Amazon EC2 linux machine with 16 GB RAM. We selected 108 elementary-level science questions (non-diagram, multiple-choice) from 4th grade New York Regents exam as our benchmark (Dev-108) and used another 68 questions from the same source as a blind test set (Unseen-68)[7].

[3] In this work, we have set the weights manually.
[4] http://i.stanford.edu/hazy/tuffy
[5] Alchemy 1.0 gave similar results.
[6] http://www.labri.fr/perso/lsimon/glucose
[7] http://allenai.org/content/data/Ariscienceexams.txt

Question Set	MLN Formulation	#Answered (some / all)	Exam Score	#MLN Rules	#Atoms	#Ground Clauses	Runtime (all)
Dev-108	FO-MLN	106 / 82	33.6%	35	384*	524*	280 s
	ER-MLN	107 / 107	34.5%	41	284	2,308	188 s
	PRALINE	108	**48.8%**	51	182	219	**17 s**
Unseen-68	FO-MLN	66	33.8%	-	-	-	288 s
	ER-MLN	68	31.3%	-	-	-	226 s
	PRALINE	68	**46.3%**	-	-	-	**17 s**

Table 1: QA performance of various MLN formulations. #MLN-Rules, #GroundClauses, and Runtime per multiple-choice question are averaged over the corresponding dataset. #Answered column indicates questions where at least one answer option didn't time out (left) and where no answer option timed out (right). Of the 432 Dev MLNs (108×4), #Atoms and #GroundClauses for FO-MLN are averaged over the 398 MLNs where grounding finished; 34 remaining MLNs timed out after processing 1.4M clauses.

The KB, representing roughly 47,000 sentences, was generated in advance by processing the New York Regents 4th grade science exam syllabus, the corresponding Barron's study guide, and documents obtained by querying the Internet for relevant terms. Given a question, we use a simple word-overlap based matching algorithm, referred to as the *rule selector*, to retrieve the top 30 matching sentences to be considered for the question. Textual entailment scores between words and short phrases were computed using WordNet (Miller, 1995), and converted to "desired" probabilities for soft $entails$ evidence. The accuracy reported for each approach is computed as the number of multiple-choice questions it answers correctly, with a partial credit of $1/k$ in case of a k-way tie between the highest scoring options if they include the correct answer.

4.1 MLN Formulation Comparison

Table 1 compares the effectiveness of our three MLN formulations: FO-MLN, ER-MLN, and Praline. For each question and approach, we generate an MLN program for each answer option using the most promising KB rule for that answer option.

In the case of FO-MLN, Tuffy exceeded the 6 minute time limit when generating groundings for 34 of the 108×4 MLNs for the Dev-108 question set, quitting after working with 1.4×10^6 clauses on average, despite starting with only around 35 first-order MLN rules. In the remaining MLNs, where our clause reduction technique successfully finished, the ground network size reduced dramatically to 524 clauses and 384 atoms on average.

Tuffy finished inference for all 4 answer options for 82 of the 108 questions; for other questions, it chose the most promising answer option among the ones it finished processing. Overall, this resulted in a score of 33.6% with an average of 280 seconds per multiple-choice question on Dev-108, and similar performance on Unseen-68.

ER-MLN, as expected, did not result in any timeouts during grounding. The number of ground clauses here, 2,308 on average, is dominated not by KB rules but by the binary and ternary entity resolution clauses involving the $sameAs$ predicate. ER-MLN was roughly 33% faster than FO-MLN, but overall achieved similar exam scores.

Praline resulted in a 10x speedup over ER-MLN, explained in part by much smaller ground networks with only 219 clauses on average. It boosted exam performance by roughly 15%, pushing it up to 48.8% on Dev-108 and 46.3% on Unseen-68 (statistically significantly better than FO-MLN with p-value < 0.05). This demonstrates the value that the added flexibility and control Praline brings.

4.2 Praline: Improvements and Ablation

We evaluate Praline when using multiple KB rules as a chain or multiple inference paths. Simply using the top two rules for inference turns out to be ineffective as they are often very similar. Instead, we use *incremental inference* where we add one rule, perform inference to determine which additional facts now hold and which $setup$ facts haven't yet been used, and then use this information to select the next best rule. This, as the Chain=2 entries in the first row of Table 2 show, improves Praline's accuracy on both datasets. The improvement comes at the cost of a modest runtime increase from 17 seconds per question to 38.

Finally, we evaluate the impact of Praline's additional rules to handle acyclicity (Acyclic) and the false unless proven (FUP) constraint. As Table 2 shows, Praline's accuracy drops upon removing

MLN	One rule		Chain=2	
	Dev-108	Unseen	Dev-108	Unseen
Praline	48.8%	46.3%	**50.3%**	**52.7%**
-Acyclic	44.7%	36.0%	43.6%	30.9%
-FUP	35.0%	30.9%	42.1%	29.4%
-FUP -Acyclic	37.3%	34.2%	36.6%	24.3%

Table 2: QA performance of Praline MLN variations.

	Dev-108	Unseen-68	Dev-170	Unseen-176
Praline	50.3%	**52.7%**	33.2%	36.6%
Word-based	**57.4%**	51.5%	**40.3%**	**43.3%**

Table 3: QA performance: Praline vs. word-based.

either of these constraints, highlighting their importance. Specifically, when using only one KB rule, dropping FUP clauses has a bigger influence that dropping Acyclic constraint clauses. With a single rule, there is still a possibility of cyclic inference within a rule, leading to a small drop in score there as well. When chaining multiple rules, however, the possibility of incorrect cyclic inference is higher and we see a correspondingly larger drop in score when dropping Acyclic constraints.

4.3 Comparison to baseline approaches

Table 3 compares Praline to a baseline word-based method on two question sets. The new set here is from 4th and 5th grade, with 170 Dev and 176 unseen questions. The word-based approach calculates the entailment score, using the same methods as for the soft $entails$ evidence earlier, between the words in the T/F question and words in a rule in the KB. It then uses the maximum entailment score from all selected rules as the confidence measure i.e. $\max_{r \in R} entailment(q, r)$. While the scores of the two approaches are statistically similar (p-value > 0.05), the simple word-based approach does have a slight edge over Praline. Automatic extraction of knowledge from text provides additional information (e.g., rule structure) that MLNs are capable of exploiting. However, we found this additional flexibility to not pay off with the current knowledge-base and questions.

5 Conclusions

Reasoning with automatically extracted knowledge presents significant challenges. Our investigation of MLN-based formulations for elementary-level science exams highlights two key issues: 1) Natural translations of text derived knowledge into first-order representations are highly inefficient, resulting in large ground networks. 2) When the logical elements in the rules largely mirror the constructs in the source text, reasoning is hampered because of structural variability. In response, we proposed, Praline, an alignment based solution that is both efficient and accurate. Praline reasons with prototypical constants, and provides greater flexibility in how inference is performed and is therefore more robust to structural mismatches.

MLNs provided a flexible, structured framework to define inference for the QA task, while also providing reasoning chains used to arrive at an answer. While models such as MLNs seem a perfect fit for textual reasoning tasks such as RTE and QA, their performance on these tasks is still not up to par with textual feature-based approaches (Beltagy and Mooney, 2014). We conjecture that the increased flexibility of complex relational models results in increased susceptibility to noisy input, and the systematic advantages of MLN models are difficult to exploit with text-derived rules. Automatically learning weights of these models may allow leveraging their flexibility to address these issues, but weight learning remains challenging with only distant supervision.

We hope our datasets, knowledge bases, and MLN models[8] will help push NLP and SRL communities towards designing improved structured reasoning QA systems.

Acknowledgments

The authors would like thank Pedro Domingos and Dan Weld for invaluable discussions and the Aristo team at AI2, especially Jesse Kinkead, for help with prototype development and evaluation.

References

Gilles Audemard and Laurent Simon. 2009. Predicting learnt clauses quality in modern SAT solvers. In *21st IJCAI*, pages 399–404, Pasadena, CA, July.

Islam Beltagy and Raymond J Mooney. 2014. Efficient Markov logic inference for natural language semantics. Quebec City, Canada, July.

Islam Beltagy, Cuong Chau, Gemma Boleda, Dan Garrette, Katrin Erk, and Raymond Mooney. 2013. Montague meets Markov: Deep semantics with probabilistic logical form. *2nd Joint Conference on*

[8] Available at http://allenai.org/software.html

Lexical and Computational Semantics: Proceeding of the Main Conference and the Shared Task, Atlanta, pages 11–21.

Islam Beltagy, Katrin Erk, and Raymond J. Mooney. 2014. Probabilistic soft logic for semantic textual similarity. In *52nd ACL*, pages 1210–1219, Baltimore, MD, June.

Peter Clark, Phil Harrison, and Niranjan Balasubramanian. 2013. A study of the AKBC/requirements for passing an elementary science test. In *Proc. of the AKBC-WEKEX workshop at CIKM*.

Peter Clark, Niranjan Balasubramanian, Sumithra Bhakthavatsalam, Kevin Humphreys, Jesse Kinkead, Ashish Sabharwal, and Oyvind Tafjord. 2014. Automatic construction of inference-supporting knowledge bases. In *4th Workshop on Automated Knowledge Base Construction (AKBC)*, Montreal, Canada, December.

James Curran, Stephen Clark, and Johan Bos. 2007. Linguistically motivated large-scale NLP with C&C and Boxer. In *45th ACL*, pages 33–36, Prague, Czech Republic, June.

Rodrigo de Salvo Braz, Eyal Amir, and Dan Roth. 2005. Lifted first-order probabilistic inference. In *19th IJCAI*, pages 1319–1325.

Lise Getoor and Ben Taskar, editors. 2007. *Introduction to Statistical Relational Learning*. MIT Press.

Vibhav Gogate and Pedro Domingos. 2011. Probabilistic theorem proving. pages 256–265, Barcelona, Spain, July.

Stanley Kok and Pedro Domingos. 2008. Extracting semantic networks from text via relational clustering. In *19th ECML*, pages 624–639, Antwerp, Belgium, September.

Douglas Lenat. 1995. Cyc: A large-scale investment in knowledge infrastructure. *Communications ACM*, 38(11):33–38.

George Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Stephen Muggleton. 1996. Stochastic logic programs. In *Advances in Inductive Logic Programming*.

Feng Niu, Christopher Ré, AnHai Doan, and Jude W. Shavlik. 2011. Tuffy: Scaling up statistical inference in Markov Logic Networks using an RDBMS. In *37th VLDB*, pages 373–384.

Tivadar Papai, Parag Singla, and Henry Kautz. 2011. Constraint propagation for efficient inference in Markov logic. In *17th CP*, pages 691–705. Springer, Perugia, Italy.

James Park. 2002. MAP complexity results and approximation methods. pages 388–396, Edmonton, Canada, August.

Hoifung Poon and Pedro Domingos. 2006. Sound and efficient inference with probabilistic and deterministic dependencies. In *21st AAAI*, pages 458–463, Boston, MA.

Hoifung Poon and Pedro Domingos. 2009. Unsupervised semantic parsing. In *EMNLP*, pages 1–10.

Luc De Raedt, Angelika Kimmig, and Hannu Toivonen. 2007. Problog: A probabilistic Prolog and its application in link discovery. In *International Joint Conference on Artificial Intelligence*.

Taisuke Sato and Yoshitaka Kameya. 2001. Parameter learning of logic programs for symbolic-statistical modeling. *Journal of Artificial Intelligence Research*, pages 391–454.

Jude Shavlik and Sriraam Natarajan. 2009. Speeding up inference in Markov logic networks by preprocessing to reduce the size of the resulting grounded network. In *21st IJCAI*, pages 1951–1956, Pasadena, CA.

Parag Singla and Pedro Domingos. 2006a. Entity resolution with Markov logic. In *6th ICDM*, pages 572–582, Hong Kong, China, December.

Parag Singla and Pedro Domingos. 2006b. Memory-efficient inference in relational domains. In *21st AAAI*, pages 488–493, Boston, MA.

Deepak Venugopal and Vibhav Gogate. 2012. On lifting the gibbs sampling algorithm. pages 1664–1672, Lake Tahoe, NV, December.

Language and Domain Independent Entity Linking with Quantified Collective Validation

Han Wang[*,1], **Jin Guang Zheng**[*,2], **Xiaogang Ma**[1], **Peter Fox**[1], **and Heng Ji**[2]
{Tetherless World Constellation[1], Computer Science Department[2]}
Rensselaer Polytechnic Institute
Troy, NY, USA
{wangh17, zhengj6, max7, pfox, jih}@rpi.edu

Abstract

Linking named mentions detected in a source document to an existing knowledge base provides disambiguated entity referents for the mentions. This allows better document analysis, knowledge extraction and knowledge base population. Most of the previous research extensively exploited the linguistic features of the source documents in a supervised or semi-supervised way. These systems therefore cannot be easily applied to a new language or domain. In this paper, we present a novel unsupervised algorithm named Quantified Collective Validation that avoids excessive linguistic analysis on the source documents and fully leverages the knowledge base structure for the entity linking task. We show our approach achieves state-of-the-art English entity linking performance and demonstrate successful deployment in a new language (Chinese) and two new domains (Biomedical and Earth Science). Experiment datasets and system demonstration are available at http://tw.rpi.edu/web/doc/hanwang_emnlp_2015 for research purpose.

1 Introduction and Motivation

The entity linking (EL) task aims at analyzing each named entity mention in a source document and linking it to its referent in a knowledge base (KB). Consider the following example: "*One day after released by the **Patriots**, **Florida** born **Caldwell** visited the **Jets**. ..... **The New York Jets** have six receivers on the roster: **Cotchery**, **Coles**, ...*". Here "*Caldwell*" is an ambiguous mention because not only are there thousands of people with different professions named "*Caldwell*", but even if as an American football player, as most people would recognize it from the context, there are several "*Caldwell*"s who are/were associated with either "*the Patriots*" or "*the Jets*". An EL system should be able to disambiguate the mention by carefully examining the context and then identify the correct KB referent, which is `Reche Caldwell` in this case.

Although EL has attracted a lot of community attention in the recent years, most research efforts have been focused on developing systems only effective for generic English corpora. When these systems are migrated to a new language or domain, their performance will usually suffer from a noticeable decline due to the following reasons:

1) State-of-the-art EL systems have developed comprehensive linguistic features from the source documents to generate advanced representations of the mentions and their context. While this methodology has been proved rewarding for a resource-rich language such as English, it prevents the systems from being adopted to a new language, especially to one with limited linguistic resources. One can imagine that it would be very difficult, if not impossible, for an English EL system that benefits from the part-of-speech tagging, dependency parsing, and named entity recognition to be deployed to a new language such as Chinese that has quite different linguistic characteristics.

2) The current EL approaches mostly target at people, organizations, and geo-political entities which are widely present in a general KB such as Wikipedia. However, domain-specific EL tends to pay more attention to entities beyond the above three types. For instance, in the biomedical science domain, protein is a major class of entities that greatly interest scientists. Conventional EL systems are very likely to fail in linking protein mentions in the text due to the lack of labeled

[*]These authors contributed equally to this work.

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 695–704,
Lisbon, Portugal, 17-21 September 2015.

training data. Moreover, their reliance on general reference KBs seems insufficient for a specific domain. Take "*A20*", a type of protein as an example. Wikipedia has more than a few items listed under the name of "*A20*" and their types range from aircrafts to roads. This diversified information inevitably introduces noise for a biomedical EL application.

One potential solution to tackle these limitations is, instead of concentrating on the source documents, to conduct more deliberate study on the KB. Structured KBs such as DBpedia[1] typically offer detailed descriptions about entities, a large collection of named relations between entities, and a growing number of multi-lingual entity surface forms. By embracing these ready-for-use information and linked structures, we will be able to obtain sufficient contextual information for disambiguation without generating a full list of linguistic features from the source documents, and therefore eliminate the language dependency. Moreover, currently there exist numerous publicly available domain ontology repositories such as BioPortal[2] and OBO Foundry[3] which provide significantly more domain knowledge than general KBs for EL to leverage. By incorporating these domain ontologies, we can easily increase the entity coverage and reduce noise for deploying EL in various new domains.

In order to make the most of the KB structure, the mention context should be matched against the KB such that the relevant KB information can be extracted. A collective way of aligning co-occurred mentions to the KB graph has been proved to be a successful strategy to better represent the source context (Pennacchiotti and Pantel, 2009; Fernández et al., 2010; Cucerzan, 2011; Han et al., 2011; Ratinov et al., 2011; Dalton and Dietz, 2013; Zheng et al., 2014; Pan et al., 2015). We take a further step to consider quantitatively differentiating entity relations in the KB in order to evaluate entity candidates more precisely. Meanwhile, we jointly validate these candidates by aligning them back to the source context and integrating multiple ranking results. This novel EL framework deeply exploits the KB structure with a light weight representation of the source context, and thus enables a smooth migration to new languages and domains.

The main novel contributions of this paper are summarized as follows: 1) We design an unsupervised EL algorithm, namely, Quantified Collective Validation (QCV) that builds KB entity candidate graphs with quantified relations for the purpose of collective disambiguation and inference. 2) We develop a procedure of building language and domain independent EL systems by incorporating various ontologies into the QCV component. 3) We demonstrate that our system is able to achieve state-of-the-art performance in English EL, and it can also produce promising results for Chinese EL as well as EL in Biomedical Science and Earth Science.

2 Baseline Collective EL

As a baseline, we adopt a competitive unsupervised collective EL system (Zheng et al., 2014) utilizing structured KBs. It defines entropy based weights for the KB relations, and embeds them in a two-step candidate ranking process to produce the EL results.

Structured KB Terminologies: In a structured KB, a fact is usually expressed in the form of a triple: (e_h, r, e_t) where e_h, e_t are called the head entity and the tail entity, respectively, and r is the relation between e_h and e_t.

Entropy Based KB Relation Weights: The goal is to leverage various levels of granularity of KB relations. The calculation of the relation weight $H(r)$ is given in Equation (1):

$$H(r) = -\sum_{e_t \in E_t(r)} P(e_t) \log(P(e_t)) \qquad (1)$$

where $E_t(r)$ is the tail entity set for r in the KB, and $P(e_t)$ is the probability of e_t appearing as the tail entity for r in the KB.

Salience Ranking: As the first ranking step, we examine the candidates without the context and prefers those with higher importance in the KB. Equation (2) computes the salience score $S_a(c)$ for a candidate c:

$$S_a(c) = \sum_{r \in R(c), e_t \in E_t(r)} H(r) \frac{S_a(e_t)}{L(e_t)} \qquad (2)$$

where $R(c)$ is the relation set for c in the KB; $H(r)$ is given by Equation (1); $E_t(r)$ is the tail entity set with c being the head entity and r being the connecting relation in the KB; $L(e_t)$ denotes the

[1] http://wiki.dbpedia.org
[2] http://bioportal.bioontology.org
[3] http://www.obofoundry.org

cardinality of the tail entity set with e_t being the head entity in the KB. $S_a(c)$ is recursively computed until convergence.

Collective Ranking: The similarity $Sim^F(m, c)$ between a candidate c and its mention m is defined using Equation (3) as the final ranking score:

$$Sim^F(m,c) = \alpha \cdot JS(m,c) \cdot S_a(c) + \beta \cdot \sum_{r \in R(c)} H(r) \cdot \sum_{n \in E_t(r) \cap C(m)} S_a(n) \quad (3)$$

where $JS(m, c)$ is the Jaccard similarity between the string surface forms of m and c; $S_a(c)$ and $S_a(n)$ are both evaluated by Equation (2); $C(m)$ denotes the candidate set for mention m; α and β are hyperparameters.

3 Quantified Collective Validation

Incorporating the KB relation weighing mechanism of the baseline system, our QCV algorithm constructs a number of candidate graphs for a given set of collaborative mentions, and then performs a two-level ranking followed by a collective validation on those candidate graphs to acquire the linking results. Because this procedure minimally relies on linguistic analysis of the source documents while mainly uses the KB structure which by nature keeps detached from any specific language or domain, we claim that QCV comes with language and domain independence.

3.1 Candidate Graph Construction

The KB entity candidate graphs are constructed based on a mention context graph and a KB graph. We will introduce them in order as follows.

Mention Context Graph: To avoid abusing linguistic knowledge from the source documents, we construct a mention context graph G_m simply involving mention co-occurrence. Figure 1 depicts a constructed G_m for the *Caldwell* example at the beginning of Section 1. In this figure, mentions "*New York Jets*", "*Cotchery*" and "*Coles*" are brought into G_m through the coreference between "*Jets*" and "*New York Jets*" since the three of them are outside the context window of "*Caldwell*", "*Florida*", "*Patriots*", and "*Jets*". G_m contains a set of vertices representing the mentions extracted from the source document and a set of undirected edges. There will be an edge between two mention vertices if both of them fall into a context window with width w_m in the source document. Ideally, w_m should cover a single discourse according to the one sense per discourse assumption (Gale et al., 1992), but for simplicity we heuristically set w_m to be 7-sentence wide as a hyperparameter. Two mention vertices will be connected via a dashed edge if they are coreferential but are not located in the same context window. Here we determine the coreference by performing substring matching and abbreviation expansion. The dashed edge indicates the out-of-context coreferential mention together with its neighbors will be indirectly included in G_m as extended context to later facilitate the candidate graph collective validation. Note that all of these loose settings comply with our intention of generating a light-weight source context representation born with domain and language independence.

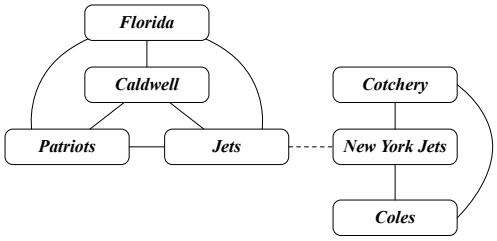


Figure 1: Mention context graph for the *Caldwell* example.

KB Graph: A structured KB such as DBpedia can be represented as a weighted graph G_k that consists of a set of vertices representing the entities and a set of directed edges labeled with relations between entities. The weights of relations are computed using Equation (1). In order to further enrich the KB relations, we add a type of relation named "wiki link" between two entities if one of them appears in the Wikipedia article of the other. Figure 2 presents a subgraph of the DBpedia KB graph containing the relevant entities in the *Caldwell* example.

Candidate Graph: The candidate graph is a set of graphs G_c^i $(i = 1, 2, ...)$ used for computing ranking scores for the KB entity candidates. For each of the mentions extracted from the source context, we first select a list of entity candidates from G_k with heuristic rules such as fuzzy string matching, synonyms, Wikipedia redirect, etc. Then we pick one candidate for each of the mentions to constitute the vertices of a G_c^i. In each G_c^i, we add an edge between two vertices if they are connected in G_k by some relation r and their mentions are connected in G_m. The edge label r from G_k is transferred to G_c^i. Upon comple-

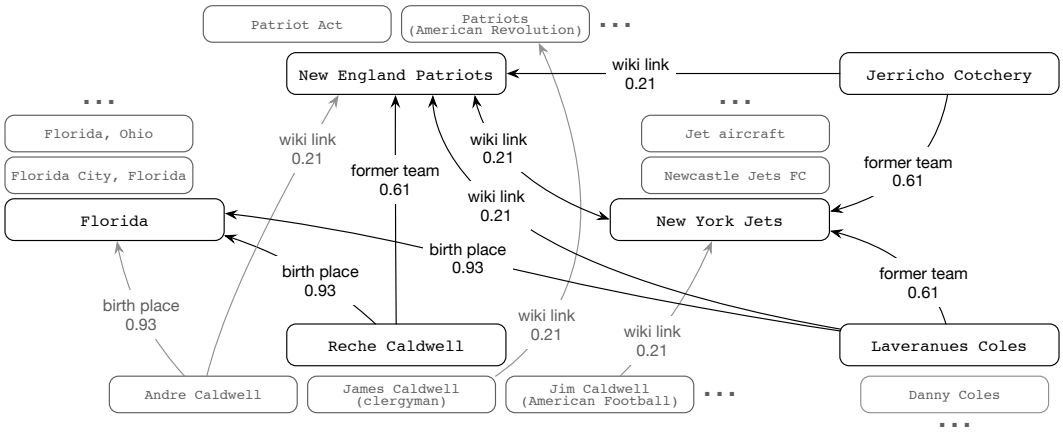


Figure 2: KB graph for the *Caldwell* example.

tion, every G_c^i represents a collective linking solution to the given mention set. Figure 3 shows three of the constructed candidate graphs for the *Caldwell* example. One can see that the first two graphs are very likely to be good solutions since they inherit many of the relation edges from G_K, while the third one is probably a poor collection as the candidates barely connect to one another. In the next section, we will more formally reveal how to rank these candidate graphs to obtain the optimal linking results.

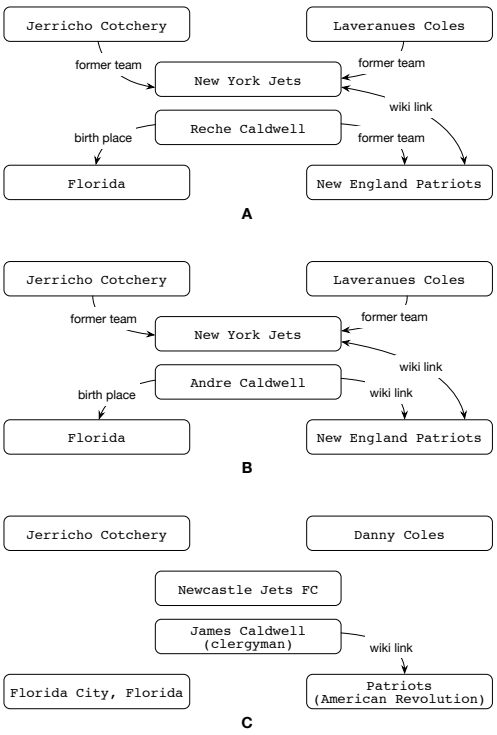


Figure 3: Candidate graphs for the *Caldwell* example.

3.2 Candidate Ranking

With the constructed candidate graphs, QCV performs two levels of ranking. First, it uses Equation (2) to compute the candidates' salience scores as a priori ranking. Then it compares each candidate graph with the mention context graph, and evaluates their vertex set similarity for context similarity ranking. Finally, by considering the relation weights in the candidate graphs as well as previous ranking scores, QCV collectively validates all the candidates and assembles the linking results. Below we will focus on introducing the context similarity ranking and the collective validation since the salience ranking resembles that of our baseline system.

Context Similarity Ranking: As shown in Figure 3, among the constructed candidate graphs, some of them contain many connected vertices while some are otherwise quite disconnected. Intuitively we would like to measure this structure difference by comparing each candidate graph G_c^i with its mention context graph G_m. Granted, we can only assert co-occurrence between two connected mentions in G_m, but it should be of great probability that two co-occurring mentions have their entity referents connected by some relation in the KB. In other words, the more a G_c^i is structurally similar to its G_m, the better the candidates in this G_c^i represent their mentions in G_m. Therefore, we define the context similarity $S_m(m_c, c)$ between a candidate c and its mention m_c using Jaccard similarity in Equation (4):

$$S_m(m_c, c) = \frac{|\Theta^{G_m}(m_c) \cap \Theta^{G_c^i}(c)|}{|\Theta^{G_m}(m_c) \cup \Theta^{G_c^i}(c)|} \quad (4)$$

where $\Theta^{G_m}(m_c)$ and $\Theta^{G_c^i}(c)$ denote m_c's neighbor set in G_m and c's neighbor set in G_c^i, respectively. The intersection takes the candidates of those mentions in $\Theta^{G_m}(m_c)$ that appear in $\Theta^{G_c^i}(c)$, and the union is equivalent to $\Theta^{G_m}(m_c)$ due to the way we construct G_c^i. We rank G_c^i using the summation of the context similarity of every c in G_c^i. Note that our baseline system uses Jaccard similarity to achieve approximate string match between the surface forms of a mention and a candidate, while we alternatively use it to capture the graph's structural similarity. After ranking with the context similarity, those G_c^i with more connected vertices such as Figure 3A and Figure 3B will get closer to the top of the ranked candidate graph list.

Candidate Graph Collective Validation: Besides the salience, the context similarity provides another ranking score for each candidate c in G_c^i, and it promotes those candidates remaining connected in G_c^i. However, it fails to differentiate how two candidates are connected. In Figure 3A, `Reche Caldwell` is a former player of `New England Patriots`, and in Figure 3B, `Andre Caldwell`'s Wikipedia article includes a hyperlink pointing to `New England Patriots`. The former seems a "tighter" relation than the latter. Although these two distinct relations imply that these two candidate pairs are related with different relation types, the context similarity rankings for these two candidate graphs are identical. Based on this observation, assuming that a "tighter" relation between two candidates is more likely to be an appropriate representation of the relation between their co-occurring mentions in the source context, we propose a novel validation step that not only considers the two previous ranking scores of each candidate but also quantitatively examines the relations between candidates. We transfer the calculated relation weights from G_k to G_c^i as positive indicators of how tightly two candidates are related, and then define the composite graph weight $W(G_c^i)$ for each G_c^i in Equation (5) as the final ranking metric:

$$W(G_c^i) = \sum_{c \in V(G_c^i)} S_a(c) S_m(m_c, c) + \sum_{r \in E(G_c^i)} H(r) \quad (5)$$

where $V(G_c^i)$ and $E(G_c^i)$ are the vertex set and the edge set of G_c^i; $S_a(c)$, $S_m(m_c, c)$, and $H(r)$ are given by Equation (2), Equation (4), and Equation (1), respectively. With this composite graph weight, since the relation "*former team*" has a greater weight than "*wiki link*", the candidate graph in Figure 3A outweighs that in Figure 3B, and therefore is ranked to the top.

4 Experiments

In this section, we first show QCV's performance on generic English corpora and compare it with our baseline together with other state-of-the-art EL systems. Then we move to a new language (Chinese) and two new domains (Biomedical Science and Earth Science) to demonstrate the language and domain independent nature of our algorithm.

4.1 EL on Generic English Corpora

For this evaluation, we used the TAC-KBP2013 EL dataset[1], which contains 2,190 mentions extracted from English newswire, web blogs, and discussion forums. We selected a subset of 1,090 linkable mentions that have entity referents in the KB for our experiment. DBpedia 3.9, which was generated from the Wikipedia dump in early 2013 and includes more than 4 million entities and more than 470 million facts[2], was used as our KB. We followed the KBP EL track using `B-Cubed+` (Ji et al., 2011) as the evaluation metric. Table 1 presents the results of QCV, our baseline system, as well as the top 3 supervised participant systems[3]and the top 3 unsupervised participant systems[3] of the TAC-KBP2013 EL track.

System	B^3+ F_1
Supervised 1st	0.724[4]
Supervised 2nd	0.721[4]
Supervised 3rd	0.718[4]
Unsupervised 1st	0.632[4]
Unsupervised 2nd	0.576[4]
Unsupervised 3rd	0.573[4]
Baseline (unsupervised)	0.697
QCV (unsupervised)	**0.749**

Table 1: Performance on the TAC-KBP2013 EL Dataset (1,090 linkable mentions).

[1]http://www.nist.gov/tac/2013/KBP/data.html

[2]http://wiki.dbpedia.org/services-resources/datasets/dataset-39

[3]Due to NIST policy, the names of the TAC-KBP2013 participant systems are not revealed.

[4]http://www.nist.gov/tac/publications/2013/papers.html

As shown in Table 1, QCV not only substantially outperforms the best unsupervised systems but also beats the best supervised systems from the KBP participants. In order to understand this notable advancement, we broke down our system into components and evaluated them accumulatively using the same dataset as above. The experiment results are summarized in Table 2.

Components	B^3+ P	B^3+ R	B^3+ F_1
SR	0.680	0.598	0.636
SR + CS	0.699	0.624	0.659
SR + CS + CV	**0.789**	**0.712**	**0.749**

Table 2: QCV Performance by Component.

In Table 2, *SR*, *CS*, and *CV* correspond to the Salience Ranking, the Context Similarity Ranking, and the Collective Validation in our QCV algorithm, respectively. It can be seen that *SR* already outperforms the best KBP unsupervised systems from Table 1. This is mainly attributed to the engagement of the entropy based relation weights which injects the impact of different relations into the entity salience. Notwithstanding being somewhat effective, *SR* solely depends on the KB and plays its role without the source context. It should be straightforward that the system performance gets improved after enabling *CS* since the source context has been incorporated. However, it was a little puzzling that the performance boost by enabling *CS* turned out to be relatively small. We took a careful look at the intermediate experiment results and discovered that although *CS* did not produce a lot more correct linking results than *SR* did, it did promote a great number of good candidates to the top of the ranking list. For example, in the *Caldwell* case, *CS* successfully raised the rankings of the context-related candidates such as `Reche Caldwell`, `Andre Caldwell`, and `Jim Caldwell`, despite the fact that it delivered `Andre Caldwell` instead of `Reche Caldwell` as the final linking result. This convincingly implies that *CS* is able to well capture the context of the target mentions, but meanwhile it is deficient in recognizing the subtle contextual difference among similar candidates. In Table 2 there is a significant performance gain after enabling *CV*. As described in Section 3.2, *CV* collectively validates the candidates of the target mention "*Caldwell*" and the mentions in its context such as "*Florida*", "*Patriots*", and "*Jets*" by integrating their *SR* and *CS* scores as well as the weights of the KB relations between them. Therefore this improvement is reasonably substantial.

By investigating the remaining errors, we identified several potential causes: 1) Our system occasionally could not capture enough context for the target mention. This happened more frequently for web blogs and discussion forums, where the language was informal and casual. Without any linguistic analysis on the source documents, it was difficult for us to extract additional context words. 2) Our simple coreference rules sometimes failed to work correctly and introduced false candidates, which, without clear context to disambiguate, could lead to linking errors. 3) Our KB had limited knowledge about some entities in a way that certain relations were missing. This kept us from creating necessary links in the candidate graphs and further effectively validating the graphs.

4.2 EL on Generic Chinese Corpora

Using Chinese as a case study, we evaluate the language portability of our approach. We used the TAC-KBP2012 Chinese EL dataset[1], and selected a subset of 1,240 linkable mentions out of the total 2,122 mentions extracted from Chinese newswire, web blogs, and discussion forums. For KB, we still used DBpedia because it contains multilingual surface forms for its entities. For instance, the entity `Barack Obama` has surface forms in over 30 languages including the Chinese one: "贝拉克·奥巴马". This cross-lingual surface form mapping naturally provides us with a convenient translation tool. Table 3 shows the linking performance comparison among QCV, our baseline system, and the top 3 participant systems of the KBP Chinese EL track. Again, we employed the `B-Cubed+` metric.

System	B^3+ F_1
Clarke et al. (2012) (supervised)	0.493
Monahan and Carpenter (2012) (supervised)	0.660
Fahrni et al. (2012) (supervised)	**0.736**
Baseline (unsupervised)	0.648
QCV (unsupervised)	0.671

Table 3: Performance on the TAC-KBP2012 Chinese EL Dataset (1240 linkable mentions).

As shown in Table 3, the best performance is

[1]http://www.nist.gov/tac/2012/KBP/data.html

achieved by Fahrni et al. (2012), a supervised system using over 20 fine-tuned features and many linguistic resources. In contrast, our QCV is an unsupervised approach without using any labeled data or linguistic resources. During the error analysis, we found that in this dataset multiple mentions are often the variants of the surface form of a single KB entity. For example, "奥巴马" and "欧巴马", being just different Chinese transliterations, both refer to "*Obama*". This fact tends to result in a low recall for our system because one or more of the mention variants may not exist in the KB. We decided to heuristically apply a substring matching in addition to the Wikipedia redirection mapping to boost the recall. However, as one can imagine, this simple strategy will impair the system precision due to the introduced noise. Take "奥巴马" again for example. If we only match its second and third characters, "欧巴马" will be correctly picked, but "巴马镇" (a small town in China) will also be falsely included. Fortunately, our QCV algorithm was able to select and rank candidates complying with the source context. Consequently most of this kind of noise got filtered out, and we thus could produce balanced precision and recall.

We acknowledge that, without performing deeper linguistic analysis on the source documents, the cross-language surface form mapping of the KB plays a crucial role in our approach. One can replace it with any machine translation product which, however, is not always available especially for a low-resource language. We should take advantage of the existing KBs where such cross-lingual mapping has already been widely created. The latest DBpedia provides localized versions in 125 languages[1], for instance.

4.3 EL in Biomedical Science

To demonstrate the domain portability of our approach, we first take the biomedical science domain as a case study. We conducted our experiment using the evaluation dataset created by Zheng et al. (2014) which contains 208 linkable mentions extracted from several biomedical publications. We built our KB with over 300 domain ontologies downloaded from BioPortal. Table 4 compares the linking accuracy of QCV and our baseline system.

As shown in Table 4, our approach achieves similar performance to our baseline system which is the state-of-the-art to our knowledge. However, we were curious why QCV did not improve the baseline system in the biomedical domain as much as it did in the general domain. After some in-depth analysis of the experiment results, we discovered that in this dataset the candidates of the related mentions (*i.e.* those mentions within the same context window) mostly have similar relations in the KB. In other words, for each mention, the candidate entity types are not as diverse as those in the general domain. As a consequence, the collective validation step in QCV does not take much effect since the weights of the involved relations are quite close to one another. On such a dataset, the context similarity ranking will play a major part for the disambiguation, and QCV will not be able to function at its full power. Nonetheless, from the results we can see that our approach can be efficiently and effectively adapted to this new domain.

System	Correct	Total	Accuracy
Baseline	173	208	83.17%
QCV	177	208	**85.10%**

Table 4: Biomedical Science EL Performance.

4.4 EL in Earth Science

Now we move to another new domain, Earth Science. As far as we know, we are the first to study EL in this domain. In order to create an evaluation dataset, our domain expert selected three scientific papers about Early Triassic discovery, Global Stratotype Section, and Triassic crisis, which are three different aspects of Earth Science related discovery, and then identified 296 mentions that can be linked to DBpedia entities. Table 5 presents the linking accuracy comparison between QCV and our baseline system. We can see that QCV provided significant gains.

System	Correct	Total	Accuracy
Baseline	221	296	74.66%
QCV	236	296	**79.73%**

Table 5: Earth Science EL Performance.

The linking errors were mainly caused by the following reasons: 1) As a general KB, DBpedia has introduced certain noise for our domain-

[1] http://wiki.dbpedia.org/about

specific EL. For example, in Geology, the term "*Beds*" mostly refers to "*Geology Bed*", which is a division of a geologic formation. But in general, "*Beds*" usually means the beds people sleep on. Much more common in the KB, the latter had such a significantly higher salience score than the former that the final ranking score of our system got biased. 2) Some relations between Earth Science related entities are not clearly defined in DBpedia. For instance, in geology time scale, the period "*Chattian*" is immediately preceded by the period "*Rupelian*". An explicit relation such as "*preceded by*" should be inserted between these two period entities. Instead, only a vague "*wiki link*" relation is present in our KB. This directly diminishes the differentiating power of our system on the KB relations.

It is worth mentioning that there exists a large number of well established ontologies for different sub-domains of Earth Science. SWEET ontologies[1], for example, widely capture Earth and Environmental terminologies. By adopting these ontologies, we will be able to considerably improve our domain EL performance, and the benefits of EL in the domain will further get revealed.

4.5 System Complexity

We indexed our KB and ontologies in the format of triples using Apache Lucene[2] such that retrieving entity candidates of a mention is $O(1)$. We precomputed all the entropy-based relation weights and entity salience scores with complexities of $O(n_r \cdot n_e)$ and $O(n_e \cdot k)$, respectively, where n_r is the number of KB relations, n_e is the number of KB entities, and k is the number of iterations it took for the salience score to get converged. For the final QCV score computation, the upper bound of the computing time to link all the mentions in a document is $O(n_m \cdot n_c \cdot n_{n_c} \cdot n_{n_m})$, where n_m is the number of linkable mentions in the document, n_c is the number of candidates for each mention, and n_{n_c} is the number of neighbor nodes of a candidate, and n_{n_m} is the number of neighbors of a mention.

5 Related Work

In recent years, collective inference methods for EL have become increasingly popular. Many efforts have been devoted to encoding linguistic features from the source documents in order to precisely select collaborator mentions for collective inference. These features include topic modeling (Xu et al., 2012; Cassidy et al., 2012), relation constraint (Cheng and Roth, 2013), coreferential chaining (Nguyen et al., 2012; Huang et al., 2014), and dependency restriction (Ling et al., 2014). Some recent work utilized multi-layer linguistic analysis integration to capture contextual properties for better mention collection (Pan et al., 2015). While many of these approaches have been proved to be effective, the dependency on deep linguistic knowledge makes it difficult to migrate them to a new language or domain. In contrast to these methods, we establish a very loose setting for the mention selection, and rely on the quantified information computed from the structured KB to collectively evaluate and validate the entity candidates. Since the KB is relatively universal to languages and domains, our approach inherently is language and domain independent.

Recent cross-lingual EL approaches can be divided into two types. The first type (McNamee et al., 2011; Cassidy et al., 2011; McNamee et al., 2012; Guo et al., 2012; Miao et al., 2013) translated entity mentions and source documents from the new language into English and then ran English mono-lingual EL to link to English KB. The second type (Monahan et al., 2011; Fahrni et al., 2011; Fahrni et al., 2012; Monahan and Carpenter, 2012; Clarke et al., 2012; Fahrni et al., 2013) developed EL systems on the new language and used cross-lingual KB links to map the link results back to English KB. While the bottleneck of the former method usually is on translation errors, the latter approach heavily relies on the linguistic resources and the KB of the new language. In comparison, our system mainly uses the English KB and a mention surface form mapping that can either come from translation or cross-lingual KB links, and requires minimal linguistic resources from the new language.

There is a limited amount of research work in the literature that focused solely on domain-specific EL (Zheng et al., 2014). In the biomedical domain, a few studies have been found on EL-related tasks such as scientific name discovery (Akella et al., 2012), gene name normalization (Hirschman et al., 2005; Fang et al., 2006; Dai et al., 2010), biomedical named entity recognition (Usami et al., 2011; Van Landeghem et al.,

[1] http://sweet.jpl.nasa.gov

[2] https://lucene.apache.org/

2012) and concept mention extraction (Tsai et al., 2013). The baseline system (Zheng et al., 2014) in this paper is the work most similar to ours in a sense of collectively aligning mentions to structured KBs. However, our system differs by integrating a context similarity ranking and a candidate validation to conduct a two-way collective inference with better performance.

6 Conclusions and Future Work

Language and domain independence is a new requirement to EL systems and this capability is particularly welcome by low-resource language related applications and domain scientists. In this paper we demonstrated a high-performance EL approach that can be easily migrated to new languages and domains due to the minimal reliance on linguistic analysis and the deep utilization of structured KBs. In the future, we plan to improve the source document processing such that the system can better extract the mention context without involving extensive linguistic knowledge. We are also experimenting with our collective validation algorithm to incorporate the impact of more distant KB entities other than just the neighbors.

7 Acknowledgement

This work was supported by the U.S. DARPA DEFT Program No. FA8750-13-2-0041, ARL NS-CTA No. W911NF-09-2-0053, NSF CAREER Award IIS-1523198, DARPA LORELEI, AFRL DREAM project, gift awards from IBM, Google, Disney and Bosch. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

References

Lakshmi Manohar Akella, Catherine N. Norton, and Holly Miller. 2012. NetiNeti: Discovery of Scientific Names from Text Using Machine Learning Methods. *BMC Bioinformatics*, 13:211.

Taylor Cassidy, Zheng Chen, Javier Artiles, Heng Ji, Hongbo Deng, Lev-Arie Ratinov, Jing Zheng, Jiawei Han, and Dan Roth. 2011. CUNY-UIUC-SRI TAC-KBP2011 Entity Linking System Description. In *Proceedings of Text Analysis Conference 2011*.

Taylor Cassidy, Heng Ji, Lev-Arie Ratinov, Arkaitz Zubiaga, and Hongzhao Huang. 2012. Analysis and Enhancement of Wikification for Microblogs with Context Expansion. In *Proceedings of the 25th International Conference on Computational Linguistics*.

Xiao Cheng and Dan Roth. 2013. Relational Inference for Wikification. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*.

James Clarke, Yuval Merhav, Ghalib Suleiman, Shuai Zheng, and David Murgatroyd. 2012. Basis Technology at TAC 2012 Entity Linking. In *Proceedings of Text Analysis Conference 2012*.

Silviu Cucerzan. 2011. TAC Entity Linking by Performing Full-Document Entity Extraction and Disambiguation. In *Proceedings of Text Analysis Conference 2011*.

Hong-Jie Dai, Po-Ting Lai, and Richard Tzong-Han Tsai. 2010. Multistage Gene Normalization and SVM-Based Ranking for Protein Interactor Extraction in Full-Text Articles. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 7(3):412–420.

Jeffrey Dalton and Laura Dietz. 2013. A Neighborhood Relevance Model for Entity Linking. In *Proceedings of the 10th Conference on Open Research Areas in Information Retrieval*.

Angela Fahrni, Vivi Nastase, and Michael Strube. 2011. HITS' Cross-lingual Entity Linking System at TAC2011: One Model for All Languages. In *Proceedings of Text Analysis Conference 2011*.

Angela Fahrni, Thierry Göckel, and Michael Strube. 2012. HITS' Monolingual and Cross-lingual Entity Linking System at TAC 2012: A Joint Approach. In *Proceedings of the Text Analysis Conference 2012*.

Angela Fahrni, Benjamin Heinzerling, Thierry Göckel, and Michael Strube. 2013. HITS' Monolingual and Cross-lingual Entity Linking System at TAC 2013. In *Proceedings of Text Analysis Conference 2013*.

Haw-ren Fang, Kevin Murphy, Yang Jin, Jessica S. Kim, and Peter S. White. 2006. Human Gene Name Normalization Using Text Matching with Automatically Extracted Synonym Dictionaries. In *Proceedings of the Workshop on Linking Natural Language Processing and Biology: Towards Deeper Biological Literature Analysis*, pages 41–48.

Norberto Fernández, Jesus A. Fisteus, Luis Sánchez, and Eduardo Martín. 2010. WebTLab: A Cooccurence-Based Approach to KBP 2010 Entity-Linking Task. In *Proceedings of Text Analysis Conference 2010*.

William A. Gale, Kenneth W. Church, and David Yarowsky. 1992. One Sense per Discourse. In *Proceedings of the Fifth DARPA Speech and Natural Language Workshop*.

Filipe Guo, Ying Xu, Filipe Mesquita, Denilson Barbosa, and Grzegorz Kondrak. 2012. ualberta at TAC-KBP 2012: English and Cross-Lingual Entity Linking. In *Proceedings of Text Analysis Conference 2012*.

Xianpei Han, Le Sun, and Jun Zhao. 2011. Collective Entity Linking in Web Text: A Graph-Based Method. In *Proceedings of the 34th Annual ACM SIGIR Conference*.

Lynette Hirschman, Marc Colosimo, Alexander Morgan, and Alexander Yeh. 2005. Overview of BioCreAtIvE Task 1B: Normalized Gene Lists. *BMC Bioinformatics*, 6.

Hongzhao Huang, Yunbo Cao, Xiaojiang Huang, Heng Ji, and Chin-Yew Lin. 2014. Collective Tweet Wikification based on Semi-supervised Graph Regularization. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*.

Heng Ji, Ralph Grishman, and Hoa Trang Dang. 2011. Overview of the TAC 2011 Knowledge Base Population Track. In *Proceedings of Text Analysis Conference 2011*.

Xiao Ling, Sameer Singh, and Daniel S. Weld. 2014. Context Representation for Named Entity Linking. In *Proceedings of the 3rd Pacific Northwest Regional NLP Workshop*.

Paul McNamee, James Mayfield, Dawn Lawrie, Douglas W. Oard, and David Doermann. 2011. Cross-Language Entity Linking. In *Proceedings of the 5th International Joint Conference on Natural Language Processing*.

Paul McNamee, Veselin Stoyanov, James Mayfield, Tim Finin, Tim Oates, Tan Xu, Douglas W. Oard, and Dawn Lawrie. 2012. HLTCOE Participation at TAC 2012: Entity Linking and Cold Start Knowledge Base Construction. In *Proceedings of Text Analysis Conference 2012*.

Qingliang Miao, Ruiyu Fang, Yao Meng, and Shu Zhang. 2013. FRDC's Cross-lingual Entity Linking System at TAC 2013. In *Proceedings of Text Analysis Conference 2013*.

Sean Monahan and Dean Carpenter. 2012. Lorify: A Knowledge Base from Scratch. In *Proceedings of Text Analysis Conference 2012*.

Sean Monahan, John Lehmann, Timothy Nyberg, Jesse Plymale, and Arnold Jung. 2011. Cross-Lingual Cross-Document Coreference with Entity Linking. In *Proceedings of Text Analysis Conference 2011*.

Hien T. Nguyen, Huy H. Minha, Tru H. Cao, and Trong T. Nguyen. 2012. JVN-TDT Entity Linking Systems at TAC-KBP2012. In *Proceedings of Text Analysis Conference 2012*.

Xiaoman Pan, Taylor Cassidy, Ulf Hermjakob, Heng Ji, and Kevin Knight. 2015. Unsupervised Entity Linking with Abstract Meaning Representation. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics – Human Language Technologies*.

Marco Pennacchiotti and Patrick Pantel. 2009. Entity Extraction via Ensemble Semantics. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing EMNLP2009*.

Lev-Arie Ratinov, Dan Roth, Doug Downey, and Mike Anderson. 2011. Local and Global Algorithms for Disambiguation to Wikipedia. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*.

Chen-Tse Tsai, Gourab Kundu, and Dan Roth. 2013. Concept-Based Analysis of Scientific Literature. In *Proceedings of 22nd ACM International Conference on Information and Knowledge Management (CIKM 2013)*.

Yu Usami, Han-Cheol Cho, Naoaki Okazaki, and Jun'ichi Tsujii. 2011. Automatic Acquisition of Huge Training Data for Bio-medical Named Entity Recognition. In *Proceedings of BioNLP 2011 Workshop*.

Sofie Van Landeghem, Jari Björne, Thomas Abeel, Bernard De Baets, Tapio Salakoski, and Yves Van de Peer. 2012. Semantically Linking Molecular Entities in Literature through Entity Relationships. *BMC Bioinformatics*, 13.

Jian Xu, Qin Lu, Jie Liu, and Ruifeng Xu. 2012. NLPComp in TAC 2012 Entity Linking and Slot-Filling. In *Proceedings of Text Analysis Conference 2012*.

Jin Guang Zheng, Daniel Howsmon, Boliang Zhang, Juergen Hahn, Deborah McGuinness, James Hendler, and Heng Ji. 2014. Entity Linking for Biomedical Literature. In *Proceedings of the ACM 8th International Workshop on Data and Text Mining in Bioinformatics*.

Modeling Relation Paths for Representation Learning of Knowledge Bases

Yankai Lin[1], **Zhiyuan Liu**[1] *, **Huanbo Luan**[1], **Maosong Sun**[1], **Siwei Rao**[2], **Song Liu**[2]

[1] Department of Computer Science and Technology, State Key Lab on Intelligent Technology and Systems, National Lab for Information Science and Technology, Tsinghua University, Beijing, China

[2] Samsung R&D Institute of China, Beijing, China

Abstract

Representation learning of knowledge bases aims to embed both entities and relations into a low-dimensional space. Most existing methods only consider direct relations in representation learning. We argue that multiple-step relation paths also contain rich inference patterns between entities, and propose a path-based representation learning model. This model considers relation paths as translations between entities for representation learning, and addresses two key challenges: (1) Since not all relation paths are reliable, we design a path-constraint resource allocation algorithm to measure the reliability of relation paths. (2) We represent relation paths via semantic composition of relation embeddings. Experimental results on real-world datasets show that, as compared with baselines, our model achieves significant and consistent improvements on knowledge base completion and relation extraction from text. The source code of this paper can be obtained from `https://github.com/mrlyk423/relation_extraction`.

1 Introduction

People have recently built many large-scale knowledge bases (KBs) such as Freebase, DBpedia and YAGO. These KBs consist of facts about the real world, mostly in the form of triples, e.g., (*Steve Jobs*, `FounderOf`, *Apple Inc.*). KBs are important resources for many applications such as question answering and Web search. Although typical KBs are large in size, usually containing thousands of relation types, millions of entities and billions of facts (triples), they are far from complete. Hence, many efforts have been invested in relation extraction to enrich KBs.

Recent studies reveal that, neural-based representation learning methods are scalable and effective to encode relational knowledge with low-dimensional representations of both entities and relations, which can be further used to extract unknown relational facts. TransE (Bordes et al., 2013) is a typical method in the neural-based approach, which learns vectors (i.e., embeddings) for both entities and relations. The basic idea behind TransE is that, the relationship between two entities corresponds to a translation between the embeddings of the entities, that is, $\mathbf{h} + \mathbf{r} \approx \mathbf{t}$ when the triple (h, r, t) holds. Since TransE has issues when modeling 1-to-N, N-to-1 and N-to-N relations, various methods such as TransH (Wang et al., 2014) and TransR (Lin et al., 2015) are proposed to assign an entity with different representations when involved in various relations.

Despite their success in modeling relational facts, TransE and its extensions only take direct relations between entities into consideration. It is known that there are also substantial multiple-step relation paths between entities indicating their semantic relationships. The relation paths reflect complicated inference patterns among relations in KBs. For example, the relation path $h \xrightarrow{\texttt{BornInCity}} e_1 \xrightarrow{\texttt{CityInState}} e_2 \xrightarrow{\texttt{StateInCountry}} t$ indicates the relation `Nationality` between h and t, i.e., (h, `Nationality`, t).

In this paper, we aim at extending TransE to model relation paths for representation learning of KBs, and propose path-based TransE (PTransE). In PTransE, in addition to direct connected relational facts, we also build triples from KBs using entity pairs connected with relation paths. As shown in Figure 1, TransE only considers direct relations between entities, e.g., $h \xrightarrow{r} t$, builds a triple (h, r, t), and optimizes the objec-

*Corresponding author: Z. Liu (liuzy@tsinghua.edu.cn)

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 705–714, Lisbon, Portugal, 17-21 September 2015.

tive $\mathbf{h} + \mathbf{r} = \mathbf{t}$. PTransE generalizes TransE by regarding multiple-step relation paths as connections between entities. Take the 2-step path $h \xrightarrow{r_1} e_1 \xrightarrow{r_2} t$ for example as shown in Figure 1. Besides building triples (h, r_1, e_1) and (e_1, r_2, t) for learning as in TransE, PTransE also builds a triple $(h, r_1 \circ r_2, t)$, and optimizes the objective $\mathbf{h} + (\mathbf{r}_1 \circ \mathbf{r}_2) = \mathbf{t}$, where $\circ$ is an operation to join the relations r_1 and r_2 together into a unified relation path representation.

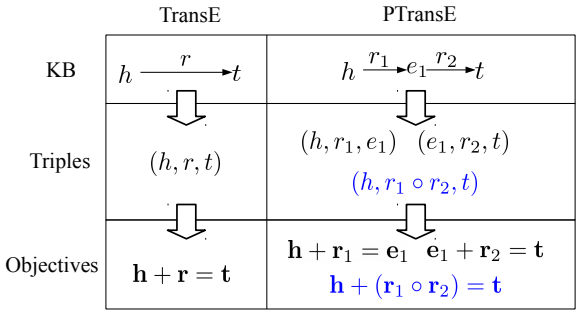


Figure 1: TransE and PTransE.

As compared with TransE, PTransE takes rich relation paths in KBs for learning. There are two critical challenges that make PTransE nontrivial to learn from relation paths:

Relation Path Reliability. Not all relation paths are meaningful and reliable for learning. For example, there is often a relation path $h \xrightarrow{\texttt{Friend}} e_1 \xrightarrow{\texttt{Profession}} t$, but actually it does not indicate any semantic relationship between h and t. Hence, it is inappropriate to consider all relation paths in our model. In experiments, we find that those relation paths that lead to lots of possible tail entities are mostly unreliable for the entity pair. In this paper, we propose a path-constraint resource allocation algorithm to measure the reliability of relation paths. Afterwards, we select the reliable relation paths for representation learning.

Relation Path Representation. In order to take relation paths into consideration, relation paths should also be represented in a low-dimensional space. It is straightforward that the semantic meaning of a relation path depends on all relations in this path. Given a relation path $p = (r_1, \ldots, r_l)$, we will define and learn a binary operation function ($\circ$) to obtain the path embedding $\mathbf{p}$ by recursively composing multiple relations, i.e., $\mathbf{p} = \mathbf{r}_1 \circ \ldots \circ \mathbf{r}_l$.

With relation path selection and representation, PTransE learns entity and relation embeddings by regarding relation paths as translations between the corresponding entities. In experiments, we select a typical KB, Freebase, to build datasets and carry out evaluation on three tasks, including entity prediction, relation prediction and relation extraction from text. Experimental results show that, PTransE significantly outperforms TransE and other baseline methods on all three tasks.

2 Our Model

In this section, we introduce path-based TransE (PTransE) that learns representations of entities and relations considering relation paths. In TransE and PTransE, we have entity set $\mathbf{E}$ and relation set $\mathbf{R}$, and learn to encode both entities and relations in $\mathbb{R}^k$. Given a KB represented by a set of triples $\mathbf{S} = \{(h, r, t)\}$ with each triple composed of two entities $h, t \in \mathbf{E}$ and their relation $r \in \mathbf{R}$. Our model is expected to return a low energy score when the relation holds, and a high one otherwise.

2.1 TransE and PTransE

For each triple (h, r, t), TransE regards the relation as a translation vector $\mathbf{r}$ between two entity vectors $\mathbf{h}$ and $\mathbf{t}$. The energy function is defined as

$$E(h, r, t) = ||\mathbf{h} + \mathbf{r} - \mathbf{t}||, \quad (1)$$

which is expected to get a low score when (h, r, t) holds, and high otherwise.

TransE only learns from direct relations between entities but ignores multiple-step relation paths, which also contain rich inference patterns between entities. PTransE take relation paths into consideration for representation learning.

Suppose there are multiple relation paths $P(h, t) = \{p_1, \ldots, p_N\}$ connecting two entities h and t, where relation path $p = (r_1, \ldots, r_l)$ indicates $h \xrightarrow{r_1} \ldots \xrightarrow{r_l} t$. For each triple (h, r, t), the energy function is defined as

$$G(h, r, t) = E(h, r, t) + E(h, P, t), \quad (2)$$

where $E(h, r, t)$ models correlations between relations and entities with direct relation triples, as defined in Equation (1). $E(h, P, t)$ models the inference correlations between relations with multiple-step relation path triples, which is defined as

$$E(h, P, t) = \frac{1}{Z} \sum_{p \in P(h,t)} R(p|h, t) E(h, p, t), \quad (3)$$

where $R(p|h, t)$ indicates the reliability of the relation path p given the entity pair (h, t), Z =

$\sum_{p \in P(h,t)} R(p|h,t)$ is a normalization factor, and $E(h,p,t)$ is the energy function of the triple (h,p,t).

For the energy of each triple (h,p,t), the component $R(p|h,t)$ concerns about relation path reliability, and $E(h,p,t)$ concerns about relation path representation. We introduce the two components in detail as follows.

2.2 Relation Path Reliability

We propose a path-constraint resource allocation (PCRA) algorithm to measure the reliability of a relation path. Resource allocation over networks was originally proposed for personalized recommendation (Zhou et al., 2007), and has been successfully used in information retrieval for measuring relatedness between two objects (Lü and Zhou, 2011). Here we extend it to PCRA to measure the reliability of relation paths. The basic idea is, we assume that a certain amount of resource is associated with the head entity h, and will flow following the given path p. We use the resource amount that eventually flows to the tail entity t to measure the reliability of the path p as a meaningful connection between h and t.

Formally, for a path triple (h,p,t), we compute the resource amount flowing from h to t given the path $p = (r_1, \ldots, r_l)$ as follows. Starting from h and following the relation path p, we can write the flowing path as $S_0 \xrightarrow{r_1} S_1 \xrightarrow{r_2} \ldots \xrightarrow{r_l} S_l$, where $S_0 = h$ and $t \in S_l$.

For any entity $m \in S_i$, we denote its direct predecessors along relation r_i in S_{i-1} as $S_{i-1}(\cdot, m)$. The resource flowing to m is defined as

$$R_p(m) = \sum_{n \in S_{i-1}(\cdot,m)} \frac{1}{|S_i(n,\cdot)|} R_p(n), \quad (4)$$

where $S_i(n,\cdot)$ is the direct successors of $n \in S_{i-1}$ following the relation r_i, and $R_p(n)$ is the resource obtained from the entity n.

For each relation path p, we set the initial resource in h as $R_p(h) = 1$. By performing resource allocation recursively from h through the path p, the tail entity t eventually obtains the resource $R_p(t)$ which indicates how much information of the head entity h can be well translated. We use $R_p(t)$ to measure the reliability of the path p given (h,t), i.e., $R(p|h,t) = R_p(t)$.

2.3 Relation Path Representation

Besides relation path reliability, we also need to define energy function $E(h,p,t)$ for the path triple (h,p,t) in Equation (2). Similar with the energy function of TransE in Equation (1), we will also represent the relation path p in the embedding space.

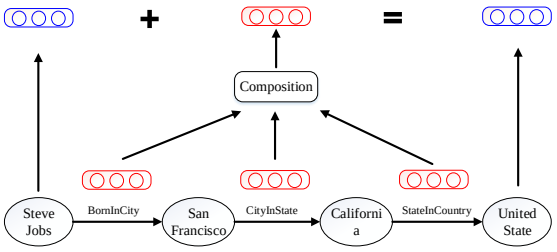


Figure 2: Path representations are computed by semantic composition of relation embeddings.

The semantic meaning of a relation path considerably relies on its involved relations. It is thus reasonable for us to build path embeddings via semantic composition of relation embeddings. As illustrated in Figure 2, the path embedding $\mathbf{p}$ is composed by the embeddings of `BorninCity`, `CityInState` and `StateInCountry`.

Formally, for a path $p = (r_1, \ldots, r_l)$, we define a composition operation $\circ$ and obtain path embedding as $\mathbf{p} = \mathbf{r}_1 \circ \ldots \circ \mathbf{r}_l$. In this paper, we consider three types of composition operation:

Addition (ADD). The addition operation obtains the vector of a path by summing up the vectors of all relations, which is formalized as

$$\mathbf{p} = \mathbf{r}_1 + \ldots + \mathbf{r}_l. \quad (5)$$

Multiplication (MUL). The multiplication operation obtains the vector of a path as the cumulative product of the vectors of all relations, which is formalized as

$$\mathbf{p} = \mathbf{r}_1 \cdot \ldots \cdot \mathbf{r}_l. \quad (6)$$

Both addition and multiplication operations are simple and have been extensively investigated in semantic composition of phrases and sentences (Mitchell and Lapata, 2008).

Recurrent Neural Network (RNN). RNN is a recent neural-based model for semantic composition (Mikolov et al., 2010). The composition operation is realized using a matrix W:

$$c_i = f(W[c_{i-1}; r_i]), \quad (7)$$

where f is a non-linearity or identical function, and $[a;b]$ represents the concatenation of two vec-

tors. By setting $c_1 = r_1$ and recursively performing RNN following the relation path, we will finally obtain $p = c_n$. RNN has also been used for representation learning of relation paths in KBs (Neelakantan et al., 2015).

For a multiple-step relation path triple (h, p, t), we could have followed TransE and define the energy function as $E(h, p, t) = ||\mathbf{h} + \mathbf{p} - \mathbf{t}||$. However, since we have minimized $||\mathbf{h} + \mathbf{r} - \mathbf{t}||$ with the direct relation triple (h, r, t) to make sure $\mathbf{r} \approx \mathbf{t} - \mathbf{h}$, we may directly define the energy function of (h, p, t) as

$$E(h, p, t) = ||\mathbf{p} - (\mathbf{t} - \mathbf{h})|| = ||\mathbf{p} - \mathbf{r}|| = E(p, r), \tag{8}$$

which is expected to be a low score when the multiple-relation path p is consistent with the direct relation r, and high otherwise, without using entity embeddings.

2.4 Objective Formalization

We formalize the optimization objective of PTransE as

$$L(\mathbf{S}) = \sum_{(h,r,t)\in\mathbf{S}} \Big[L(h, r, t) + \frac{1}{Z} \sum_{p\in P(h,t)} R(p|h, t) L(p, r) \Big]. \tag{9}$$

Following TransE, $L(h, r, t)$ and $L(p, r)$ are margin-based loss functions with respect to the triple (h, r, t) and the pair (p, r):

$$L(h, r, t) = \sum_{(h',r',t')\in\mathbf{S}^-} [\gamma + E(h, r, t) - E(h', r', t')]_+, \tag{10}$$

and

$$L(p, r) = \sum_{(h,r',t)\in\mathbf{S}^-} [\gamma + E(p, r) - E(p, r')]_+, \tag{11}$$

where $[x]_+ = \max(0, x)$ returns the maximum between 0 and x, γ is the margin, $\mathbf{S}$ is the set of valid triples existing in a KB and $\mathbf{S}^-$ is the set of invalid triples. The objective will favor lower scores for valid triples as compared with invalid triples.

The invalid triple set with respect to (h, r, t) is defined as

$$\mathbf{S}^- = \{(h', r, t)\} \cup \{(h, r', t)\} \cup \{(h, r, t')\}. \tag{12}$$

That is, the set of invalid triples is composed of the original valid triple (h, r, t) with one of three components replaced.

2.5 Optimization and Implementation Details

For optimization, we employ stochastic gradient descent (SGD) to minimize the loss function. We randomly select a valid triple from the training set iteratively for learning. In the implementation, we also enforce constraints on the norms of the embeddings h, r, t. That is, we set

$$\|\mathbf{h}\|_2 \leq 1, \quad \|\mathbf{r}\|_2 \leq 1, \quad \|\mathbf{t}\|_2 \leq 1. \quad \forall h, r, t. \tag{13}$$

There are also some implementation details that will significantly influence the performance of representation learning, which are introduced as follows.

Reverse Relation Addition. In some cases, we are interested in the reverse version of a relation, which may not be presented in KBs. For example, according to the relation path $e_1 \xrightarrow{\texttt{BornInCity}} e_2 \xleftarrow{\texttt{CityOfCountry}} e_3$ we expect to infer the fact that $(e_1, \texttt{Nationality}, e3)$. In this paper, however, we only consider the relation paths following one direction. Hence, we add reverse relations for each relation in KBs. That is, for each triple (h, r, t) we build another (t, r^{-1}, h). In this way, our method can consider the above-mentioned path as $e_1 \xrightarrow{\texttt{BornInCity}} e_2 \xrightarrow{\texttt{CityOfCountry}^{-1}} e_3$ for learning.

Path Selection Limitation. There are usually large amount of relations and facts about each entity pair. It will be impractical to enumerate all possible relation paths between head and tail entities. For example, if each entity refers to more than 100 relations on average, which is common in Freebase, there will be billions of 4-step paths. Even for 2-step or 3-step paths, it will be time-consuming to consider all of them without limitation. For computational efficiency, in this paper we restrict the length of paths to at most 3-steps and consider those relation paths with the reliability score larger than 0.01.

2.6 Complexity Analysis

We denote N_e as the number of entities, N_r as the number of relations and K as the vector dimension. The model parameter size of PTransE is $(N_e K + N_r K)$, which is the same as TransE. PTransE follows the optimization procedure introduced by (Bordes et al., 2013) to solve Equation (9). We denote S as the number of triples for learning, P as the expected number of relation paths between two entities, and L as the expected

length of relation paths. For each iteration in optimization, the complexity of TransE is $O(SK)$ and the complexity of PTransE is $O(SKPL)$ for ADD and MUL, and $O(SK^2PL)$ for RNN.

3 Experiments and Analysis

3.1 Data Sets and Experimental Setting

We evaluate our method on a typical large-scale KB Freebase (Bollacker et al., 2008). In this paper, we adopt two datasets extracted from Freebase, i.e., FB15K and FB40K. The statistics of the datasets are listed in Table 1.

Table 1: Statistics of data sets.

Dataset	#Rel	#Ent	#Train	#Valid	# Test
FB15K	1,345	14,951	483,142	50,000	59,071
FB40K	1,336	39,528	370,648	67,946	96,678

We evaluate the performance of PTransE and other baselines by predicting whether testing triples hold. We consider two scenarios: (1) Knowledge base completion, aiming to predict the missing entities or relations in given triples only based on existing KBs. (2) Relation extraction from texts, aiming to extract relations between entities based on information from both plain texts and KBs.

3.2 Knowledge Base Completion

The task of knowledge base completion is to complete the triple (h, r, t) when one of h, t, r is missing. The task has been used for evaluation in (Bordes et al., 2011; Bordes et al., 2012; Bordes et al., 2013). We conduct the evaluation on FB15K, which has $483,142$ relational triples and $1,345$ relation types, among which there are rich inference and reasoning patterns.

In the testing phase, for each testing triple (h, r, t), we define the following score function for prediction

$$S(h, r, t) = G(h, r, t) + G(t, r^{-1}, h), \quad (14)$$

and the score function $G(h, r, t)$ is further defined as

$$G(h, r, t) = ||\mathbf{h} + \mathbf{r} - \mathbf{t}|| + \frac{1}{Z} \sum_{p \in P(h,t)} \Pr(r|p) R(p|h, t) ||\mathbf{p} - \mathbf{r}||. \quad (15)$$

The score function is similar to the energy function defined in Section 2.1. The difference is that, here we consider the reliability of a path p is also related to the inference strength given r, which is quantified as $\Pr(r|p) = \Pr(r, p)/\Pr(p)$ obtained from the training data.

We divide the stage into two sub-tasks, i.e., entity prediction and relation prediction.

3.2.1 Entity Prediction

In the sub-task of entity prediction, we follow the setting in (Bordes et al., 2013). For each testing triple with missing head or tail entity, various methods are asked to compute the scores of $G(h, r, t)$ for all candidate entities and rank them in descending order.

We use two measures as our evaluation metrics: the mean of correct entity ranks and the proportion of valid entities ranked in top-10 (Hits@10). As mentioned in (Bordes et al., 2013), the measures are desirable but flawed when an invalid triple ends up being valid in KBs. For example, when the testing triple is (*Obama*, `PresidentOf`, *USA*) with the head entity *Obama* is missing, the head entity *Lincoln* may be regarded invalid for prediction, but in fact it is valid in KBs. The evaluation metrics will under-estimate those methods that rank these triples high. Hence, we can filter out all these valid triples in KBs before ranking. The first evaluation setting was named as "Raw" and the second one as "Filter".

For comparison, we select all methods in (Bordes et al., 2013; Wang et al., 2014) as our baselines and use their reported results directly since the evaluation dataset is identical.

Ideally, PTransE has to find all possible relation paths between the given entity and each candidate entity. However, it is time consuming and impractical, because we need to iterate all candidate entities in $|E|$ for each testing triple. Here we adopt a re-ranking method: we first rank all candidate entities according to the scores from TransE, and then re-rank top-500 candidates according to the scores from PTransE.

For PTransE, we find the best hyperparameters according to the mean rank in validation set. The optimal configurations of PTransE we used are $\lambda = 0.001$, $\gamma = 1$, $k = 100$ and taking L_1 as dissimilarity. For training, we limit the number of epochs over all the training triples to 500.

Evaluation results of entity prediction are shown in Table 2. The baselines include RESCAL (Nickel et al., 2011), SE (Bordes et al., 2011), SME (linear) (Bordes et al., 2012), SME (bilinear)

Table 2: Evaluation results on entity prediction.

Metric	Mean Rank		Hits@10 (%)	
	Raw	Filter	Raw	Filter
RESCAL	828	683	28.4	44.1
SE	273	162	28.8	39.8
SME (linear)	274	154	30.7	40.8
SME (bilinear)	284	158	31.3	41.3
LFM	283	164	26.0	33.1
TransE	243	125	34.9	47.1
TransH	212	87	45.7	64.4
TransR	**198**	77	48.2	68.7
TransE (Our)	205	63	47.9	70.2
PTransE (ADD, 2-step)	200	**54**	**51.8**	83.4
PTransE (MUL, 2-step)	216	67	47.4	77.7
PTransE (RNN, 2-step)	242	92	50.6	82.2
PTransE (ADD, 3-step)	207	58	51.4	**84.6**

(Bordes et al., 2012), LFM (Jenatton et al., 2012), TransE (Bordes et al., 2013) (original version and our implementation considering reverse relations), TransH (Wang et al., 2014), and TransR (Lin et al., 2015).

For PTransE, we consider three composition operations for relation path representation: addition (ADD), multiplication (MUL) and recurrent neural networks (RNN). We also consider relation paths with at most 2-steps and 3-steps. With the same configurations of PTransE, our TransE implementation achieves much better performance than that reported in (Bordes et al., 2013).

From Table 2 we observe that: (1) PTransE significantly and consistently outperforms other baselines including TransE. It indicates that relation paths provide a good supplement for representation learning of KBs, which have been successfully encoded by PTransE. For example, since both *George W. Bush* and *Abraham Lincoln* were presidents of *the United States*, they exhibit similar embeddings in TransE. This may lead to confusion for TransE to predict the spouse of *Laura Bush*. In contrast, since PTransE models relation paths, it can take advantage of the relation paths between *George W. Bush* and *Laura Bush*, and leads to more accurate prediction. (2) For PTransE, the addition operation outperforms other composition operations in both Mean Rank and Hits@10. The reason is that, the addition operation is compatible with the learning objectives of both TransE and PTransE. Take $h \xrightarrow{r_1} e_1 \xrightarrow{r_2} t$ for example. The optimization objectives of two direct relations $\mathbf{h} + \mathbf{r}_1 = \mathbf{e}_1$ and $\mathbf{e}_1 + \mathbf{r}_2 = \mathbf{t}$ can easily derive the path objective $\mathbf{h} + \mathbf{r}_1 + \mathbf{r}_2 = \mathbf{t}$. (3) PTransE of considering relation paths with at most 2-step and 3-step achieve comparable results. This indicates that it may be unnecessary to consider those relation paths that are too long.

As defined in (Bordes et al., 2013), relations in KBs can be divided into various types according to their mapping properties such as 1-to-1, 1-to-N, N-to-1 and N-to-N. Here we demonstrate the performance of PTransE and some baselines with respect to different types of relations in Table 3. It is observed that, on all mapping types of relations, PTransE consistently achieves significant improvement as compared with TransE.

3.2.2 Relation Prediction

Relation prediction aims to predict relations given two entities. We also use FB15K for evaluation. In this sub-task, we can use the score function of PTransE to rank candidate relations instead of reranking like in entity prediction.

Since our implementation of TransE has achieved the best performance among all baselines for entity prediction, here we only compare PTransE with TransE due to limited space. Evaluation results are shown in Table 4, where we report Hits@1 instead of Hits@10 for comparison, because Hits@10 for both TransE and PTransE exceeds 95%. In this table, we report the performance of TransE without reverse relations (TransE), with reverse relations (+Rev) and considering relation paths for testing like that in PTransE (+Rev+Path). We report the performance of PTransE with only considering relation paths (-TransE), only considering the part in Equation (1) (-Path) and considering both (PTransE).

The optimal configurations of PTransE for relation prediction are identical to those for entity prediction: $\lambda = 0.001$, $\gamma = 1$, $k = 100$ and taking L_1 as dissimilarity.

From Table 4 we observe that: (1) PTransE outperforms TransE+Rev+Path significantly for relation prediction by reducing 41.8% prediction errors. (2) Even for TransE itself, considering relation paths for testing can reduce 17.3% errors as compared with TransE+Rev. It indicates that encoding relation paths will benefit for predicting relations. (3) PTransE with only considering relation paths (PTransE-TransE) gets surprisingly high mean rank. The reason is that, not all entity pairs in testing triples have relation paths, which will lead to random guess and the expectation of rank of these entity pairs is $|R|/2$. Meanwhile, Hits@1 of PTransE-TransE is relatively reasonable, which indicates the worth of modeling rela-

Table 3: Evaluation results on FB15K by mapping properties of relations. (%)

Tasks	Predicting Head Entities (Hits@10)				Predicting Tail Entities (Hits@10)			
Relation Category	1-to-1	1-to-N	N-to-1	N-to-N	1-to-1	1-to-N	N-to-1	N-to-N
SE	35.6	62.6	17.2	37.5	34.9	14.6	68.3	41.3
SME (linear)	35.1	53.7	19.0	40.3	32.7	14.9	61.6	43.3
SME (bilinear)	30.9	69.6	19.9	38.6	28.2	13.1	76.0	41.8
TransE	43.7	65.7	18.2	47.2	43.7	19.7	66.7	50.0
TransH	66.8	87.6	28.7	64.5	65.5	39.8	83.3	67.2
TransR	78.8	89.2	34.1	69.2	79.2	37.4	**90.4**	72.1
TransE (Our)	74.6	86.6	43.7	70.6	71.5	49.0	85.0	72.9
PTransE (ADD, 2-step)	**91.0**	**92.8**	**60.9**	83.8	**91.2**	**74.0**	88.9	86.4
PTransE (MUL, 2-step)	89.0	86.8	57.6	79.8	87.8	71.4	72.2	80.4
PTransE (RNN, 2-step)	88.9	84.0	56.3	84.5	88.8	68.4	81.5	86.7
PTrasnE (ADD, 3-step)	90.1	92.0	58.7	**86.1**	90.7	70.7	87.5	**88.7**

tion paths. As compared with TransE, the inferior of PTransE-TransE also indicates that entity representations are informative and crucial for relation prediction.

Table 4: Evaluation results on relation prediction.

Metric	Mean Rank		Hits@1 (%)	
	Raw	Filter	Raw	Filter
TransE (Our)	2.8	2.5	65.1	84.3
+Rev	2.6	2.3	67.1	86.7
+Rev+Path	2.4	1.9	65.2	89.0
PTransE (ADD, 2-step)	**1.7**	**1.2**	69.5	93.6
-TransE	135.8	135.3	51.4	78.0
-Path	2.0	1.6	**69.7**	89.0
PTransE (MUL, 2-step)	2.5	2.0	66.3	89.0
PTransE (RNN, 2-step)	1.9	1.4	68.3	93.2
PTransE (ADD, 3-step)	1.8	1.4	68.5	**94.0**

3.3 Relation Extraction from Text

Relation extraction from text aims to extract relational facts from plain text to enrich existing KBs. Many works regard large-scale KBs as distant supervision to annotate sentences as training instances and build relation classifiers according to features extracted from the sentences (Mintz et al., 2009; Riedel et al., 2010; Hoffmann et al., 2011; Surdeanu et al., 2012). All these methods reason new facts only based on plain text. TransE was used to enrich a text-based model and achieved a significant improvement (Weston et al., 2013), and so do TransH (Wang et al., 2014) and TransR (Lin et al., 2015). In this task, we explore the effectiveness of PTransE for relation extraction from text.

We use New York Times corpus (NYT) released by (Riedel et al., 2010) as training and testing data. NYT aligns Freebase with the articles in New York Times, and extracts sentence-level features such as part-of-speech tags, dependency tree paths for each mention. There are 53 relations (including non-relation denoted as NA) and 121,034 training mentions. We use FB40K as the KB, consisting all entities mentioned in NYT and 1,336 relations.

In the experiments, we implemented the text-based model Sm2r presented in (Weston et al., 2013). We combine the ranking scores from the text-based model with those from KB representations to rank testing triples, and generate precision-recall curves for both TransE and PTransE. For learning of TransE and PTransE, we set the dimensions of entities/relations embeddings $k = 50$, the learning rate $\lambda = 0.001$, the margin $\gamma = 1.0$ and dissimilarity metric as L1. We also compare with MIMLRE (Surdeanu et al., 2012) which is the state-of-art method using distant supervision. The evaluation curves are shown in Figure 3.

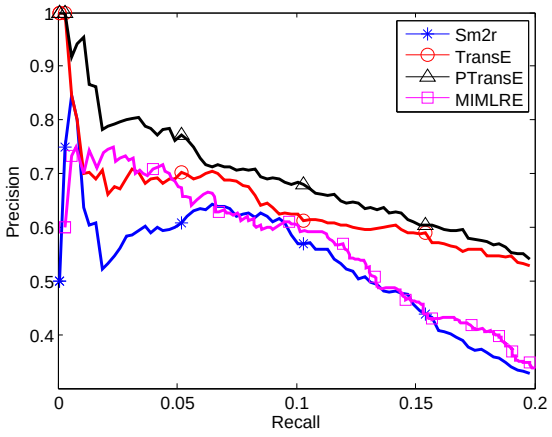


Figure 3: Precision-recall curves of Sm2r, TransE and PTransE combine with Sm2r.

From Figure 3 we can observe that, by combining with the text-based model Sm2r, the precision of PTransE significantly outperforms TransE especially for the top-ranked triples. This indicates that encoding relation paths is also useful for relation extraction from text.

Note that TransE used here does not consider reverse relations and relation paths because the performance does not change much. We analyze the reason as follows. In the task of knowledge base completion, each testing triple has at least one valid relation. In contrast, many testing triples in this task correspond to non-relation (NA), and there are usually several relation paths between two entities in these non-relation triples. TransE does not encode relation paths during the training phase like PTransE, which results in worse performance for predicting non-relation when considering relation paths in the testing phase, and compensates the improvement on those triples that do have relations. This indicates it is non-trivial to encode relation paths, and also confirms the effectiveness of PTransE.

3.4 Case Study of Relation Inference

We have shown that PTransE can achieve high performance for knowledge base completion and relation extraction from text. In this section, we present some examples of relation inference according to relation paths.

According to the learning results of PTransE, we can find new facts from KBs. As shown in Figure 4, two entities *Forrest Gump* and *English* are connected by three relation paths, which give us more confidence to predict the relation between the two entities to `LanguageOfFilm`.

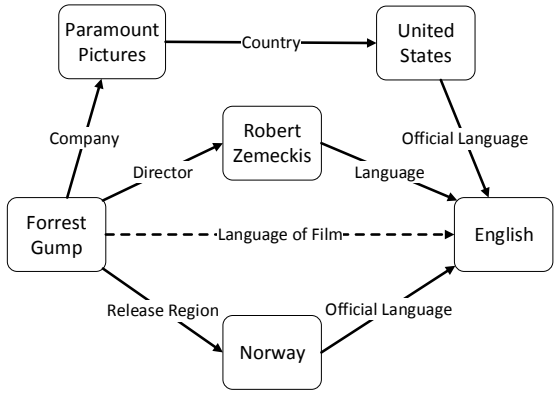


Figure 4: An inference example in Freebase.

4 Related Work

Recent years have witnessed great advances of modeling multi-relational data such as social networks and KBs. Many works cope with relational learning as a multi-relational representation learning problem, encoding both entities and relations in a low-dimensional latent space, based on Bayesian clustering (Kemp et al., 2006; Miller et al., 2009; Sutskever et al., 2009; Zhu, 2012), energy-based models (Bordes et al., 2011; Chen et al., 2013; Socher et al., 2013; Bordes et al., 2013; Bordes et al., 2014), matrix factorization (Singh and Gordon, 2008; Nickel et al., 2011; Nickel et al., 2012) . Among existing representation models, TransE (Bordes et al., 2013) regards a relation as translation between head and tail entities for optimization, which achieves a good trade-off between prediction accuracy and computational efficiency. All existing representation learning methods of knowledge bases only use direct relations between entities, ignoring rich information in relation paths.

Relation paths have already been widely considered in social networks and recommender systems. Most of these works regard each relation and path as discrete symbols, and deal with them using graph-based algorithms, such as random walks with restart (Tong et al., 2006). Relation paths have also been used for inference on large-scale KBs, such as Path Ranking algorithm (PRA) (Lao and Cohen, 2010), which has been adopted for expert finding (Lao and Cohen, 2010) and information retrieval (Lao et al., 2012). PRA has also been used for relation extraction based on KB structure (Lao et al., 2011; Gardner et al., 2013). (Neelakantan et al., 2015) further learns a recurrent neural network (RNN) to represent unseen relation paths according to involved relations. We note that, these methods focus on modeling relation paths for relation extraction without considering any information of entities. In contrast, by successfully integrating the merits of modeling entities and relation paths, PTransE can learn superior representations of both entities and relations for knowledge graph completion and relation extraction as shown in our experiments.

5 Conclusion and Future Work

This paper presents PTransE, a novel representation learning method for KBs, which encodes relation paths to embed both entities and relations

in a low-dimensional space. To take advantages of relation paths, we propose path-constraint resource allocation to measure relation path reliability, and employ semantic composition of relations to represent paths for optimization. We evaluate PTransE on knowledge base completion and relation extraction from text. Experimental results show that PTransE achieves consistent and significant improvements as compared with TransE and other baselines.

In future, we will explore the following research directions: (1) This paper only considers the inference patterns between direct relations and relation paths between two entities for learning. There are much complicated patterns among relations. For example, the inference form $\texttt{Queen}(e) \xRightarrow{\text{Inference}} \texttt{Female}(e)$ cannot be handled by PTransE. We may take advantages of first-order logic to encode these inference patterns for representation learning. (2) There are some extensions for TransE, e.g., TransH and TransR. It is non-trivial for them to adopt the idea of PTransE, and we will explore to extend PTransE to these models to better deal with complicated scenarios of KBs.

6 Acknowledgments

Zhiyuan Liu and Maosong Sun are supported by the 973 Program (No. 2014CB340501) and the National Natural Science Foundation of China (NSFC No. 61133012) and Tsinghua-Samsung Joint Lab. Huanbo Luan is supported by the National Natural Science Foundation of China (NSFC No. 61303075). We sincerely thank Yansong Feng for insightful discussions, and thank all anonymous reviewers for their constructive comments.

References

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of KDD*, pages 1247–1250.

Antoine Bordes, Jason Weston, Ronan Collobert, Yoshua Bengio, et al. 2011. Learning structured embeddings of knowledge bases. In *Proceedings of AAAI*, pages 301–306.

Antoine Bordes, Xavier Glorot, Jason Weston, and Yoshua Bengio. 2012. Joint learning of words and meaning representations for open-text semantic parsing. In *Proceedings of AISTATS*, pages 127–135.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Proceedings of NIPS*, pages 2787–2795.

Antoine Bordes, Xavier Glorot, Jason Weston, and Yoshua Bengio. 2014. A semantic matching energy function for learning with multi-relational data. *Machine Learning*, 94(2):233–259.

Danqi Chen, Richard Socher, Christopher D Manning, and Andrew Y Ng. 2013. Learning new facts from knowledge bases with neural tensor networks and semantic word vectors. *Proceedings of ICLR*.

Matt Gardner, Partha Pratim Talukdar, Bryan Kisiel, and Tom M Mitchell. 2013. Improving learning and inference in a large knowledge-base using latent syntactic cues. In *Proceedings of EMNLP*, pages 833–838.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of ACL-HLT*, pages 541–550.

Rodolphe Jenatton, Nicolas L Roux, Antoine Bordes, and Guillaume R Obozinski. 2012. A latent factor model for highly multi-relational data. In *Proceedings of NIPS*, pages 3167–3175.

Charles Kemp, Joshua B Tenenbaum, Thomas L Griffiths, Takeshi Yamada, and Naonori Ueda. 2006. Learning systems of concepts with an infinite relational model. In *Proceedings of AAAI*, volume 3, page 5.

Ni Lao and William W Cohen. 2010. Relational retrieval using a combination of path-constrained random walks. *Machine learning*, 81(1):53–67.

Ni Lao, Tom Mitchell, and William W Cohen. 2011. Random walk inference and learning in a large scale knowledge base. In *Proceedings of EMNLP*, pages 529–539.

Ni Lao, Amarnag Subramanya, Fernando Pereira, and William W Cohen. 2012. Reading the web with learned syntactic-semantic inference rules. In *Proceedings of EMNLP-CoNLL*, pages 1017–1026.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of AAAI*, pages 2181–2187.

Linyuan Lü and Tao Zhou. 2011. Link prediction in complex networks: A survey. *Physica A: Statistical Mechanics and its Applications*, 390(6):1150–1170.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Proceedings of Interspeech*, pages 1045–1048.

Kurt Miller, Michael I Jordan, and Thomas L Griffiths. 2009. Nonparametric latent feature models for link prediction. In *Proceedings of NIPS*, pages 1276–1284.

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

Jeff Mitchell and Mirella Lapata. 2008. Vector-based models of semantic composition. In *Proceedings of ACL*, pages 236–244.

Arvind Neelakantan, Benjamin Roth, and Andrew McCallum. 2015. Compositional vector space models for knowledge base inference. In *2015 AAAI Spring Symposium Series*.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data. In *Proceedings of ICML*, pages 809–816.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2012. Factorizing yago: scalable machine learning for linked data. In *Proceedings of WWW*, pages 271–280.

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

Ajit P Singh and Geoffrey J Gordon. 2008. Relational learning via collective matrix factorization. In *Proceedings of KDD*, pages 650–658.

Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *Proceedings of NIPS*, pages 926–934.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D Manning. 2012. Multi-instance multi-label learning for relation extraction. In *Proceedings of EMNLP*, pages 455–465.

Ilya Sutskever, Joshua B Tenenbaum, and Ruslan Salakhutdinov. 2009. Modelling relational data using bayesian clustered tensor factorization. In *Proceedings of NIPS*, pages 1821–1828.

Hanghang Tong, Christos Faloutsos, and Jia-Yu Pan. 2006. Fast random walk with restart and its applications. In *Proceedings of ICDM*, pages 613–622.

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

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

Tao Zhou, Jie Ren, Matúš Medo, and Yi-Cheng Zhang. 2007. Bipartite network projection and personal recommendation. *Physical Review E*, 76(4):046115.

Jun Zhu. 2012. Max-margin nonparametric latent feature models for link prediction. *In Proceedings of ICML*, pages 719–726.

Corpus-level Fine-grained Entity Typing Using Contextual Information

Yadollah Yaghoobzadeh and **Hinrich Schütze**
Center for Information and Language Processing
University of Munich, Germany
yadollah@cis.lmu.de

Abstract

This paper addresses the problem of corpus-level entity typing, i.e., inferring from a large corpus that an entity is a member of a class such as "food" or "artist". The application of entity typing we are interested in is knowledge base completion, specifically, to learn which classes an entity is a member of. We propose FIGMENT to tackle this problem. FIGMENT is embedding-based and combines (i) a global model that scores based on aggregated contextual information of an entity and (ii) a context model that first scores the individual occurrences of an entity and then aggregates the scores. In our evaluation, FIGMENT strongly outperforms an approach to entity typing that relies on relations obtained by an open information extraction system.

1 Introduction

Natural language understanding (NLU) is not possible without knowledge about the world – partly so because world knowledge is needed for many NLP tasks that must be addressed as part of NLU; e.g., many coreference ambiguities can only be resolved based on world knowledge. It is also true because most NLU applications combine a variety of information sources that include both text sources and knowledge bases; e.g., question answering systems need access to knowledge bases like gazetteers. Thus, high-quality knowledge bases are critical for successful NLU.

Unfortunately, most knowledge bases are incomplete. The effort required to create knowledge bases is considerable and since the world changes, it will always continue. Knowledge bases are therefore always in need of updates and corrections. To address this problem, we present an information extraction method that can be used for knowledge base completion. In contrast to most other work on knowledge base completion, we focus on fine-grained *classification of entities* as opposed to *relations between entities*.

The goal of knowledge base completion is to acquire knowledge in general as opposed to detailed analysis of an individual context or sentence. Therefore, our approach is *corpus-level*: We infer the types of an entity by considering the set of all of its mentions in the corpus. In contrast, named entity recognition (NER) is *context-level* or *sentence-level*: NER infers the type of an entity in a particular context. As will be discussed in more detail in the following sections, the problems of corpus-level entity typing vs. context/sentence-level entity typing are quite different. This is partly because the objectives of optimizing accuracy on the context-level vs. optimizing accuracy on the corpus-level are different and partly because evaluation measures for corpus-level and context-level entity typing are different.

We define our problem as follows. Let K be a knowledge base that models a set E of entities, a set T of fine-grained classes or *types* and a membership function $m : E \times T \mapsto \{0, 1\}$ such that $m(e, t) = 1$ iff entity e has type t. Let C be a large corpus of text. Then, the problem we address in this paper is *corpus-level entity typing*: For a given pair of entity e and type t determine – based on the evidence available in C – whether e is a member of type t (i.e., $m(e, t) = 1$) or not (i.e., $m(e, t) = 0$) and update the membership relation m of K with this information.

We investigate two approaches to entity typing: a global model and a context model.

The *global model* aggregates all contextual information about an entity e from the corpus and then based on that, makes a classification decision on a particular type t – i.e., $m(e, t) = 0$ vs. $m(e, t) = 1$.

The *context model* first scores each individual

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 715–725,
Lisbon, Portugal, 17-21 September 2015.

context of e as expressing type t or not. A final decision on the value of $m(e,t)$ is then made based on the distribution of context scores. One difficulty in knowledge base completion based on text corpora is that it is too expensive to label large amounts of text for supervised approaches. For our context model, we address this problem using *distant supervision*: we treat *all* contexts of an entity that can have type t as contexts of type t even though this assumption will in general be only true for a *subset* of these contexts. Thus, as is typical for distant supervision, the labels are incorrect in some contexts, but we will show that the labeling is good enough to learn a high-quality context model.

The global model is potentially more robust since it looks at all the available information at once. In contrast, the context model has the advantage that it can correctly predict types for which there are only a small number of reliable contexts. For example, in a large corpus we are likely to find a few reliable contexts indicating that "Barack Obama" is a bestselling author even though this evidence may be obscured in the global distribution because the vast majority of mentions of "Obama" do not occur in author contexts.

We implement the global model and the context model as well as a simple combination of the two and call the resulting system FIGMENT: FIne-Grained eMbedding-based Entity Typing. A key feature of FIGMENT is that it makes extensive use of distributed vector representations or *embeddings*. We compute embeddings for words as is standard in a large body of NLP literature, but we also compute embeddings for *entities* and for *types*. The motivation for using embeddings in these cases is (i) better generalization and (ii) more robustness against noise for text types like web pages. We compare the performance of FIGMENT with an approach based on Open Information Extraction (OpenIE).

The main contributions of this paper can be summarized as follows.

- We address the problem of corpus-level entity typing in a knowledge base completion setting. In contrast to other work that has focused on learning *relations* between entities, we learn *types* of entities.
- We show that context and global models for entity typing provide complementary information and combining them gives the best results.
- We use embeddings for words, entities and types to improve generalization and deal with noisy input.
- We show that our approach outperforms a system based on OpenIE relations when the input corpus consists of noisy web pages.

In the following, we first discuss related work. Then we motivate our approach and define the problem setting we adopt. We then introduce our models in detail and report and analyze experimental results. Finally, we discuss remaining challenges and possible future work and present our conclusions.

2 Related work

Named entity recognition (NER) is the task of detecting and classifying named entities in text. While most NER systems (e.g., Finkel et al. (2005)) only consider a small number of entity classes, recent work has addressed fine-grained NER (Yosef et al., 2012; Ling and Weld, 2012). These methods use a variety of lexical and syntactic features to segment and classify entity mentions. Some more recent work assumes the segmentation is known and only classifies entity mentions. Dong et al. (2015) use distributed representations of words in a hybrid classifier to classify mentions to 20 types. Yogatama et al. (2015) classify mentions to more fine-grained types by using different features for mentions and embedding labels in the same space. These methods as well as standard NER systems try to maximize correct classification of mentions in individual contexts whereas we aggregate individual contexts and evaluate on accuracy of entity-type assignments inferred from the entire corpus. In other words, their evaluation is *sentence-level* whereas ours is *corpus-level*.

Entity set expansion (ESE) is the problem of finding entities in a class (e.g., medications) given a seed set (e.g., {"Ibuprofen", "Maalox", "Prozac"}). The standard solution is pattern-based bootstrapping (Thelen and Riloff, 2002; Gupta and Manning, 2014). ESE is different from the problem we address because ESE starts with a small seed set whereas we assume that a large number of examples from a knowledge base (KB) is available. Initial experiments with the system of Gupta

and Manning (2014) showed that it was not performing well for our task – this is not surprising given that it is designed for a task with properties quite different from entity typing.

More closely related to our work are the OpenIE systems NNPLB (Lin et al., 2012) and PEARL (Nakashole et al., 2013) for *fine-grained typing of unlinkable and emerging entities*. Both systems first extract relation tuples from a corpus and then type entities based on the tuples they occur in (where NNPLB only uses the subject position for typing). To perform typing, NNPLB propagates activation from known members of a class to other entities whereas PEARL assigns types to the argument slots of relations. The main difference to FIGMENT is that we do not rely on relation extraction. In principle, we can make use of any context, not just subject and object positions. FIGMENT also has advantages for noisy text for which relation extraction can be challenging. This will be demonstrated in our evaluation on web text. Finally, our emphasis is on making yes-no decisions about possible types (as opposed to just ranking possibilities) for all entities (as opposed to just emerging or unlinkable entities). Our premise is that even existing entities in KBs are often not completely modeled and have entries that require enhancement. We choose NNPLB as our baseline.

The fine-grained typing of entities performed by FIGMENT can be used for *knowledge base completion (KBC)*. Most KBC systems focus on *relations* between entities, not on *types* as we do. Some generalize the patterns of relationships within the KB (Nickel et al., 2012; Bordes et al., 2013) while others use a combination of within-KB generalization and information extraction from text (Weston et al., 2013; Socher et al., 2013; Jiang et al., 2012; Riedel et al., 2013; Wang et al., 2014). Neelakantan and Chang (2015) address entity typing in a way that is similar to FIGMENT. Their method is based on KB information, more specifically entity descriptions in Wikipedia and Freebase. Thus, in contrast to our approach, their system is not able to type entities that are not covered by existing KBs. We infer classes for entities from a large corpus and do not assume that these entities occur in the KB.

Learning embeddings for words is standard in a large body of NLP literature (see Baroni et al. (2014) for an overview). In addition to words, we also learn *embeddings for entities and types*. Most prior work on entity embeddings (e.g., Weston et al. (2013), Bordes et al. (2013)) and entity and type embeddings (Zhao et al., 2015) has mainly used KB information as opposed to text corpora. Wang et al. (2014) learn embeddings of words and entities in the same space by replacing Wikipedia anchors with their corresponding entities. For our global model, we learn entity embedding in a similar way, but on a corpus with automatically annotated entities. For our context model, we learn and use type embeddings jointly with corpus words to improve generalization, a novel contribution of this paper to the best of our knowledge. We learn all our embeddings using `word2vec` (Mikolov et al., 2013).

Our problem can be formulated as *multi-instance multi-label* (MIML) learning (Zhou and Zhang, 2006), similar to the formulation for relation extraction by Surdeanu et al. (2012). In our problem, each example (entity) can have several instances (contexts) and each instance can have several labels (types). Similar to Zhou and Zhang (2006)'s work on scene classification, we also transform MIML into easier tasks. The global model transforms MIML into a multi-label problem by merging all instances of an example. The context model solves the problem by combining the instance-label scores to example-label scores.

3 Motivation and problem definition

3.1 Freebase

Large scale KBs like Freebase (Bollacker et al., 2008), YAGO (Suchanek et al., 2007) and Google knowledge graph are important NLP resources. Their structure is roughly equivalent to a graph in which entities are nodes and edges are relations between entities. Each node is also associated with one or more semantic classes, called types. These types are the focus of this paper.

We use Freebase, the largest available KB, in this paper. In Freebase, an entity can belong to several classes, e.g., "Barack Obama" is a member of 37 types including "US president" and "author". One *notable type* is also defined for each entity, e.g., "US-president" for "Obama" since it is regarded as his most prominent characteristic and the one that would be used to disambiguate references to him, e.g., to distinguish him from somebody else with the same name.

There are about 1500 types in Freebase, or-

ganized by domain; e.g., the domain "food" has types like "food", "ingredient" and "restaurant". Some types like "location" are very general, some are very fine-grained, e.g., "Vietnamese urban district". There are types that have a large number of instances like "citytown" and types that have very few like "camera_sensor". Entities are defined as instances of types. They can have several types based on the semantic classes that the entity they are referring to is a member of – as in the above example of Barack Obama.

The types are not organized in a strict taxonomy even though there exists an *included type* relationship between types in Freebase. The reason is that for a user-generated KB it is difficult to maintain taxonomic consistency. For example, almost all instances of "author" are also instances of "person", but sometimes organizations author and publish documents. We follow the philosophy of Freebase and assume that the types do not have a hierarchical organization.

3.2 Incompleteness of knowledge base

Even though Freebase is the largest publicly available KB of its kind, it still has significant coverage problems; e.g., 78.5% of persons in Freebase do not have *nationality* (Min et al., 2013).

This is unavoidable, partly because Freebase is user-generated, partly because the world changes and Freebase has to be updated to reflect those changes. All existing KBs that attempt to model a large part of the world suffer from this incompleteness problem. Incompleteness is likely to become an even bigger problem in the future as the number of types covered by KBs like Freebase increases. As more and more fine-grained types are added, achieving good coverage for these new types using only human editors will become impossible.

The approach we adopt in this paper to address incompleteness of KBs is extraction of information from large text corpora. Text can be argued to be the main repository of the type of knowledge represented in KBs, so it is reasonable to attempt completing them based on text. There is in fact a significant body of work on corpus-based methods for extracting knowledge from text; however, most of it has addressed relation extraction, not the acquisition of type information – roughly corresponding to unary relations (see Section 2). In this paper, we focus on typing entities.

3.3 Entity linking

The first step in extracting information about entities from text is to reliably identify mentions of these entities. This problem of *entity linking* has some mutual dependencies with entity typing. Indeed, some recent work shows large improvements when entity typing and linking are jointly modeled (Ling et al., 2015; Durrett and Klein, 2014). However, there are constraints that are important for high-performance entity linking, but that are of little relevance to entity typing. For example, there is a large literature on entity linking that deals with coreference resolution and inter-entity constraints – e.g., "Naples" is more likely to refer to a US (resp. an Italian) city in a context mentioning "Fort Myers" (resp. "Sicily").

Therefore, we will only address entity typing in this paper and consider entity linking as an independent module that provides contexts of entities for FIGMENT. More specifically, we build FIGMENT on top of the output of an existing entity linking system and use FACC1,[1] an automatic Freebase annotation of ClueWeb (Gabrilovich et al., 2013). According to the FACC1 distributors, precision of annotated entities is around 80-85% and recall is around 70-85%.

3.4 FIGER types

Our goal is fine-grained typing of entities, but types like "Vietnamese urban district" are too fine-grained. To create a reliable setup for evaluation and to make sure that all types have a reasonable number of instances, we adopt the FIGER type set (Ling and Weld, 2012) that was created with the same goals in mind. FIGER consists of 112 tags and was created in an attempt to preserve the diversity of Freebase types while consolidating infrequent and unusual types through filtering and merging. For example, the Freebase types "dish", "ingredient", "food" and "cheese" are mapped to one type "food". See (Ling and Weld, 2012) for a complete list of FIGER types. We use "type" to refer to FIGER types in the rest of the paper.

4 Global, context and joint models

We address a problem setting in which the followings are given: a KB with a set of entities E, a set of types T and a membership function $m : E \times T \mapsto \{0, 1\}$ such that $m(e, t) = 1$ iff entity e has type t; and a large annotated corpus C

[1] lemurproject.org/clueweb12/FACC1

in which mentions of E are linked. As mentioned before, we use FACC1 as our corpus.

In this problem setting, we address the task of *corpus-level fine-grained entity typing*: we want to infer from the corpus for each pair of entity e and type t whether $m(e,t) = 1$ holds, i.e., whether entity e is a member of type t.

We use three scoring models in FIGMENT: global model, context model and joint model. The models return a score $S(e,t)$ for an entity-type pair (e,t). $S(e,t)$ is an assessment of the extent to which it is true that the semantic class t contains e and we learn it by training on a subset of E. The trained models can be applied to large corpora and the resulting scores can be used for learning new types of entities covered in the KB as well as for typing new or unknown entities – i.e., entities not covered by the KB. To work for new or unknown entities, we would need an entity linking system such as the ones participating in TAC KBP (McNamee and Dang, 2009) that identifies and clusters mentions of them.

4.1 Global model

The global model (GM) scores possible types of entity e based on a *distributed vector representation* or *embedding* $\vec{v}(e) \in \mathbb{R}^d$ of e. $\vec{v}(e)$ can be learned from the entity-annotated corpus C.

Embeddings of words have been widely used in different NLP applications. The embedding of a word is usually derived from the distribution of its context words. The hypothesis is that words with similar meanings tend to occur in similar contexts (Harris, 1954) and therefore cooccur with similar context words. By extension, the assumption of our model is that entities with similar types tend to cooccur with similar context words.

To learn a score function $S_{\text{GM}}(e,t)$, we use a multilayer perceptron (MLP) with one shared hidden layer and an output layer that contains, for each type t in T, a logistic regression classifier that predicts the probability of t:

$$S_{\text{GM}}(e,t) = G_t\Big(\tanh\left(\mathbf{W}_{\text{input}}\vec{v}(e)\right)\Big)$$

where $\mathbf{W}_{\text{input}} \in \mathbb{R}^{h\times d}$ is the weight matrix from $\vec{v}(e) \in \mathbb{R}^d$ to the hidden layer with size h. G_t is the logistic regression classifier for type t that is applied on the hidden layer. The shared hidden layer is designed to exploit the dependencies among labels. Stochastic gradient descent (SGD) with AdaGrad (Duchi et al., 2011) and minibatches are used to learn the parameters.

4.2 Context model

For the context model (CM), we first learn a scoring function $S_{\text{c2t}}(c,t)$ for individual contexts c in the corpus. $S_{\text{c2t}}(c,t)$ is an assessment of how likely it is that an entity occurring in context c has type t. For example, consider the contexts c_1 = "he served SLOT cooked in wine" and c_2 = "she loves SLOT more than anything". SLOT marks the occurrence of an entity and it also shows that we do not care about the entity mention itself but only its context. For the type t = "food", $S_{\text{c2t}}(c_1,t)$ is high whereas $S_{\text{c2t}}(c_2,t)$ is low. This example demonstrates that some contexts of an entity like "beef" allow specific inferences about its type whereas others do not. We aim to learn a scoring function S_{c2t} that can distinguish these cases.

Based on the context scoring function S_{c2t}, we then compute the corpus-level CM scoring function S_{CM} that takes the scores $S_{\text{c2t}}(c_i,t)$ for all contexts of entity e in the corpus as input and returns a score $S_{\text{CM}}(e,t)$ that assesses the appropriateness of t for e. In other words, S_{CM} is:

$$S_{\text{CM}}(e,t) = g(U_{e,t}) \tag{1}$$

where $U_{e,t} = \{S_{\text{c2t}}(c_1,t), \ldots, S_{\text{c2t}}(c_n,t)\}$ is the set of scores for t based on the n contexts $c_1 \ldots c_n$ of e in the corpus. The function g is a summary function of the distribution of scores, e.g., the mean, median or maximum. We use the mean in this paper.

We now describe how we learn S_{c2t}. For training, we need contexts that are labeled with types. We do not have such a dataset in our problem setting, but we can use the contexts of linked entities as distantly supervised data. Specifically, assume that entity e has n types. For each mention of e in the corpus, we generate a training example with n labels, one for each of the n types of e.

For training S_{c2t}, a context c of a mention is represented as the concatenation of two vectors. One vector is the *average* of the embeddings of the $2l$ words to the left and right of the mention. The other vector is the *concatenation* of the embeddings of the $2k$ words to the left and right of the mention. E.g., for $k = 2$ and $l = 1$ the context c is represented as the vector: $\Phi(c) = \left[x_{-2}, x_{-1}, x_{+1}, x_{+2}, \text{avg}(x_{-1}, x_{+1})\right]$ where $x_i \in \mathbb{R}^d$ is the embedding of the context word at position i relative to the entity in position 0.

We train S_{c2t} on context representations that consist of embeddings because our goal is a robust model that works well on a wide variety of genres, including noisy web pages. If there are other entities in the contexts, we first replace them with their notable type to improve generalization. We learn word and type embeddings from the corpus C by replacing train entities with their notable type.

The next step is to score these examples. We use an MLP similar to the global model to learn S_{c2t}, which predicts the probability of type t occurring in context c:

$$S_{\text{c2t}}(c,t) = G_t\Big(\tanh\left(\mathbf{W}_{\text{input}}\Phi(c)\right)\Big)$$

where $\Phi(c) \in \mathbb{R}^n$ is the feature vector of the context c as described above, $n = (2k+1) * d$ and $\mathbf{W}_{\text{input}} \in \mathbb{R}^{h \times n}$ is the weight matrix from input to hidden layer with h units. Again, we use SGD with AdaGrad and minibatch training.

4.3 Joint model

Global model and context model have complementary strengths and weaknesses.

The strength of CM is that it is a direct model of the only source of reliable evidence we have: the context in which the entity occurs. This is also the way a human would ordinarily do entity typing: she would determine if a specific context in which the entity occurs implies that the entity is, say, an author or a musician and type it accordingly. The order of words is of critical importance for the accurate assessment of a context and CM takes it into account. A well-trained CM will also work for cases for which GM is not applicable. In particular, if the KB contains only a small number of entities of a particular type, but the corpus contains a large number of contexts of these entities, then CM is more likely to generalize well.

The main weakness of CM is that a large proportion of contexts does not contain sufficient information to infer all types of the entity; e.g., based on our distant supervised training data, we label every context of "Obama" with "author", "politician" and Obama's other types in the KB. Thus, CM is trained on a noisy training set that contains only a relatively small number of informative contexts.

The main strength of GM is that it bases its decisions on the entire evidence available in the corpus. This makes it more robust. It is also more efficient to train since its training set is by a factor of $|M|$ smaller than the training set of CM where $|M|$ is the average number of contexts per entity.

The disadvantage of GM is that it does not work well for rare entities since the aggregated representation of an entity may not be reliable if it is based on few contexts. It is also less likely to work well for non-dominant types of an entity which might be swamped by dominant types; e.g., the author contexts of "Obama" may be swamped by the politician contexts and the overall context signature of the entity "Obama" may not contain enough signal to infer that he is an author. Finally, methods for learning embeddings like `word2vec` are bag-of-word approaches. Therefore, word order information – critical for many typing decisions – is lost.

Since GM and CM models are complementary, a combination model should work better. We test this hypothesis for the simplest possible joint model (JM), which adds the scores of the two individual models:

$$S_{\text{JM}}(e,t) = S_{\text{GM}}(e,t) + S_{\text{CM}}(e,t)$$

5 Experimental setup and results

5.1 Setup

Baseline: Our baseline system is the OpenIE system no-noun-phrase-left-behind (NNPLB) by Lin et al. (2012) (see Section 2). Our reimplementation performs on a par with published results.[2] We use NNPBL as an alternative way of computing scores $S(e,t)$. Scores of the four systems we compare – NNPBL, GM, CM, JM – are processed the same way to perform entity typing (see below).

Corpus: We select a subset of about 7.5 million web pages, taken from the first segment of ClueWeb12,[3] from different crawl types: 1 million Twitter links, 120,000 WikiTravel pages and 6.5 million web pages. This corpus is preprocessed by eliminating HTML tags, replacing all numbers with "7" and all web links and email addresses with "HTTP", filtering out sentences with length less than 40 characters, and finally doing a simple tokenization. We merge the text with the FACC1 annotations. The resulting corpus has 4 billion tokens and 950,000 distinct entities. We use the 2014-03-09 Freebase data dump as our KB.

[2] The precision of our implementation on the dataset of three million relation triples distributed by (Lin et al., 2012) is 60.7% compared to 59.8% and 61% for tail and head entities reported by Lin et al. (2012).

[3] `http://lemurproject.org/clueweb12`

Entity datasets: We consider all entities in the corpus whose notable types can be mapped to one of the 112 FIGER types, based on the mapping provided by FIGER. 750,000 such entities form our set of entities. 10 out of 112 FIGER types have no entities in this set.[4]

We run the OpenIE system Reverb (Fader et al., 2011) to extract relation triples of the form <subject, relation, object>. Since NNPLB only considers entities in the subject position, we filter out triples whose subject is not an entity. The size of the remaining set of triples is 4,000,000. For a direct comparison with NNPLB, we divide the 750,000 entities into those that occur in subject position in one of the extracted triples (about 250,000 *subject entities* or SE) and those that do not (about 500,000 *non-subject entities* or NSE). We split SE 50:20:30 into train, dev and test sets. The average and median number of FIGER types of the training entities are 1.8 and 2, respectively. We use NSE to evaluate performance of FIGMENT on entities that do not occur in subject position.[5]

Context sampling: For S_{c2t}, we create train', dev' and test' sets of *contexts* that correspond to train, dev and test sets of *entities*. Because the number of contexts is unbalanced for both entities and types and because we want to accelerate training and testing, we downsample contexts. For the set train', we use the notable type feature of Freebase: For each type t, we take contexts from the mentions of those entities whose notable type is t. Recall, however, that each context is labeled with all types of its entity – see Section 4.2.

Then if the number of contexts for t is larger than a minimum, we sample the contexts based on the number of training entities of t. We set the minimum to 10,000 and constrain the number of samples for each t to 20,000. Also, to reduce the effect of distant supervision, entities with fewer distinct types are preferred in sampling to provide discriminative contexts for their notable types. For test' and dev' sets, we sample 300 and 200 random contexts, respectively, for each entity.

System setup: As the baseline, we apply NNPLB to the 4 million extracted triples. To learn entity embeddings for GM, we run `word2vec` (skipgram, 200 dimensions, window size 5) on a version of the corpus in which entities have been replaced by their Freebase IDs, based on the FACC1 annotation. We then train MLP with number of hidden units h = 200 on the embeddings of training entities until the error on dev entities stops decreasing.

Our reasoning for the unsupervised training setup is that we do not use any information about the types of entities (e.g., no entities annotated by humans with types) when we run an unsupervised algorithm like `word2vec`. In a real-world application of FIGMENT to a new corpus, we would first run `word2vec` on the merger of our corpus and the new corpus, retrain GM on training entities and finally apply it to entities in the new corpus. This scenario is simulated by our setup.

Recall that the input to CM consists of $2k$ unit embeddings and the average of $2l$ unit embeddings where we use the term *unit* to refer to both words and types. We set k to 4 and l to 5. To learn embeddings for units, we first exclude lines containing test entities, and then replace each entity with its notable type. Then, we run `word2vec` (skipgram, 100 dimensions, window size 5) on this new corpus and learn embeddings for words and types.

Using the embeddings as input representations, we train S_{c2t} on train' until error on dev' stops decreasing. We set the number of hidden units to 300. We then apply the trained scoring function S_{c2t} to test' and get the scores $S_{\text{c2t}}(c,t)$ for test' contexts. As explained in Section 4.2, we compute the corpus-level scores $S_{\text{CM}}(e,t)$ for each entity by averaging its context-level scores (see Equation 1).

Ranking evaluation: This evaluation shows how well the models rank types for entities. The ranking is based on the scores $S(e,t)$ produced by the different models and baselines. Similar to the evaluation performed by Lin et al. (2012), we use precision at 1 (P@1) and breakeven point (BEP, Boldrin and Levine (2008)). BEP is F_1 at the point in the ranked list at which precision and recall have the same value.

Classification evaluation: This evaluation demonstrates the quality of the thresholded assignment decisions produced by the models. These measures more directly express how well FIGMENT would succeed in enhancing the KB with new information since for each pair (e,t), we have to make a binary decision about whether to put it in the KB or not. We compare our decisions with the gold KB information.

[4]The reason is that the FIGER mapping uses Freebase user-created classes. The 10 missing types are not the notable type of any entity in Freebase.

[5]The entity datasets are available at `http://cistern.cis.lmu.de/figment`

	all types	head types	tail types
NNPLB	.092	.246	.066
CM	.406	.662	.268
GM	.533	.725	.387
JM	**.545**	**.734**	**.407**

Table 2: Type macro average F_1 for all, head and tail types

2k	0	2	4	6	8	10	12	14
h	50	100	200	250	300	400	450	450
micro	.576	.613	.672	.673	.668	.674	**.680**	.674
P@1	.663	.685	.687	.718	.694	**.744**	.722	.742

Table 3: Effect of the context size $2k$ in CM (2k: context size, h: number of hidden units in MLP)

Our evaluation measures are (i) accuracy: an entity is correct if all its types and no incorrect types are assigned to it; (ii) micro average: F_1 of all type-entity assignment decisions; (iii) entity macro average F_1: F_1 of types assigned to an entity, averaged over entities; (iv) type macro average F_1: F_1 of entities assigned to a type, averaged over types.

The assignment decision is made based on thresholds, one per type, for each $S(e, t)$. We select the threshold that maximizes F_1 of entities assigned to the type on dev.

5.2 Results

Table 1 presents results for the ranking evaluation as well as for the first three measures of the classification evaluation. MFT is the most frequent type baseline that ranks types according to their frequency in train. We also show the results for head entities (frequency higher than 100) and tail entities (frequency less than 5). The performance of the systems is in this order: JM $>$ GM $>$ CM $>$ NNPLB $>$ MFT.

Table 2 shows the results of the fourth classification measure, type macro average F_1, for all, head (more than 3000 train entities, 11 types), and tail (less than 200 train entities, 36 types) types. The ordering of models for Table 2 is in line with Table 1: JM $>$ GM $>$ CM $>$ NNPLB $>$ MFT.

We can easily run FIGMENT for **non-subject entities** (NSE) exactly the same way we have run it for subject entities. We test our JM on the 67,000 NSE entities with a frequency of more than 10. The top ranked type returned for 73.5% of entities was correct. Thus, due to our ability to deal with NSE, we can type an additional 50,000 entities correctly.

6 Analysis

Effect of window size in CM: Table 3 explores the effect of using different context sizes. Recall that CM was trained with $2k$ = 8 for the concatenation and $2l$ = 10 for the average window size. We change $2k$ from 0 to 14 while keeping $2l$ = 10. The number of hidden units used in each model is also reported. The table shows that CM can leverage larger context sizes well.

Poor results of NNPLB: NNPLB is mostly hampered by Reverb, which did not work well on the noisy web corpus. As a result, the quality of the extracted relations – which NNPLB entity typing is based on – is too low for reliable typing decisions. The good results of NNPLB on their non-noisy published relation triples confirm that. On the three million relation triples, when mapping Freebase types to FIGER, P@1 of NNPLB is .684; when limiting entities to those with more than 10 relations, the results improve to .776.

GM performs better than CM and JM performs best: The fact that GM outperforms CM shows that decisions based on one global vector of an entity work better than aggregating multiple weak decisions on their contexts. That is clearest for tail entities – where one bad context can highly influence the final decision – and for tail types, which CM was not able to distinguish from other similar types. However, the good results of the simple JM confirm that the score distributions in CM do help. As an example, consider one of the test entities that is an "author". GM and CM wrongly predict "written_work" and "artist", respectively, but JM correctly outputs "author".

Errors of CM: Many CM errors are caused by its simple input representation: it has to learn all linguistic abstractions that it wants to rely on from the training set. One manifestation of this problem is that CM confuses the types "food" and "restaurant". There are only few linguistic contexts in which entities of these types can be exchanged for each other. On the other hand, the context words they cooccur with in a bag-of-words (BOW) sense are very similar. Thus, this indicates that CM pays too much attention to BOW information and that its representation of contexts is limited in terms of generalization.

	all entities					head entities					tail entities				
	P@1	BEP	acc	mic	mac	P@1	BEP	acc	mic	mac	P@1	BEP	acc	mic	mac
MFT	.101	.406	-	-	-	.111	.410	-	-	-	.097	.394	-	-	-
NNPLB	.365	.480	.000	.099	.096	.378	.503	.000	.114	.109	.368	.474	.000	.086	.084
CM	.694	.734	.299	.668	.635	.713	.751	.385	.738	.702	.608	.661	.118	.487	.452
GM	.805	.856	.426	.733	.688	.869	.899	.489	.796	.769	.665	.757	.299	.578	.510
JM	**.816**	**.860**	**.435**	**.743**	**.699**	**.874**	**.900**	**.500**	**.803**	**.776**	**.688**	**.764**	**.306**	**.601**	**.532**

Table 1: Ranking and classification results for SE entities. P@1 and BEP are ranking measures. Accuracy (acc), micro (mic) and macro (mac) are classification measures.

Assumptions that result in errors: The performance of all models suffers from a number of assumptions we made in our training / evaluation setup that are only approximately true.

The first assumption is that FACC1 is correct. But it has a precision of only 80-85% and this caused many errors. An example is the lunar crater "Buffon" in Freebase, a "location". Its predicted type is "athlete" because some FACC1 annotations of the crater link it to the Italian goalkeeper.

The second assumption of our evaluation setup is the completeness of Freebase. There are about 2,600 entities with the single type "person" in SE test. For 62% of the errors on this subset, the top predicted type is a subtype of person: "author", "artist" etc. We manually typed a random subset of 50 and found that the predicted type is actually correct for 44 of these entities.

The last assumption is the mapping from Freebase to FIGER. Some common Freebase types like "award-winner" are not mapped. This negatively affects evaluation measures for many entities. On the other hand, the resulting types do not have a balanced number of instances. Based on our training entities, 11 types (e.g., "law") have less than 50 instances while 26 types (e.g., "software") have more than 1000 instances. Even sampling the contexts could not resolve this problem and this led to low performance on tail types.

7 Future work

The performance of FIGMENT is poor for tail types and entities. We plan to address this in the future (i) by running FIGMENT on larger corpora, (ii) by refining the FIGER type set to cover more Freebase entities, (iii) by exploiting a hierarchy over types and (iv) by exploring more complex input representations of the context for the CM.

FIGMENT's context model can in principle be based on any system that provides entity-type assessment scores for individual contexts. Thus, as an alternative to our scoring model $S_{\text{c2t}}(c,t)$, we could use sentence-level entity classification systems such as FIGER (Ling and Weld, 2012) and (Yogatama et al., 2015)'s system. These systems are based on linguistic features different from the input representation we use, so a comparison with our embedding-based approach is interesting. Our assumption is that FIGMENT is more robust against noise, but investigation is needed.

The components of the version of FIGMENT we presented, in particular, context model and global model, do not use features derived from the mention of an entity. Our assumption was that such features are less useful for fine-grained entity typing. However, there are clearly some types for which mention-based features are useful (e.g., medications or organizations referred to by abbreviations), so we will investigate the usefulness of such features in the future.

8 Conclusion

We presented FIGMENT, a corpus-level system that uses contextual information for entity typing. We designed two scoring models for pairs of entities and types: a global model that scores based on aggregated context information and a context model that aggregates the scores of individual contexts. We used embeddings of words, entities and types to represent contextual information. Our experimental results show that global model and context model provide complementary information for entity typing. We demonstrated that, comparing with an OpenIE-based system, FIGMENT performs well on noisy web pages.

Acknowledgements. Thanks to the anonymous reviewers for their valuable comments. This work was supported by Deutsche Forschungsgemeinschaft (grant DFG SCHU 2246/8-2, SPP 1335).

References

Marco Baroni, Georgiana Dinu, and Germán Kruszewski. 2014. Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014*, pages 238–247.

Michele Boldrin and David K. Levine. 2008. *Against intellectual monopoly*. Cambridge University Press Cambridge.

Kurt D. Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2008, Vancouver, BC, Canada, June 10-12, 2008*, pages 1247–1250.

Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Irreflexive and hierarchical relations as translations. *CoRR*, abs/1304.7158.

Li Dong, Furu Wei, Hong Sun, Ming Zhou, and Ke Xu. 2015. A hybrid neural model for type classification of entity mentions. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 1243–1249.

John C. Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159.

Greg Durrett and Dan Klein. 2014. A joint model for entity analysis: Coreference, typing, and linking. *TACL*, 2:477–490.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, EMNLP 2011, 27-31 July 2011, John McIntyre Conference Centre, Edinburgh, UK, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1535–1545.

Jenny Rose Finkel, Trond Grenager, and Christopher D. Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *ACL 2005, 43rd Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, 25-30 June 2005, University of Michigan, USA*.

Evgeniy Gabrilovich, Michael Ringgaard, and Amarnag Subramanya. 2013. Facc1: Freebase annotation of clueweb corpora.

Sonal Gupta and Christopher D. Manning. 2014. Improved pattern learning for bootstrapped entity extraction. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning, CoNLL 2014, Baltimore, Maryland, USA, June 26-27, 2014*, pages 98–108.

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

Xueyan Jiang, Volker Tresp, Yi Huang, and Maximilian Nickel. 2012. Link prediction in multi-relational graphs using additive models. In *Proceedings of the International Workshop on Semantic Technologies meet Recommender Systems & Big Data, Boston, USA, November 11, 2012*, pages 1–12.

Thomas Lin, Mausam, and Oren Etzioni. 2012. No noun phrase left behind: Detecting and typing unlinkable entities. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, EMNLP-CoNLL 2012, July 12-14, 2012, Jeju Island, Korea*, pages 893–903.

Xiao Ling and Daniel S. Weld. 2012. Fine-grained entity recognition. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, July 22-26, 2012, Toronto, Ontario, Canada.*

Xiao Ling, Sameer Singh, and Daniel S. Weld. 2015. Design challenges for entity linking. *TACL*, 3:315–328.

Paul McNamee and Hoa Trang Dang. 2009. Overview of the tac 2009 knowledge base population track. In *Text Analysis Conference (TAC)*, volume 17, pages 111–113.

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

Bonan Min, Ralph Grishman, Li Wan, Chang Wang, and David Gondek. 2013. Distant supervision for relation extraction with an incomplete knowledge base. In *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, June 9-14, 2013, Westin Peachtree Plaza Hotel, Atlanta, Georgia, USA*, pages 777–782.

Ndapandula Nakashole, Tomasz Tylenda, and Gerhard Weikum. 2013. Fine-grained semantic typing of emerging entities. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, ACL 2013, 4-9 August 2013, Sofia, Bulgaria, Volume 1: Long Papers*, pages 1488–1497.

Arvind Neelakantan and Ming-Wei Chang. 2015. Inferring missing entity type instances for knowledge base completion: New dataset and methods. In *NAACL HLT 2015, The 2015 Conference of the North American Chapter of the Association for*

Computational Linguistics: Human Language Technologies, Denver, Colorado, USA, May 31 - June 5, 2015, pages 515–525.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2012. Factorizing YAGO: scalable machine learning for linked data. In *World Wide Web Conference*, pages 271–280.

Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M. Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, June 9-14, 2013, Westin Peachtree Plaza Hotel, Atlanta, Georgia, USA*, pages 74–84.

Richard Socher, Danqi Chen, Christopher D. Manning, and Andrew Y. Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States.*, pages 926–934.

Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *Proceedings of the 16th International Conference on World Wide Web, WWW 2007, Banff, Alberta, Canada, May 8-12, 2007*, pages 697–706.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D. Manning. 2012. Multi-instance multi-label learning for relation extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, EMNLP-CoNLL 2012, July 12-14, 2012, Jeju Island, Korea*, pages 455–465.

Michael Thelen and Ellen Riloff. 2002. A bootstrapping method for learning semantic lexicons using extraction pattern contexts. In *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing, EMNLP 2002, Stroudsburg, PA, USA*, pages 214–221.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph and text jointly embedding. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1591–1601.

Jason Weston, Antoine Bordes, Oksana Yakhnenko, and Nicolas Usunier. 2013. Connecting language and knowledge bases with embedding models for relation extraction. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1366–1371.

Dani Yogatama, Daniel Gillick, and Nevena Lazic. 2015. Embedding methods for fine grained entity type classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 2: Short Papers*, pages 291–296.

Mohamed Amir Yosef, Sandro Bauer, Johannes Hoffart, Marc Spaniol, and Gerhard Weikum. 2012. HYENA: hierarchical type classification for entity names. In *COLING 2012, 24th International Conference on Computational Linguistics, Proceedings of the Conference: Posters, 8-15 December 2012, Mumbai, India*, pages 1361–1370.

Yu Zhao, Zhiyuan Liu, and Maosong Sun. 2015. Representation learning for measuring entity relatedness with rich information. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 1412–1418.

Zhi-Hua Zhou and Min-Ling Zhang. 2006. Multi-instance multi-label learning with application to scene classification. In *Advances in Neural Information Processing Systems 19, Proceedings of the Twentieth Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 4-7, 2006*, pages 1609–1616.

Knowledge Base Unification via Sense Embeddings and Disambiguation

Claudio Delli Bovi
Department of Computer Science
Sapienza University of Rome
dellibovi@di.uniroma1.it

Luis Espinosa-Anke
Department of Information and Communication Technologies
Universitat Pompeu Fabra
luis.espinosa@upf.edu

Roberto Navigli
Department of Computer Science
Sapienza University of Rome
navigli@di.uniroma1.it

Abstract

We present KB-UNIFY, a novel approach for integrating the output of different Open Information Extraction systems into a single unified and fully disambiguated knowledge repository. KB-UNIFY consists of three main steps: (1) disambiguation of relation argument pairs via a sense-based vector representation and a large unified sense inventory; (2) ranking of semantic relations according to their degree of specificity; (3) cross-resource relation alignment and merging based on the semantic similarity of domains and ranges. We tested KB-UNIFY on a set of four heterogeneous knowledge bases, obtaining high-quality results. We discuss and provide evaluations at each stage, and release output and evaluation data for the use and scrutiny of the community[1].

1 Introduction

The breakthrough of the Open Information Extraction (OIE) paradigm opened up a research area where Web-scale unconstrained Information Extraction systems are developed to acquire and formalize large quantities of knowledge. However, while successful, to date most state-of-the-art OIE systems have been developed with their own type inventories, and no portable ontological structure. In fact, OIE systems can be very different in nature. Early approaches (Etzioni et al., 2008; Wu and Weld, 2010; Fader et al., 2011) focused on extracting a large number of relations from massive unstructured corpora, mostly relying on dependencies at the level of surface text. Systems like NELL (Carlson et al., 2010) combine a hand-crafted taxonomy of entities and relations with self-supervised large-scale extraction from the Web, but they require additional processing for linking and integration (Dutta et al., 2014).

More recent work has focused, instead, on deeper language understanding, especially at the level of syntax and semantics (Nakashole et al., 2012; Moro and Navigli, 2013). By leveraging semantic analysis, knowledge gathered from unstructured text can be adequately integrated and used to enrich existing knowledge bases, such as YAGO (Mahdisoltani et al., 2015), FREEBASE (Bollacker et al., 2008) and DBPEDIA (Lehmann et al., 2014). A large amount of reliable structured knowledge is crucial for OIE approaches based on distant supervision (Mintz et al., 2009; Riedel et al., 2010), even when multi-instance multi-learning algorithms (Surdeanu et al., 2012) or matrix factorization techniques (Riedel et al., 2013; Fan et al., 2014) come into play to deal with noisy extractions. For this reason a recent trend of research has focused on Knowledge Base (KB) completion (Nickel et al., 2012; Bordes et al., 2013), exploiting the fact that distantly supervised OIE and structured knowledge can complement each other. However, the majority of integration approaches nowadays are not designed to deal with many different resources at the same time.

We propose an approach where the key idea is to bring together knowledge drawn from an arbitrary number of OIE systems, regardless of whether these systems provide links to some general-purpose inventory, come with their own ad-hoc structure, or have no structure at all. Knowledge from each source, in the form of ⟨subject, predicate, object⟩ triples, is disambiguated and linked to a single large sense inventory. This enables us to discover alignments at a semantic level between relations from different KBs, and to generate a unified, fully disambiguated KB of entities and semantic relations. KB-UNIFY achieves state-of-the-art disambiguation and provides a general, resource-independent representation of semantic relations, suitable for any kind of KB.

[1] http://lcl.uniroma1.it/kb-unify

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 726–736,
Lisbon, Portugal, 17-21 September 2015.

The remainder of this paper is structured as follows: Section 2 reviews relevant related work; Sections 3, 4, 5 and 6 describe in detail each stage of the approach; Sections 7 and 8 describe the experiments carried out and the results obtained; and finally Section 9 summarizes our findings and discusses potential directions for future work.

2 Related Work

The integration of knowledge drawn from different sources has received much attention over the last decade. Among the most notable examples are resources like BabelNet (Navigli and Ponzetto, 2012), UBY (Gurevych et al., 2012) and YAGO (Mahdisoltani et al., 2015). While great effort has been put into aligning knowledge at the *concept level*, most approaches do not tackle the problem of integrating heterogeneous knowledge at the *relation level*, nor do they exploit effectively the huge amount of information harvested with OIE systems, even when this information is unambiguously linked to a structured resource, as in (Nakashole et al., 2012), or (Moro and Navigli, 2013). In fact, as the number of resources increases, KB alignment is already becoming an emergent research field: Dutta et al. (2014) describe a method for linking arguments in NELL triples to DBPEDIA by combining First Order Logic and Markov Networks; Grycner and Weikum (2014) semantify PATTY's pattern synsets and connect them to WordNet verbs; Lin et al. (2012) propose a method to propagate FREEBASE types across REVERB and deal with the problem of unlinkable entities. All these approaches achieve very competitive results in their respective settings, but unlike the approach being proposed here, they limit the task to 1-to-1 alignments. A few contributions have tried to broaden the scope and include different resources at the same time, but with rather different goals from ours. For example, Riedel et al. (2013) propose a universal schema that integrates structured data with OIE data by learning latent feature vectors for entities and relations; the KNOWLEDGE VAULT (Dong et al., 2014) uses a graph-based probabilistic framework where prior knowledge from existing resources (e.g. FREEBASE) improves Web extractions by predicting their reliability. However, in both cases the main objective is distantly supervised extraction from unstructured text, rather than KB unification. A recent trend of research focuses on learning embedding models for structured knowledge and their application to tasks like relation extraction and KB completion (Socher et al., 2013; Weston et al., 2013; Bordes et al., 2013). These approaches, however, leverage embeddings at surface level, which are suboptimal for our task, as will be discussed in Section 3. Since we require a common semantic framework for KB unification, we use vector representations based on word senses, which are mapped to a very large sense inventory. This shared sense inventory, then, constitutes the common ground in which disambiguation, alignment and final unification occurs.

3 Knowledge Base Unification: Overview

KB-UNIFY takes as input a set of KBs $\mathbf{K} = \{KB_1, ..., KB_n\}$ and outputs a single, unified and fully disambiguated KB, denoted as $\mathbf{KB}^*$. For our purposes we can define a KB KB_i as a triple $\langle E_i, R_i, T_i \rangle$, where E_i is a set of entities, R_i is a set of semantic relations, and T_i is a set of triples (facts) $\langle e_d, r, e_g \rangle$ with subject and object e_d, $e_g \in E_i$ and predicate $r \in R_i$. Depending on the nature of each KB_i, entities in E_i might be disambiguated and linked to an external inventory (e.g. the entity *Washington* linked to the Wikipedia page GEORGE WASHINGTON), or unlinked and only available as ambiguous mentions (e.g. the bare word *washington* might refer to the president, the city or the state). We can thus partition $\mathbf{K}$ into a subset of linked resources $\mathbf{K}_D$, and one of unlinked resources $\mathbf{K}_U$. In order to align very different and heterogeneous KBs at the semantic level, KB-UNIFY exploits:

- A unified sense inventory S, which acts as a superset for the inventories of individual KBs. We choose BabelNet (Navigli and Ponzetto, 2012) for this purpose: by merging complementary knowledge from different resources (e.g. Wikipedia, WordNet, Wikidata and Wiktionary, among others), BabelNet provides a wide coverage of entities and concepts whilst at the same time enabling convenient inter-resource mappings for KB_i in $\mathbf{K}_D$. For instance, each Wikipedia page (or Wikidata item) has a corresponding synset in BabelNet, which enables a one-to-one mapping between BabelNet's synsets and entries in, e.g., DBPEDIA or FREEBASE;

- A vector space model V_S that enables a semantic representation for every item in S. Current distributional models, like word em-

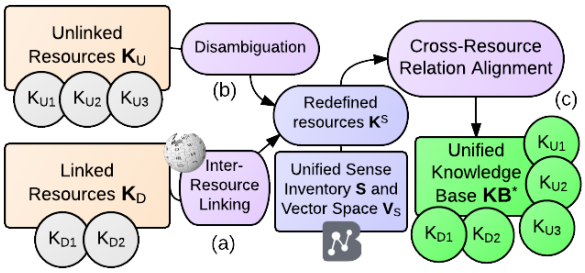


Figure 1: Unification algorithm workflow

beddings (Mikolov et al., 2013), are not suitable to our setting: they are constrained to surface word forms, and hence they inherently retain ambiguity of polysemous words and entity mentions. We thus leverage SENSEMBED (Iacobacci et al., 2015), a novel semantically-enhanced approach to embeddings. SENSEMBED is trained on a large annotated corpus and produces continuous representations for individual word senses (*sense embeddings*), according to an underlying sense inventory.

Figure 1 illustrates the workflow of our KB unification approach. Entities coming from any $KB_i \in \mathbf{K}_D$ can be directly (and unambiguously) mapped to the corresponding entries in S via BabelNet inter-resource linking (Figure 1(a)): in the above example, the entity *Washington* linked to the Wikipedia page GEORGE WASHINGTON is included in the BabelNet synset $\texttt{Washington}^4_{bn}$. In contrast, unlinked (and potentially ambiguous) entities need an explicit disambiguation step (Figure 1(b)) connecting them to appropriate entries, i.e. synsets, in S: this is the case, in the above example, for the ambiguous mention *washington* that has to be linked to either the president, the city or the state. Therefore, our approach comprises two successive stages:

- A **disambiguation** stage (Section 5) where all $KB_i \in \mathbf{K}$ are linked to S, either by inter-resource mapping (Figure 1(a)) or disambiguation (Figure 1(b)), and all E_i are merged into a unified set of entities E^*. As a result of this process we obtain a set $\mathbf{K}^S$ comprising all the KBs in $\mathbf{K}$ redefined using the common sense inventory S;
- An **alignment** stage (Section 6, Figure 1(c)) where, for each pair of KBs $KB_i^S, KB_j^S \in \mathbf{K}^S$, we compare any relation pair $\langle r_i, r_j \rangle$, $r_i \in R_i^S$ and $r_j \in R_j^S$, in order to identify cross-resource alignments and merge relations sharing equivalent semantics into relation clusters (*relation synsets*). This process yields a unified set of relation synsets R^*. The overall result is $\mathbf{KB}^* = \langle E^*, R^*, T^* \rangle$, where T^* is the set of all disambiguated triples redefined over E^* and R^*.

4 Background

The disambiguation stage of our approach is based on the interplay between two core components: a vector space model V_S, as introduced in Section 3, which provides an unambiguous semantic representation for each item in S; and a Word Sense Disambiguation/Entity Linking system, working on the same sense inventory S, which discovers and disambiguates concepts and entity mentions within a given input text. In this section we briefly describe our choice for these two components: SENSEMBED (Iacobacci et al., 2015) and BABELFY (Moro et al., 2014).

SENSEMBED is a knowledge-based approach for obtaining latent continuous representations of individual word senses. Unlike other sense-based embeddings approaches, like (Huang et al., 2012), which address the inherent polysemy of word-level representations relying solely on text corpora, SENSEMBED exploits the structured knowledge of a large sense inventory along with the distributional information gathered from text corpora. In order to do this, SENSEMBED requires a sense-annotated corpus; for each target word sense, then, a representation is computed by maximizing the log likelihood of the word sense with respect to its context within the annotated text, similarly to the word-based embeddings model. Following Iacobacci et al. (2015), we trained SENSEMBED using the English Wikipedia and, as sense inventory, BabelNet.

BABELFY[2] is a joint state-of-the-art approach to multilingual Entity Linking and Word Sense Disambiguation. Given the BabelNet lexicalized semantic network as underlying structure, BABELFY first models each concept in the network through its corresponding *semantic signature* by leveraging a graph random walk algorithm. Then, given an input text, the generated semantic signatures are used to construct a subgraph

[2] http://babelfy.org

of the semantic network representing the meaning of the content words in that text. BABELFY then searches this subgraph for the intended sense of each content word, by means of a densest-subgraph heuristic that identifies high-coherence interpretations. Given its unified approach that covers concepts and named entities alike, and its flexibility in disambiguating both bag-of-words and proper text, BABELFY constitutes the most convenient choice for linking relation triples to a high-coverage sense inventory like BabelNet.

5 Disambiguation

In the disambiguation phase (Figure 1(b)), all $KB_i \in \mathbf{K}_U$ are linked to the unified sense inventory S and added to the set of redefined KBs $\mathbf{K}^S$. As explained in Section 3, while each KB in $\mathbf{K}_D$ can be unambiguously redefined via BabelNet inter-resource links and added to $\mathbf{K}^S$, KBs in $\mathbf{K}_U$ require an explicit disambiguation step. Given $KB_i \in \mathbf{K}_U$, our disambiguation module (Figure 2) takes as input its set of unlinked triples T_i and outputs a set $T_i^S \subseteq T_i$ of disambiguated triples with subject-object pairs linked to S. The triples in T_i^S, together with their corresponding entity sets and relation sets, constitute the redefined KB_i^S which is then added to $\mathbf{K}^S$. However, applying a straightforward approach that disambiguates all triples in isolation might lead to very imprecise results, due to the lack of available context for each individual triple. We thus devised a disambiguation strategy that comprises three successive steps:

1. We identify a set of high-confidence seeds from T_i (Section 5.1), i.e. triples $\langle e_d, r, e_g \rangle$ where subject e_d and object e_g are highly semantically related, and disambiguate them using the senses that maximize their similarity in our vector space V_S;

2. We use the seeds to generate a ranking of the relations in R_i according to their degree of specificity (Section 5.2). We represent each $r \in R_i$ in our vector space V_S and assign higher specificity to relations whose arguments are closer in V_S;

3. We finally disambiguate the remaining non-seed triples in T_i (Section 5.3) starting from the most specific relations, and jointly using all participating argument pairs as context.

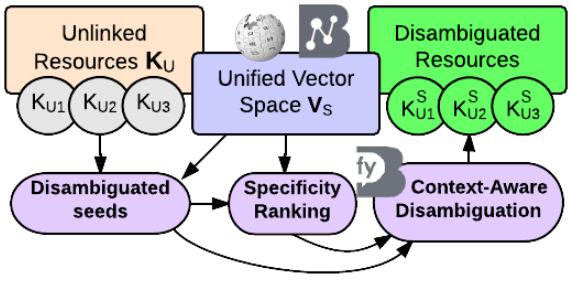


Figure 2: Disambiguation algorithm workflow

5.1 Identifying Seed Argument Pairs

The first stage of our disambiguation approach aims at extracting reliable seeds from T_i, i.e. triples $\langle e_d, r, e_g \rangle$ where subject e_d and object e_g can be confidently disambiguated without additional context. In order to do this we leverage the sense embeddings associated with each candidate disambiguation for e_d and e_g. We consider all the available senses for both e_d and e_g in S, namely $\mathbf{s}_d = \{s_d^1, ..., s_d^m\}$ and $\mathbf{s}_g = \{s_g^1, ..., s_g^{m'}\}$, and the corresponding sets of sense embeddings $\mathbf{v}_d = \{v_d^1, ..., v_d^m\}$ and $\mathbf{v}_g = \{v_g^1, ..., v_g^{m'}\}$. We then select, among all possible pairs of senses, the pair $\langle s_d^*, s_g^* \rangle$ that maximizes the cosine similarity between the corresponding embeddings $\langle v_d^*, v_g^* \rangle$:

$$\langle v_d^*, v_g^* \rangle = \text{argmax}_{v_d \in \mathbf{v}_d,\, v_g \in \mathbf{v}_g} \frac{v_d \cdot v_g}{\|v_d\| \, \|v_g\|} \quad (1)$$

For each disambiguated triple $\langle s_d^*, r, s_g^* \rangle$, the cosine similarity value associated with $\langle v_d^*, v_g^* \rangle$ represents the disambiguation confidence ζ_{dis}. We rank all such triples according to their confidence, and select those above a given threshold δ_{dis}. The underlying assumption is that, for high-confidence subject-object pairs, the embeddings associated with the correct senses s_d^* and s_g^* will be closest in V_S compared to any other candidate pair. Intuitively, the more the relation r between e_d and e_g is semantically well defined, the more this assumption is justified. As an example, consider the triple $\langle$*Armstrong*, *worked for*, *NASA*$\rangle$: among all the possible senses for *Armstrong* (the astronaut, the jazz musician the cyclist, etc.) and *NASA* (the space agency, the racing organization, a Swedish band, etc.) we expect the vectors corresponding to the astronaut and the space agency to be closest in the vector space model V_S.

5.2 Relation Specificity Ranking

The assumption that, given an ambiguous subject-object pair, correct argument senses are

the closest pair in the vector space (Section 5.1) is easily verifiable for general relations (e.g. *is a*, *is part of*). However, as a semantic relation becomes specific, its arguments are less guaranteed to be semantically related (e.g. *is a professor in the university of*) and a disambiguation approach based exclusively on similarity is prone to errors. On the other hand, specific relations tend to narrow down the scope of possible entity types occurring as subject and object. In the above example, *is a professor in the university of* requires entity pairs with professors as subjects and cities as objects. Our disambiguation strategy should thus vary according to the specificity of the relations taken into account. In order to consider this observation in our disambiguation pipeline, we first need to estimate the degree of specificity for each relation in the relation set R_i of the target KB to be disambiguated. Given R_i and a set of seeds from the previous stage (Section 5.1), we apply a specificity ranking policy and sort relations in R_i from the most general to the most specific. We compute the generality $Gen(r)$ of a given relation r by looking at the spatial dispersion of the sense embeddings associated with its seed subjects and objects. Let $\mathbf{v}_D$ ($\mathbf{v}_G$) be the set of sense embeddings associated with the domain (range) seed arguments of r. For both $\mathbf{v}_D$ and $\mathbf{v}_G$, we compute the corresponding centroid vectors μ_D and μ_G as:

$$\mu_k = \frac{1}{|\mathbf{v}_k|} \sum_{v \in \mathbf{v}_k} \frac{v}{\|v\|}, \quad k \in \{D, G\} \qquad (2)$$

Then, the variances σ_D^2 and σ_G^2 are given by:

$$\sigma_k^2 = \frac{1}{|\mathbf{v}_k|} \sum_{v \in \mathbf{v}_k} (1 - cos\,(v, \mu_k))^2 \qquad (3)$$

with $k \in \{D, G\}$ as before. We finally compute $Gen(r)$ as the average of σ_D^2 and σ_G^2. The result of this procedure is a *relation specificity ranking* that associates each relation r with its generality $Gen(r)$. Intuitively, we expect more general relations to show higher variance (hence higher $Gen(r)$), as their subjects and objects are likely to be rather disperse throughout the vector space; instead, arguments of very specific relations are more likely to be clustered together in compact regions, yielding lower values of $Gen(r)$.

5.3 Disambiguation with Relation Context

In the third step, both the specificity ranking and the seeds are exploited to disambiguate the remaining triples in T_i. To do this we leverage BABELFY (Moro et al., 2014) (introduced in Section 4). As we observed in Section 5.2, specific relations impose constraints on their subject-object types and tend to show compact domains and ranges in the vector space. Therefore, given a triple $\langle e_d, r, e_g \rangle$, knowing that r is specific enables us to put together all the triples in T_i where r occurs, and use them to provide meaningful context for disambiguation. If r is general, instead, its subject-object types are less constrained and additional triples do not guarantee to provide semantically related context.

At this stage, our algorithm takes as input the set of triples T_i, along with the associated disambiguation seeds (Section 5.1), the specificity ranking (Section 5.2) and a specificity threshold δ_{spec}. T_i is first partitioned into two subsets: T_i^{spec}, comprising all the triples for which $Gen(r) < \delta_{spec}$, and $T_i^{gen} = T_i \setminus T_i^{spec}$. We then employ two different disambiguation strategies:

- For each distinct relation r occurring in T_i^{spec}, we first retrieve the subset $T_{i,r}^{spec} \subset T_i^{spec}$ of triples where r occurs, and then disambiguate $T_{i,r}^{spec}$ as a whole with BABELFY. For each triple in $T_{i,r}^{spec}$, context is provided by all the remaining triples along with the disambiguated seeds extracted for r.

- We disambiguate the remaining triples in T_i^{gen} one by one in isolation with BABELFY, providing for each triple only the predicate string r as additional context.

6 Cross-Resource Relation Alignment

After disambiguation (Section 5) each KB in $\mathbf{K}$ is linked to the unified sense inventory S and added to $\mathbf{K}^S$. However, each $KB_i^S \in \mathbf{K}^S$ still provides its own relation set $R_i^S \subseteq R_i$. Instead, in the unified $\mathbf{KB}^*$, relations with equivalent semantics should be considered as part of a single relation synset even when they come from different KBs. Therefore, at this stage, we apply an alignment algorithm to identify pairs of relations from different KBs having equivalent semantics. We exploit the fact that each relation r is now defined over entity pairs linked to S, and we generate a semantic representation of r in the vector space V_S based on the centroid vectors of its domain and range. Due to representing the semantics of relations on this common ground, we can compare them by computing their domain and range similarity in V_S. We

first consider each $KB_i^S \in \mathbf{K}^S$ and, for each relation r_i in R_i^S, we compute the corresponding centroid vectors $\mu_d^{r_i}$ and $\mu_g^{r_i}$ using formula (2). Then, for each pair of KBs $\langle KB_i^S, KB_j^S \rangle \in \mathbf{K}^S \times \mathbf{K}^S$, we compare all relation pairs $\langle r_i, r_j \rangle \in R_i^S \times R_j^S$ by computing the cosine similarity between domain centroids s_D and between range centroids s_G:

$$s_k = \frac{\mu_k^{r_i} \cdot \mu_k^{r_j}}{\|\mu_k^{r_i}\| \, \|\mu_k^{r_j}\|} \quad (4)$$

where μ_k^r denotes the centroid associated with relation r and $k \in \{D, G\}$. The average of s_D and s_G gives us an *alignment confidence* ζ_{align} for the pair $\langle r_i, r_j \rangle$. If confidence is above a given threshold δ_{align} then r_i and r_j are merged into the same relation synset. Relations for which no alignment is found are turned into singleton relation synsets. As a result of this alignment procedure we obtain the unified set of relations R^*.

7 Experimental Setup

The setting for our experimental evaluation was the following:

- We used BabelNet 3.0[3] as our unified sense inventory for the unification procedure as well as the underlying inventory for both BABELFY and SENSEMBED. Currently, BabelNet contains around 14M synsets and represents the largest single multilingual repository of entities and concepts;
- We selected PATTY (Nakashole et al., 2012) and WISENET (Moro and Navigli, 2013) as linked resources. We used PATTY with FREEBASE types and pattern synsets derived from Wikipedia, and WISENET 2.0 with Wikipedia relational phrases;
- We selected NELL (Carlson et al., 2010) and REVERB (Fader et al., 2011) as unlinked resources. We used KB beliefs updated to November 2014 for the former, and the set of relation instances from ClueWeb09 for the latter.

Comparative statistics in Table 1 show that the input KBs are rather different in nature: NELL is based on 298 predefined relations and contains beliefs for about 2 million entities. The distribution of entities over relations is however very skewed, with 80.33% of the triples being instances of the `generalizations` relationship. In contrast, REVERB contains a highly sparse relation set (1,299,844 distinct relations) and more than 3 million distinct entities. PATTY features the largest (and, together with WISENET, sparsest) set of triples, with 1,631,531 distinct relations and less than 10 triples per relation on average.

[3] `http://babelnet.org`

	$\mathbf{K}_U$		$\mathbf{K}_D$	
	NELL	REVERB	PATTY	WISENET
# relations	298	1 299 844	1 631 531	245 935
# triples	2 245 050	14 728 268	15 802 946	2 271 807
# entities	1 996 021	3 327 425	1 087 907	1 636 307

Table 1: Statistics on the input KBs

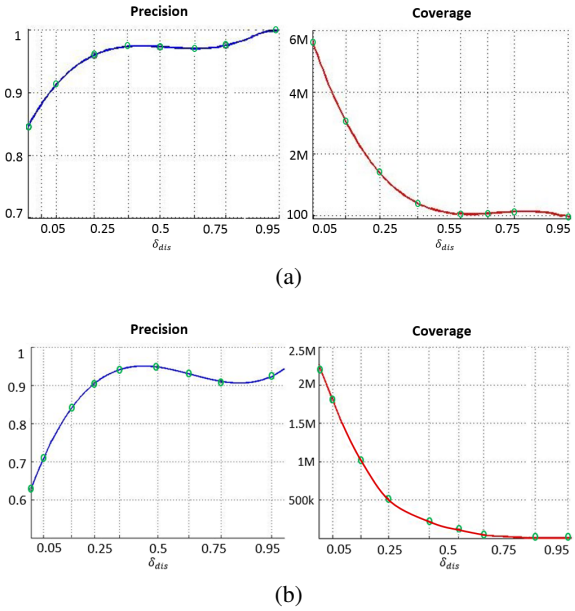


Figure 3: Precision (left) and coverage (right) of disambiguated seeds at different values of δ_{dis} for (a) the whole set of triples in PATTY and (b) the subset of ambiguous triples

8 Experiments

8.1 Disambiguation

We tested our disambiguation approach experimentally in terms of both disambiguated seed quality (Section 8.1.1) and overall disambiguation performance (Section 8.1.2). We created a development set by extracting a subset of 6 million triples from the largest linked KB in our experimental setup, i.e. PATTY. Triples in PATTY are automatically linked to YAGO, which is in turn linked to WordNet and DBPEDIA. Since both resources are also linked by BabelNet, we mapped the original triples to the BabelNet sense inventory and used them to tune our disambiguation module. We also provide two baseline approaches: (1) di-

	SENSEMBED			Baseline		
ζ_{dis}	0.5-0.7	0.7-0.9	0.9-1.0	0.5-0.7	0.7-0.9	0.9-1.0
PATTY	.980	.980	1.000	.793	.780	1.000
WISENET	.958	.960	.973	.726	.786	.791
NELL	.955	.995	1.000	.800	.770	.885
REVERB	.930	.940	.950	.775	.725	.920

Table 2: Disambiguation precision for all KBs

	$\delta_{spec} = 0.8$		$\delta_{spec} = 0.5$		$\delta_{spec} = 0.3$	
	all	only seeds	all	only seeds	all	only seeds
PATTY	62.15	26.60	52.49	24.06	40.75	21.41
WISENET	60.00	37.46	54.44	22.26	53.58	16.62
NELL	76.97	62.98	50.95	20.71	44.70	4.36
REVERB	41.20	38.57	25.14	23.70	13.37	12.75

Table 3: Coverage results (%) for all KBs

rect disambiguation on individual triples with BABELFY alone (without the seeds) and (2) direct disambiguation of the seeds only (without BABELFY).

8.1.1 Results: Disambiguated Seeds

We tuned our disambiguation algorithm by studying the quality of the disambiguated seeds (Section 5.1) extracted from the surface text triples of PATTY. Figure 3 shows precision and coverage for increasing values of the confidence threshold δ_{dis}. We computed precision by checking each disambiguated seed against the corresponding linked triple in the development set, and coverage as the ratio of covered triples. We analyzed results for both the whole set of triples in PATTY (Fig. 3a) and the subset of ambiguous triples (Fig. 3b), i.e. those triples whose subjects and objects have at least two candidate senses each in the BabelNet inventory. In both cases, precision of disambiguated seeds increases rapidly with δ_{dis}, stabilizing above 90% with $\delta_{dis} > 0.25$. Coverage displays the opposite behavior, decreasing exponentially with more confident outcomes, from 6 million triples to less than a thousand (for seeds with confidence $\delta_{dis} > 0.95$). As a result, we chose $\delta_{dis} = 0.25$ as optimal threshold value throughout the rest of the evaluations.

In addition, we manually evaluated the disambiguated seeds extracted from both linked KBs (PATTY and WISENET) and unlinked KBs (NELL and REVERB). For each KB, we extracted up to three random samples of 150 triples according to different levels of confidence ζ_{dis}: the first sample included extraction with $0.5 \leq \zeta_{dis} < 0.7$, the second with $0.7 \leq \zeta_{dis} < 0.9$, and the third with $\zeta_{dis} \geq 0.9$. Each sample was evaluated by two human judges: for each disambiguated triple

	KB-UNIFY		Dutta et al.	Baseline
	all	only seeds	($\alpha = 0.5$)	
Precision	.852	**.957**	.931	.749
Recall	**.875**	.117	.799	.608
F-score	**.864**	.197	.857	.671

Table 4: Disambiguation results over NELL gold standard

$\langle e_d, r, e_g \rangle$, we presented our judges with the surface text arguments e_d, e_g and the relation string r, along with the two BabelNet synsets corresponding to the disambiguated arguments s_d^*, s_g^*, and we asked whether the association of each subject and object with the proposed BabelNet synset was correct. We then estimated precision as the average proportion of correctly disambiguated triples. For each sample we compared disambiguation precision using SENSEMBED, as in Section 5.1, against the first baseline with BABELFY alone. Results, reported in Table 2, show that our approach consistently outperforms the baseline and provides high precision over all samples and KBs.

8.1.2 Results: Disambiguation with Relation Context

We then evaluated the overall disambiguation output after specificity ranking (Section 5.2) and disambiguation with relation context using BABELFY (Section 5.3). We analyzed three configurations of the disambiguation pipeline, namely $\delta_{spec} \in \{0.8, 0.5, 0.3\}$. We ran the algorithm over both linked and unlinked KBs of our experimental setup, and computed the coverage for each KB as the overall ratio of disambiguated triples. Results are reported in Table 3 and compared to the coverage obtained from the disambiguated seeds only: context-aware disambiguation substantially increases coverage over all KBs. Table 3 also shows that a restrictive δ_{spec} results in lower coverage values, due to the increased number of triples disambiguated without context.

Finally, we evaluated the quality of disambiguation on a publicly available dataset (Dutta et al., 2014) comprising manual annotations for NELL. This dataset provides a gold standard of 1200 triples whose subjects and objects are manually assigned a proper DBpedia URI. We again used BabelNet's inter-resource links to express DBpedia annotations with our sense inventory and then sought, for each annotated triple in the dataset, the corresponding triple in our disambiguated version of NELL with $\delta_{dis} = 0.25$ and $\delta_{spec} = 0.8$. We then repeated this process con-

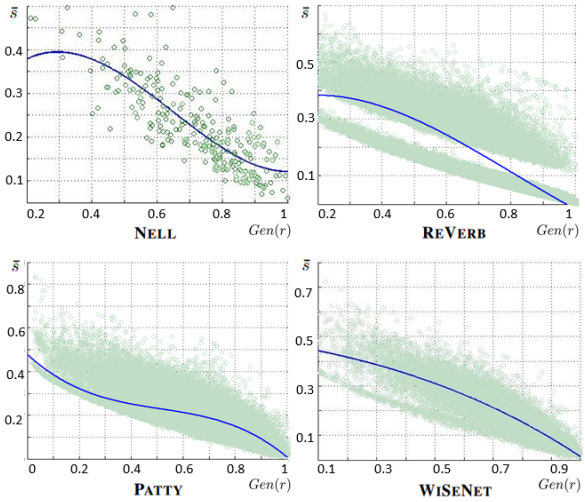


Figure 4: Average argument similarity against $Gen(r)$

	NELL	REVERB	PATTY	WISENET
Precision	.660	.715	.625	.750
Cohen's kappa	-	.430	.620	.600

Table 5: Specificity ranking evaluation

	NELL
High $Gen(r)$	`agent created` `at location`
Low $Gen(r)$	`person in economic sector` `restaurant in city`
	REVERB
High $Gen(r)$	`is for` `is in`
Low $Gen(r)$	`enter Taurus in` `carry oxygen to`
	PATTY
High $Gen(r)$	`located in` `later served to`
Low $Gen(r)$	`starting pitcher who played` `league coach for`
	WISENET
High $Gen(r)$	`include` `is a type of`
Low $Gen(r)$	`lobe-finned fish lived during` `took part in the Eurovision contest`

Table 6: Examples of general and specific relations for all KBs

sidering only the disambiguated seeds instead of the whole disambiguation pipeline. In line with (Dutta et al., 2014), we computed precision, recall and F-score for each setting. Results are reported in Table 4 and compared against those of Dutta et al. (2014) and against our first baseline with BABELFY alone. KB-UNIFY achieves the best result, showing that a baseline based on state-of-the-art disambiguation is negatively affected by the lack of context for each individual triple. In contrast, an approach that relies only on the disambiguated seeds affords very high precision, but suffers from dramatically lower coverage.

8.2 Specificity Ranking

We evaluated the specificity ranking (Section 5.2) generated by KB-UNIFY for all KBs of our experimental setup. First of all, we empirically validated our scoring function $Gen(r)$ over each resource: for each relation we computed the average cosine similarity among all its domain arguments $\bar{s}_D$ and among all its range arguments $\bar{s}_G$. We then plotted the average $\bar{s}$ of $\bar{s}_D$ and $\bar{s}_G$ against $Gen(r)$ for each relation r (Figure 4). As observed in Section 5.2, the average similarity among domain and range arguments decreases for increasing values of $Gen(r)$, indicating that more general relations allow less semantically constrained subject-object types. We then used human judgement to assess the quality of our specificity rankings. First, each ranking was split into four quartiles, and two human evaluators were presented with a sample from the top quartile (i.e. a relation falling into the most general category) and a sample from the bottom quartile (i.e. a relation falling into the most specific category). We shuffled each relation pair, showed it to our human judges, and then asked which of the two relations they considered to be the more specific. Ranking precision was computed by considering those pairs where human choice agreed with the ranking. Finally, we computed inter-annotator agreement on each specificity ranking (except for NELL, due to the small sample size) with Cohen's kappa coefficient (Cohen, 1968). Results for each ranking are reported in Table 5, while some examples of general and specific relations for each KB are shown in Table 6. Disagreement between human choice and ranking is higher in NELL (where the set of relations is quite small compared to other KBs) and in PATTY (due to a sparser set of relations, biased towards very specific patterns). Inter-annotator agreement is instead lower for REVERB, where unconstrained Web harvesting often results in ambiguous relation strings.

8.3 Alignment

Due to the novelty of our approach, and hence the lack of widely accepted gold standards and testbeds, we evaluated our cross-resource relation alignment algorithm (Section 6) by exploiting human judgement once again. Given the results of

	PATTY-WISENET		PATTY-REVERB		NELL-REVERB	
δ_{align}	0.7	0.9	0.7	0.9	0.7	0.9
Prec.	.68	**.80**	.58	**.74**	.61	**.75**
# Align.	128k	1.2k	47k	643	2.6k	88
	PATTY-NELL		**WISENET-NELL**		**WISENET-REVERB**	
δ_{align}	0.7	0.9	0.7	0.9	0.7	0.9
Prec.	.66	**1.00**	.70	**.84**	.59	**.87**
# Align.	2.6k	57	381	34	9.9k	169

Table 7: Cross-resource alignment evaluation

PATTY-WISENET		ζ_{align}
portrayed	's character	0.84
debuted in	first appeared in	0.86
PATTY-REVERB		ζ_{align}
language in	is spoken in	0.81
mostly known for	plays the role of	0.70
NELL-REVERB		ζ_{align}
bookwriter	is a novel by	0.88
personleadscity	is the mayor of	0.60
NELL-PATTY		ζ_{align}
worksfor	was hired by	0.72
riveremptiesintoriver	tributary of	0.89
NELL-WISENET		ζ_{align}
animaleatfood	feeds on	0.72
teamhomestadium	play their home games at	0.88
REVERB-WISENET		ζ_{align}
has a selection of	offers	0.82
had grown up in	was born and raised in	0.85

Table 8: Examples of cross resource relation alignments

Section 8.1, we considered the top 10k frequent relations for each KB and ran the algorithm over each possible pair of KBs with two different configurations: $\delta_{align} = 0.7$ and $\delta_{align} = 0.9$. From each pair of KBs $\langle KB_i, KB_j \rangle$ we obtained a list of candidate alignments, i.e. pairs of relations $\langle r_i, r_j \rangle$ where $r_i \in KB_i$ and $r_j \in KB_j$. From each list we then extracted a random sample of 150 candidate alignments. We showed each alignment[4] $\langle r_i, r_j \rangle$ to two human judges, and asked whether r_i and r_j represented the same relation. The problem was presented in terms of paraphrasing: for each pair, we asked whether exchanging r_i and r_j within a sentence would have changed that sentence's meaning. In line with Section 8.2 we computed precision based on the agreement between human choice and automatic alignments. Results are reported in Table 7. Our alignment algorithm shows high precision in all pairings where $\delta_{align} = 0.9$. Alignment reliability decreases for lower δ_{align}, as relation pairs where r_i is a generalization of r_j (or vice versa) tend to have similar centroids in V_S. The same holds for pairs where r_i is the negation of r_j (or vice versa). Even though we could have utilized measures based on relation string similarity (Dutta et al., 2015) to reduce wrong alignments in these cases, by relying on a purely semantic criterion we removed any prior assumption on the format of input KBs. Some examples of alignments are shown in Table 8.

To conclude, we report statistics regarding the unified **KB*** produced from the initial set of resources in our experimental setup (cf. Section 7). We validated our thresholds for high-precision, and selected $\delta_{dis} = 0.25$, δ_{spec} = 0.8 and δ_{align} = 0.8. Our alignment algorithm produced 56,673 confident alignments, out of which 2,207 relation synsets were derived, with an average size of 16.82 individual relations per synset. As a result, we obtained a unified **KB*** comprising 24,221,856 disambiguated triples defined over 1,952,716 distinct entities and 2,675,296 distinct relations.

9 Conclusion and Future Work

We have presented KB-UNIFY, a novel, general approach for disambiguating and seamlessly unifying KBs produced by different OIE systems. KB-UNIFY represents entities and relations using a shared semantic representation, leveraging a unified sense inventory together with a semantically-enhanced vector space model and a disambiguation algorithm. This enables us to disambiguate unlinked resources (like NELL and REVERB) with high precision and coverage, and to discover relation-level cross-resource alignments effectively. One of the key features of our strategy is its generality: by representing each KB on a common ground, we need no prior assumption on the nature and format of the knowledge it encodes. We tested our approach experimentally on a set of four very different KBs, both linked and unlinked, and we evaluated disambiguation and alignment results extensively at every stage, exploiting both human evaluations and public gold standard datasets (when available). This work opens compelling avenues for future work. We plan to further exploit sense-enhanced unified representations of relations in various ways: providing an ontological structure for the unified KB, exploring complementary approaches for capturing semantic relation alignments, and incorporating multilinguality.

Acknowledgments

The authors gratefully acknowledge the support of the ERC Starting Grant MultiJEDI No. 259234.

[4] In the case of relation synsets, such as PATTY and WISENET, we selected up to three random relation strings from each synset.

References

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: A Collaboratively Created Graph Database For Structuring Human Knowledge. In *Proceedings of SIGMOD*, pages 1247–1250.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating Embeddings for Modeling Multi-relational Data. In *Advances in NIPS*, volume 26, pages 2787–2795.

Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R. Hruschka Jr., and Tom M. Mitchell. 2010. Toward an Architecture for Never-Ending Language Learning. In *Proceedings of AAAI*, pages 1306–1313.

Jacob Cohen. 1968. Weighted Kappa: Nominal Scale Agreement Provision for Scaled Disagreement or Partial Credit. *Psychological Bulletin*, 70(4):213–220.

Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. 2014. Knowledge Vault: A Web-scale Approach to Probabilistic Knowledge Fusion. In *Proceedings of the SIGKDD*, pages 601–610.

Arnab Dutta, Christian Meilicke, and Simone Paolo Ponzetto. 2014. A Probabilistic Approach for Integrating Heterogeneous Knowledge Sources. In *Proceedings of ESWC*, pages 286–301.

Arnab Dutta, Christian Meilicke, and Heiner Stuckenschmidt. 2015. Enriching Structured Knowledge with Open Information. In *Proceedings of WWW*, pages 267–277.

Oren Etzioni, Michele Banko, Stephen Soderland, and Daniel S. Weld. 2008. Open Information Extraction from the Web. *Commun. ACM*, 51(12):68–74.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying Relations for Open Information Extraction. In *Proceedings of EMNLP*, pages 1535–1545.

Miao Fan, Deli Zhao, Qiang Zhou, Zhiyuan Liu, Thomas Fang Zheng, and Edward Y. Chang. 2014. Distant Supervision for Relation Extraction with Matrix Completion. In *Proceedings of ACL*, pages 839–849.

Adam Grycner and Gerhard Weikum. 2014. HARPY: Hypernyms and Alignment of Relational Paraphrases. In *Proceedings of ACL*, pages 2195–2204.

Iryna Gurevych, Judith Eckle-Kohler, Silvana Hartmann, Michael Matuschek, Christian M. Meyer, and Christian Wirth. 2012. Uby: A large-scale unified lexical-semantic resource based on LMF. In *Proceedings of ACL*, pages 580–590.

Eric H. Huang, Richard Socher, Christopher D. Manning, and Andrew Y. Ng. 2012. Improving Word Representations via Global Context and Multiple Word Prototypes. In *Proceedings of ACL*, pages 873–882.

Ignacio Iacobacci, Mohammad Taher Pilehvar, and Roberto Navigli. 2015. SensEmbed: Learning Sense Embeddings for Word and Relational Similarity. In *Proceedings of ACL*, pages 95–105.

Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N. Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, and Christian Bizer. 2014. DBpedia - A Large-scale, Multilingual Knowledge Base Extracted from Wikipedia. *Semantic Web Journal*, pages 1–29.

Thomas Lin, Mausam, and Oren Etzioni. 2012. No Noun Phrase Left Behind: Detecting and Typing Unlinkable Entities. In *Proceedings of EMNLP-CoNLL*, pages 893–903.

Farzaneh Mahdisoltani, Joanna Biega, and Fabian M. Suchanek. 2015. YAGO3: A Knowledge Base from Multilingual Wikipedias. In *CIDR*.

Tomas Mikolov, Kal Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. In *Proceedings of Workshop at ICLR*.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant Supervision for Relation Extraction Without Labeled Data. In *Proceedings of ACL-IJCNLP*, pages 1003–1011.

Andrea Moro and Roberto Navigli. 2013. Integrating Syntactic and Semantic Analysis into the Open Information Extraction Paradigm. In *Proceedings of IJCAI*, pages 2148–2154.

Andrea Moro, Alessandro Raganato, and Roberto Navigli. 2014. Entity Linking meets Word Sense Disambiguation: a Unified Approach. *TACL*, 2:231–244.

Ndapandula Nakashole, Gerhard Weikum, and Fabian M. Suchanek. 2012. PATTY: A Taxonomy of Relational Patterns with Semantic Types. In *Proceedings of EMNLP-CoNLL*, pages 1135–1145.

Roberto Navigli and Simone Paolo Ponzetto. 2012. BabelNet: The Automatic Construction, Evaluation and Application of a Wide-Coverage Multilingual Semantic Network. *Artificial Intelligence*, 193:217–250.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2012. Factorizing YAGO: Scalable Machine Learning for Linked Data. In *Proceedings of WWW*, pages 271–280.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling Relations and Their Mentions Without Labeled Text. In *Proceedings of ECML-PKDD*, pages 148–163.

Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M. Marlin. 2013. Relation Extraction with Matrix Factorization and Universal Schemas. In *Proceedings of NAACL*, pages 74–84.

Richard Socher, Danqi Chen, Christopher D. Manning, and Andrew Y. Ng. 2013. Reasoning with Neural Tensor Networks for Knowledge Base Completion. In *Advances in NIPS*, pages 926–934.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D. Manning. 2012. Multi-instance Multi-label Learning for Relation Extraction. In *Proceedings of EMNLP-CoNLL*, pages 455–465.

Jason Weston, Antoine Bordes, Oksana Yakhnenko, and Nicolas Usunier. 2013. Connecting Language and Knowledge Bases with Embedding Models for Relation Extraction. In *Proceedings of EMNLP*, pages 1366–1371.

Fei Wu and Daniel S. Weld. 2010. Open Information Extraction using Wikipedia. In *Proceedings of ACL*, pages 118–127.

Open-Domain Name Error Detection using a Multi-Task RNN

Hao Cheng **Hao Fang** **Mari Ostendorf**
Department of Electrical Engineering
University of Washington
{chenghao,hfang,ostendor}@uw.edu

Abstract

Out-of-vocabulary name errors in speech recognition create significant problems for downstream language processing, but the fact that they are rare poses challenges for automatic detection, particularly in an open-domain scenario. To address this problem, a multi-task recurrent neural network language model for sentence-level name detection is proposed for use in combination with out-of-vocabulary word detection. The sentence-level model is also effective for leveraging external text data. Experiments show a 26% improvement in name-error detection F-score over a system using n-gram lexical features.

1 Introduction

Most spoken language processing or dialogue systems are based on a finite vocabulary, so occasionally a word used will be out of the vocabulary (OOV), in which case the automatic speech recognition (ASR) system chooses the best matching in-vocabulary sequence of words to cover that region (where acoustic match dominates the decision). The most difficult OOV words to cover are names, since they are less likely to be covered by morpheme-like subword fragments and they often result in anomalous recognition output, e.g.

REF: what can we get at **Litanfeeth**
HYP: what can we get *it leaks on feet*

While these errors are rare, they create major problems for language processing, since names tend to be important for many applications. Thus, it is of interest to automatically detect such error regions for additional analysis or human correction.

Named entity recognition (NER) systems have been applied to speech output (Palmer and Ostendorf, 2005; Sudoh et al., 2006), taking advantage of local contextual cues to names (e.g. titles for person names), but as illustrated above, neighboring words are often affected, which obscures lexical cues to name regions. Parada *et al.* (2011) reduce this problem somewhat by applying an NER tagger to a word confusion network (WCN) based on a hybrid word/fragment ASR system.

In addition to the problem of noisy context, automatic name error detection is challenging because name errors are rare for a good recognizer. To learn the cues to name errors, it is necessary to train from the output of the target recognizer, so machine learning is faced with infrequent positive examples for which training data is very sparse. In addition, in an open domain system, automatically-learned lexical context features from one domain may be useless in another.

In this paper, we address these general problems – detecting rare events in an open-domain task – specifically for name error detection. Prior work addressed the problem of skewed priors by artificially increasing the error rate by holding names out of the vocabulary (Chen et al., 2013) or by factoring the problem into sentence-level name detection and OOV word detection (He et al., 2014) (since OOV errors in general are more frequent than name errors). Sentence-level features are also shown to be more robust than local context in direct name error prediction (Marin, 2015). While these techniques provide some benefit, the use of discrete lexical context cues is sensitive to the limited amount of training data available.

Our work leverages the factored approach, but improve performance by using a continuous-space sentence representation for predicting presence of a name. Specifically, we modify a recurrent neural network (RNN) language model (LM) to predict both the word sequence and a sentence-level name indicator. Combining the LM objective with name prediction provides a regularization effect in training that leads to improved sentence-level

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 737–746,
Lisbon, Portugal, 17-21 September 2015.

name prediction. The continuous-space model is also effective for leveraging external text resources to improve generalization in the open-domain scenario.

The overall framework for speech recognition and baseline name error detection system is outlined in Section 2, and the multi-task (MT) RNN approach for sentence-level name prediction is introduced in Section 3. Experimental results for both sentence-level name detection and name error detection are presented in Section 4, demonstrating the effectiveness of the approach on test data that varies in its match to the training data. As discussed in Section 5, the sentence-level model is motivated by similar models for other applications. The paper summarizes key findings and discusses potential areas for further improvement in Section 6.

2 System Overview and Tasks

The name error detection task explored in this work is a component in a bidirectional speech-to-speech translation system for English to/from Iraqi Arabic with human-computer dialogue interaction for error resolution (Ayan et al., 2013), developed during the DARPA BOLT project. The training data consists of a range of topics associated with activities of military personnel, including traffic control, military training, civil affairs, medical checkups, and so on. However, the system is expected to handle open-domain tasks, and thus the evaluation data covers a broader range of topics, including humanitarian aid and disaster relief, as well as more general topics such as sports, family and weather. The dialogues often contain mentions of names and places, many of which are OOV words to the ASR system. As illustrated in the previous section, ASR hypotheses necessarily have errors in OOV regions, but because specific names are infrequent, even in-vocabulary names can have these types of error patterns. Therefore, developing a robust name error detector for ASR hypotheses is an important component of the system to resolve errors and ambiguity.

Detecting OOV errors requires combining evidence of recognizer uncertainty and anomalous word sequences in a local region. For name errors, lexical cues to names are also useful, e.g. a person's title, location prepositions, or keywords such as "name". The baseline system for this work uses structural features extracted from a confusion network of ASR hypotheses plus ASR word confidence to represent recognizer uncertainty, and word n-gram context to the left and right of the target confusion network slot. These features are combined in a maximum entropy (ME) classifier trained to predict name errors directly. This is the same as the baseline used in (He et al., 2014; Marin, 2015; Marin et al., 2015), but with a different ASR system.

Training a classifier to predict whether a sentence has a name is easier than direct name error prediction, because the positive class is less rare, and it does not require recognizer output so more data can be used (e.g. speech transcripts without recognizer output, or written text). In addition, since the words abutting the name are less reliable in a recognition hypothesis, the information lost by working at the sentence level is minimal. The idea of using sentence-level name prediction is proposed in (He et al., 2014), but in that work the sentence name posterior is a feature in the ME model (optionally with word cues learned by the sentence-level predictor). In our work, the problem is factored to use the acoustic confusibility and local word class features for OOV error prediction, which is combined with the sentence-level name posterior for name error prediction. In other words, only two features (posteriors) are used in training with the sparse name error prediction data. An ME classifier is then used to combine the two features to predict the word-level name error. The word-level OOV detector is another ME binary classifier; the full set of features used for the word-level OOV detector can be found in (Marin, 2015).

The main innovation in this work is that we propose to use a multi-task RNN model for the sentence-level name prediction, where the training objective takes into account both the language modeling task and the sentence-level name prediction task, as described in the next section.

3 Multi-task Recurrent Neural Network

The RNN is a powerful sequential model and has proven to be useful in many natural language processing tasks, including language modeling (Mikolov et al., 2010) and word similarity (Mikolov et al., 2013c). It also achieves good results for a variety of text classification problems when combined with a convolutional neural network (CNN) (Lai et al., 2015). In this paper, we

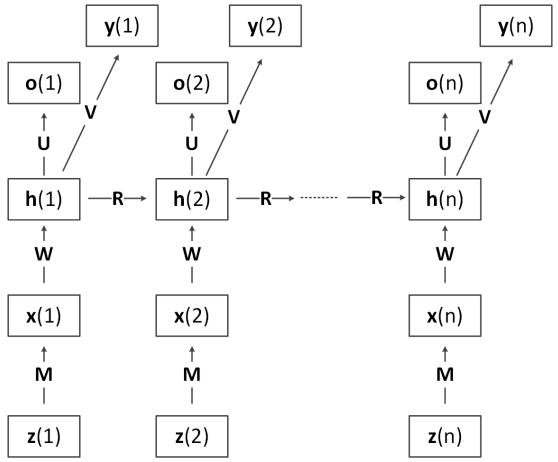


Figure 1: The structure of the proposed MT RNN model, which predicts both the next word $o(t)$ and whether the sentence contains a name $y(t)$ at each time step.

propose an MT RNN for the sentence-level name prediction task, which augments the word prediction in the RNN language model with an additional output layer for sentence-level name prediction. Formally, the MT RNN is defined over $t = 1, \ldots, n$ for a sentence of length n as:

$$\begin{aligned} \mathbf{x}(t) &= M\mathbf{z}(t), \\ \mathbf{h}(t) &= f\left(W\mathbf{x}(t) + R\mathbf{h}(t-1)\right), \\ \mathbf{o}(t) &= s\left(U\mathbf{h}(t) + \mathbf{b}_1\right), \\ \mathbf{y}(t) &= s\left(V\mathbf{h}(t) + \mathbf{b}_2\right). \end{aligned}$$

where $\mathbf{z}(t) \in \mathbb{R}^V$ is the 1-of-V encoding of the t-th word in the sentence; $\mathbf{x}(t) \in \mathbb{R}^d$ is the d-dimensional continuous word embedding corresponding to the t-th word; $\mathbf{h}(t) \in \mathbb{R}^k$ is a k-dimensional embedding that summarizes the word sequence through time t; and $\mathbf{o}(t)$ and $\mathbf{y}(t)$ are respectively the output layers for the language modeling task and the sentence-level prediction task. The parameters of the model (learned in multi-task training) include: $M \in \mathbb{R}^{d \times V}$, which is usually referred to as the word embedding matrix; projection matrices $U \in \mathbb{R}^{V \times d}$ and $V \in \mathbb{R}^{2 \times d}$; and bias terms $\mathbf{b}_1, \mathbf{b}_2 \in \mathbb{R}^d$. f and s are respectively the sigmoid and softmax functions. Note that the word sequence associated with a sentence includes start and end symbols.

The structure of the proposed MT RNN is shown in Fig. 1. At each time step, the hidden vector is used to predict both the next word and a sentence-level indicator of the presence of a name, providing a probability distribution for both variables. Thus, the hidden vector h_t provides a continuous representation of the word history that emphasizes words that are important for predicting the presence of a name. The vector at time n can be thought of as a sentence embedding. The sentence-level output y_t differs from the word-dependent label predictor typically used in named entity detection (or part-of-speech tagging) in that it is providing a sequentially updated prediction of a sentence-level variable, rather than a word-level indicator that specifies the location of a named entity in the sentence. The final prediction y_n is used as a feature in the name error detection system. The sentence-level output y_t does not always converge to y_n gradually nor is it always monotonic, since the prediction can change abruptly (either positively or negatively) as new words are processed. The sentence-level variable provides a mechanism for capturing long distance context, which is particularly useful for speech applications, where both the name of interest and the words in its immediate context may be in error.

The training objective is the combination of the log-likelihood of the word sequence and that of the sentence-level name prediction:

$$\sum_{t=1}^{n} \left[(1-\lambda)\log P(w(t)|h(t)) + \lambda \log P(y(t)|h(t))\right], \quad (1)$$

where $h(t) = [w(1), \ldots, w(t-1)]$ and λ is the weight on the log-likelihood of the sentence-level name labels.

Another way to train the model is to predict the sentence-level name label *only* at the end of the sentence, rather than at every time step. Preliminary experiment results show that this model has inferior performance. We argue that training with only the sentence-final name label output can result in unbalanced updates, i.e., information from the language modeling task is used more often than that from the sentence-level name prediction task, implying that balancing the use of information sources is an important design choice for multi-task models.

The training objective is optimized using stochastic gradient descent (SGD). We also experiment with AdaGrad (Duchi et al., 2011), which has shown to be more stable and converge faster for non-convex SGD optimization. Since language model training requires a normalization operation each time over the whole vocabulary (~60K) which is computationally intensive, we further speed up training by using noise contrastive esti-

mation (NCE) (Mnih and Teh, 2012). For all models using NCE, we fix the number of negative samples to 50.

All the weights are randomly initialized in the range of $[-0.1, 0.1]$. The hidden layer size k is selected from {50, 100, 200} and the task mixing weight λ is select in {0.2, 0.4, 0.6, 0.8}, based on development set performance. We set the initial learning rate to 1 and 0.1 for SGD with and without Adagrad respectively. In our experiments, we observe that models trained with AdaGrad achieve better performance, so we only report the models with AdaGrad in this paper. At each training epoch, we validate the objective on the development set. The learning rate is reduced after the first time the development set loglikelihood decreases, and the whole training procedure terminates when the development set loglikelihood decreases for the second time.

Domain	Topics
TRANSTAC	Traffic Control, Facilities Inspection, Civil Affairs, Medical, Combined Training, Combined Operations
HADR	Humanitarian Aid, Disaster Relief
General	Family, Gardening, Sports, Pets, Books, Weather, Language, Phone

Table 1: Topics in different domains.

Target Class	BOLT-Train	Dev1	Dev2	Test1	Test2
name sentences	7.6	7.5	8.2	8.1	14.0
name errors	0.8	1.1	0.9	0.7	0.8
OOVs	1.7	1.8	1.8	1.8	1.7

Table 2: Data splits and statistics, including the percentage of sentences containing names (name sentences), the percentage of hypothesized words that are name errors (name errors), and the percentage of words that are OOVs (OOVs).

4 Experiments

4.1 Data

There are two types of datasets used in this paper: the BOLT dataset and a collection of Reddit discussions. The first dataset was collected during the DARPA TRANSTAC and BOLT projects. The ASR hypotheses of totally 7088 spoken sentences makes up of the training dataset (BOLT-Train) for both sentence-level name prediction and word-level name error detection. There are two development sets: Dev1 and Dev2. Dev1 is used for parameter tuning for all models, and Dev2 is used for training the ME-based word-level name error detector using the word-level OOV posterior and sentence-level name posterior. For RNN models, we tune the hidden layer size and the multi-task weight λ; for the ME-based word-level name error detector, we tune the regularization parameter. Two test sets are used to evaluate sentence-level name prediction (Test1) and word-level name error detection (Test2), based on the BOLT phase 2 and 3 evaluations, respectively.

As shown in Table 1, the BOLT topics are categorized into three domains: TRANSTAC, HADR and General. The BOLT-Train, Dev1 and Dev2 sets contain only speech from the TRANSTAC domain, whereas the Test1 and Test2 sets contain all three domains. Detailed data split statistics and domain information are summarized in Table 2. Note that there are very few positive samples of name errors (roughly 1%), whereas for the sentence-level name prediction task and the word-level OOV prediction task, the data skewness is somewhat less severe (roughly 8% sentences with names in most of our data sets).

In order to address the issue of domain mismatch between the training data and the broad-domain subsets of the test data, we collect text from *Reddit.com* which is a very active discussion forum where users can discuss all kinds of topics. Reddit has thousands of user-created and user-moderated *subreddits*, each emphasizing a different topic. For example, there is a general ASKREDDIT subreddit for people to ask any questions, as well as subreddits targeted for specific interests, like ASKMEN, ASKWOMEN, ASKSCIENCE, etc. Although the Reddit discussions are different from BOLT data, they have a conversational nature and names are often observed within the discussions. Therefore, we hypothesize that they can help improve sentence-level name prediction in general. We collect data from 14 subreddits that cover different kinds of topics (such as politics, news, book suggestions) and vary in community size. The Stanford Name Entity Recognizer (Finkel et al., 2005) is used to detect names in each sentence, and a sentence-level name label is assigned if there are any names present.[1] The data are tokenized using Stanford

[1]The Stanford Name Entity Recognizer achieves 82.3% F-score on the references of Dev1. Thus, it is expected to

Model	TRANSTAC	HADR	General	All
BOW + ME	52.9	32.5	11.1	40.8
RNN + ME	11.2	21.7	30.5	15.0
SG + ME	14.7	12.6	16.3	14.6
ST RNN	40.3	37.5	**37.8**	39.3
MT RNN	**59.8**	**52.0**	23.8	**51.1**

Table 3: F-scores on Test1 for sentence-level name prediction for models trained on BOLT data.

CoreNLP tools (Manning et al., 2014) and are lowered cased after running the Stanford Name Entity Recognizer. In total, we obtain 135K sentences containing at least one name and 360K sentences without names.

4.2 Sentence-level Name Prediction

To evaluate the effectiveness of the proposed MT RNN model, we first apply it to the sentence-level name prediction task on ASR hypotheses. For this task, each sample corresponds to a hypothesized sentence generated by the ASR system and a ground-truth label indicating whether there are names in that sentence. We compare the MT RNN with four contrasting models for predicing whether a sentence includes a name.

- **BOW + ME**. An ME classifier using a bag-of-words (BOW) sentence representation.
- **SG + ME**. An ME classifier is used with the sentence embedding as features, where the embedding uses the skip-gram (SG) model (Mikolov et al., 2013a) to get word-level embeddings (with window size 10) and sentence embeddings are composed by averaging embeddings of all words in the sentence.
- **RNN + ME**. A simple RNN LM is trained (i.e., λ in (1) is set to 0), and the hidden layer for the last word in a sentence provides a sentence-level embedding that is used in an ME classifier trained to predict the sentence-level name label.
- **ST RNN**. A single-task (ST) RNN model is trained to directly predict the sentence-level name for each word (i.e., λ in (1) is set to 1).

All models are trained on either BOLT-Train or BOLT-Train + Reddit, and tuned on Dev1 including the dimension of embeddings, ℓ_2 regularization parameters for the ME classifiers, and so on.

The domain-specific F-scores on Test1 are summarized in Table 3 for training only with the BOLT

give useful labels for the Reddit data.

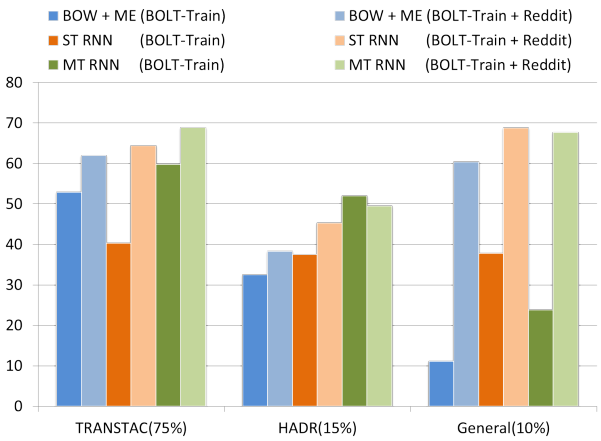


Figure 2: F-scores on Test1 for sentence-level name prediction for models trained with and without Reddit data. The number in the parenthesis indicates the portion of the domain in the data.

data. The proposed MT RNN achieves the best results on the TRANSTAC and HADR domains, and it has significant overall performance improvement over all baseline models. Not surprisingly, the two approaches that used unsupervised learning to obtain a sentence-level embedding (SG + ME, RNN + ME) have the worst performance on the TRANSTAC and HADR domains, with both having very low precision (6-18%), with best precision on the general domain. The BOW + ME and the ST RNN have similar performance in terms of overall F-score, but the BOW + ME model is much better on the TRANSTAC domain, whereas the ST RNN achieves the best results on the General domain. The main failing of the BOW + ME model is in recall on the General domain, though it also has relatively low recall on the HADR domain. Note that the General domain only accounts for 10% of the Test1 set, so the ST RNN gets lower F-score overall compared with the MT RNN, which performs best on the other two domains (90% of the Test1 set). On all domains, the MT RNN greatly improves precision compared to the ST RNN (at the expense of recall), and it improves recall compared to the BOW + ME approach (at the expense of precision).

In order to study the effectiveness of using external data, we also train all models on an enlarged training set including extra name-tagged sentences from Reddit. The improvement for each domain due to also using the Reddit data is shown in Fig. 2 for the three best configurations. (There is no benefit in the unsupervised learning cases.) Com-

pared with training only on the BOLT data, all three of these models get substantial overall performance improvement by utilizing the external domain training data. Since the external training data covers mostly topics in the General domain, the performance gain in that domain is most significant. For the MT RNN and BOW + ME classifiers, the additional training data primarily benefits recall, particulary for the General domain. In contrast, the ST RNN sees an improvement in precision for all domains. One reason that the added training text also benefits the models on the TRANSTAC domain is that over 90% of the words in the speech recognizer vocabulary are not seen in the small set of name-labeled speech training data, which means that the embeddings for these words are random in the RNNs trained only this data and the BOW + ME classifier will never use these words.

4.3 Word-level Name Error Detection

We next assess the usefulness of the resulting MT RNN model for word-level name error detection on ASR hypotheses. Here, several word-level name error detection approaches are compared, including direct name error prediction and factored name/OOV prediction approaches.

- **OOV thresholding**. This system simply uses the OOV prediction, but with the posterior threshold tuned for word-level name error based on the Dev1 set.
- **Word Context**. This is the baseline ME system described in Section 2 and also used as a baseline in (He et al., 2014; Marin et al., 2015), which directly predicts the word-level name error using WCN structural features, current word Brown class, and up-to trigram left and right word context information.
- **Word Class**. This system, from (Marin et al., 2015), also directly predicts the word-level name error, but replaces the word n-gram features with a smaller number of word class features to address the sparse training problem. The word classes are based on seed words learned from sentence-level name prediction which are expanded to classes using a nearest neighbor distance with RNN embeddings trained on the BOLT data.
- **Word Context + OOV**. This system uses an ℓ_2 regularized ME classifier to predict the word-level name error using two posteriors as features: a word-dependent posterior from the Word Context system and one from the OOV detection system.
- **MT RNN + OOV**. This system also uses an ℓ_2 regularized ME classifier to predict the word-level name error using two posteriors as features: the same word-dependent OOV posterior as above, and the posterior from the sentence-level name prediction using the MT RNN model described in Section 4.2, which is constant for all positions in the sentence.

All of the above models are trained on BOLT-Train and tuned on Dev1. The ME classifier used in the Word Context + OOV and the MT RNN + OOV systems are trained on Dev2, with regularization weights and decision boundaries tuned on Dev1.

Name error detection results (F-scores) are summarized in Table 4. The three systems that use discrete, categorical lexical context features (word or word class context) in direct name error prediction have worse results than the OOV thresholding approach overall, as well as on the TRANSTAC subset. Presumably this is due to over-fitting associated with the lexical context features. The factored MT RNN + OOV system, which uses a continuous-space representation of lexical context, achieves a gain in overall performance compared to the other systems and a substantial gain in performance on the TRANSTAC domain on which it is trained. Using the Reddit data further improves the overall F-score, with a slight loss on the TRANSTAC subset but substantial gains in the other domains. Although the performance improvement in the general domain is relatively small compared with sentence-level name prediction, utilizing external data makes the resulting representation more transferable across different domains and tasks. The best MT RNN + OOV system obtains 26% relative improvement over the baseline Word Context system (or 17% improvement over the simple OOV thresholding approach).

Looking at performance trade-offs in precision and recall, we find that the use of the RNN systems mainly improves precision, though there is also a small gain in recall overall. The added training data benefits recall for all domains, with a small loss in precision for the TRANSTAC and HADR sets. The use of the OOV posterior improves precision but limits recall, particularly for the general domain where recall of the OOV posterior alone

Model	TRANSTAC	HADR	General	All
OOV thresholding	41.6	31.2	16.5	30.9
Word Context	37.6	29.0	16.3	28.6
Word Class	33.6	35.9	11.9	27.9
Word Context + OOV	40.2	26.0	14.4	28.4
MT RNN + OOV	47.9	32.8	13.5	34.1
MT RNN† + OOV	46.4	37.5	18.0	36.2

Table 4: F-scores on Test2 for word-level name error prediction. MT RNN is trained on BOLT-Train, whereas MT RNN† is trained on BOLT-Train + Reddit.

is only 18% vs. 25% for the word context model with the OOV posterior information.

To better understand some of the challenges of this task, consider the following examples (R=reference, H=hypothesis):

R1: i'm doing good my name is captain **rodriguez**

H1: i'm doing good my name is captain *road radios*

R2: well it's got flying **lizards** knights and **zombies** and shit

H2: well it's gotta flying lives there it's nights and some bees and shia

R3: i live in a city called **omaha**

H3: i live in a city called *omar*

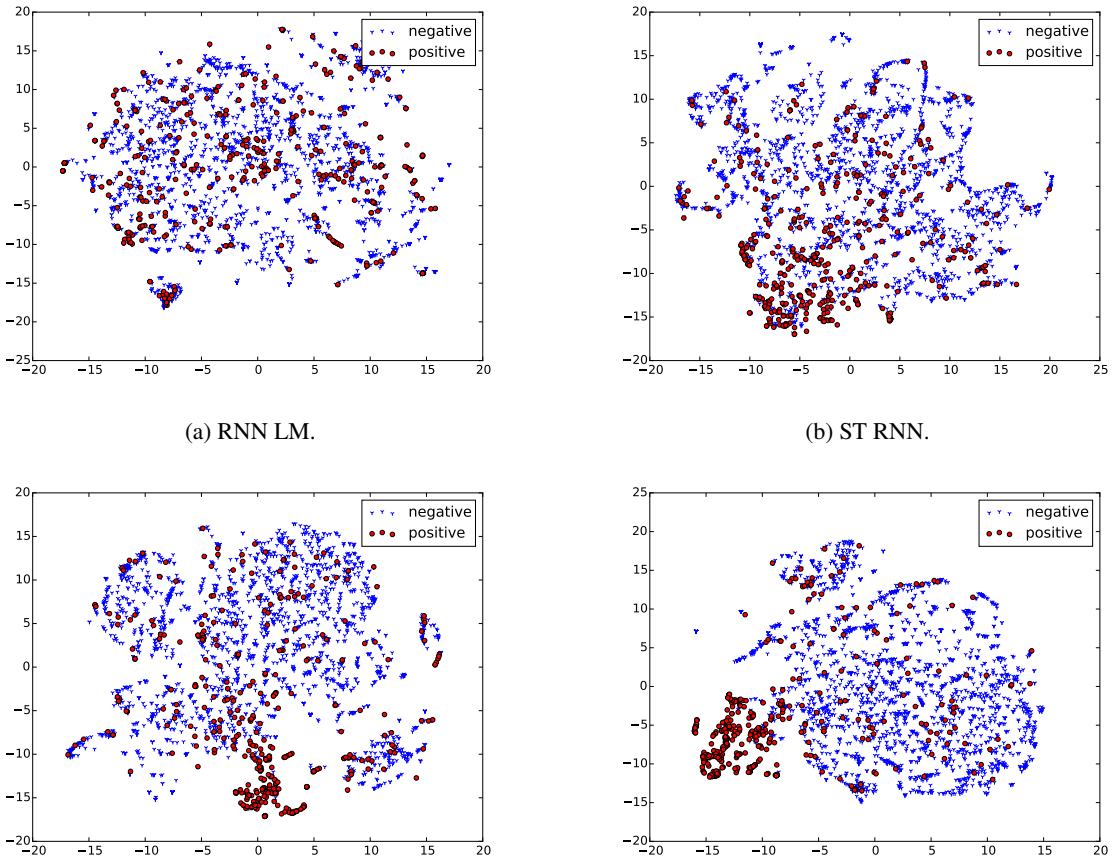
ASR tokens associated with name errors are underlined and italicized; tokens associated with non-name OOV errors are simply underlined. Name errors have a similar character to OOV erors in that they often have anomalous word sequences in the region of the OOV word (examples 1 and 2), which is why the OOV posterior is so useful. However, too much reliance on the OOV posterior leads to wrongly detecting general OOV errors as name errors ('lizards' and 'zombies' in example 2) and missed detection of name errors where the confusion network cues indicate a plausible hypothesis ('omaha' in example 3). Examples 1 and 3 illustrate the importance of lexical cues to names ('name ... captain', 'city called'), but word-based cues are unreliable for the systems trained only on the small amount of domain-specific data. Leveraging the reddit data allowed the MT RNN system to detect the error in example 1 (HADR domain) that was missed by the word context system. Example 3 was only detected by the word context system when no OOV posterior is used. Though this example was from the General domain, city names represent an important error class in the TRANSTAC data, so the term 'city' is learned as a useful cue.

4.4 Sentence Embedding

As discussed in Section 3, we postulate that by modeling words in a sentence in sequential order and simutaneously predicting sentence categorical information, the resulting hidden vector of the last word should be a good representation of the whole sentence, i.e., a sentence embedding. To provide support for this hypothesis and show the impact of external data, we present the sentence embeddings learned by the different RNN variants on Test2 using the t-Distributed Stochastic Neighbor Embedding (t-SNE) visualization (van der Maaten and Hinton, 2008) in Fig. 3. Four models are compared, including three RNNs trained on the BOLT data, and the MT RNN model trained on BOLT + Reddit data. Note the visualization method t-SNE does not take the label information into account during the learning.

As we can see in Fig. 3a, the positive and negative sentence embeddings learned by RNN LM are randomly scattered in the space, indicating that embeddings learned via unsupervised training (e.g., solely on word context) may fail to capture the sentence-level indicators associated with a particular task, in this case presence of a name. When the results are ploted in terms of domains, the embeddings are similarly broadly scattered – there is no obvious topic representation in the embeddings.

When comparing Fig. 3b to Fig. 3a, which is associated with the RNN using a sentence-final name indicator, we can see that a lot of positive vectors have moved to the bottom-left of the space, though there is still a relatively large overlap between positive and negative embeddings. In Fig. 3c, corresponding to the word-level MT RNN, there forms a separable subgroup of positive embeddings and the overlapping seems to be reduced as well in contrast to Figs. 3a and 3b. Finally, most of the positive sentence embeddings produced by the MT RNN model trained with external Reddit data gather at the botton-left of Fig. 3d. In general, the overlap between positive and negative sentences decreases from single task models to multi-task model. The external data make the proposed model produce more well-shaped groupings of sentence embeddings.



(a) RNN LM.

(b) ST RNN.

(c) MT RNN.

(d) MT RNN using extra Reddit data.

Figure 3: t-SNE visualization of sentence embeddings learned from different RNN models.

5 Related Work

Recently, due to the success of continuous representation methods, much work has been devoted to studying methods for learning word embeddings that capture semantic and syntactic meaning. Although these word embeddings are shown to be successful in some word-level tasks, such as word analogy (Mikolov et al., 2013b; Jeffery Pennington, 2014) and semantic role labeling (Collobert and Weston, 2008), it is still an open question how best to compose the word embeddings effectively and make use of them for text understanding.

Recent work on learning continuous sentence representations usually compose the word embeddings using either a convolutional neural network (CNN), a tree-structured recursive NN, or a variant of an RNN. A typical CNN-based structure composes word embeddings in a hierachical fashion (alternating between convolutional layers and pooling layers) to form the continuous sentence representation for sentence-level classification tasks (Kim, 2014; Kalchbrenner et al., 2014; Lai et al., 2015). These models usually build up the sentence representation directly from the lexical surface representation and rely on the pooling layer to capture the dependencies between words. Another popular method for continuous sentence representation is based on the recursive neural network (Socher et al., 2012; Socher et al., 2013; Tai et al., 2015). These models use a tree structure to compose a continuous sentence representation and have the advantages of capturing more fine-grained sentential structure due to the use of parsing trees. Note that the RNN-based sequential modeling used in this paper can be viewed as a linearized tree-structure model.

In this paper, we train the neural network model with a multi-task objective reflecting both the probability of the sequence and the probability that the sequence contains names. The general idea of multitask learning dates back to (Caruana, 1997), and is shown to be effective recently for neural network models in different natural language

processing tasks. Collobert and Weston (2008) propose a unified deep convolutional neural network for different tasks by using a set of task-independent word embeddings together with a set of task-specific word embeddings. For each task, it uses a *unique* neural network with its own layers and connections. Liu et al. (2015) propose a different neural network structure for search query classification and document retrieval where lower-level layers and connections are all shared but the high-level layers are task-specific. For tasks considered in (Collobert and Weston, 2008) and (Liu et al., 2015), training samples are task-dependent. Thus, both models are trained following the SGD manner by alternating tasks for each training samples with task-dependent training objectives. In this paper, we combine the language modeling task with the sentence-level name prediction task, and each training sample has labels for both tasks. Therefore, the SGD training can be done with the weighted sum of the task-specific objectives for each training sample, and the language model objective can be thought of as a regularization term. Similar settings of multitask learning for neural network models are employed in phoneme recognition for speech (Seltzer and Droppo, 2013) and speech synthesis (Wu et al., 2015) as well, but both of them use equal weights for all tasks.

6 Conclusion

In this paper, we address an open domain rare event detection problem, specifically, name error detection on ASR hypotheses. To alleviate the data skewness and domain mismatch problems, we adopt a factored approach (sentence-level name prediction and OOV error prediction) and propose an MT RNN for sentence-level name prediction. The factored model is shown to be more robust to the sparse training problem. For the problem of sentence-level name prediction, the proposed method of combining the language modeling and sentence-level name prediction objectives in an MT RNN achieves the best results among studied models for the domain represented by the training data as well as in the open-domain scenario. Visualization of sentence-level embeddings show how both the multi-task and the word-level name label update are important for achieving good results. The use of unrelated external training text (which can only be used in sentence-level name prediction) improves all models, particularly for the highly-mismatched general domain data.

The improvement in performance associated with using the external text is much smaller on the word-level name error detection task than on the sentence-level name prediction. This seems to be due to the high weight learned for the word-level posterior. For future work, it is worthwhile looking into whether continuous word and sentence representations can be combined in the name error detector to achieve further improvement.

In this work, we proposed a model for learning sentence representations that might be useful for other sentence classification tasks, such as review and opinion polarity detection, question type classification and so on. As discussed in Section 5, there are other models that have been found useful for obtaining continuous sentence embeddings. It would be of interest to investigate whether other structures are more or less sensitive to data skew and/or useful for incorporating multi-domain training data.

Acknowledgments

The authors would like to thank Alex Marin, Ji He, Wen Wang and all reviewers for useful discussion and comment. This material is based on work supported by DARPA under Contract No. HR0011-12-C-0016 (subcontract 27-001389). Any opinions, findings and conclusions or recommendations expressed herein are those of the authors and do not necessarily reflect the views of DARPA.

References

N.F. Ayan, A Mandal, M. Frandsen, Jing Zheng, P. Blasco, A Kathol, F. Bechet, B. Favre, A Marin, T. Kwiatkowski, M. Ostendorf, L. Zettlemoyer, P. Salletmayr, J. Hirschberg, and S. Stoyanchev. 2013. "Can you give me another word for hyperbaric?" Improving speech translation using targeted clarification questions. In *Proc. ICASSP*, pages 8391–8395.

Rich Caruana. 1997. Multitask learning. *Machine Learning*, 28, July.

Wei Chen, Sankaranarayanan Ananthakrishnan, Rohit Prasad, and Prem Natarajan. 2013. Variable-span out-of-vocabulary named entity detection. In *Proc. Interspeech*, pages 3761–3765.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proc. ICML*.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Machine Learning Research*, 12.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proc. ACL*.

Ji He, Alex Marin, and Mari Ostendorf. 2014. Effective data-driven feature learning for detecting name errors in automatic speech recognition. In *Proc. SLT*.

Christopher Manning Jeffery Pennington, Richard Socher. 2014. Glove: Global vectors for word representations. In *Proc. EMNLP*.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proc. ACL*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proc. EMNLP*.

Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent convolutional neural networks for text classification. In *Proc. AAAI*.

Xiaodong Liu, Jianfeng Gao, Xiaodong He, Li Deng, Kevin Duh, and Ye-Yi Wang. 2015. Representation learning using multi-task deep neural networks for semantic classification and information retrieval. In *Proc. NAACL*.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proc. ACL*.

Alex Marin, Mari Ostendorf, and Ji He. 2015. Learning phrase patterns for asr error detection using semantic similarity. In *Proc. Interspeech*.

Alex Marin. 2015. *Effective use of cross-domain parsing in automatic speech recognition and error detection*. Ph.D. thesis, University of Washington.

Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černocký, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Proc. Interspeech*.

T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. 2013a. Distributed representations of words and phrases and their compositionality. In *Proc. NIPS*.

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

Tomáš Mikolov, Wen tau Yih, and Geoffrey Zweig. 2013c. Linguistic regularities in continuous space word representations. In *Proc. NAACL-HLT*.

Andriy Mnih and Yee Whye Teh. 2012. A fast and simple algorithm for training neural probabilistic language models. In *Proc. ICML*.

David Palmer and Mari Ostendorf. 2005. Improving out-of-vocabulary name resolution. *Computer Speech and language*, 19(1):107–128.

Carolina Parada, Mark Dredze, and Frederick Jelinek. 2011. OOV sensitive named-entity recognition in speech. In *Proc. Interspeech*, pages 2085–2088.

Michael L. Seltzer and Jasha Droppo. 2013. Multitask learning in deep neural networks for improved phoneme recognition. In *Proc. ICASSP*.

Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. 2012. Semantic compositionality through recursive matrix-vector space. In *Proc. EMNLP-CoNLL*.

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

Katsuhito Sudoh, Hajime Tsukada, and Hideki Isozaki. 2006. Incorporating speech recognition confidence into discriminative named entity recognition of speech. In *Proc. ACL*.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proc. ACL*.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Machine Learning Research*, 9, November.

Zhizheng Wu, Cassia Valentini-Botinhao, Oliver Watts, and Simon King. 2015. Deep neural networks employing multi-task learning and stacked bottleneck features for speech synthesis. In *Proc. ICASSP*.

Extracting Relations between Non-Standard Entities using Distant Supervision and Imitation Learning

Isabelle Augenstein* **Andreas Vlachos#** **Diana Maynard***
* Department of Computer Science, University of Sheffield
i.augenstein@sheffield.ac.uk, d.maynard@sheffield.ac.uk
Computer Science Department, University College London
a.vlachos@cs.ucl.ac.uk

Abstract

Distantly supervised approaches have become popular in recent years as they allow training relation extractors without text-bound annotation, using instead known relations from a knowledge base and a large textual corpus from an appropriate domain. While state of the art distant supervision approaches use off-the-shelf named entity recognition and classification (*NERC*) systems to identify relation arguments, discrepancies in domain or genre between the data used for NERC training and the intended domain for the relation extractor can lead to low performance. This is particularly problematic for "non-standard" named entities such as album which would fall into the MISC category. We propose to ameliorate this issue by jointly training the named entity classifier and the relation extractor using imitation learning which reduces structured prediction learning to classification learning. We further experiment with Web features different features and compare against using two off-the-shelf supervised NERC systems, Stanford NER and FIGER, for named entity classification. Our experiments show that imitation learning improves average precision by 4 points over an one-stage classification model, while removing Web features results in a 6 points reduction. Compared to using FIGER and Stanford NER, average precision is 10 points and 19 points higher with our imitation learning approach.

1 Introduction

Factual answers to queries such as "What albums did The Beatles release?" are commonly stored in knowledge bases and can then be accessed by an information retrieval system, a commercial example for this being Google's knowledge vault (Xin et al., 2014). In order to keep knowledge bases up to date should new facts emerge, and to quickly adapt to new domains, there is a need for flexible and accurate information extraction (*IE*) approaches which do not require manual effort to be developed for new domains. A popular approach for creating IE methods to extract such relations is distant supervision (Craven and Kumlien, 1999; Mintz et al., 2009) which is a method for learning relation extractors using relations stored in a knowledge base combined with raw text to automatically generate training data.

An important first step in distant supervision is to identify named entities (*NEs*) and their types to determine if a pair of NEs is a suitable candidate for the relation. As an example, the album relation has a Musical Artist and an Album as arguments. Existing works use supervised named entity recognisers and classifiers (*NERC*) with either a small set of types such as the Stanford NER system (Manning et al., 2014), or fine-grained NE types (Ling and Weld, 2012; Liu et al., 2014). However, supervised NERCs typically focus on recognising persons, locations and organisations and perform poorly for other types of NEs, e.g. we find that Stanford NER only recognises 43% of all MISC NEs in our corpus. In addition, they do not always perform well if they are trained on a different type of text or for a different domain (Derczynski et al., 2015). This issue becomes more important as focus is shifting from using curated text collections such as Wikipedia to texts collected from the Web via search queries (Web-based distant supervision) which can provide better coverage (West et al., 2014).

In order to ameliorate this issue, we propose to recognise NEs with simple heuristics, then use the imitation learning algorithm DAGGER (Ross et al.,

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 747–757,
Lisbon, Portugal, 17-21 September 2015.

2011) to learn the NEC component jointly with relation extraction (*RE*), without requiring explicitly labeled data for NERC. Instead, training signal is obtained by assessing the predictions of the relation extraction component. In this paper we make the following contributions:

1. We learn jointly training a named entity classifier and a relation extractor for Web-based distant supervision. Our method does not rely on hand-labeled training data and is applicable to any domain, which is shown in our evaluation on 18 different relations.
2. We compare different methods for this purpose: (1) we use imitation learning to train separate classifiers for NEC and RE jointly; (2) we aggregate NEC features and RE features and train a one-stage classification model; (3) we train a one-stage classification model with only RE features; (4) we classify NEs with two supervised off-the-shelf NEC systems (Stanford NER and FIGER) and use the NE types as features in RE to achieve a soft NE type constraint.
3. We explore the effects of using different NEC and RE features, including Web features such as links and lists on Web pages, and show that Web-based features improve average precision by 7 points. We further find that high-precision, but low-frequency features perform better than low-precision and high-frequency features.
4. Our experiments show that joint learning of NEC and RE with imitation learning outperforms one-stage classification models by 4 points in average precision, and models based on Stanford NER and FIGER by 19 and 10 points respectively.

2 Distant Supervision

Distantly supervised RE is defined as automatically labelling a corpus with properties, P, and resources, R, where resources stand for entities from a knowledge base, KB, to train a classifier to learn to predict binary relations. The distant supervision paradigm is defined as (Mintz et al., 2009):

> If two entities participate in a relation, any sentence that contains those two entities might express that relation.

In general relations are of the form $(s, p, o) \in R \times P \times R$, consisting of a subject, a predicate and an object; during training, we only consider statements which are contained in a knowledge base, i.e. $(s, p, o) \in KB \subset R \times P \times R$. In any single extraction we consider only those subjects in a particular class $C \subset R$, i.e. $(s, p, o) \in KB \cap C \times P \times R$. Each resource $r \in R$ has a set of lexicalisations, $L_r \subset L$. Lexicalisations are retrieved from the KB, where they are represented as the name or alias, i.e. less frequent name of a resource.

3 Approach Overview

The input to the approach is a KB which contains entities and is partly populated with relations, the task is to complete the knowledge base. As an example, consider a KB about musical artists and their albums, which contains names of musical artists, and albums for some of them. The task is then to find albums for the remaining musical artists. Queries are automatically formulated containing C, s and o, e.g. "Musical Artist album 'The Beatles"' and we obtain Web pages using a search engine. For each sentence on the Web pages retrieved which contains s, all candidates for C are identified using NER heuristics (Section 4.2). Next, the distant supervision assumption is applied to all such sentences containing s (e.g. "Michael Jackson") and a candidate for that relation (e.g. "Music & Me"). If the candidate is an example of a relation according to the KB, it is used as a positive example, and if not, as a negative example. The examples are then used to train a model to recognise if the candidate is of the right type for the relation (NEC) and if it is of the correct relation (RE). The model is applied to the sentences of all the incomplete entries in the KB. Since different sentences could predict different answers to the query, all predictions are combined for the final answer.

4 Named Entity Recognition and Relation Extraction

The input to the learning task is a collection of training examples for a specific relation. The examples are sentences containing the subject of the relation and one further NE identified using simple heuristics. The examples are labeled as true (relation is contained in knowledge base) or as false (relation is not contained in the knowledge base).

We model the task in two binary classifica-

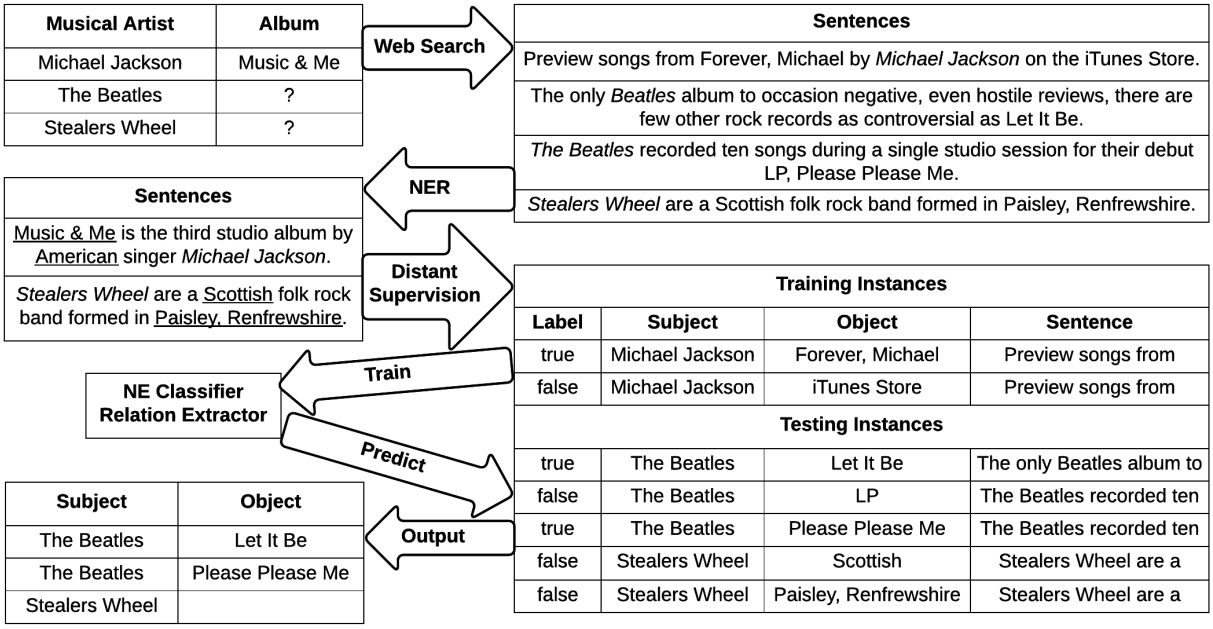


Figure 1: Overview of approach

tion stages: named entity classification (NEC) and relation extraction (RE). Existing approaches assume that named entity recognition and classification is done as part of the pre-processing. However, this is not possible domains for which NE classifiers are not readily available. To ameliorate this issue, existing approaches — e.g Mintz et al. (2009) — perform NEC to provide additional features for relation extraction. We use two such baselines with off-the-shelf NECs and add the NE labels to the relation features. The first baseline (**Stanf**) is with the Stanford NER 7-class (Time, Location, Organization, Person, Money, Percent and Date) model, the second (**FIGER**) is with the fine-grained FIGER (Ling and Weld, 2012).

An alternative approach is to simply add NEC features to relation extraction features, which we call **one-stage model (OS)** here. NEC features are typically morphological features extracted from the NE mention and features to model its context, whereas relation features typically model the path between the subject and object of the relation. While NEC features may be useful to determine if the NE has the correct type for the relation, such features are usually less sparse and also not directly related to the relation extraction task. Consider the following sentence, containing an example of the relation director:

"One of director <o>Steven Spielberg</o>'s greatest heroes was <o>Alfred Hitchcock</o>, the mastermind behind <s>Psycho</s>.

This sentence contains two relation candidates, "Steven Spielberg" and "Alfred Hitchcock", between which the decision for the final prediction has to be made. Both of the candidates are directors, but only one of them is the director of "Psycho". Because the context around "Steven Spielberg" is stronger (preceded by "director"), NEC features alone are more likely to indicate that as the correct candidate and also likely overpower relation features for the final prediction, as the latter tend to be sparser.

Ideally, we would like to train two models, one for NEC and one for RE, which would be applied in sequence. If the NEC stage concludes that the candidate is of the correct type for the relation, the RE stage determines whether the relation between the two entities is expressed. If the NEC stage concludes that the entity is not of the correct type, then the RE stage is not reached. However, distant supervision only provides positive labels for NEC, since if a sentence is labeled as false we do not know if it is due to the candidate not being of the correct type, or the relation not being true for the two entities. To overcome this, we learn models for the two stages, NEC and RE, jointly using the imitation learning algorithm DAGGER (Ross et al., 2011), as described in the next section.

4.1 Imitation Learning

Imitation learning[1] algorithms such as SEARN (Daumé III et al., 2009) and DAGGER (Ross et al., 2011) have been applied successfully to a variety of structured prediction tasks due to their flexibility in incorporating features and their ability to learn with non-decomposable loss functions. Sample applications include biomedical event extraction (Vlachos and Craven, 2011), dynamic feature selection (He et al., 2013), and machine translation (Grissom II et al., 2014).

Imitation learning algorithms for structured prediction decompose the prediction task into a sequence of actions; these actions are predicted by classifiers which are trained to take into account the effect of their predictions on the whole sequence by assessing their effect using a (possibly non-decomposable) loss function on the complete structure predicted. The dependencies between the actions are learnt via appropriate generation of training examples.

The ability to learn by assessing only the final prediction and not the intermediate steps is very useful in the face of missing labels, such as in the case of the labels for the NEC stage. Recall that the action sequence in our case consists of one NEC action and possibly one RE action, dependent on whether the NEC action is true, i.e. the entity is of the appropriate type for the relation. Following Vlachos and Clark (2014), for each training instance, we obtain supervision for the NEC stage by taking both options for this stage, true or false, obtaining the prediction from the RE stage in the former case and then comparing the outcomes against the label obtained from distant supervision. Thus the NEC stage is learned so that it enhances the performance of RE. In parallel, the RE stage is learned using only instances that actually reach this stage. The process is iterated so that the models learned adjust to each other. For more details on this we refer the reader to Vlachos and Clark (2014).

4.2 Relation Candidate Identification

To extract relations among NEs, the latter have to be detected first. Most distantly supervised approaches use supervised NER systems for this, which, especially for relations involving MISC NEs, achieve a low recall. High recall for NE identification is more important than high precision, since precision errors can be dealt with by the NEC stage. For a relation candidate identification stage with higher recall we instead rely on POS-based heuristics for detecting NEs[2] and HTML markup. We use the following POS heuristics:

- **Noun phrases:** Sequences of *N* tags
- **Capitalised phrases:** Those can be distinct from noun phrases, e.g. some album titles are capitalised verb phrases.

We further consider as relation candidates words which contain the following HTML markup:

- **Phrases from HTML markup:** All sequences of words marked as: `<ahref>` (links), `<li>` (list elements), `<h1>` or `<h2>` or `<h3>` (headers and subheaders, i.e. titles), `<strong>` or `<b>` (bold), `<em>` (emphasised), `<i>` (italics)

Different relation candidate identification strategies are then applied depending on the coarse NE types of objects of relation as defined in the KB (Table 2).

- **PER**: All capitalised noun phrases. We allow for a maximum of two characters to be surrounded by quotes to capture alternative first names, e.g. "Jerome David 'J. D.' Salinger".
- **LOC**: All capitalised noun phrases.
- **ORG**: All capitalised phrases and phrases from HTML markup. The latter is to capture ORG names which are not capitalised, e.g. the school "Woodrow Wilson School of Public and International Affairs" or the record label "Sympathy for the Record Industry".
- **MISC**: As for ORG, we use all capitalised phrases and phrases from HTML markup. MISC NEs are even more varied than ORG NEs and it is difficult to find the right balance between recognising most of them and generating unecessary candidates.

To assess how useful these strategies are, we randomly sample 30 instances of each Freebase class per coarse NE type of the object and manually examine all sentences which contain the subject of the relation. We used precision, i.e. how many of the relation candidates are appropriate

[1] Also referred to as search-based structured prediction or learning to search.

[2] The Stanford POS tagger uses Penn Treebank POS tags, see `http://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html` for a list of tags

NE type	Model	R	P	F1
PER	heuristic	0.976	0.1287	0.227
PER	Stanford	0.774	0.1781	0.29
LOC	heuristic	0.963	0.1176	0.21
LOC	Stanford	0.889	0.1611	0.272727273
ORG	heuristic	0.95	0.0265	0.0516
ORG	Stanford	0.8	0.0505	0.095
MISC	heuristic	0.854	0.0496	0.0938
MISC	Stanford	0.427	0.053	0.0943

Table 1: Results for POS-based candidate identification strategies compared to Stanford NER

for the relation, and recall to compare the relation candidate identification strategies described above against the identification of candidates by Stanford NER (ignoring the NE label). As shown in Table 1, while supervised identification of NE labels achieves a higher precision for all NE types, the recall is higher for all NE types using POS-based heuristics. The simple heuristics are especially helpful for MISC NEs, for which recall is twice as high compared to Stanford NER and precision only marginally higher. If we were to use the NE label to enforce hard constraints, recall would be reduced even further: 88% of all PER entities are correctly identified as persons, compared to 58% for locations and 87% for organisations. MISC NE are identified as PER (45%), LOC (40%) or ORG (15%). Overall, precision is not as important for candidate identification as recall, since choosing correct entities among the candidates can be dealt with in a NEC stage.

4.3 NEC features

For the one-stage and imitation learning model, we use the following **Web features** based on HTML markup, both as local features if the entity mention contains the markup, and as global features if a mention somewhere else in the document with the same lexicalisation contains that markup: is link, is list element, is header or subheader, is bold, is emphasised, is italics, is title, is in title.

In addition, the following NEC features are extracted, based on Nadeau et al. (2007) and Hoffmann et al. (2011):
Word features (mentfeats):

- Object occurrence
- Sequence and BOW of occurrence
- Sequence and bag of POS of occurrence
- Number of words, characters and digits of object
- Ends with period, is roman number, contains apostrophe, hyphen, ampersand, possessive
- Digit and capitalisation pattern

Context features, as 1-grams (1cont) and 2-grams, 2 words to left and right of occurrence (2cont): BOW, sequence, bag of POS, POS sequence.

4.4 RE Features

The following features are used for RE, based on Hoffman et al (2011) and Mintz et al. (2009):

- 1cont and 2cont features
- Flag indicating which entity came first in sentence
- Sequence of POS tags and bag of words (BOW) between the subject and the object occurrence

Parsing features as full sequences (parse):

- Dependency path between subject and object, POS tags of words on that path
- Lemmas on dependency path, same with NNP and CD tokens substituted by POS tags

4.5 Supervised NEC Features for RE

For the baselines with off-the-shelf NECs, sentences are preprocessed with the two NEC systems Stanford NER and FIGER. NE labels are then used in addition to the RE features listed in Section 4.4. For the Stanf baseline, Stanford NER 7-class labels are added as RE features. Those are: Time, Location, Organization, Person, Money, Percent, Date. FIGER classifies NEs according to 112 types, most of which are subtypes of Person, Organization, Location, Product, Art, Event and Building. Some of the types are relation types we evaluate (see Table 2 for relation types): educational institution, city, director, actor and author. Since FIGER performs multi-label classification, it annotates some of the relation candidates with more than one NE label. In that case, we add all NE labels returned as features, though more experiments on how best to integrate multiple NE labels as features could be performed, as shown by Liu et al. (2014).

Musical Artist		Politician	
Relation type	**NE type**	**Relation type**	**NE type**
album	MISC	birthplace	LOC
record label	ORG	educational institution	ORG
track	MISC	spouse	PER
Business		**Educational Institution**	
Relation type	**NE type**	**Relation type**	**NE type**
employees	PER	mascot	MISC
founders	PER	city	LOC
Film		**Book**	
Relation type	**NE type**	**Relation type**	**NE type**
director	PER	author	PER
producer	PER	characters	MISC
actor	PER		
character	MISC		
River			
Relation type	**NE type**		
origin	LOC		
mouth	LOC		

Table 2: Relation types and corresponding coarse NE types

5 Evaluation

5.1 Corpus

To create a corpus[3] for Web RE, seven Freebase classes and two to four of their relations are selected (Table 2). The selected classes are subclasses of PER (Musical Artist, Politician), LOC (River), ORG (Business (Operation)), Education(al Institution)) or MISC (Film, Book). To avoid noisy training data, we only use entities which have values for all of those properties, which resulted in 1800 to 2200 entities per class. For each entity, 10 Web pages were retrieved via the Google Search API using the search pattern "'*subject_entity*" *class_name relation_name*', e.g. "'The Beatles" Musical Artist Origin'. In total, the corpus consists of around one million pages drawn from 76,000 different websites. Text content is extracted from HTML pages using the Jsoup API[4] and processed with Stanford CoreNLP[5].

5.2 Models and Metrics

We evaluate the following models: imitation learning (**IL**) as described in Section 4.1, a one-stage model (**OS**), a one-stage model with relation features only (**RelOnly**), and using Stanford (**Stanf**) and FIGER (**FIGER**) NE labels as features (Section 4). For all models we use linear classifiers learned with passive-aggressive updates (Crammer et al., 2006). For imitation learning, we use the learning algorithm DAGGER (Ross et al., 2011), which requires two parameters: the learning rate, i.e. how quickly the learning algorithm moves away from the optimal policy, and the number of iterations. We found empirically that the best learning rate for our prediction task is 0.25 and that the best number of iterations is 12.

[3]The resources for experiments documented in this paper are available online via `http://tinyurl.com/o8ykn4y`

[4]http://jsoup.org

[5]http://nlp.stanford.edu/software/corenlp.shtml

Model	R-top	P-top	F1-top	R-all	P-all	P-avg
RelOnly	0.1943	0.404	0.255	0.223	0.309	0.373
Stanf	0.233	0.436	0.304	0.268	0.329	0.398
FIGER	0.228	0.497	0.298	0.251	0.413	0.483
OS	**0.269**	0.58	**0.356**	**0.288**	0.486	0.552
IL	0.246	**0.600**	0.329	0.271	**0.521**	**0.588**

Table 3: Results for best model for each relation, macro average over all relations.

The output of the models is a score for each relation example and stage, i.e. for the one-stage model, the output is one score and for the imitation learning model, there is a score each for the NEC stage and the RE stage. The default for deciding whether the relation label should be true or false depends on stage thresholds, which are 0 by default. Instead of using the default thresholds, we automatically pick thresholds for all models on 1/3 of the training set, which we set aside as a development set, then retrain on the whole training set and predict relations based on the learnt thresholds.

We use the metrics first best precision (P-top), first best recall (R-top), first best F1 (F1-top), all precision (P-all), all recall (P-all), and all average precision (P-avg). For top, only the top-ranked answer is considered, whereas for all all answers are returned until either the correct one is found or they are exhausted. Finally, in the all mode we evaluated precision at all recall points by varying the thresholds used in the respective classifiers and we report average precision (P-avg) (Manning et al., 2008). This evaluation measure provides an assessment of how well a system trades precision for recall. The number of all results for computing recall is the number of all relation tuples in the KB.

Relation	RelOnly		Stanf		FIGER		OS		IL	
	F1-top	P-avg	F1-top	P-avg	F1-top	P-avg	F1-top	P-avg	F1-top	P-avg
Musical Artist : album	0.071	0.175	0.079	0.109	0.116	0.203	0.158	0.409	0.115	**0.569**
Musical Artist : record label	0.090	0.182	0.100	0.345	0.179	0.636	0.404	0.758	0.376	**0.926**
Musical Artist : track	0.093	0.109	0.053	0.175	0.104	0.400	0.118	**0.471**	0.114	0.367
Politician : birthplace	0.410	0.594	0.514	0.541	0.496	0.609	0.585	**0.709**	0.516	0.548
Politician : educational institution	0.321	0.387	0.330	0.426	0.366	0.560	0.419	0.719	0.381	**0.831**
Politician : spouse	0.148	0.197	0.152	0.197	0.082	0.309	0.218	**0.319**	0.150	0.181
Business : employees	0.059	0.090	0.097	0.153	0.082	0.325	0.149	0.291	0.133	**0.493**
Business : founders	0.341	0.256	0.462	0.332	0.404	0.542	0.448	0.663	0.429	**0.693**
Education : mascot	0.148	0.362	0.195	0.483	0.226	0.500	0.225	0.506	0.206	**0.585**
Education : city	0.630	0.705	0.711	0.740	0.701	0.770	0.724	0.847	0.690	**0.872**
Film : director	0.383	0.548	0.445	0.603	0.358	0.554	0.439	0.601	0.387	**0.612**
Film : producer	0.149	0.384	0.209	0.395	0.164	0.387	0.198	0.355	0.227	**0.400**
Film : actor	0.246	0.576	0.308	0.633	0.351	0.609	0.342	0.684	0.312	**0.732**
Film : character	0.093	0.123	0.093	0.117	0.180	0.195	0.194	0.298	0.173	**0.319**
Book : author	0.629	0.852	0.703	0.852	0.781	0.878	0.773	0.867	0.781	**0.885**
Book : characters	0.224	0.127	0.193	0.127	0.262	0.328	0.268	0.315	0.231	**0.355**
River : origin	0.175	0.328	0.232	0.493	0.160	0.351	0.256	0.406	0.228	**0.550**
River : mouth	0.336	0.594	0.423	0.564	0.347	0.529	0.488	**0.709**	0.479	0.668

Table 4: Results for best model for each relation, highest P-avg in bold

6 Results and Discussion

6.1 Comparison of Models

Overall results in Table 3 show that both of our models (**IL** and **OS**) outperform the baselines with off-the-shelf supervised NEC (**Stanf**, **FIGER**) for all metrics. Detailed results for different relations (Table 4) show that IL outperforms both OS and Base in terms of average precision. FIGER results fall in between Stanf and OS results. For some relations, there is a dramatic improvement by using fine-grained FIGER NE features over coarse-grained Stanford NE features; occasionally FIGER even outperforms OS, as for the relation author. This is because FIGER has a corresponding NE type (see Section 4.5).

For most relations, including those whose objects are of type MISC, IL shows a significant improvement in terms of F1 or average precision over OS (Table 5). This confirms our hypothesis that separating the NEC and relation extraction stages using imitation learning can achieve a higher precision and recall for non-standard relations than preprocessing sentences with a supervised NEC model. Furthermore, we show that it can also be useful for most standard relations. The main relations for which Stanf, FIGER or OS can have a benefit over IL are those for which entities are easy to classify, specifically LOC NEs, but also PER NEs. This is because, if NEs are easy to classify, a separate NEC is less likely to be useful.

6.2 Imitation Learning vs One-Stage

To give more insight into why **IL** is overall more successful than **OS**, common errors made by OS are shown here, along with an explanation of how those errors are prevented by using IL. One example of IL predicting correctly but OS incorrectly is from the following sentence, expressing the director relation:

"In 2010 he appeared in a leading role in <o>Alicia Duffy</o>'s <s>All Good Children</s>.

In that example, the NEC features extracted for <o>Alicia Duffy</o> are not very strong indicators, since neither the object string itself nor the surrounding context give any direct indication for the director relation. The RE features, which are based on the dependency path, are a stronger indicator. Since in the OS model all features are combined, the NEC features overpower the RE features. The IL model, on the other hand, learns a permissive NEC as a first stage, which filters NEs with respect to if they are generally appropriate for the relation or not, and then leaves the RE to the second stage.

Another example is a sentence for which OS incorrectly predicts the relation author, whereas IL correctly predicts "false":

"<o>Laura</o> and Mary went to school for the first time in Pepin rather than Walnut Grove, which is not included in <s>Little House in the Big Woods</s>."

Relation	NEC Features	Rel Features
Musical Artist : album	2cont + 1cont + mentfeats + web	parse
Musical Artist : record label	2cont + 1cont + mentfeats + web	parse + 2contword
Musical Artist : track	parse + 2cont + 1cont + mentfeats	parse
Politician : birthplace	2cont + 1cont + mentfeats + web	parse
Politician : educational institution	parse + cont + ment	parse
Politician : spouse	parse + 2cont + 1cont + web	parse
Business : employees	2cont + 1cont + mentfeats + web	parse + 2contword
Business : founders	parse + cont + ment	parse
Education : mascot	parse + 2contwordpos	parse + cont
Education : city	parse + cont + ment	parse + 2contwordpos
Film : director	2cont + 1cont + mentfeats + web	parse + 2contword
Film : producer	parse + cont	parse + 2contwordpos
Film : actor	parse + 2cont + web	parse + 2contwordpos
Film : character	parse + cont + ment	parse + 2contword
Book : author	2cont + 1cont + mentfeats + web	parse
Book : characters	parse + cont + ment	parse
River : origin	2cont + 1cont + mentfeats + web	parse + 2contword
River : mouth	2cont + 1cont + mentfeats + web	parse

Table 5: Best feature combination for IL

NEC Features	Rel Features	P-top	R-top	F1-top	P-all	R-all	P-avg
2cont	parse	0.215	0.399	0.28	0.253	0.316	0.381
2cont + 1cont + mentfeats	parse	0.239	0.456	0.313	0.275	0.378	0.441
2cont + 1cont + mentfeats + web	parse	0.248	0.51	0.322	0.276	0.431	0.502
2cont + web	parse	0.204	0.375	0.264	0.244	0.289	0.35
2cont	parse + 2contwordpos	0.236	0.43	0.305	0.275	0.338	0.402
2cont + 1cont + mentfeats	parse + 2contwordpos	0.239	0.456	0.313	0.275	0.378	0.441
2cont + 1cont + mentfeats + web	parse + 2contwordpos	0.248	0.518	0.324	0.275	0.421	0.486
2cont + web	parse + 2contwordpos	0.24	0.402	0.3	0.279	0.305	0.371
2cont	parse + 2contword	0.215	0.394	0.278	0.258	0.309	0.372
2cont + 1cont + mentfeats	parse + 2contword	0.231	0.453	0.295	0.266	0.352	0.43
2cont + 1cont + mentfeats + web	parse + 2contword	0.25	0.54	0.325	0.284	0.433	0.505
2cont + web	parse + 2contword	0.223	0.395	0.285	0.263	0.305	0.373

Table 6: Imitation learning results for different NE and relation features, macro average over all relations.

For this example, OS relation features have small positive weights, which then overall lead to a positive prediction. For IL, the first stage predicts "false", since the one-token string <o>Laura</o> is not a likely candidate for author.

6.3 Comparison of Features

All different feature groups have an overall positive effect on the results (see Table 6). While low precision, high frequency features improve recall (1cont), they do not always improve precision. Both **OS** and **IL** benefit from high precision, low frequency features, e.g. for author and mouth, the best results are achieved with only sparse parsing features for RE.

Web features improve performance for 10 out of 18 relations. For n-ary relations the is list element feature is very useful because Web pages about musical artist, films or books often contain lists with their attributes, e.g. a Web page about a musical artist typically contains a list with their albums. For relations with persons as objects, is link and is bold is useful because Web pages often highlight persons or provide links to Web pages with more information about them. As an example, for the author relation, the strongest positive Web feature is is in title and the strongest negative feature is is list element. This makes sense since a book is frequently mentioned with its author and one of the most important attributes of a book, whereas lists on Web pages about books mention less important attributes, such as the characters.

6.4 Overall Comparison

Overall, we showed that using an off-the-shelf NEC as a pre-processing step for distant super-

vision as done by existing works often causes errors which can be prevented by instead separating NEC and RE with imitation learning. We also showed that using Web features increases precision for NEC. Finally, it is worth noting that the recall for some of the relations is quite low because they only infrequently occur in text, especially in the same sentence as the subject of the relation. These issues can be overcome by performing coreference resolution (Augenstein et al., 2014; Koch et al., 2014), by retrieving more Web pages or improving the information retrieval component of the approach (West et al., 2014) and by combining extractors operating on sentences with other extractors for semi-structured content on Web pages (Carlson et al., 2010).

7 Related Work

One of the first papers to introduce distant supervision was Mintz et al. (2009), which aims at extracting relations between entities in Wikipedia for the most frequent relations in Freebase. Most distant supervision research focuses on addressing the disadvantages of heuristic labelling, namely reducing false positive training data (Hoffmann et al., 2011; Surdeanu et al., 2012; Riedel et al., 2010; Riedel et al., 2013; Yao et al., 2010; Alfonseca et al., 2012; Roth and Klakow, 2013; Takamatsu et al., 2012; Xu et al., 2013) and dealing with false negatives due to missing entries in the knowledge base (Min et al., 2013), as well as combining distant supervision with active learning (Angeli et al., 2014)

Distant supervision has been researched for different domains, including newswire (Riedel et al., 2010; Riedel et al., 2013), Wikipedia (Mintz et al., 2009; Nguyen and Moschitti, 2011), the biomedical domain (Craven and Kumlien, 1999; Roller and Stevenson, 2014), the architecture domain (Vlachos and Clark, 2014) and the Web (Xin et al., 2014; Augenstein et al., 2014; Augenstein et al., 2015).

To date, there is very little research on improving NERC for distant supervision to extract relations between non-standard entities such as musical artists and albums. Some research has been done on improving distant supervision by using fine-grained named entity classifiers (Ling and Weld, 2012; Liu et al., 2014) and on using named entity linking for distant supervision (Koch et al., 2014). Liu et al. (2014) train a supervised fine-grained NERC on Wikipedia and show that using those types as entity contraints improves precision and recall for a distantly supervised RE on newswire. However, they assume that labeled training data is available, making it unsuitable for applying distant supervision to domains with relations involving non-standard entity types.

Vlachos and Clark (2014) also proposed a distantly supervised approach for joint learning of NEC and RE with imitation learning for the architecture domain. However, they only used two relations in their experiments which involved rather standard entity types and they did not compare against using off-the shelf NEC systems.

8 Conclusion and Future Work

In this paper, we proposed a method for extracting non-standard relations with distant supervision that learns a NEC jointly with relation extraction using imitation learning. Our proposed imitation learning approach outperforms models with supervised NEC for relations involving non-standard entities as well as relations involving persons, locations and organisations. We achieve an increase of 4 points in average precision over a simple one-stage classification model, and an increase in 10 points and 19 points over baselines with FIGER and Stanford NE labels. We further demonstrate that using specialised Web features, such as appearances of entities in lists and links to other Web pages, improves average precision by 7 points, which other Web search-based relation extraction approaches could also benefit from (Xin et al., 2014; Augenstein et al., 2014).

In future work, the proposed approach could be combined with other approaches to solve typical issues arising in the context of distant supervision, such as dealing with overlapping relations (Hoffmann et al., 2011), improving heuristic labelling of sentences (Takamatsu et al., 2012) or dealing with incomplete knowledge bases (Min et al., 2013).

References

Enrique Alfonseca, Katja Filippova, Jean-Yves Delort, and Guillermo Garrido. 2012. Pattern Learning for Relation Extraction with a Hierarchical Topic Model. In *Proceedings of ACL*, volume 2, pages 54–59, Jeju, South Korea. ACL.

Gabor Angeli, Julie Tibshirani, Jean Wu, and Christopher D. Manning. 2014. Combining distant and par-

tial supervision for relation extraction. In *Proceedings of EMNLP*, pages 1556–1567, Doha, Qatar. ACL.

Isabelle Augenstein, Diana Maynard, and Fabio Ciravegna. 2014. Relation Extraction from the Web using Distant Supervision. In *Proceedings of EKAW*, pages 26–41. Springer.

Isabelle Augenstein, Diana Maynard, and Fabio Ciravegna. 2015. Distantly Supervised Web Relation Extraction for Knowledge Base Population. *Semantic Web Journal*. to appear.

Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R. Hruschka, and Tom M. Mitchell. 2010. Toward an Architecture for Never-Ending Language Learning. In *Proceedings of AAAI*, Palo Alto, California, USA. AAAI Press.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7:551–585.

M. Craven and J. Kumlien. 1999. Constructing Biological Knowledge Bases by Extracting Information from Text Sources. In *Proceedings of ISMB*, pages 77–86, Palo Alto, California, USA. AAAI Press.

Hal Daumé III, John Langford, and Daniel Marcu. 2009. Search-based Structured Prediction. *Machine Learning*, 75:297–325.

Leon Derczynski, Diana Maynard, Giuseppe Rizzo, Marieke van Erp, Genevieve Gorrell, Raphaël Troncy, and Kalina Bontcheva. 2015. Analysis of Named Entity Recognition and Linking for Tweets. *Information Processing and Management*, 51:32–49.

Alvin Grissom II, Jordan Boyd-Graber, He He, John Morgan, and Hal Daumé III. 2014. Don't Until the Final Verb Wait: Reinforcement Learning for Simultaneous Machine Translation. In *Proceedings of EMNLP*, Doha, Qatar. ACL.

He He, Hal Daumé III, and Jason Eisner. 2013. Dynamic Feature Selection for Dependency Parsing. In *Proceedings of EMNLP*, pages 1455–1464, Seattle, Washington, USA. ACL.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke S. Zettlemoyer, and Daniel S. Weld. 2011. Knowledge-Based Weak Supervision for Information Extraction of Overlapping Relations. In *Proceedings of ACL*, pages 541–550, Portland, Oregon, USA. ACL.

Mitchell Koch, John Gilmer, Stephen Soderland, and Daniel S. Weld. 2014. Type-aware distantly supervised relation extraction with linked arguments. In Alessandro Moschitti, Bo Pang, and Walter Daelemans, editors, *Proceedings of EMNLP*, pages 1891–1901, Doha, Qatar, October. Association for Computational Linguistics.

Xiao Ling and Daniel S. Weld. 2012. Fine-Grained Entity Recognition. In *Proceedings of AAAI*, pages 94–100. AAAI Press.

Yang Liu, Kang Liu, Liheng Xu, and Jun Zhao. 2014. Exploring Fine-grained Entity Type Constraints for Distantly Supervised Relation Extraction. In *Proceedigs of COLING*, pages 2107–2116, Dublin, Ireland. ACL.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schtze. 2008. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In *Proceedings of ACL: System Demonstrations*, pages 55–60, Baltimore, Maryland. ACL.

Bonan Min, Ralph Grishman, Li Wan, Chang Wang, and David Gondek. 2013. Distant Supervision for Relation Extraction with an Incomplete Knowledge Base. In *Proceedings of HLT-NAACL*, pages 777–782, Atlanta, Georgia, USA. ACL.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of ACL-IJCNLP*, volume 2, pages 1003–1011, Suntec, Singapore. ACL.

David Nadeau and Satoshi Sekine. 2007. A Survey of Named Entity Recognition and Classification. *Linguisticae Investigationes*, 30(1):3–26.

Truc-Vien T. Nguyen and Alessandro Moschitti. 2011. End-to-End Relation Extraction Using Distant Supervision from External Semantic Repositories. In *Proceedings of ACL-HLT*, pages 277–282, Portland, Oregon, USA. ACL.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling Relations and Their Mentions without Labeled Text. In *Proceedings of ECML-PKDD (3)*, volume 6323, pages 148–163. Springer.

Sebastian Riedel, Limin Yao, Benjamin M. Marlin, and Andrew McCallum. 2013. Relation Extraction with Matrix Factorization and Universal Schemas. In *Proceedings of HLT-NAACL*, pages 74–84, Atlanta, Georgia, USA. ACL.

Roland Roller and Mark Stevenson. 2014. Self-supervised Relation Extraction Using UMLS. In Evangelos Kanoulas, Mihai Lupu, Paul D. Clough, Mark Sanderson, Mark M. Hall, Allan Hanbury, and Elaine G. Toms, editors, *Proceedings of CLEF*, volume 8685 of *Lecture Notes in Computer Science*, pages 116–127. Springer.

Stéphane Ross, Geoffrey J. Gordon, and Drew Bagnell. 2011. A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning. In

Geoffrey J. Gordon, David B. Dunson, and Miroslav Dudk, editors, *Proceedings of AISTATS*, volume 15, pages 627–635. JMLR.

Benjamin Roth and Dietrich Klakow. 2013. Combining Generative and Discriminative Model Scores for Distant Supervision. In *Proceedings of EMNLP*, pages 24–29, Seattle, Washington, USA. ACL.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D. Manning. 2012. Multi-instance Multi-label Learning for Relation Extraction. In *Proceedings of EMNLP-CoNLL*, pages 455–465, Jeju Island, Korea. ACL.

Shingo Takamatsu, Issei Sato, and Hiroshi Nakagawa. 2012. Reducing Wrong Labels in Distant Supervision for Relation Extraction. In *Proceedings of ACL*, pages 721–729, Jeju Island, Korea. ACL.

Andreas Vlachos and Stephen Clark. 2014. Application-Driven Relation Extraction with Limited Distant Supervision. In *Proceedings of Aha!*, pages 1–6, Dublin, Ireland. ACL.

Andreas Vlachos and Mark Craven. 2011. Search-based structured prediction applied to biomedical event extraction. In *Proceedings of CoNLL*, pages 49–57, Portland, Oregon, USA. ACL.

Robert West, Evgeniy Gabrilovich, Kevin Murphy, Shaohua Sun, Rahul Gupta, and Dekang Lin. 2014. Knowledge Base Completion via Search-Based Question Answering. In *Proceedings of WWW*, pages 515–526, New York, NY, USA. ACM.

Luna Dong Xin, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. 2014. Knowledge Vault: A Web-scale approach to probabilistic knowledge fusion. In *Proceedings of KDD*, pages 601–610, New York, NY, USA. ACM.

Wei Xu, Raphael Hoffmann, Le Zhao, and Ralph Grishman. 2013. Filling Knowledge Base Gaps for Distant Supervision of Relation Extraction. In *Proceedings of ACL*, pages 665–670, Sofia, Bulgaria. ACL.

Limin Yao, Sebastian Riedel, and Andrew McCallum. 2010. Collective Cross-document Relation Extraction Without Labelled Data. In *Proceedings of EMNLP*, pages 1013–1023, Cambridge, MA, USA. ACL.

Sieve-Based Spatial Relation Extraction with Expanding Parse Trees

Jennifer D'Souza and **Vincent Ng**
Human Language Technology Research Institute
University of Texas at Dallas
Richardson, TX 75083-0688
{jld082000,vince}@hlt.utdallas.edu

Abstract

A key challenge introduced by the recent SpaceEval shared task on spatial relation extraction is the identification of MOVELINKs, a type of spatial relation in which up to eight spatial elements can participate. To handle the complexity of extracting MOVELINKs, we combine two ideas that have been successfully applied to information extraction tasks, namely *tree kernels* and *multi-pass sieves*, proposing the use of an expanding parse tree as a novel structured feature for training MOVELINK classifiers. Our approach yields state-of-the-art results on two key tasks in SpaceEval.

1 Introduction

Spatial relation extraction is the task of determining the relation among a set of spatial elements. Although it has thus far received much less attention than temporal relation extraction, there has been a surge of interest in it in recent years, as evidenced by the organization of the three shared tasks on spatial relation extraction, namely the spatial role labeling tasks in 2012 (Kordjamshidi et al., 2012) and 2013 (Kolomiyets et al., 2013), as well as this year's SpaceEval task (Pustejovsky et al., 2015). The task has also evolved over the years, with new types of spatial elements and/or spatial relations being defined in each shared task. For instance, while the first two shared tasks have focused on extracting spatial relations between *stationary* objects, SpaceEval examines for the first time spatial relations on objects *in motion*.

Extracting spatial relations on objects in motion, or MOVELINKs, is very challenging. The challenge stems in part from the fact that a MOVELINK involves two *mandatory* participants (with roles $mover$ and $trigger$) and up to six *optional* participants (with other semantic roles). As an example, consider the MOVELINK that can be extracted from the sentence "John walked from Boston to Cambridge". As shown in Figure 1, this MOVELINK involves six spatial elements: "John" as the $mover$, "walked" as the $trigger$, "Boston" as the $source$, "Cambridge" as the $goal$, and "from" and "to" as the $motion_signal$s.

Figure 1: A MOVELINK example.

Given the complexity of MOVELINKs, any approach that attempts to *jointly* identify all the spatial elements involved in a MOVELINK and their roles is computationally infeasible. On the other extreme, one can tackle the task by identifying each element involved in a MOVELINK *independently* of the other elements. In fact, this is roughly the approach adopted by our participating system in SpaceEval (D'Souza and Ng, 2015), which achieved the best results in one of the SpaceEval tasks involving MOVELINK extraction. Specifically, this system trains one classifier for each *optional* role r to identify the filler for r in a MOVELINK independently of the other optional roles. Although this approach has achieved state-of-the-art performance, it is arguably not ideal: intuitively, dependencies exist among elements of different roles, and not capturing them may harm system performance.

Our goal in this paper is to advance the state of the art in spatial relation extraction, focusing on the extraction of MOVELINKs by addressing the aforementioned weakness. The key question is: how can we capture the dependencies among the spatial elements involved without sacrificing computational tractability? To address this question, we combine two ideas that have been successfully applied to a variety of information extraction tasks, namely multi-pass sieves (Raghu-

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 758–768,
Lisbon, Portugal, 17-21 September 2015.

nathan et al., 2010; Lee et al., 2013) and tree kernels (Moschitti, 2004; Moschitti, 2006). Recall that a sieve-based approach is composed of a pipeline of *sieves* ordered by precision, where the decisions made by earlier sieves can be exploited by later sieves in order to incrementally construct a complex structure. When applied to MOVELINK identification, we can create a sieve for identifying each role, so that (1) spatial elements corresponding to different roles are incrementally added to a MOVELINK, and (2) earlier attachment decisions can be exploited as additional contextual information by later sieves. Hence, compared to a joint approach, a sieve-based approach achieves computational tractability by modeling *partial*, rather than *full* dependencies among the spatial elements.

While a sieve-based approach allows us to exploit additional contextual information provided by earlier sieves, we still have to specify how we encode such contextual information. Motivated by the successful application of tree kernels to relation extraction (e.g., Zelenko et al. (2003), Culotta and Sorensen (2004), Bunescu and Mooney (2005), Zhou et al. (2007)), we propose to (1) encode the *syntactic* context in which the spatial elements extracted by the sieves appear using a syntactic parse tree, and then (2) employ the tree as an (additional) structured feature for training the classifier associated with each sieve. This novel combination of sieves and tree-based structured features results in what we call an *expanding* parse tree. Specifically, as a spatial element for a MOVELINK is extracted by a (role-specific) sieve, it will be added to the structured feature for the classifier associated with the following sieve. In other words, the parse tree corresponding to the structured feature will keep expanding as we move along the sieves in the pipeline. This contrasts with previous applications of tree kernels, where a structured feature is created from a static parse subtree for extracting *exactly two* arguments involved in a relation. To our knowledge, this is the first attempt to combine sieves and parse trees to create expanding trees to extract complex relations involving multiple arguments.

2 Corpus and Task Definition

In this section, we introduce our corpus and the spatial relation extraction task. Owing to space limitations, we will only discuss those aspects that are relevant to the SpaceEval tasks we focus on.

2.1 The SpaceEval Corpus

We use as our corpus the SpaceEval training corpus, which is a subset of ISO-SpaceBank (Pustejovsky and Yocum, 2013). The corpus consists of 59 travel narratives annotated with seven types of spatial elements (Table 1) and three types of spatial relations (Table 2), following the ISO-Space (2012) annotation specifications. Different types of spatial elements have different attributes. The only attribute that is relevant to our work is *semantic type*, which is one of the attributes of a *spatial entity*. *Semantic type* expresses the type of the relation it triggers and can take one of three values: topological, directional, or both.

What is missing in Table 2 about spatial relations is that each element participating in a relation has a *role*. Each QSLINK/OLINK involves exactly three elements participating as $trajector$ (the object of interest), $landmark$ (the grounding location), and $trigger$ (the relation indicator). On the other hand, a MOVELINK has two fixed participants and up to six optional participants. The two mandatory MOVELINK participants are $mover$ (object in motion) and $trigger$ (verb denoting motion). Five of the optional participants express different aspects of the $mover$ in space, namely, $source$ (the spatial element at the beginning of the motion path), $midpoint$ (the spatial elements along the motion path), $goal$ (the spatial element at the end of the motion path), $path$ (the spatial element that reflects the path of motion), and $landmark$ (the grounding location). The sixth optional participant, $motion\_signal$, connects the spatial aspect to the $mover$. Note that all spatial relations are *intra-sentential*.

2.2 The Spatial Relation Extraction Task

Given a set of n spatial elements, the spatial relation extraction task aims to (1) determine whether the elements form a spatial relation of a particular type, and if so, (2) classify the roles of each participating element. For example, from the sentence "The cup is on the table", two relations can be extracted: QSLINK(cup$_{trajector}$, table$_{landmark}$, on$_{trigger}$) and OLINK(cup$_{trajector}$, table$_{landmark}$, on$_{trigger}$). As another example, from the sentence "John walked from Boston to Cambridge", a MOVELINK with participants John$_{mover}$, walked$_{trigger}$, from$_{motion_signal}$, Boston$_{source}$, to$_{motion_signal}$, and Cambridge$_{goal}$ can be extracted.

Type	Description
place	A geographic entity or region (e.g., *lakes*, *mountains*) or an administrative entity (e.g., *towns*, *countries*)
path	A location where the focus is on the potential for traversal (e.g., *road*)
spatial entity	A spatially relevant entity that is neither a *place* nor a *path* (e.g., *car*)
non-motion event	An event that does not involve movement but is directly related to another spatial element
motion event	A species of event that involves movement (e.g., *arrived*)
motion signal	A particle, preposition, verb, or adverb that encodes path or manner information about a *motion event*
spatial signal	A preposition/prepositional phrase that reveals the relationship between two locations (e.g., *north of*)

Table 1: Seven types of spatial elements in SpaceEval.

Relation	Description
QSLINK	Exists between stationary spatial elements with a regional connection. E.g., in "The cup is on the table", the regions of "cup" and "table" are *externally connected* and hence they are involved in a QSLINK.
OLINK	Exists between stationary spatial elements expressing their relative or absolute orientations. E.g., in the above sentence, "cup" and "table" are involved in an OLINK, which conveys that "cup" is oriented *above* "table".
MOVELINK	Exists between spatial elements in motion. E.g., in "John walked from Boston to Cambridge", there is a MOVELINK involving mover "John", motion verb "walked", source "Boston", and goal "Cambridge".

Table 2: Three types of spatial relations in SpaceEval.

3 Related Work

Broadly speaking, existing spatial relation extraction systems have adopted either a *pipeline* approach or a *joint* approach to these subtasks. Given a set of spatial elements, a pipeline spatial relation extraction system (1) extracts the $triggers$, (2) determines whether a spatial relation exists between each extracted $trigger$ and each of the remaining spatial elements, and (3) classifies the role of each non-$trigger$ in each pair of spatially-related elements (Kordjamshidi et al., 2011; Bastianelli et al., 2013; Kordjamshidi and Moens, 2015).

The major weakness of pipeline approaches is that errors in trigger identification can propagate to the relation classification component, whose errors can in turn propagate to the role labeling component. To address this weakness, Roberts et al. (2012; 2013) investigated *joint* approaches. Given a set of spatial elements with an assignment of roles to each element, a joint spatial relation extraction system uses a binary classifier to determine whether these elements form a spatial relation with the roles correctly assigned to all participating elements. In other words, the classifier will output a 1 if and only if (1) the elements in the set form a relation and (2) their roles in the relation are correct. The systems participating in SpaceEval all seem to be in favor of joint approaches (D'Souza and Ng, 2015; Nichols and Botros, 2015; Salaberri et al., 2015).

4 Baseline System

To ensure that we have a state-of-the-art baseline system for spatial relation extraction, we employ our SpaceEval participating system (D'Souza and Ng, 2015), which achieved the best results on task 3a in the official SpaceEval evaluation.[1]

This Baseline system performs *joint* role labeling and relation classification using an ensemble of classifiers. Specifically, it trains one classifier for extracting QSLINKs and OLINKs, and seven classifiers for extracting MOVELINKs. Creating these eight classifiers permits (1) separating the treatment of MOVELINKs from QSLINKs and OLINKs (because the former involves objects in motion while the latter involve stationary objects); and (2) simplifying the extraction of MOVELINKs (because its optional participants are extracted *independently* of each other by these classifiers).

4.1 Training the Baseline Classifiers

In this subsection, we describe how we train the Baseline classifiers, which include one classifier for identifying QSLINKs and OLINKs (Section 4.1.1) and seven classifiers for identifying MOVELINKs (Section 4.1.2).

4.1.1 The LINK Classifier

We collapse QSLINKs and OLINKs to a single relation type, LINK, identifying these two types of links using the LINK classifier. To understand why we can do this, recall from Section 2.1 that in QSLINKs and OLINKs, the $trigger$ has to be a *spatial signal* element having a semantic type attribute. If its semantic type is topological, it triggers a QSLINK; if it is directional, it triggers an OLINK; and

[1] Since the official annotated test data is not available to us, we cannot compare our results with the shared task systems' official results, but comparing against a state-of-the-art baseline will enable us to determine whether our approach is better than the best existing spatial relation extraction system.

if it is both, it triggers both relation types. Hence, if a LINK is identified by our classifier, we can simply use the semantic type of the relation's $trigger$ to determine whether the relation is a QSLINK, an OLINK, or both.

We create training instances for training a LINK classifier as follows. Following the joint approach described above, we create one training instance for each possible role labeling of each triplet of distinct spatial elements in each sentence in a training document. The role labels assigned to the spatial elements in each triplet are subject to the following constraints: (1) each triplet contains a $trajector$, a $landmark$, and a $trigger$; (2) neither the $trajector$ nor the $landmark$ are of type *spatial signal* or *motion signal*; and (3) the $trigger$ is a *spatial signal*. These role constraints are derived from the data annotation scheme. Note that a LINK may have at most one *implicit* participant. For instance, the relation LINK(balloon$_{trajector}$, up$_{trigger}$) extracted from the sentence "The balloon went up" has an implicit $landmark$. To allow for implicit participants, from each training instance we have created thus far, we create three additional training instances, where exactly one of the three participants has the value IMPLICIT.

A training instance is labeled as positive if and only if the elements in the triplet form a relation and their roles are correct. As an example, for the QSLINK and OLINK sentence in Table 2, exactly one positive instance, LINK(cup$_{trajector}$, table$_{landmark}$, on$_{trigger}$), will be created.

Each instance is represented using the 31 features, which can be broadly divided into seven types: lexical, grammatical, semantic, positional, distance, entity attributes, and entity roles.[2] We train the LINK classifier using the SVM learning algorithm as implemented in the SVM^{light} software package (Joachims, 1999). To optimize classifier performance, we tune two parameters, the regularization parameter C and the cost-factor parameter J, to maximize F-score on the development data.[3] Since joint tuning of these parameters is computationally expensive, we employ a hill-climbing algorithm to find a local maximum, altering one parameter at a time to optimize F-score by holding the other parameter fixed.

4.1.2 The Seven MOVELINK Classifiers

If we adopted the aforementioned joint method as is for extracting MOVELINKs, each instance would correspond to an octuple of the form ($trigger_i$, $mover_j$, $source_k$, $midpoint_m$, $goal_n$, $landmark_o$, $path_p$, $motion\_signal_r$), where each participant in the octuple is either a distinct spatial element with a role or the NULL element (if it is not present in the relation). However, generating role permutations for octuples from all spatial elements in a sentence is computationally infeasible. For this reason, we simplify MOVELINK extraction as follows. First, we decompose the MOVELINK octuple into seven smaller tuples including one pair and six triplets. These seven tuples are: [1] ($trigger_i$, $mover_j$); [2] ($trigger_i$, $mover_j$, $source_k$); [3] ($trigger_i$, $mover_j$, $midpoint_m$); [4] ($trigger_i$, $mover_j$, $goal_n$); [5] ($trigger_i$, $mover_j$, $landmark_o$); [6] ($trigger_i$, $mover_j$, $path_p$); and [7] ($trigger_i$, $mover_j$, $motion\_signal_r$). Then, we create seven separate classifiers for identifying these seven MOVELINK tuples, respectively.

Using this decomposition for MOVELINK instances, we can generate instances for each classifier using the aforementioned joint approach as is. For instance, to train classifier [1], we generate pairs of the form ($trigger_i$, $mover_j$), where $trigger_i$ and $mover_j$ are spatial elements proposed as a candidate $trigger$ and a candidate $mover$, respectively. Positive training instances are those ($trigger_i$, $mover_j$) pairs annotated as being part of a MOVELINK in the training data, while the rest of the candidate pairs are negative training instances. The instances for training the remaining six classifiers are generated similarly.

As in the LINK classifier, we enforce global role constraints when creating training instances for the MOVELINK classifiers. Specifically, the roles assigned to the spatial elements in each training instance of each MOVELINK classifier are subject to six constraints: (1) the $trigger$ has type *motion event*; (2) the $mover$ has type *place*, *path*, *spatial entity*, or *non-motion event*; (3) the $source$, the $goal$, and the $landmark$ can be NULL or have type *place*, *path*, *spatial entity*, or *non-motion event*; (4) the $midpoint$ can be NULL or have type *place*, *path*, or *spatial entity*; (5) the $path$ can be NULL or have type *path*; and (6) the $motion\_signal$ can

[2] Space limitations preclude a description of these features. See D'Souza and Ng (2015) for details.

[3] C is chosen from the set {0.01, 0.05, 0.1, 0.5, 1.0, 10.0, 50.0, 100.0}, and J is chosen from the set {0.01, 0.05, 0.1, 0.5, 1.0, 2.0, 4.0, 6.0}. All other learning parameters are set to their default values. In particular, a linear kernel is used.

be NULL or have type *motion signal*.

Our method for decomposing the octuple by role can be justified as follows. Since $trigger$ and $mover$ are mandatory MOVELINK participants, we have a classifier for classifying this core aspect of a MOVELINK. The next six classifiers, [2] to [7], aim to improve the core MOVELINK extraction by exploiting the contextual dependencies with each of its unique spatial aspects, namely $source$, $midpoint$, $goal$, $landmark$, $path$, and $motion\_signal$.

As an example, for the MOVELINK sentence in Table 2, we will create three positive instances: (John$_{mover}$, walked$_{trigger}$) for classifier [1], (John$_{mover}$, walked$_{trigger}$, Boston$_{source}$) for classifier [2], and (John$_{mover}$, walked$_{trigger}$, Cambridge$_{goal}$) for classifier [4].

We represent each training instance using the 31 features that were used to train the LINK classifier. We train each of the MOVELINK classifiers using SVM^{light}. We tune the C and J parameters to maximize F-score on the development data using the hill-climbing algorithm described earlier.[4]

4.2 Applying the Baseline Classifiers

After training, we apply the resulting classifiers to classify the test instances, which are created in the same way as the training instances. As noted before, each LINK extracted from a test document by the LINK classifier is further qualified as QSLINK, OLINK, or both based on the semantic type of its $trigger$. The MOVELINKs are extracted from a test document by combining the outputs from the seven MOVELINK classifiers.

There is a caveat, however: different MOVELINK classifiers can make conflicting decisions. For instance, classifier [1] might misclassify (John$_{mover}$, walked$_{trigger}$) as negative, whereas classifier [2] might correctly classify (John$_{mover}$, walked$_{trigger}$, Boston$_{source}$) as positive. To resolve these conflicting decisions, we give preference to positive decisions, meaning that in this case we will posit "John" and "walked" as having the roles of $mover$ and $trigger$ respectively. There are two more sources of conflicts. First, a spatial element may be assigned different roles for a given MOVELINK by different classifiers. Second, the global constraint that each MOVELINK can have at most one $source$, at most one $goal$, and at most one $landmark$ can be violated. To resolve these conflicts, we select for each spatial element the role that was predicted with highest confidence by the SVM classifiers subject to the global constraint.[5]

5 Our Multi-Pass Sieve Approach

In this section, we describe two methods for employing sieves for extracting spatial relations.

5.1 Using Sieves without Trees

To motivate our first method, recall that the Baseline resolves conflicting decisions in a heuristic manner. For instance, it prefers positive to negative decisions, effectively favoring recall over precision for spatial relation extraction. It is not clear whether this ad-hoc decision is good or not. As another example, when more than one role is assigned to the same spatial element in a MOVELINK, it favors the role associated with the highest SVM confidence. This, however, is also an ad-hoc decision: recall that each classifier's parameters are tuned independently of the others, so different confidence values assigned by different classifiers are not directly comparable.

Employing sieves for spatial relation extraction obviates the need for such ad-hoc decisions. In our implementation, we have eight sieves, each of which corresponds to exactly one of the eight classifiers employed by the Baseline. Recall from the introduction that these sieves are ordered as a pipeline. So, whenever a conflict arises, earlier sieves' decisions have precedence over later sieves' decisions. Returning to the conflicting decisions mentioned before, if sieve 1 misclassifies (John$_{mover}$, walked$_{trigger}$) as negative, whereas sieve 2 correctly classifies (John$_{mover}$, walked$_{trigger}$, Boston$_{source}$) as positive, then we will posit that no MOVELINK exists between "John" and "walked" because we have more confidence in sieve 1's decision than sieve 2's decision. As another example, if two classifiers assign different roles to the same spatial element, then we will choose the role assigned by the classifier associated with the earlier sieve.

Given the above discussion, it should be clear that the ordering of the sieves is important. Typically, sieves are ordered by *precision*, with the hope of reducing the number of erroneous deci-

[4] See Footnote 3 for the set of values of C and J used for parameter tuning.

[5] We use the distance from the hyperplane as a measure of an SVM classifier's confidence.

sions passed from the earlier sieves to the later sieves (e.g., Raghunathan et al. (2010), Lee et al. (2013), Chambers et al. (2014)). Motivated by this observation, we order the sieves as follows. We set sieve 0 to be the LINK classifier and sieve 1 to be the ($trigger$, $mover$) classifier, and then order the remaining sieves by precision. Specifically, we compute the precision of each sieve on the development data, then add sieves into the pipeline in decreasing order of precision.

5.2 Using Sieves with Trees

Next, we describe our second method for applying sieves to spatial relation extraction. To motivate this method, recall that one important property of a sieve-based approach is that later sieves can exploit earlier sieves' decisions when making their own decisions. However, our first method of using sieves makes limited use of the decisions made by earlier sieves. In particular, while each sieve exploits the knowledge of whether a spatial element has been assigned a role by an earlier sieve, it does not exploit the knowledge of what the role is.

Our second method exploits the role decisions made by earlier sieves, but another question arises: how can we encode these role decisions so that they can be best exploited by later sieves? One possibility is to employ them as additional features for training the classifiers associated with later sieves. Motivated by previous work on tree kernels for relation extraction, we employ parse trees as a *structured* feature to encode the syntactic relationships among the roles extracted so far for a given MOVELINK.

We create the structured feature as follows. To strike a better balance between having a rich representation of the context surrounding a spatial relation and improving the learner's ability to generalize, we extract a *subtree* from a parse tree and use it as the value of the structured feature. Specifically, given relation candidate triplet (e_1,e_2,e_3), where spatial elements e_1, e_2, and e_3 are posited in roles r_1, r_2, and r_3, respectively, and the associated syntactic parse tree T, we extract our parse subtree from T as follows. First, we identify the smallest subtree that covers all three spatial elements and call its root r. Second, for each path from each spatial element to r, we include in the parse subtree all the nodes that lie on the path and their immediate children. Third, we simplify the subtree by removing the POS nodes above each spatial element, effectively attaching it to its grandparent. Finally, for better generalization, we replace each spatial element with its role.

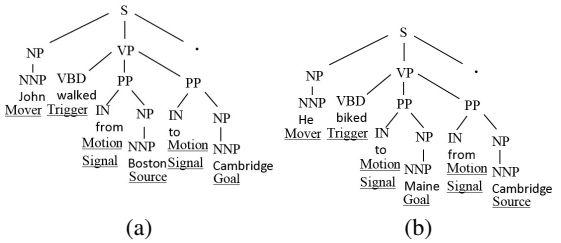


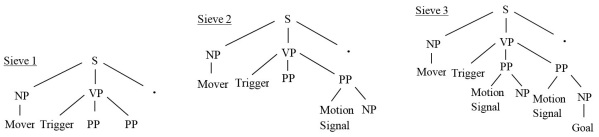
Figure 2: Syntactic parse trees for two example sentences containing MOVELINKs.

Since this subtree centers on a spatial relation, we call it a *spatial relation centered tree* (SRCT). As mentioned before, we will use SRCTs in combination with our sieve-based approach. This results in a novel application of structured features: to our knowledge, all trees that were previously used as structured features were *static*. By contrast, SRCTs expand as we move along the sieve pipeline. We will discuss examples of how to create SRCTs below.

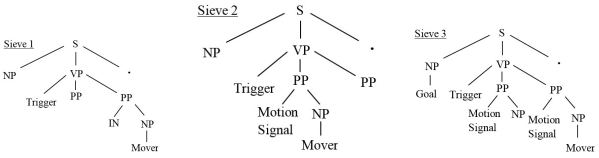
Creating training instances with structured features is straightforward. Recall that a structured feature is used as an additional feature when training each classifier. In other words, it will be used in combination with the original 31 features. The training instance creation method used by the Baseline remains unchanged. All we need to do is to add a SRCT as a structured feature to each training instance.

Consider the sentence "John walked from Boston to Cambridge", whose parse tree is shown in Figure 2(a). Since only one MOVELINK can be extracted from it, only one positive training instance will be created for each classifier. Assume that sieve 1 contains the ($trigger$, $mover$) classifier; sieve 2 contains the ($trigger$, $mover$, $motion\_signal$) classifier; and sieve 3 contains the ($trigger$, $mover$, $goal$) classifier. Figure 3(a) shows the SRCTs used in the corresponding positive instances, while Figure 3(b) shows the SRCTs associated with randomly chosen negative instances used to train the classifiers in these sieves.

To train a classifier on instances containing the original 31 features *and* SRCTs, we employ SVM$^{light-TK}$ (Moschitti, 2004; Moschitti, 2006), which (1) trains an SVM classifier using the 31 features with a linear kernel; (2) trains an SVM classifier using only the SRCTs with a convolution

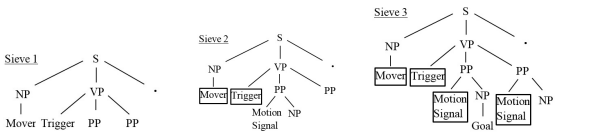


(a) SRCTs associated with positive training instances in sieves 1, 2, and 3.

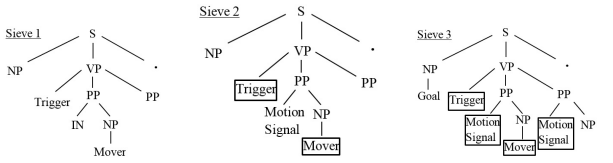


(b) SRCTs associated with randomly chosen negative training instances in sieves 1, 2, and 3.

Figure 3: SRCTs associated with training instances created for the sentence in Figure 2a.



(a) SRCTs associated with correctly classified test instances in sieves 1, 2, and 3.



(b) SRCTs associated with incorrectly classified test instances in sieves 1, 2, and 3.

Figure 4: SRCTs associated with test instances for the sentence in Figure 2b.

kernel; and (3) combines these two kernels using a composite kernel. Specifically, we define composite kernel K_c for combining linear kernel K_l and convolution kernel K_t as follows:

$$K_c(F_1, F_2) = K_l(F_1, F_2) + T \cdot K_t(T_1, T_2),$$

where F_1 and F_2 are the set of 31 features from two training instances, T_1 and T_2 are their SRCTs, and T is the combination parameter. We employ the hill-climbing algorithm described before to tune the C, J and T parameters to maximize F-score on the development data.[6]

The test instances with structured features are created in the same way as the training instances, with one notable difference. Note that roles are used to create structured features. While they are available in training, they are not available in the test documents. Hence, to create SRCTs for a test instance, we have to employ the roles *predicted* by preceding sieves. This is precisely how we exploit the role decisions made by earlier sieves in later sieves. This also explains why the SRCTs expand: more and more roles will be attached to a SRCT as it passes through the sieve pipeline.

As an example, consider the test sentence "He biked to Maine from Cambridge", whose parse tree is shown in Figure 2(b). Figure 4(a) shows the SRCTs generated when each sieve makes the correct decisions. Specifically, sieve 1 correctly identifies "He" as the $mover$ and "biked" as the $trigger$. The roles (correctly) extracted by sieve 1 (shown in boxes) are incorporated into the SRCT created in sieve 2. Similarly, the $motion_signal$s (correctly) extracted by sieve 2 are incorporated into the SRCT created in sieve 3. Figure 4(b) shows the SRCTs generated for misclassified instances. Assume that sieve 1 misclassifies the test instance underlying the SRCT (as positive). As we can see, this mistake is propagated to the SRCTs generated in later sieves.

Note that the precision of a sieve may change with the addition of SRCTs. For this reason, the sieves need to be reordered using the algorithm described in the previous subsection.

6 Evaluation

6.1 Experimental Setup

SpaceEval tasks. We evaluate our approach in tasks 1d and 3a of SpaceEval. These two tasks evaluate a system's ability to extract ($trajector$, $landmark$, $trigger$) triplets in QSLINKs and OLINKs as well as ($trigger$, $mover$) pairs in MOVELINKs using gold spatial elements (1d) and automatically extracted spatial elements (3a). To extract the spatial elements needed for task 3a, we follow Bastianelli et al.'s (2013) sequence labeling approach, except that we train the sequence labeler using a CRF rather than an HMM.[7]

Dataset. Since the annotated test set used in SpaceEval's official evaluation is not available

[6]To tune T, we attempted values between 0 and 2 in increments of 0.2. To tune C and J, we attempted the values specified in Footnote 3.

[7]We train two CRF models using CRF++ (`https://taku910.github.io/crfpp/`), one to extract *motion signal*s and the other to extract the remaining six types of spatial elements (see Table 1). The reason is that *motion signal* is the only type of spatial element that can overlap with other types. When predicting *spatial signal*s, we also predict their semantic types, since the LINK classifier needs this attribute to distinguish between QSLINKs and OLINKs (see Section 4.1.1). To increase recall, we use the 10-best outputs returned by the CRFs as candidate spatial elements.

	QSLINK						OLINK						MOVELINK						**OVERALL**		
	False			True			False			True			False			True					
	R	P	F	R	P	F	R	P	F	R	P	F	R	P	F	R	P	F	R	P	F
Baseline	99.5	99.4	99.5	46.9	48.9	47.9	100	99.4	**99.7**	50.3	100	**66.9**	91.3	99.8	95.3	84.8	61.5	71.3	78.8	84.8	81.7
Sieve	99.5	99.4	99.5	46.9	48.9	47.9	100	99.4	**99.7**	50.3	100	**66.9**	94.7	99.8	97.2	79.4	72.1	75.5	78.5	86.6	82.3
+ SRCTs	99.8	99.4	**99.6**	43.1	66.4	**52.3**	100	99.4	**99.7**	43.7	100	60.8	97.1	99.8	**98.4**	77.3	82.3	**79.7**	76.8	91.2	**83.4**

(a) Results obtained using gold spatial elements.

	QSLINK						OLINK						MOVELINK						**OVERALL**		
	False			True			False			True			False			True					
	R	P	F	R	P	F	R	P	F	R	P	F	R	P	F	R	P	F	R	P	F
Baseline	99.8	99.9	**99.9**	28.5	11.5	16.4	100	99.9	**100**	31.2	100	**47.5**	92.3	50.0	64.9	54.0	17.1	26.0	67.6	63.1	65.3
Sieve	99.8	99.9	**99.9**	28.5	11.5	16.4	100	99.9	**100**	31.2	100	**47.5**	96.1	50.0	**65.8**	42.5	25.3	31.7	66.4	64.4	65.4
+ SRCTs	99.8	99.9	**99.9**	28.3	12.9	**17.8**	100	99.9	**100**	31.2	100	**47.5**	94.7	50.0	65.5	56.2	24.5	**34.2**	68.4	64.6	**66.4**

(b) Results obtained using extracted spatial elements.

Table 3: Results for extracting spatial relations.

to us at the time of writing, we conduct our evaluation on the SpaceEval training corpus, which contains 1890 spatial relations (886 QSLINKs, 225 OLINKs, and 779 MOVELINKs) and 1139 MOVELINK optional roles (95 $sources$, 65 $midpoints$, 310 $goals$, 77 $landmarks$, 100 $paths$, and 492 $motion\_signals$). We partition the 59 narratives in the corpus into five folds and report five-fold cross-validation results. In each fold experiment, we employ three folds for training, one fold for development, and one fold for testing.

Evaluation metrics. To evaluate the results for the two SpaceEval tasks, we employ the official SpaceEval scoring program, which reports results in terms of recall, precision, and F-score on the three types of spatial relations in isolation and in combination. To evaluate the results for extracting MOVELINK optional roles, we compute the recall, precision, and F-score for each role.

6.2 Results and Discussion

Tables 3a and 3b show the spatial relation extraction results for three systems — the Baseline (row 1), our Sieve approach without SRCTs (row 2), and our Sieve approach with SRCTs (row 3) — obtained using gold and extracted spatial elements, respectively.

The official scoring program reports spatial relation extraction results in terms of recall (R), precision (P), and F-score (F). For QSLINKs and OLINKs, it reports results on (1) extracting spatially-related ($trajector$, $landmark$, $trigger$) triplets (see the "True" columns) and (2) identifying that no spatial relation exists among a (candidate $trajector$, candidate $landmark$, candidate $trigger$) triplet (see the "False" columns). For MOVELINKs, it reports results on (1) extracting spatially-related ($trigger$, $mover$) pairs (see the "True" columns) and (2) identifying that no spatial relation exists between a (candidate $trigger$, candidate $mover$) pair (see the "False" columns). Since the pairs/triplets without links considerably outnumber those with links, the OVERALL scores are dominated by the performances on "False", which, as expected, are high.

Of particular interest are the MOVELINK scores under the "True" columns. When gold spatial elements are used, Sieve significantly outperforms Baseline ($p < 0.001$) owing to substantial gains in precision with smaller losses in recall.[8] Adding SRCTs to Sieve further boosts performance significantly ($p < 0.005$). As we can see, Sieve+SRCTs outperforms Baseline by 8.4% absolute F-score on extracting ($trigger$, $mover$) pairs. With respect to the OVERALL score, which also takes into account QSLINKs and OLINKs, Sieve+SRCTs outperforms Baseline significantly by 1.7% absolute F-score. Similar trends can be observed for the results obtained using extracted spatial elements. Note that Baseline and Sieve have the same QSLINK and OLINK results because the LINK classifier is associated with the first sieve.

Tables 4a and 4b show the results on extracting MOVELINK optional roles using gold and extracted spatial elements, respectively. When gold elements are used, Sieve insignificantly outperforms Baseline on $source$, $midpoint$, and $motion\_signal$. When used with SRCTs, Sieve insignificantly outperforms Baseline on $source$, $midpoint$, $path$, and $motion\_signal$. Overall, Sieve+SRCTs insignificantly outperforms Baseline by 3.0% absolute F-score.[9] While the results

[8] All statistical significance tests are paired t-tests, with p set to 0.05 unless otherwise stated.

[9] A closer examination of the results reveals why the improvement is insignificant: since many roles occur infrequently in the corpus, the parameters learned from the devel-

	source			*midpoint*			*goal*			*landmark*			*path*			*motion-signal*			**OVERALL**		
	R	P	F	R	P	F	R	P	F	R	P	F	R	P	F	R	P	F	R	P	F
Baseline	34.7	37.1	35.9	43.1	26.9	33.1	46.1	73.7	**56.8**	19.5	31.9	**24.2**	44.0	62.0	51.5	72.6	67.4	69.9	54.4	59.9	57.0
Sieve	29.5	62.2	**40.0**	30.8	39.2	34.5	43.6	79.4	56.3	13.0	37.0	19.2	36.0	69.2	47.4	67.5	74.8	71.0	49.3	71.1	58.2
+ SRCTs	23.2	100	37.6	30.8	71.4	**43.0**	41.3	85.3	55.7	5.2	66.7	9.6	40.0	85.1	**54.4**	64.0	84.5	**72.8**	46.5	84.5	**60.0**

(a) Results obtained using gold spatial elements.

	source			*midpoint*			*goal*			*landmark*			*path*			motion signal			**OVERALL**		
	R	P	F	R	P	F	R	P	F	R	P	F	R	P	F	R	P	F	R	P	F
Baseline	25.3	12.0	16.3	35.4	8.9	14.3	38.7	15.2	21.8	18.2	2.5	4.4	14.0	12.9	13.4	27.7	24.0	25.7	29.1	13.3	18.2
Sieve	10.5	23.3	14.5	20.0	20.2	20.2	30.3	24.1	26.9	5.2	3.3	4.1	10.0	20.0	13.3	27.4	25.3	**26.3**	23.4	22.1	22.7
+ SRCTs	15.8	51.7	**24.2**	16.9	34.4	**22.7**	32.3	25.4	**28.4**	5.2	8.3	**6.4**	12.0	18.5	**14.6**	28.1	22.9	25.2	24.6	23.9	**24.3**

(b) Results obtained using extracted spatial elements.

Table 4: Results for extracting MOVELINK optional roles.

obtained using extracted elements exhibit different trends, Sieve+SRCTs's OVERALL improvement of 6.1% absolute F-score over Baseline is significant ($p < 0.001$).

Table 5 shows the results of these systems on extracting entire MOVELINKs, where a MOVELINK is considered correctly extracted if *all* of its participating elements and their roles are correct. Note that none of the SpaceEval tasks employ this stringent but informative evaluation measure. As we can see, Sieve+SRCTs insignificantly outperforms Baseline by 2.3–3.0% absolute F-score, regardless of whether gold or extracted elements are used.

6.3 Error Analysis

In this subsection, we analyze the errors made by the best-performing system, Sieve+SRCTs, with respect to the extraction of MOVELINKs given our focus on this type of spatial relation.

Extracting ($mover$, $trigger$) pairs. The major source of recall error stems from the system's inability to extract $mover$s that are unseen in the training data. This error could be addressed using a named entity recognizer and WordNet categories related to people, places, animals, etc. The major source of precision error arises from *missing* gold annotations. Consider the sentence "I found myself biking..." Our system correctly extracted "biking" as the $trigger$ and "I" as the $mover$, but was considered wrong because "myself", not "I", was annotated as the $mover$ in the gold standard.

Extracting optional roles. A major source of recall/precision error stems from the system's inability to exploit contextual cues that are reliable indicators of a particular role. For instance, a statistical analysis of the training data reveals that $source$s are commonly preceded by prepositions such as "from" and "or", whereas $goal$s are commonly associated with verbs such as "return", "visit", "arrive", and "reach". This problem could be addressed by encoding these cues explicitly as additional features for training the role-specific classifiers. Another source of recall error can be attributed to the lack of background knowledge. Consider the sentence "We had only 70 more km to Cluj taking this way, but if getting back to Ciucea and on to Cluj the normal way would have been 25 km longer". Despite correctly extracting "Cluj" as the $goal$, the system failed to extract "Ciucea" as the $midpoint$. This problem could be alleviated by exploiting geographical knowledge concerning these cities in external knowledge sources such as Wikipedia.

	Gold			Extracted		
	R	P	F	R	P	F
Baseline	40.6	50.4	45.0	21.7	15.3	18.0
Sieve	40.7	50.4	45.0	20.9	14.9	17.4
+ SRCTs	40.5	59.0	**48.0**	23.3	18.0	**20.3**

Table 5: Results for extracting entire MOVELINKs using gold and extracted elements.

7 Conclusions

We have examined the under-studied task of spatial relation extraction, focusing on spatial relations of objects in motion. Our approach exploited expanding parse trees, which resulted from a novel combination of multi-pass sieves and tree kernels, achieving state-of-the-art results on two key SpaceEval tasks. To facilitate comparison with future work on this task, we released the source code of our spatial relation extraction system.[10]

Acknowledgments

We thank the three anonymous reviewers for their comments. This work was supported in part by NSF Grants IIS-1147644 and IIS-1219142.

opment data are not necessarily the same as those that yield the best results on the test data.

[10] See our website at `http://www.hlt.utdallas.edu/~jld082000/spatial-relations/` for details.

References

Emanuele Bastianelli, Danilo Croce, Daniele Nardi, and Roberto Basili. 2013. Unitor-HMM-TK: Structured kernel-based learning for spatial role labeling. In *Proceedings of the 7th International Workshop on Semantic Evaluation*, pages 573–579.

Razvan Bunescu and Raymond Mooney. 2005. A shortest path dependency kernel for relation extraction. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 724–731.

Nathanael Chambers, Taylor Cassidy, Bill McDowell, and Steven Bethard. 2014. Dense event ordering with a multi-pass architecture. *Transactions of the Association for Computational Linguistics*, 2:273–284.

Aron Culotta and Jeffrey Sorensen. 2004. Dependency tree kernels for relation extraction. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics, Main Volume*, pages 423–429.

Jennifer D'Souza and Vincent Ng. 2015. UTD: Ensemble-based spatial relation extraction. In *Proceedings of the 9th International Workshop on Semantic Evaluation*, pages 862–869.

Thorsten Joachims. 1999. Making large-scale SVM learning practical. In Bernhard Scholkopf and Alexander Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, pages 44–56. MIT Press.

Oleksandr Kolomiyets, Parisa Kordjamshidi, Steven Bethard, and Marie-Francine Moens. 2013. SemEval-2013 Task 3: Spatial role labeling. In *Proceedings of the 7th International Workshop on Semantic Evaluation*, pages 255–266.

Parisa Kordjamshidi and Marie-Francine Moens. 2015. Global machine learning for spatial ontology population. *Web Semantics: Science, Services and Agents on the World Wide Web*, 30(C):3–21.

Parisa Kordjamshidi, Martijn Van Otterlo, and Marie-Francine Moens. 2011. Spatial role labeling: Towards extraction of spatial relations from natural language. *ACM Transactions on Speech and Language Processing*, 8(3):4.

Parisa Kordjamshidi, Steven Bethard, and Marie-Francine Moens. 2012. SemEval-2012 Task 3: Spatial role labeling. In *Proceedings of the 6th International Workshop on Semantic Evaluation*, pages 365–373.

Heeyoung Lee, Angel Chang, Yves Peirsman, Nathanael Chambers, Mihai Surdeanu, and Dan Jurafsky. 2013. Deterministic coreference resolution based on entity-centric, precision-ranked rules. *Computational Linguistics*, 39(4):885–916.

Alessandro Moschitti. 2004. A study on convolution kernels for shallow statistic parsing. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics, Main Volume*, pages 335–342.

Alessandro Moschitti. 2006. Making tree kernels practical for natural language processing. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics*, pages 113–120.

Eric Nichols and Fadi Botros. 2015. SpRL-CWW: Spatial relation classification with independent multi-class models. In *Proceedings of the 9th International Workshop on Semantic Evaluation*, pages 895–901.

James Pustejovsky and Zachary Yocum. 2013. Capturing motion in ISO-SpaceBank. *Proceedings of the 9th Joint ISO - ACL SIGSEM Workshop on Interoperable Semantic Annotation*, pages 25–34.

James Pustejovsky, Jessica Moszkowicz, and Marc Verhagen. 2012. A linguistically grounded annotation language for spatial information. *TAL*, 53(2):87–113.

James Pustejovsky, Parisa Kordjamshidi, Marie-Francine Moens, Aaron Levine, Seth Dworman, and Zachary Yocum. 2015. SemEval-2015 Task 8: SpaceEval. In *Proceedings of the 9th International Workshop on Semantic Evaluation*, pages 884–894.

Karthik Raghunathan, Heeyoung Lee, Sudarshan Rangarajan, Nate Chambers, Mihai Surdeanu, Dan Jurafsky, and Christopher Manning. 2010. A multi-pass sieve for coreference resolution. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 492–501.

Kirk Roberts and Sanda M. Harabagiu. 2012. UTD-SpRL: A joint approach to spatial role labeling. In *Proceedings of the 6th International Workshop on Semantic Evaluation*, pages 419–424.

Kirk Roberts, Michael A. Skinner, and Sanda M. Harabagiu. 2013. Recognizing spatial containment relations between event mentions. In *Proceedings of the 10th International Conference on Computational Semantics – Long Papers*, pages 216–227.

Haritz Salaberri, Olatz Arregi, and Beñat Zapirain. 2015. IXAGroupEHUSpaceEval: (X-Space) A WordNet-based approach towards the automatic recognition of spatial information following the ISO-Space annotation scheme. In *Proceedings of the 9th International Workshop on Semantic Evaluation*, pages 856–861.

Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. 2003. Kernel methods for relation extraction. *Journal of Machine Learning Research*, 3:1083–1106.

GuoDong Zhou, Min Zhang, DongHong Ji, and QiaoMing Zhu. 2007. Tree kernel-based relation extraction with context-sensitive structured parse tree information. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 728–736.

Mr. Bennet, his coachman, and the Archbishop walk into a bar but only one of them gets recognized: On The Difficulty of Detecting Characters in Literary Texts

Hardik Vala[1], **David Jurgens**[1], **Andrew Piper**[2], **Derek Ruths**[1]
[1]School of Computer Science
[2]Department of Languages, Literatures, and Cultures
McGill University, Montreal, QC, Canada
hardik.vala@mail.mcgill.ca, jurgens@cs.mcgill.ca
andrew.piper@mcgill.ca, derek.ruths@mcgill.ca

Abstract

Characters are fundamental to literary analysis. Current approaches are heavily reliant on NER to identify characters, causing many to be overlooked. We propose a novel technique for character detection, achieving significant improvements over state of the art on multiple datasets.

1 Introduction

How many literary characters appear in a novel? Despite the seeming simplicity of the question, precisely identifying which characters appear in a story remains an open question in literary and narrative analysis. Characters form the core of many computational analyses, from inferring prototypical character types (Bamman et al., 2014) to identifying the structure of social networks in literature (Elson et al., 2010; Lee and Yeung, 2012; Agarwal et al., 2013; Ardanuy and Sporleder, 2014; Jayannavar et al., 2015). These current approaches have largely assumed that characters can be reliably identified in text using standard techniques such as Named Entity Recognition (NER) and that the variations in how a character is named can be found through coreference resolution. However, such treatment of character identity often overlooks minor characters that serve to enrich the social structure and serve as foils for the identities of major characters (Eder et al., 2010).

This work provides a comprehensive examination of literary character detection, with three key contributions. First, we formalize the task with evaluation criteria and offer two datasets, including a complete, manually-annotated list of all characters in 58 literary works. Second, we propose a new technique for character detection based on inducing character prototypes, and in comparisons with three state-of-the-art methods, demonstrate superior performance, achieving significant improvements in F1 over the next-best method. Third, as practical applications, we analyze literary trends in character density over 20 decades and revisit the character-based literary hypothesis tested by Elson et al. (2010).

2 Related Work

Character detection has primarily been performed in the context of mining literary social networks. Elson et al. (2010) extract character mentions from conversational segments, using the Stanford CoreNLP NER system to discover character names (Manning et al., 2014). To account for variability in character naming, alternate forms of a name are generated using the method of Davis et al. (2003) and merged together as a single character. Furthermore, the set of aliases for a character is expanded by creating coreference chains originating from these proper names and merging all coreferent expressions. Agarwal et al. (2013) also rely on the CoreNLP NER and coreference resolution systems for character detection; however for literary analysis, they use gold character mentions that have been marked and resolved by a team of trained annotators, highlighting the difficulty of the task.

He et al. (2013) propose an alternate approach for identifying speaker references in novels, using a probabilistic model to identify which character is speaking. However, to account for the multiple aliases used to refer to a character, the authors first manually constructed a list of characters and their aliases, which is the task proposed in this work and underscores the need for automated methods.

Two approaches mined social interaction net-

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 769–774,
Lisbon, Portugal, 17-21 September 2015.

works without relying on dialogue, unlike the methods of Elson et al. (2010) and He et al. (2013). Lee and Yeung (2012) build social networks by recognizing characters from explicit markers (e.g., kinship) and implicit markers (e.g., physical collocation). Similarly, Agarwal and Rambow (2010) build character networks using tree kernels on parse trees to identify interacting agents.

In the two most-related works, Bamman et al. (2014) and Ardanuy and Sporleder (2014), character names are extracted and clustered under a set of constraints. In the BookNLP system developed by Bamman et al. (2014), NER-identified names are retained and merged based on animacy, determined through dependencies with "sentient" lemmas from a small dictionary (including for example, *say* and *smile*), and gender, assigned through pronomial resolution and a dictionary of gender-specific honorifics. Ardanuy and Sporleder (2014) similarly use NER to identify character name mentions. These names are grouped through the application of a series of deterministic rules, beginning with recognizing gender constraints, where gender assignments are based off of gender-specific honorifics and names. If a gender can't be assigned, then one is derived from the majority count of gender-specific pronouns (e.g. he, herself) appearing in the immediate context of the name mentions. The extracted names are then clustered, while respecting the gender impositions, based on a sieve of name variant heuristics. In the final step, any remaining ambiguous referents , i.e., those that can be matched to multiple characters, are assigned to the more prominent character in the story. The authors achieve F1-scores > 0.9 for extracting the 10 most relevant characters in a small collection of novels, but the performance on all characters is unknown.

3 Detecting Characters

We propose an eight stage pipeline for detecting characters, which builds a graph where nodes are names and edges connect names belonging to the same character. The vertices in the graph are initially populated by running NER over the corpus and also incorporating names following an honorific. Second, coreference resolution is run to identify names that occur together in a coreference chain and edges are added where two nodes' names co-occur in a chain. Stanford CoreNLP is used for both NER and co-reference. Third, we apply a series of name variation rules to link

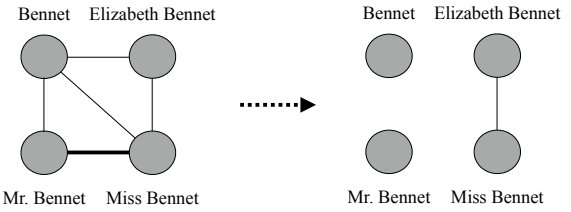


Figure 1: Resolving names in a character graph. The circles represent individual names and the thin and thick lines denote edges and anti-edges, respectively.

names potentially referring to the same character (e.g., by removing an honorific). Fourth, a gazetteer of 1859 hypocorisms for 560 names is used to link variations (e.g., Tim and Timmy).

Stages 2–4 potentially introduce edges connecting names of different characters. Therefore, in the fifth stage, three heuristics are applied to add prohibitions on merging two names into the same character. Two vertices cannot be merged if (1) the inferred genders of both names differ, (2) both names share a common surname but different first names, or (3) the honorific of both names differ, e.g., "Miss" and "Mrs." Similarly, the sixth stage inserts prohibitions by extracting pairs of names from the novel where (1) both names appear connected by a conjunction, (2) one name appears as the speaker mentioning the other name in direct speech, and (3) both names appear together in a quotation.

Together, Stage 1–6 are applied by first linking all nodes by edges and following, identifying pairs prohibited from being connected and remove the edges along the shortest path between those two nodes, effectively creating two new disconnected components in the name graph. Figure 1 illustrates this transformation on a simple character graph.

Next, the seventh step attempts to identify characters whose names may not be recognized by NER. For example, many minor characters do not appear as named entities and instead have general role-based referents such as "the governor" or "the archbishop." However, despite the lack of proper names, such characters behave and interact in similar ways as major characters, including having dialogue. Therefore, to discover such characters, we adopt a bootstrapping technique aimed at uncovering prototypical character behaviors from the novels themselves, inspired by the semantic predicate work of Flati and Navigli (2013). The Project Gutenberg fiction corpus was dependency parsed to identify all verbs in a dependency relation with nouns, where each noun was categorized as (a) a

named entity (b) having its first sense in WordNet refer to an animate entity (Fellbaum, 1998), or (c) neither of the above. All verbs associated with these were then ranked according to their ratio of types (a) and (c) to identify verbs strongly associated with character-like behaviors, which avoids including the behavior of nouns in (b) which may refer to minor characters. Ultimately, 2,073 verbs-and-dependency pairs with a ratio of $\geq$0.25 were retained as predicates selecting for character-like entities, after limited experimental testing showed this threshold extracted sensible verbs such as "rejoice," "accost," and "frown." Using this set of predicates, nouns appearing with the verb in the appropriate dependency relation are added as characters. We prohibit adding names contained in a small stop list of 22 generic nouns (e.g., "man").

Finally, the eighth and last stage removes nodes that are disconnected from the rest of the graph and represent a name that is a portion of one or more names for other nodes. These nodes are typically ambiguous first or last names. Thus, the remaining set of nodes are merged to create sets of names, each associated with a different character.

4 Experiments

Given a literary novel, our objective is to produce a list of characters, where each character may be associated with one or more names.

Datasets Two datasets are used. The first is a manually-annotated collection of 58 works with complete lists of all characters and their possible referents in texts. Of these works, 56 were generated as a part of an on-going longitudinal study of author style for all *Sherlock Holmes* stories written by Sir Arthur Conan Doyle. The remaining two works are the full length novels *Pride and Prejudice* by Jane Austen and *The Moonstone* by Wilkie Collins. Characters and their alias were manually coded by expert annotators, with multiple passes to ensure completeness. *The Moonstone* was treated as truly held-out test data and results were only generated once prior to submission.

The second dataset consists of 30 novels listed on Sparknotes (`sparknotes.com`) and their corresponding lists of characters, with supplemental naming variations of these characters provided by our annotators. These character lists often contain only the major characters in a novel; for example, their list for Pride and Prejudice contains only 17 characters, where as our manually-annotated list identifies 73 characters. Nevertheless, the Sparknotes data serves as a baseline of those characters any method should be able to detect.

Evaluation Character recognition systems produce a list of sets, each containing the names associated with one character, denoted $E = \{E_1, \ldots, E_n\}$ where E_i is a set of names for a character. These lists are evaluated against a gold standard list, denoted G, containing all naming variations for each character. To evaluate, we formalize the problem as finding a maximum bipartite matching where the sets of names in E and those in G constitute the two node types. For precision, matching is measured in the purity of an extracted set of names, E_i, with respect to the gold-standard names, G_j: $1 - \frac{|E^i - G^j|}{|E^i|}$; simply, a match is maximal when the set of extracted names is a subset of the gold standard names, with penalties for including wrong names. Recall uses a looser definition of matching with the aim of measuring whether a character G_j was found at all; matching is measured as a binary function that is 1 if $E_i \cap G_j \neq \emptyset$ and 0 otherwise.

Comparison Systems The task of character recognition has largely been subsumed into the task of extracting the social network of novels. Therefore, three state-of-the-art systems for social network extraction were selected: the method described in Elson et al. (2010), BookNLP (Bamman et al., 2014), and the method described in Ardanuy and Sporleder (2014). For each method, we follow their procedures for identifying the characters in the social network, which produces sets of one or more aliases associated with each identified character. As a baseline, we use the output of Stanford NER, where every name is considered a separate character; this baseline represents the upper-bound in recall from any system using only NER to identify character names.

Experiment 1: Accuracy Table 1 shows the results for the manually-annotated and SparkNotes corpora. The Sherlock Holmes corpus presents a notable challenge due to the presence of many minor characters, which are not detected by NER. An error analysis for our approach revealed that while many characters were extracted, the coreference resolution did not link a characters' different referents together and hence, each name was reported as a separate character, which caused a drop in performance. Nevertheless, our system provided the highest performance for character recognition.

The Pride and Prejudice novel presents a dif-

	Sherlock Holmes Stories			Pride and Prejudice			The Moonstone			SparkNotes
System	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1	Recall
NER Baseline	0.3327	0.6535	0.4332	0.3910	0.8356	0.5328	0.2460	0.5441	0.3388	0.6762
Elson et al. (2010)	0.3757	0.6042	0.4485	0.3100	0.5205	0.3886	0.2612	0.4931	0.3415	0.4723
BookNLP (Bamman et al., 2014)	0.6084	0.4832	0.5219	0.4855	0.5205	0.5024	0.3662	0.4706	0.4119	0.5880
Ardanuy and Sporleder (2014)	0.5744	0.4719	0.5181	0.4610	0.5108	0.4846	0.3623	0.4691	0.4088	0.5898
This work	0.5109	0.6099	**0.5404**	0.7245	0.7945	**0.7579**	0.3673	0.5735	**0.4478**	0.5990

Table 1: Accuracy of character detection on different portions of the two datasets.

	Precision	Recall	F1
Sherlock Holmes Stories	0.5910	0.5335	0.5608
Pride and Prejudice	0.7635	0.6879	0.7237
The Moonstone	0.3943	0.4613	0.4294

Table 2: Accuracy of proposed system without stage 7.

ferent set of challenges due to multiple characters sharing the same last name or the same first name. Here, coreference resolution frequently creates incorrect links between the similar names of different characters, creating a drop in precision for most systems. Our precision value particularly benefited from the heuristics for distinguishing characters by gender and stringent name-merging constraints. BookNLP and the approach of Ardanuy and Sporleder (2014) performed quite similarly in identifying characters, which is expected given the overlap in rules applied by both systems.

Moonstone contains a unique novel structure with multiple first-person narrators, group-based characters (e.g., "the jugglers") that present a challenge to co-reference systems, and 419 different names for the 78 unique characters. An error analysis of our system revealed that majority of mistakes were due to the multiple names for a character not being merged into a single identity. Nevertheless, our system performs best of those tested.

For the SparkNotes data, the NER baseline achieves the highest recall, indicating that many of the major character names listed in SparkNotes' data can be directly found by NER. Nevertheless, in reality, the baseline's performance is offset by its significantly lower precision, as shown in its performance on the other novels; indeed the baseline grossly overestimates the number of characters for the SparkNotes novels, reporting 339 characters per novel on average.

Table 2 shows our system's performance without stage 7, which involved the extraction of minor characters. Stage 7 overall improves recall with a slight hindrance to precision. For the Sherlock Holmes corpus, stage 7 is slightly detrimental to overall performance, which as we stipulated earlier is caused by missing co-referent links.

Finally, returning to the initially-posed question of how many characters are present, we find that despite the detection error in our method, the overall predicted number of characters is quite close to the actual: for Sherlock Holmes stories, the number of characters was estimated within 2.4 on average, for *Pride and Prejudice* our method predicted 72 compared with 73 actual characters, and for *The Moonstone* our method predicted 87 compared with 78. Thus, we argue that our procedure can provide a reasonable estimate for the total number of characters. (For comparison, BookNLP, the next best system, extracted 69 and 72 characters for Pride and Prejudice and The Moonstone, respectively, and within 1.2, on average, on the Sherlock Holmes set.)

Experiment 2: Literary Theories Elson et al. (2010) analyze 60 novels to computationally test literary theories for novels in urban and rural settings (Williams, 1975; Moretti, 1999). Recently, Jayannavar et al. (2015) challenged this analysis, showing their improved method for social network extraction did not support the same conclusions. While our work focuses only on character detection, we are nevertheless able to test the related hypothesis of whether the number of characters in novels with urban settings is more than those in rural. Character detection was run on the same novels from Elson et al. (2010) and we found no statistically-significant difference in the mean number of characters in urban and rural settings, even when accounting for text size. Thus, our work raises questions about how these character interact and whether the setting influences the structure of the social network, despite similar numbers of characters.

Experiment 3: Historical Trends As a second application of our technique, we examine historical trends in how many characters appear in a novel. All fiction novels listed on Project Gutenberg were compiled and publication dates were automatically extracted for 1066 and manually entered for an additional 637. This set was combined with a corpus of 6333 novels, including works such as *To The Lighthouse* by Virginia Woolf, not available on Project Gutenberg. Books were then partitioned into the decade in which they were au-

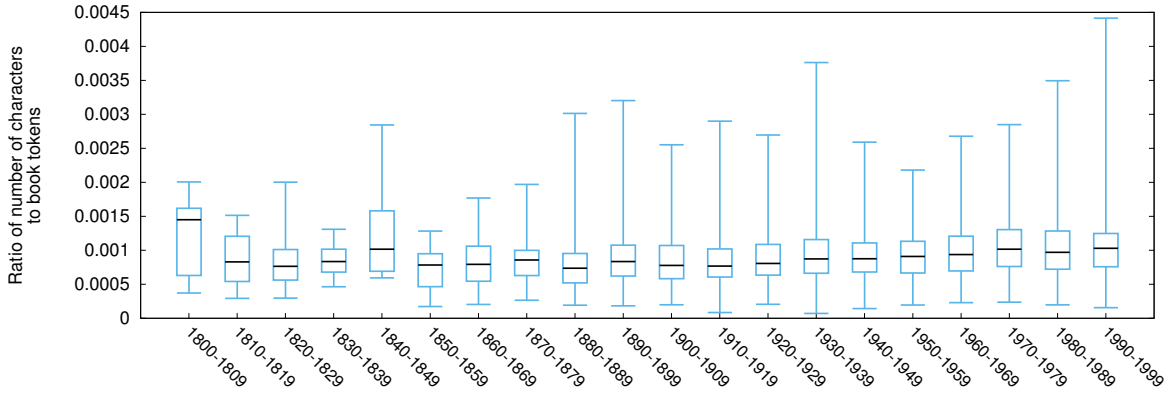


Figure 2: Distributions of the size-normalized number of characters per novel per decade.

thored. We limit our focus to trends starting in 1800 to 1990, when at least 11 books are available for each decade.

To account for variability in novel length, we normalize the novel's number of characters by its number of tokens. Figure 2 shows the box-and-whisker plot of the normalized number of characters per novel, where the box denotes the first and third quartile and the bar denotes the median. Surprisingly, we did not observe any significant change in the relative number of characters per novel, despite the underlying socio-economic changes that accompanied this time period. While novels written before 1850 had slightly more characters on average, this effect may be due to the smaller number of works available from this period. However, our finding raises many questions about whether the social networks for these characters obey similar trends in their size and density.

5 Conclusion

Although a fundamental task to character analysis, identifying the number of characters in a literary novel presents a significant challenge to current state of the art. To lay the foundation towards solving the task, we provide three contributions: (1) an annotated corpus of 58 books, (2) an evaluation framework for measuring performance on the task, (3) a new state-of-the-art method for character extraction. Furthermore, to promote future work we make all software and data available upon request.

6 Acknowledgements

We would like to thank the three annotators for their diligent reading and coding of novels.

References

Apoorv Agarwal and Owen Rambow. 2010. Automatic detection and classification of social events. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1024–1034.

Apoorv Agarwal, Anup Kotalwar, and Owen Rambow. 2013. Automatic extraction of social networks from literary text: A case study on alice in wonderland. In *the Proceedings of the 6th International Joint Conference on Natural Language Processing*.

Mariona Coll Ardanuy and Caroline Sporleder. 2014. Structure-based clustering of novels. In *Proceedings of the EACL Workshop on Computational Linguistics for Literature*, pages 31–39.

David Bamman, Ted Underwood, and Noah A Smith. 2014. A bayesian mixed effects model of literary character. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 370–379.

Peter T Davis, David K Elson, and Judith L Klavans. 2003. Methods for precise named entity matching in digital collections. In *Proceedings of the 3rd ACM/IEEE-CS joint conference on Digital libraries*, pages 125–127.

Jens Eder, Fotis Jannidis, and Ralf Schneider. 2010. *Characters in fictional worlds: Understanding imaginary beings in literature, film, and other media*, volume 3. Walter de Gruyter.

David K Elson, Nicholas Dames, and Kathleen R McKeown. 2010. Extracting social networks from literary fiction. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 138–147.

Christiane Fellbaum. 1998. *WordNet*. Wiley Online Library.

Tiziano Flati and Roberto Navigli. 2013. Spred: Large-scale harvesting of semantic predicates. In *Proceedings of the The 51st Annual Meeting of the Association for Computational Linguistics*.

Hua He, Denilson Barbosa, and Grzegorz Kondrak. 2013. Identification of speakers in novels. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 1312–1320.

Prashant Arun Jayannavar, Apoorv Agarwal, Melody Ju, and Owen Rambow. 2015. Validating literary theories using automatic social network extraction. In *Proceedings of the NAACL-2015 Workshop on Computational Linguistics for Literature*.

John Lee and Chak Yan Yeung. 2012. Extracting networks of people and places from literary texts. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60.

Franco Moretti. 1999. *Atlas of the European novel, 1800-1900*. Verso.

Raymond Williams. 1975. *The country and the city*, volume 423. Oxford University Press.

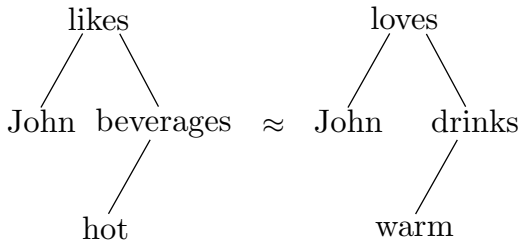
Convolutional Sentence Kernel from Word Embeddings for Short Text Categorization

Jonghoon Kim **François Rousseau** **Michalis Vazirgiannis**

LIX, École Polytechnique, France
john.jonghoon.kim@gmail.com

Abstract

This paper introduces a convolutional sentence kernel based on word embeddings. Our kernel overcomes the sparsity issue that arises when classifying short documents or in case of little training data. Experiments on six sentence datasets showed statistically significant higher accuracy over the standard linear kernel with n-gram features and other proposed models.



(a) 'John likes hot beverages' (b) 'John loves warm drinks'

Figure 1: Dependency tree representations of semantically related sentences yet with little overlap.

1 Introduction

With the proliferation of text data available online, text categorization emerged as a prominent research topic. Traditionally, words (unigrams) and phrases (n-grams) have been considered as document features and subsequently fed to a classifier such as an SVM (Joachims, 1998). In the SVM dual formulation that relies on kernels, i. e. similarity measures between documents, a linear kernel can be interpreted as the number of exact matching n-grams between two documents. Consequently, for short documents or when little training data is available, sparsity issues due to word synonymy arise, e. g., the sentences 'John likes hot beverages' and 'John loves warm drinks' have little overlap and therefore low linear kernel value (only 1) in the n-gram feature space, even with dependency tree representations and downward paths for n-grams as illustrated in Figure 1.

We propose to relax the exact matching between words by capitalizing on distances in word embeddings. We smooth the implicit delta word kernel, i. e. a Dirac similarity function between unigrams, behind the traditional linear document kernel to capture the similarity between words that are different, yet semantically close. We then aggregate these word and phrase kernels into sentence and documents kernels through convolution resulting in higher kernel values between semantically related sentences (e. g., close to 7 compared to 1 with bigram downward paths in Figure 1). Experiments on six standard datasets for sentiment analysis, subjectivity detection and topic spotting showed statistically significant higher accuracy for our proposed kernel over the bigram approaches. Our main goal is to demonstrate empirically that word distances from a given word vector space can easily be incorporated in the standard kernel between documents for higher effectiveness and little additional cost in efficiency.

The rest of this paper is structured as follows. Section 2 reviews the related work. Section 3 gives the detailed formulation of our kernel. Section 4 describes the experimental settings and the results we obtained on several datasets. Finally, Section 5 concludes our paper and mentions future work.

2 Related work

Siolas and d'Alché Buc (2000) pioneered the idea of *semantic kernels* for text categorization, capitalizing on WordNet (Miller, 1995) to propose continuous word kernels based on the inverse of the path lengths in the tree rather than the common delta word kernel used so far, i. e. exact matching between unigrams. Bloehdorn et al. (2006) extended it later to other tree-based similarity measures from WordNet while Mavroeidis et al. (2005) exploited its hierarchical structure to define a Generalized Vector Space Model kernel.

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 775–780,
Lisbon, Portugal, 17-21 September 2015.

In parallel, Collins and Duffy (2001) developed the first *tree kernels* to compare trees based on their topology (e. g., shared subtrees) rather than the similarity between their nodes. Culotta and Sorensen (2004) used them as Dependency Tree Kernels (DTK) to capture *syntactic similarities* while Bloehdorn and Moschitti (2007) and Croce et al. (2011) used them on parse trees with respectively Semantic Syntactic Tree Kernels (SSTK) and Smoothing Partial Tree Kernels (SPTK), adding node similarity based on WordNet to capture *semantic similarities* but limiting to comparisons between words of the same POS tag.

Similarly, Gärtner et al. (2003) developed *graph kernels* based on random walks and Srivastava et al. (2013) used them on dependency trees with Vector Tree Kernels (VTK), adding node similarity based on word embeddings from SENNA (Collobert et al., 2011) and reporting improvements over SSTK. The change from WordNet to SENNA was supported by the recent progress in low-dimension Euclidean vector space representations of words that are better suited for computing distances between words. Actually, in our experiments, word2vec by Mikolov et al. (2013a) led to better results than with SENNA for both VTK and our kernels. Moreover, it possesses an additional additive compositionality property obtained from the Skip-gram training setting (Mikolov et al., 2013b), e. g., the closest word to 'Germany' + 'capital' in the vector space is found to be 'Berlin'.

More recently, for short text similarity, Song and Roth (2015) and Kenter and de Rijke (2015) proposed additional semantic meta-features based on word embeddings to enhance classification.

3 Formulation

We denote the embedding of a word w by $\mathbf{w}$.

3.1 Word Kernel (WK)

We define a kernel between two words as a polynomial kernel over a cosine similarity in the word embedding space:

$$\mathrm{WK}(w_1, w_2) = \left[\frac{1}{2}\left(1 + \frac{\langle \mathbf{w}_1, \mathbf{w}_2 \rangle}{\|\mathbf{w}_1\|\|\mathbf{w}_2\|}\right)\right]^{\alpha} \quad (1)$$

where α is a scaling factor. We also tried Gaussian, Laplacian and sigmoid kernels but they led to poorer results in our experiments. Note that a delta word kernel, i. e. the Dirac function $\mathbb{1}_{w_1=w_2}$, leads to a document kernel corresponding to the standard linear kernel over n-grams.

3.2 Phrase Kernel (PhK)

Next we define a kernel between phrases consisting of several words. In our work, we considered two types of phrases: (1) *co-occurrence phrases* defined as contiguous sequences of words in the text; and (2) *syntactic phrases* defined as downward paths in the dependency tree representation, e. g., respectively 'hot beverages' and 'beverages hot' in Figure 1. With this dependency tree involved, we expect to have phrases that are syntactically more meaningful. Note that VTK considers *random walks* in dependency trees instead of downward paths, i. e. potentially taking into account same nodes multiple times for phrase length greater than two, phenomenon known as tottering.

Once we have phrases to compare, we may construct a kernel between them as the product of word kernels if they are of the same length l. That is, we define the Product Kernel (PK) as:

$$\mathrm{PK}(p_1, p_2) = \prod_{i=1}^{l} \mathrm{WK}(w_i^1, w_i^2) \quad (2)$$

where w_i^j is the i-th word in phrase p_j of length l. Alternatively, in particular for phrases of different lengths, we may embed phrases into the embedding space by taking a composition operation on the constituent word embeddings. We considered two common forms of composition (Blacoe and Lapata, 2012): vector addition ($+$) and element-wise multiplication ($\odot$). Then we define the Composition Kernel (CK) between phrases as:

$$\mathrm{CK}(p_1, p_2) = \mathrm{WK}(\mathbf{p}_1, \mathbf{p}_2) \quad (3)$$

where $\mathbf{p}_j$, the embedding of the phrase p_j, can be obtained either by addition ($\mathbf{p}_j = \sum_{i=1}^{l} \mathbf{w}_i^j$) or by element-wise multiplication ($\mathbf{p}_j = \bigodot_{i=1}^{l} \mathbf{w}_i^j$) of its word embeddings. For CK, we do not require the two phrases to be of the same length so the kernel has a desirable property of being able to compare 'Berlin' with 'capital of Germany' for instance.

3.3 Sentence Kernel (SK)

We can then formulate a sentence kernel in a similar way to Zelenko et al. (2003). It is defined through convolution as the sum of all local phrasal similarities, i. e. kernel values between phrases contained in the sentences:

$$\mathrm{SK}(s_1, s_2) = \sum_{\substack{p_1 \in \phi(s_1),\\ p_2 \in \phi(s_2)}} {\lambda_1}^{\epsilon}\, {\lambda_2}^{\eta}\, \mathrm{PhK}(p_1, p_2) \quad (4)$$

where $\phi(s_k)$ is the set of either statistical or syntactic phrases (or set of random walks for VTK) in sentence s_k, λ_1 is a decaying factor penalizing longer phrases, $\epsilon = \max\{|p_1|, |p_2|\}$ is the maximum length of the two phrases, λ_2 is a distortion parameter controlling the length difference η between the two phrases ($\eta = ||p_1| - |p_2||$) and PhK is a phrase kernel, either PK, CK$^+$ or CK$^\odot$.

Since the composition methods we consider are associative, we employed a dynamic programming approach in a similar fashion to Zelenko et al. (2003) to avoid duplicate computations.

3.4 Document Kernel

Finally, we sum sentence kernel values for all pairs of sentences between two documents to get the document kernel. Once we have obtained all document kernel values K_{ij} between documents i and j, we may normalize them by $\sqrt{K_{ii}K_{jj}}$ as the length of input documents might not be uniform.

4 Experiments

We evaluated our kernel with co-occurrence and syntactic phrases on several standard text categorization tasks.

4.1 Datasets

We considered four tasks: (1) binary *sentiment analysis* with a movie review dataset of 10,662 sentences (PL05) (Pang and Lee, 2005) and a product review dataset (Amazon) of 2,000 multi-line documents for 4 different product groups (Blitzer et al., 2007) (we will report the average effectiveness over the 4 sub-collections); (2) ternary *sentiment analysis* with the SemEval 2013 Task B dataset (Twitter) containing 12,348 tweets classified as positive, neutral or negative (Nakov et al., 2013); (3) binary *subjectivity detection* with a dataset of 10,000 sentences (PL04) (Pang and Lee, 2004) and another of 11,640 sentences (MPQA) (Wiebe et al., 2005); and (4) seven-class *topic spotting* with a news dataset (News) of 32,602 one-line news summaries (Vitale et al., 2012).

4.2 Experimental settings

In all our experiments, we used the FANSE parser (Tratz and Hovy, 2011) to generate dependency trees and the pre-trained version of word2vec[1], a 300 dimensional representation of 3 million English words trained over a Google News dataset of 100 billion words using the Skip-gram model and a context size of 5. While fine-tuning the embeddings to a specific task or on a given dataset may improve the result for that particular task or dataset (Levy et al., 2015), it makes the expected results less generalizable and the method harder to use as an off-the-shelf solution – re-training the neural network to obtain task-specific embeddings requires a certain amount of training data, admittedly unlabeled, but still not optimal under our scenario with short documents and little task-specific training data available. Moreover, tuning the hyperparameters to maximize the classification accuracy needs to be carried out on a validation set and therefore requires additional labeled data. Here, we are more interested in showing that distances in a given word vector space can enhance classification in general. As for the dependency-based word embeddings proposed by Levy and Goldberg (2014), we do not think they are better suited for the problem we are tackling. As we will see in the results, we do benefit from the dependency tree structure in the phrase kernel but we still want the word kernel to be based on topical similarity rather than functional similarity.

To train and test the SVM classifier, we used the LibSVM library (Chang and Lin, 2011) and employed the one-vs-one strategy for multi-class tasks. To prevent overfitting, we tuned the parameters using cross-validation on 80% of PL05 dataset ($\alpha = 5$, $\lambda_1 = 1$ for PK since there is no need for distortion as the phrases are of the same length by definition, and $\lambda_1 = \lambda_2 = 0.5$ for CK) and used the same set of parameters on the remaining datasets. We performed normalization for our kernel and baselines only when it led to performance improvements on the training set (PL05, News, PL04 and MPQA).

We report accuracy on the remaining 20% for PL05, on the standard test split for Twitter (25%) and News (50%) and from 5-fold cross-validation for the other datasets (Amazon, PL04 and MPQA). We only report accuracy as the macro-average F1-scores led to similar conclusions (and except for Twitter and News, the class label distributions are balanced). Results for phrase lengths longer than two were omitted since they were marginally different at best. Statistical significance of improvement over the bigram baseline with the same phrase definition was assessed using the micro sign test ($p < 0.01$) (Yang and Liu, 1999).

[1] https://code.google.com/p/word2vec

Table 1: Accuracy results on the test set for PL05 (20%), standard test split for Twitter (25%) and News (50%) and from 5-fold CV for the other datasets (Amazon, PL04 and MPQA). Bold font marks the best performance in the column. * indicates statistical significance at $p < 0.01$ using micro sign test against the bigram baseline (delta word kernel) of the same column and with the same phrase definition.

phrase definition	phrase kernel	phrase length	word kernel	PL05	Amazon	Twitter	News	PL04	MPQA
co-occurrence	PK	1	delta	0.742	0.768	0.623	0.769	0.904	0.754
co-occurrence	PK	2	delta	0.739	0.765	0.611	0.766	0.907	0.754
syntactic	PK	2	delta	0.748	0.791	0.646	0.767	0.910	0.757
random walk	PK	2	poly	0.799	0.810	0.698	0.802	**0.927**	**0.797**
co-occurrence	PK	1	poly	0.789*	0.797	0.776*	**0.806***	0.923*	0.793*
co-occurrence	PK	2	poly	0.784*	0.798	0.762*	0.801*	0.926*	0.794*
co-occurrence	CK$^+$	2	poly	0.796*	0.778	0.613	0.792*	0.917*	0.796*
co-occurrence	CK$^\odot$	2	poly	**0.801***	0.783	0.757*	0.793*	0.918*	0.794*
syntactic	PK	2	poly	0.796*	**0.813***	**0.808***	0.805*	**0.927***	0.796*
syntactic	CK$^+$	2	poly	0.794*	0.780	0.741*	0.788*	0.918*	0.794*
syntactic	CK$^\odot$	2	poly	0.797*	0.774	0.744*	0.792*	0.918*	0.794*

4.3 Results

Table 1 presents results from our convolutional sentence kernel and the baseline approaches. Note again that a delta word kernel leads to the typical unigram and bigram baseline approaches (first three rows). The 3rd row corresponds to DTK (Culotta and Sorensen, 2004) and the 4th one to VTK (Srivastava et al., 2013) – the difference with our model on the 9th row lies in the function $\phi(\cdot)$ that enumerates all random walks in the dependency tree representation following Gärtner et al. (2003) whereas we only consider the downward paths.

Overall, we obtained better results than the n-gram baselines, DTK and VTK, especially with syntactic phrases. VTK shows good performance across all datasets but its computation was more than 700% slower than with our kernel. Regarding the phrase kernels, PK generally produced better results than CK, implying that the semantic linearity and ontological relation encoded in the embedding is not sufficient enough and treating them separately is more beneficial. However, we believe CK has more room for improvement with the use of more accurate phrase embeddings such as the ones from Le and Mikolov (2014), Yin and Schütze (2014) and Yu and Dredze (2015).

There was little contribution to the accuracy from non-unigram features, indicating that large part of the performance improvement is credited to the word embedding resolving the sparsity issue.

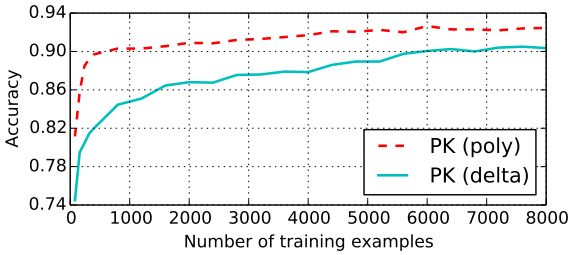


Figure 2: Test accuracy vs. number of training examples for our kernel and the bigram baseline.

This can be well observed with the following experiment on the number of training examples. Figure 2 shows the accuracy on the same test set (20% of the dataset) when the learning was done on 1% to 100% of the training set (80% of the dataset) for the bigram baseline and our bigram PK phrase kernel, both with dependency tree representation, on PL04. We see that our kernel starts to plateau earlier in the learning curve than the baseline and also reaches the maximum baseline accuracy with only about 1,500 training examples.

4.4 Computational complexity

Solving the SVM in the primal for the baselines requires $\mathcal{O}(NnL)$ time where N is the number of training documents, n is the number of words in the document and L is the maximum phrase length considered. The computation of VTK reduces down to power series computation of the

adjacency matrix of the product graph, and since we require kernel values between all documents, it requires $\mathcal{O}(N^2(n^2d + n^4L))$ time where d is the dimension of the word embedding space.

Our kernel is the sum of phrase kernels (PhK) starting from every pair of nodes between two sentences, for all phrase lengths (l) and distortions (λ_2) under consideration. By storing intermediate values of composite vectors, a phrase kernel can be computed in $\mathcal{O}(d)$ time regardless of the phrase length, therefore the whole computation process has $\mathcal{O}(N^2n^2L^2d)$ complexity. Although our kernel has the squared terms of the baseline's complexity, we are tackling the sparsity issue that arises with short text (small n) or when little training data is available (small N). Moreover, we were able to get better results with only bigrams (small L). Hence, the loss in efficiency is acceptable considering significant gains in effectiveness.

5 Conclusion

In this paper, we proposed a novel convolutional sentence kernel based on word embeddings that overcomes the sparsity issue, which arises when classifying short documents or when little training data is available. We described a general framework that can encompass the standard n-gram baseline approach as well as more relaxed versions with smoother word and phrase kernels. It achieved significant improvements over the baselines across all datasets when taking into account the additional information from the latent word similarity (word embeddings) and the syntactic structure (dependency tree).

Future work might involve designing new kernels for syntactic parse trees with appropriate similarity measures between non-terminal nodes as well as exploring recently proposed phrase embeddings for more accurate phrase kernels.

References

William Blacoe and Mirella Lapata. 2012. A Comparison of Vector-based Representations for Semantic Composition. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 546–556. ACL.

John Blitzer, Mark Dredze, and Fernando Pereira. 2007. Biographies, bollywood, boomboxes and blenders: Domain adaptation for sentiment classification. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, ACL '07, pages 440–447. ACL.

Stephan Bloehdorn and Alessandro Moschitti. 2007. Structure and Semantics for Expressive Text Kernels. In *Proceedings of the 16th ACM international conference on Information and knowledge management*, CIKM '07, pages 861–864. ACM.

Stephan Bloehdorn, Roberto Basili, Marco Cammisa, and Alessandro Moschitti. 2006. Semantic kernels for text classification based on topological measures of feature similarity. In *Proceedings of the 6th IEEE International Conference on Data Mining*, ICDM '06, pages 808–812. IEEE Computer Society.

Chih-Chung Chang and Chih-Jen Lin. 2011. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2(3):27:1–27:27.

Michael Collins and Nigel Duffy. 2001. Convolution Kernels for Natural Language. In *Advances in Neural Information Processing Systems 14*, NIPS '01, pages 625–632. The MIT Press.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural Language Processing (Almost) from Scratch. *Journal of Machine Learning Research*, 12:2493–2537, November.

Danilo Croce, Alessandro Moschitti, and Roberto Basili. 2011. Structured Lexical Similarity via Convolution Kernels on Dependency Trees. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 1034–1046. ACL.

Aron Culotta and Jeffrey Sorensen. 2004. Dependency Tree Kernels for Relation Extraction. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, ACL '04, pages 423–429. ACL.

Thomas Gärtner, Peter Flach, and Stefan Wrobel. 2003. On graph kernels: Hardness results and efficient alternatives. In *Proceedings of the Annual Conference on Computational Learning Theory*, COLT '03, pages 129–143.

Thorsten Joachims. 1998. Text categorization with Support Vector Machines: Learning with many relevant features. In *Proceedings of the 10th European Conference on Machine Learning*, ECML '98, pages 137–142.

Tom Kenter and Maarten de Rijke. 2015. Short text similarity with word embeddings. In *Proceedings of the 24th ACM international conference on Information and knowledge management*, CIKM '15. ACM.

Quoc Le and Tomas Mikolov. 2014. Distributed Representations of Sentences and Documents. In *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *ICML '14*, pages

1188–1196. JMLR Workshop and Conference Proceedings.

Omer Levy and Yoav Goldberg. 2014. Dependency-Based Word Embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, volume 2 of *ACL '14*, pages 302–308. ACL.

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

Dimitrios Mavroeidis, George Tsatsaronis, Michalis Vazirgiannis, Martin Theobald, and Gerhard Weikum. 2005. Word Sense Disambiguation for Exploiting Hierarchical Thesauri in Text Classification. In *Proceedings of the 9th European conference on Principles and Practice of Knowledge Discovery in Databases*, ECML PKDD '05, pages 181–192. Springer-Verlag Berlin.

Tomas Mikolov, Kai Chen, Greg S. Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. In *Proceedings of Workshop at International Conference on Learning Representations*, ICLR '13.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeffrey Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26*, NIPS '13, pages 3111–3119. Neural Information Processing Systems.

George A. Miller. 1995. WordNet: A Lexical Database for English. *Communications of the ACM*, 38(11):39–41, November.

Preslav Nakov, Zornitsa Kozareva, Alan Ritter, Sara Rosenthal, Veselin Stoyanov, and Theresa Wilson. 2013. Semeval-2013 task 2: Sentiment analysis in twitter. In *Proceedings of the 7th International Workshop on Semantic Evaluation*, SemEval-2013.

Bo Pang and Lilian Lee. 2004. A sentimental education: sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, ACL '04, pages 271–278. ACL.

Bo Pang and Lilian Lee. 2005. Seeing stars: exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 115–124. ACL.

Georges Siolas and Florence d'Alché Buc. 2000. Support Vector Machines Based on a Semantic Kernel for Text Categorization. In *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks*, volume 5 of *IJCNN '00*, pages 205–209. IEEE Computer Society.

Yangqiu Song and Dan Roth. 2015. Unsupervised Sparse Vector Densification for Short Text Similarity. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL-Short '15, pages 1275–1280. ACL.

Shashank Srivastava, Dirk Hovy, and Eduard H. Hovy. 2013. A Walk-Based Semantically Enriched Tree Kernel Over Distributed Word Representations. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, EMNLP '13, pages 1411–1416. ACL.

Stephen Tratz and Eduard H. Hovy. 2011. A Fast, Accurate, Non-projective, Semantically-enriched Parser. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 1257–1268. ACL.

Daniele Vitale, Paolo Ferragina, and Ugo Scaiella. 2012. Classification of Short Texts by Deploying Topical Annotations. In *Proceedings of the 34th European Conference on Information Retrieval*, ECIR'12, pages 376–387. Springer-Verlag.

Janyce M. Wiebe, Theresa Wilson, and Claire Cardie. 2005. Annotating expressions of opinions and emotions in language. *Language resources and evaluation*, 39(2-3):165–210.

Yiming Yang and Xin Liu. 1999. A Re-examination of Text Categorization Methods. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '99, pages 42–49. ACM.

Wenpeng Yin and Hinrich Schütze. 2014. An Exploration of Embeddings for Generalized Phrases. In *Proceedings of the ACL Student Research Workshop*, ACLstudent '14, pages 41–47. ACL.

Mo Yu and Mark Dredze. 2015. Learning Composition Models for Phrase Embeddings. *Transactions of the Association for Computational Linguistics*, 3:227–242.

Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. 2003. Kernel Methods for Relation Extraction. *Journal of Machine Learning Research*, 3:1083–1106.

Predicting the Structure of Cooking Recipes

Jermsak Jermsurawong and **Nizar Habash**
Computational Approaches to Modeling Language Lab
Computer Science
New York University Abu Dhabi
United Arab Emirates
{jermsak.jermsurawong,nizar.habash}@nyu.edu

Abstract

Cooking recipes exist in abundance; but due to their unstructured text format, they are hard to study quantitatively beyond treating them as simple bags of words. In this paper, we propose an ingredient-instruction dependency tree data structure to represent recipes. The proposed representation allows for more refined comparison of recipes and recipe-parts, and is a step towards semantic representation of recipes. Furthermore, we build a parser that maps recipes into the proposed representation. The parser's edge prediction accuracy of 93.5% improves over a strong baseline of 85.7% (54.5% error reduction).

1 Introduction

Cooking recipes are a specific genre of how-to instructions which have been gaining interest in recent years as they may allow us to discover insights into culinary and cultural preferences. Most of the work studying recipes relies on simple ingredient bag-of-word representations. While such representations may suffice for many purposes, they fail to capture much of the recipes' internal structure. A smaller number of efforts focus on semantic representations of cooking recipes with an eye toward more complex and deep understanding. Natural language processing (NLP) techniques have been used to interpret cooking instructions; however, the results have not been as successful as other language genres due to the unique aspects of cooking recipes.

This paper makes two contributions. First, we propose an ingredient-instruction dependency tree representation of recipe structure. This representation abstracts textual recipes more expressively than bag-of-word representations; and it allows for more nuanced comparisons of recipes. Second, we present a cooking recipe parser that maps text recipes into our proposed ingredient-instruction tree structures. The overall accuracy of predicting edges of our ingredient-instruction trees is 93.5%, beating a strong baseline of 85.7%, and achieving a relative error reduction of 54.5%.

We present next some related work (Section 2) followed by a discussion of the structure of recipes and our representation (Section 3). Section 4 details the recipe parser design, implementation and evaluation.

2 Related Work

There have been many efforts on the processing of cooking recipes using models that range from bags of words to complex semantic representations.

Among the approaches to studying recipes as ingredient bags of words, Ahn et al. (2011) constructed a data-driven flavor network relating ingredients together. Jain et al. (2015) adopted Ahn et al. (2011)'s framework to further analyze culinary practices of specific cultures. Nedovic (2013) examined underlying ingredient groupings from their recipe co-occurrences, using topic modeling techniques (latent Dirichlet allocation), and further improvised novel ingredient combinations using deep belief networks.

Among the structured representation approaches, Tasse and Smith (2008) proposed MILK (Minimal Instruction Language for the Kitchen), a formal language to describe actions required in directive cooking instructions. They used MILK in developing CURD (Carnegie Mellon University Recipe Database), a corpus of manually annotated recipes. Mori et al. (2014) also manually annotated the procedural flow of Japanese cooking recipes using directed acyclic graphs (DAGs) where graph nodes correspond to food ingredients, cooking instruments, and actions. Tasse and Smith (2008) had limited success in parsing into MILK; and Mori et al. (2014) did not report on parsing experiments.

Other studies explored different machine learning and NLP techniques to processing recipes.

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 781–786,
Lisbon, Portugal, 17-21 September 2015.

Text Recipe	
ingredients	
0	6 (2 inch thick) slices French bread
1	1/4 cup ricotta cheese
2	1/4 cup cottage cheese, whipped
3	2 tablespoons lowfat cream cheese
4	2 teaspoons white sugar
5	2 teaspoons vanilla extract
6	3 cups egg substitute
7	1/4 cup evaporated milk
instructions	
0	Cut a pocket in each slice of bread.
1	Open carefully
2	In a large bowl, combine the ricotta, cottage cheese and cream cheese.
3	Add the sugar and flavoring extract and beat until smooth.
4	Spread the mixture evenly into each bread pocket.
5	Beat together the egg substitutes and milk.
6	Dip the slices of bread in the egg mixture.
7	Heat a nonstick pan over medium-high heat.
8	Coat with cooking spray.
9	Cook the toast on each side for about 3 to 4 minutes per side until golden brown.

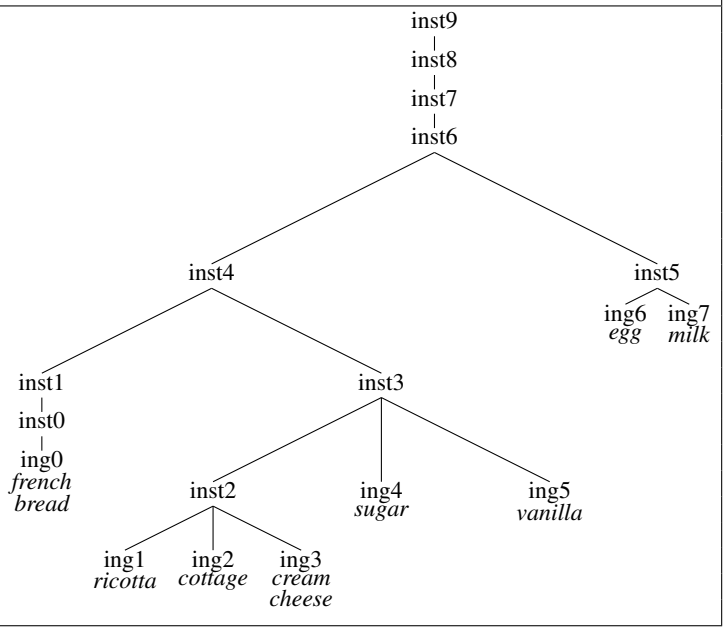


Figure 1: Example of a text recipe for *Surprise-inside French Toast* and its SIMMR representation. **ing**<*index*> and **inst**<*index*> refer to specific ingredients and instructions, respectively.

Mori et al. (2012) applied word segmentation, named entity recognition, and syntactic analysis to extract predicate-argument structures from Japanese recipe instructions as part of an effort to develop complete recipe flow representations. Malmaud et al. (2014) proposed a Markov Decision Process, in which the context of ingredients and tools is propagated along the temporal order of cooking instructions.

Most recently, Abend et al. (2015) proposed an *edge-factored* model to determine the likely temporal order of events based solely on the identity of their predicates and arguments. They demonstrated their approach on recipe text, under the simplifying assumption that such text is also temporally ordered.

In this paper, we present an ingredient-instruction dependency tree representation of recipe structure, which we call SIMMR (Simplified Ingredient Merging Map in Recipes). The SIMMR representation captures the high-level flow of ingredients but without modeling the semantics in each individual instruction unlike other efforts (Tasse and Smith, 2008; Mori et al., 2012; Mori et al., 2014). We create a corpus in our representation by converting the recipes in the CURD corpus (Tasse and Smith, 2008) from MILK to SIMMR. We also develop a parser to generate SIMMR trees from input recipes.

3 Recipe Representation

3.1 Text Recipes

The prototypical text recipe consists of two parts: an ingredient list that declares the food items to process, and a set of instructions that mostly describe the transformations of the ingredients or the actions using the kitchen tools. The instructions relocate, process, combine, and separate ingredients, as well as heat or cool utensils in the recipes. For the most part, the output produced from one instruction feeds as input into another instruction. The list of ingredients can be generalized as special *fetch* instructions whose output feeds as input to one of the cooking instructions.

3.2 SIMMR

Our proposed representation is SIMMR: Simplified Ingredient Merging Map in Recipes. SIMMR represents a recipe as a dependency tree whose leaves (terminal nodes) are the recipe ingredients, and whose internal nodes are the recipe instructions. Figure 1 exemplifies the SIMMR tree of a recipe for *Surprise-inside French Toast*. Indices for all ingredients and instructions are provided here to illustrate mapping between the different parts of the text recipe and its SIMMR tree. The subtree headed by *inst2* indicates that ingredients #1, #2 and #3 (three cheeses) are inputs to instruction #2, whose output is then an input to instruc-

tion #3, together with ingredients #4 and #5 (sugar and vanilla). The SIMMR tree provides additional insights into the structure of recipes, suggesting in the case of this example, that multiple actions can take place in different orders without changing the recipe so long as the order of combinations is not changed. Some instructions simply transforms their input to produce their output, e.g. instructions #1, #7, #8 and #9.

3.3 From MILK to SIMMR

We use MILK commands from the CMU CURD database (Tasse and Smith, 2008) to construct a database of SIMMR trees. The MILK language is a lot more expressive that SIMMR. For example, instruction #2 in Figure 1 translates into create_tool(t0, "large bowl"); combine(ing1, ing2, ing3, ing9, "cheeses", ""); and put(ing9, t0). These details of ingredient processing are abstracted away in SIMMR. To accomplish SIMMR instruction and ingredient linking, we process MILK instructions in order, tracing with MILK id numbers when each ingredient or its transformed or combined form at one instruction node is called for by a subsequent instruction node. The MILK intermediate names for instruction outputs (e.g., "cheeses" above) are not adopted in SIMMR. Additionally, food items that appear in instructions but are not part of the recipe ingredient list text are not included in SIMMR although MILK assigns ingredient ids to them. For example, instruction #8 mentions *cooking spray*, which is not on the ingredient list. As a result *inst8* looks like it has no ingredients coming into it other than the output of *inst7*.

We assume in SIMMR that at most one output is produced from each instruction. One MILK command, *separate*, violates this assumption. For example, the instruction *drain off the fat, and place the mixture into a slow cooker* is MILK-tagged with a *separate* command to dispose fat, and retain the mixture. We ignore the *separate* command which occurs in 3% of all recipes (and 1.3% of the time involves disposing of a separated ingredient, such as *draining off fat*).

Instructions that do not interact with food, such as those calling for preheating the oven or lining the baking sheets, are considered to take the ingredient mixture of the cooking instruction immediately prior and produce the *same* output.

4 Recipe Parser

In this section, we present our SIMMR tree recipe parser. The input and output of the parser correspond to the left hand and right hand sides of Figure 1. We split the parsing process into two phases: first we link ingredients to instructions where the ingredients are first used; then we link instructions to other instructions, where the target instruction uses the source instruction's output. We present the different challenges and solutions employed in each phase below and present evaluation results.

4.1 Experimental Setup

4.1.1 Data

We use a total of 260 recipes downloaded from CMU Recipe Database. MILK tags are used to construct the SIMMR trees as mentioned above. The dataset is randomly split (along recipe boundaries) into training, development, and test sets with ratios of 50%, 20%, and 30%, respectively. The development set is used for tuning parameters. We report here on the test set only. We preprocess the data using standard NLP packages to tokenize, stem, and POS tag the words (De Marneffe et al., 2006; Bird et al., 2009; De Smedt and Daelemans, 2012). We further remove stop words and quantity measures.[1]

4.1.2 Metrics

The evaluation metric is the accuracy of predicting edges in the SIMMR tree. This is comparable to the *attachment score* in dependency parsing. The overall accuracy of the task is computed at the edge level (counting all edges in the data set), and at the recipe level (average accuracy over all recipes).

4.2 Ingredient-Instruction Linking

4.2.1 Challenges

There are several characteristics of recipe text that do not reveal explicit linking of ingredients to instructions that first use them. The ingredients are sometimes referred to by their qualifiers or ontological classes. For example, the ingredient *1 (15 oz) can sliced peaches, drained* may be referred to in the instruction as *canned fruit*. In addition,

[1] Since we start from the MILK representations, the set of ingredients and instructions are clearly identified in each recipe. However, we do not expect this to be the case when starting from raw text recipes, which need to be processed to segment the different ingredients and instructions. We do not attempt this step in this paper and leave it as future work.

	Edge-level Accuracy	Recipe-level Accuracy
Baseline	84.3	84.8
SVMrank	**95.3**	**95.8**

Table 1: Performance of different learning models for ingredient-instruction linking

a number of ingredients are sometimes referred to as a group entity without specific mentions, e.g., *Add the remaining ingredients*, or *mix together the dry ingredients*.

4.2.2 Baseline

For each instruction, in order, we compute all the maximum *stemmed* n-gram chunk matches against all ingredients. For each matching token chunk of the instruction, if it matches only one ingredient, we link the ingredient to the instruction and mark the ingredient as used. Otherwise, we compute the Levenshtein distance between the *unstemmed* surface text of both the ingredient candidate matches and the instruction token chunk, and take the highest matching ingredient. Once an ingredient is used, it is no longer available for linking with subsequent instructions.

4.2.3 Features and Linking

For each ingredient-instruction edge, we provide 1,136 features to classify whether the edge exists or not. The following are some of the most important features we use:

- The baseline decision for the ingredient-instruction edge.
- The number of distinct unigram matches.
- The largest match size.
- The sum of the relative frequency (in the list of instructions) of every word in the largest match (henceforth SRFM).
- The degree of similarity between SRFM and the instruction relative position (InstRP) in the instruction list. The intuition for this feature is that if an ingredient is mentioned more often in the recipe, its first mention is likely to be in an earlier instruction. This feature is computed as $(1 - |SRFM - InstRP|)$.
- The degree of similarity between the ingredient relative position (IngRP) in the ingredient list and the InstRP. This feature captures an observation that earlier listed ingredients are used by earlier instructions. We compute the feature as $(1 - |IngRP - InstRP|)$.
- To model the first word of each instruction (i.e., the directive verb such as *heat* or *mix*), we use a binary term vector whose vocabulary is constructed from all the instruction first words in the training data.
- We model the maximum ingredient-instruction word match using a binary term vector whose vocabulary covers *non-directive recipe words* (NDRW). We construct the NDRW vocabulary by taking the union of the set of words from the compiled food item list of Ahn et al. (2011) and the non-first words of the instructions in the training data.
- If there is a match, we model its surrounding words using a similar binary term vector (NDRW vocabulary). If there is no match, the vector will be empty.
- We use a binary term vector (NDRW vocabulary also) to model the non-first words in the instruction.

We train using Linear SVMrank (Joachims, 2006).[2] The test set results are shown in Table 1. Ranking edges using SVMrank and then picking the best one significantly outperforms the baseline.

4.2.4 Linking Errors

Among the linking errors, three patterns appear. First, stemming reduces the specificity of some of the terms, e.g., one prediction links *baking powder* to an action *bake for 30 minutes*. Second, our approach does not handle negated mentions, e.g. the instruction *mix together the dry ingredients,* **except** *the candies* is incorrectly linked to ingredient *candy*. Finally, the ingredients *flour* and *butter* are specifically part of many erroneous links because they are often used in small quantities to facilitate the cooking process such as board flouring and pan greasing. MILK does not always account for this trivial use of these ingredients and neither does SIMMR.

4.3 Instruction-Instruction Linking

4.3.1 Challenges

Although cooking instructions are written with an implied temporal order, they are not linked in a linear chain. Rather, the instructions describe different cooking *stages*, where the output of apply-

[2] We also experimented Linear SVM and Gaussian Kernel SVM (Pedregosa et al., 2011). We used the distance from the hyperplane to select the best edge among the set of edges connecting an ingredient. However, these techniques underperformed compared to SVMrank.

ing a number of instructions waits until other instructions are finished to be used again, e.g., the output of *instruction #1* in Figure 1 waits until instructions #2 and #3 are done. Sometimes stage switching is explicitly stated as in *Set aside the flour mixture, and combine eggs and oil together*; however, this is not the common case. Furthermore, the waiting output of an earlier stage may be referred to collectively or using the main ingredient which makes linking harder, e.g., referring to the output of an instruction combining *chicken, salt and pepper* as *chicken*.

4.3.2 Baseline

We link the instructions in a linear chain. In the training set, 89% of the instruction-instruction links are to the immediate neighbor.

4.3.3 Features and Linking

For each possible instruction-instruction edge, we provide 1,573 features to classify if the edge exists or not. The features group into three categories.

First are words that suggest cooking stage switching, e.g., ingredient mixture words such as *mixture*, *dough*, and *batter*, or the simple mention of new containers and utensils. They are represented as binary features indicating whether the words appear in either or both instructions along the considered edge, as well as in immediate neighboring instructions. Logic operations (and, or) are further applied to all pairs of these features. Another feature in this group is a binary verb conjugation feature that marks the presence of a past participle verb in the target instruction and a non-past-participle form of the same verb in the source instruction. The intuition here is that an instruction that asks to *chop* an ingredient would be followed later by an instruction containing the word *chopped* when referring to the ingredient.

The second group consists of features that encode whether the n previous or future instructions has m linked ingredients, where n ranges from 1 to 3 and m ranges from 0 to 4.

Finally, the third group deals with term vectors describing the source and target instructions, as well as, the instructions' first words (the directives). The term vector vocabularies used are the same as those discussed above in the ingredient-instruction linking section.

We additionally consider decoupling the features for the case of immediate neighboring instructions from the *further* apart instructions. In the decoupled mode, we effectively double the number of features for each edge with *nil*s used to fill in the gaps. This allows us to learn different models for adjacent and apart instructions. The results are in Table 2 and show that decoupling features in SVMrank is our best setting.

	Edge-level Accuracy	Recipe-level Accuracy
Baseline	87.6	89.5
SVMrank	90.5	92.0
SVMrank - decoupled	**91.3**	**92.4**

Table 2: Performance of different learning models for *instruction*-instruction linking

Examining the weights learned for adjacent and long-distance edge features gives some interesting insights. One example is that the verb conjugation feature has a higher weight for long-distance edges compared to adjacent edges. This is understandable as most adjacent pairs of instructions would not exhibit this feature to express cooking progress: it is redundant to state, *cook the pasta*, and immediately refer to that pasta as the *cooked pasta*.

4.3.4 Linking Errors

As the distance between linked instructions increases, the likelihood of error also increases: while long-distance (i.e., non-adjacent) links constitute 12.3% of the reference, they are 91.7% of the errors. We expect that more training examples of long-distance linking can help address this issue.

4.4 Overall Accuracy

The overall edge-level accuracy of constructing the **full** SIMMR tree is 93.5%, outperforming a baseline of 85.7%, and achieving error reduction of 54.5%.

5 Conclusions and Future Work

We proposed a new ingredient-instruction dependency tree representation to capture the internal structure of cooking recipes, and built a parser for it. Our overall parsing accuracy is 93.5%, outperforming a strong baseline of 85.7%.

We will make our SIMMR database, and our SIMMR parser publicly available. Further, we plan to build on SIMMR to get closer to the MILK representation. We also plan on parsing a large corpus of text recipes to provide structural features on a large scale that will allow us to discover new patterns of similarity across and within cuisines, as well as generate new recipes.

References

Omri Abend, Shay B Cohen, and Mark Steedman. 2015. Lexical event ordering with an edge-factored model. In *Proceedings of NAACL*.

Yong-Yeol Ahn, Sebastian E Ahnert, James P Bagrow, and Albert-László Barabási. 2011. Flavor network and the principles of food pairing. *Scientific reports*, 1.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python*. "O'Reilly Media, Inc.".

Marie-Catherine De Marneffe, Bill MacCartney, Christopher D Manning, et al. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC*, volume 6, pages 449–454.

Tom De Smedt and Walter Daelemans. 2012. Pattern for python. *The Journal of Machine Learning Research*, 13(1):2063–2067.

Anupam Jain, Ganesh Bagler, et al. 2015. Spices form the basis of food pairing in indian cuisine. *arXiv preprint arXiv:1502.03815*.

Thorsten Joachims. 2006. Training linear svms in linear time. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 217–226. ACM.

Jon Malmaud, Earl J Wagner, Nancy Chang, and Kevin Murphy. 2014. Cooking with semantics. *ACL 2014*, page 33.

Shinsuke Mori, Tetsuro Sasada, Yoko Yamakata, and Koichiro Yoshino. 2012. A machine learning approach to recipe text processing. In *Proc. of the 1st Cooking with Computer Workshop*, pages 29–34.

Shinsuke Mori, Hirokuni Maeta, Yoko Yamakata, and Tetsuro Sasada. 2014. Flow graph corpus from recipe texts. In *Proceedings of the Nineth International Conference on Language Resources and Evaluation*, pages 2370–2377.

V Nedovic. 2013. Learning recipe ingredient space using generative probabilistic models. In *Proceedings of Cooking with Computers Workshop (CwC)*, volume 1, pages 13–18.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Dan Tasse and Noah A Smith. 2008. Sour cream: Toward semantic processing of recipes. Technical report, Technical Report CMU-LTI-08-005, Carnegie Mellon University, Pittsburgh, PA.

TSDPMM: Incorporating Prior Topic Knowledge into Dirichlet Process Mixture Models for Text Clustering

Linmei Hu[†], **Juanzi Li**[†], **Xiaoli Li**[‡], **Chao Shao**[†], **Xuzhong Wang**[§]
[†] Dept. of Computer Sci. and Tech., Tsinghua University, China
[‡] Institute for Infocomm Research(I2R), A*STAR, Singapore
[§] State Key Laboratory of Math. Eng. and Advanced Computing, China
{hulinmei1991, lijuanzi2008}@gmail.com
xlli@i2r.a-star.edu.sg, {birdlinux, koodoneko}@gmail.com

Abstract

Dirichlet process mixture model (DPMM) has great potential for detecting the underlying structure of data. Extensive studies have applied it for text clustering in terms of topics. However, due to the unsupervised nature, the topic clusters are always less satisfactory. Considering that people often have some prior knowledge about which potential topics should exist in given data, we aim to incorporate such knowledge into the DPMM to improve text clustering. We propose a novel model TSDPMM based on a new seeded Pólya urn scheme. Experimental results on document clustering across three datasets demonstrate our proposed TSDPMM significantly outperforms state-of-the-art DPMM model and can be applied in a lifelong learning framework.

1 Introduction

Dirichlet process mixture model (DPMM) (Neal, 2000) has been used in detecting the underlying structure in data. For example, (Vlachos et al., 2008; Vlachos et al., 2009) applied it to lexical-semantic verb clustering. (Wang et al., 2011; Huang et al., 2013; Yin and Wang, 2014) applied it for text clustering in terms of their topics. While DPMM achieved some promising results, it can still sometimes produce unsatisfactory topic clusters due to its unsupervised nature.

On the other hand, people often have prior knowledge about what potential topics should exist in a given text corpus. Take an earthquake event corpus as an example. The topics, such as "*casualties and damages*", "*rescue*" and "*government reaction*", called prior topics, are expected to occur in the corpus according to our common knowledge (e.g., the topics automatically learned from previous events using topic modeling (Ahmed and Xing, 2008)) or external resources (e.g., table of contents at Wikipedia event pages [1]). Similarly, in academic fields, "call for papers (CFP)" of conferences [2] lists main topics that conference organizers would like to focus on. Clearly, these prior topics can be represented as sets of words, which are available in many real-world applications. They can serve as weakly supervised information to enhance the unsupervised DPMM for text clustering.

Standard DPMM (Neal, 2000; Ranganathan, 2006) lacks a mechanism for incorporating prior knowledge. Some existing work (Vlachos et al., 2008; Vlachos et al., 2009) added knowledge of observed *instance*-level constraints (*must-links* and *cannot-links* between documents) to DPMM. (Ahmed and Xing, 2008) proposed recurrent Chinese Restaurant Process to incorporate *previous* documents with known topic clusters. We focus on incorporating *topic*-level knowledge, which is more challenging, as seed/prior topics could be latent rather than observable.

Particularly, we construct our novel TSDPMM (Topic Seeded DPMM) based on a principled seeded Pólya urn (sPU) scheme. Our model inherits the nonparametric property of DPMM and has additional technical merits. Importantly, our model is encouraged but not forced to find evidences of seed topics. Therefore, it has freedom to discover new topics beyond prior topics, as well as to detect which prior topics are not covered by current data. It is thus convenient to observe topic variations between prior topics and newly mined topics. Experimental results on document clustering across three corpora demonstrate that our model effectively incorporates prior topics, and significantly outperforms state-of-the-art DPMM model. Particularly, our TSDPMM can be applied in a lifelong learning framework which enables the prior

[1] e.g., http://en.wikipedia.org/wiki/2010_Chile_earthquake
[2] e.g., https://nips.cc/Conferences/2014/CallForPapers

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 787–792,
Lisbon, Portugal, 17-21 September 2015.

topic knowledge to evolve as more and more data are observed.

2 Topic Seeded DPMM

In this section, we first introduce the standard DPMM model for document clustering in terms of topics. Then we describe how to incorporate seed/prior topics into the model using a seeded Pólya urn (sPU) scheme, which gives us our novel TSDPMM model (Topic Seeded DPMM). Finally, we present the model inference.

2.1 DPMM

The DPMM (Antoniak, 1974) as a non-parametric model assumes the given data is governed by an infinite number of components where only a fraction of these components are activated by the data. Figure 1 illustrates the DPMM graphical model and its generative process of a document x_i. First, we sample a topic $\theta_i = \{\theta_{ij}\}_{j=1}^{j=|V|}$ (a multinomial distribution over words belonging to the vocabulary V) for the document x_i according to a Dirichlet Process (DP) $G \sim DP(\alpha, G_0)$, where $\alpha > 0$ is a concentration parameter and the base measure $G_0 = Dir(\vec{\beta})$ can be considered as a prior distribution for θ. Consider the document x_i as a bag of words, given the topic θ_i, the generative distribution F is a given likelihood function parameterized by θ. We define F as $p(x_i|\theta_i) = \prod_{j=1}^{|x_i|} p(x_{ij}|\theta_i)$, where x_{ij} is the j_{th} word in x_i. Note that the DPMM assumes each document can be assigned to one topic cluster only.

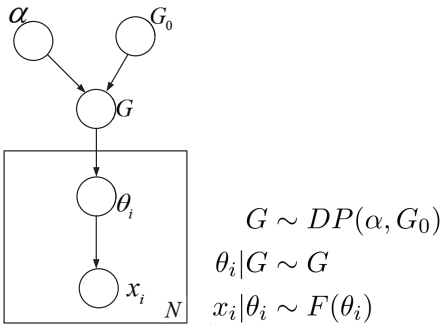


Figure 1: Graphical Representation of DPMM.

The DP process of DPMM, according to which topic θ_i for a document x_i is drawn, can be explained by the popular metaphor of Pólya urn (PU) scheme (Blackwell and MacQueen, 1973), equivalent to the Chinese Restaurant Process (Ahmed and Xing, 2008). The PU scheme works on balls (documents) and colors (topics). It starts with an empty urn. With probability proportional to α, we draw $\theta_i \sim G_0$, and add a ball of this color to the urn. With probability proportional to $i-1$ (i.e., the current number of balls in the urn), we draw a ball at random from the urn, observe its color θ_i and replace the ball with two balls of the same color. In this way, we draw topic θ_i for document x_i. As shown in the process, the prior probability of assigning a document to a topic is proportional to the number of documents already assigned to the topic. As a result, the DPMM exhibits the "rich get richer" property.

2.2 TSDPMM: Incorporating Seed Topics

In this section, we describe our proposed algorithm to incorporate prior seed topics into the DPMM. A prior/seed topic k is represented by a vector $\vec{N}_k^{(0)}$ (word frequencies under the topic). We can obtain the prior topics represented by $\vec{N}_k^{(0)}$ from past learning of topic models or external resources such as Wikipedia and "CFP". Assuming we have $K^{(0)}$ prior topics, we use the parameter $\vec{\alpha}^{(0)} = \{\alpha_k^{(0)}\}_{k=1}^{K^{(0)}}$ to control our confidence about how likely each prior topic exists. Let us go back to Pólya urn (PU) scheme, where a prior topic can be taken as a known color. We extend the PU scheme to incorporate prior topics, which gives the **sPU** (seeded Pólya Urn) scheme. The sPU scheme can be described as follows:

- We start with an urn with $\alpha_k^{(0)}$ balls of each known color $k \in \{1, ..., K^{(0)}\}$.
- With a probability proportional to α, we draw $\theta_i \sim G_0$ and add a ball of this color to the urn.
- With probability proportional to $i - 1 + \sum_{k=1}^{K^{(0)}} \alpha_k^{(0)}$, we draw a random ball from the urn, and replace the ball with two balls of the same color.

As shown in the above process, instead of starting with an empty urn in DPMM, we assume that the urn already has certain balls of known colors. In this way, we incorporate the prior seed topics. The number of initial balls (documents) $\alpha_k^{(0)}$ controls how likely the topic k exists. We can use different values of $\alpha_k^{(0)}$ for prior topics with different confidence levels. This sPU scheme gives our novel model TSDPMM (Topic Seeded DPMM) incorporating prior topics. The TSDPMM has

similar graphical representation as DPMM (Figure 1), except the introduction of hyper-parameter $\vec{\alpha}^{(0)}$. We then present a collapsed gibbs sampling algorithm for model inference as follows.

TSDPMM Inference. The model inference is described in detail in Algorithm 1. It first initializes all documents with random topic clusters. Then it iteratively updates the topic cluster assignments of documents according to the conditional probabilities (Eq.1) until convergence. Eq.1 can be derived as:

$$p(z_i|\vec{Z}_{-i},\vec{X}) \propto p(z_i|\vec{Z}_{-i},\alpha,\vec{\alpha}^{(0)})p(x_i|\vec{X}_{-i},\vec{Z},\vec{\beta}) \quad (1)$$

where z_i is the topic assignment of observation x_i, $\vec{X}$ is the given document corpus, and $\vec{Z}_{-i}$ are $\vec{X}_{-i}$ are the set of topic assignments and the corpus excluding the i_{th} observation x_i, respectively.

Algorithm 1: Collapsed Gibbs Sampling

Input: Document dataset $\vec{X} = \{x_i\}_{i=1}^{m}$, prior topics $\{\vec{N}_k^{(0)}\}_{k=1}^{K^{(0)}}$, parameter $\vec{\alpha}^{(0)}$

Output: Topic assignments $\vec{Z}$ of all documents

Initialize the topic assignments $\vec{Z}$ based on prior topics randomly;

repeat

  Select a document $x_i \in \vec{X}$ randomly

  Fix the other topic assignments $\vec{Z}_{-i}$

  Assign a new value to z_i:$z_i \sim p(z_i|\vec{Z}_{-i},\vec{X})$(Eq. 1)

until *Convergence*;

In Eq.1, the first item $p(z_i{=}k|\vec{Z}_{-i},\alpha,\vec{\alpha}^{(0)})$ denotes a prior probability of $z_i{=}k$, which is proportional to the number of documents already assigned to it. If k is a **prior** topic, it is proportional to $n_{k,-i} + \alpha_k^{(0)}$, where $n_{k,-i}$ is the number of documents of topic k excluding the current document x_i. If k is an **existing** (not prior) topic, it is proportional to $n_{k,-i}$. If k is a **new** topic, the probability is proportional to α. The second item $p(x_i|\vec{X}_{-i},\vec{Z}_{-i},z_i = k,\vec{\beta})$ is the likelihood of x_i given $\vec{X}_{-i}$, $\vec{Z}_{-i}$ and $z_i{=}k$. They can be derived as $p(x_i|\vec{X}_{-i},\vec{Z}_{-i},z_i = k,\vec{\beta}) \propto \frac{p(\vec{X}|\vec{Z},\vec{\beta})}{p(\vec{X}_{-i}|\vec{Z}_{-i},\vec{\beta})}$ where $p(\vec{X}|\vec{Z},\vec{\beta}) = \int p(\vec{X}|\vec{Z},\Theta)p(\Theta|\vec{\beta})d\Theta$. As $p(\Theta|\vec{\beta})$ is a Dirichlet distribution and $p(\vec{X}|\vec{Z},\Theta)$ is a multinomial distribution, we can get $p(\vec{X}|\vec{Z}){=}\prod_{k=1}^{K}\frac{\Delta(\vec{N}_k+\vec{\beta})}{\Delta(\vec{\beta})}$, where $\vec{N}_k = \{N_{k,w}\}_{w=1}^{V}$ and $N_{k,w}$ is the number of occurrences of word w in the k_{th} topic. Here, we adopt the function Δ in (Heinrich, 2009), and we have $\Delta(\vec{\beta}) = \frac{\prod_{w=1}^{V}\Gamma(\beta)}{\Gamma(\sum_{w=1}^{V})\beta}$ and $\Delta(\vec{N}_k+\vec{\beta}) = \frac{\prod_{w=1}^{V}\Gamma(N_{k,w}+\beta)}{\Gamma(\sum_{w=1}^{V}(N_{k,w}+\beta))}$. Finally, we can derive:

$$p(z_i = k|\vec{Z}_{-i},\vec{X}) \propto \begin{cases} (n_{k,-i}+\alpha^{(0)})\cdot\frac{\Delta(\vec{N}_{.,i}+\vec{N}_{k,-i}+\vec{N}_k^{(0)}+\vec{\beta})}{\Delta(\vec{N}_{k,-i}+\vec{N}_k^{(0)}+\vec{\beta})} & \text{prior} \\ n_{k,-i}\cdot\frac{\Delta(\vec{N}_{.,i}+\vec{N}_{k,-i}+\vec{\beta})}{\Delta(\vec{N}_{k,-i}+\vec{\beta})} & \text{existing} \\ \alpha\cdot\frac{\Delta(\vec{N}_{.,i}+\vec{\beta})}{\Delta(\vec{\beta})} & \text{new}, \end{cases}$$

where $\vec{N}_{k,-i}$ is a vector with the word counts for all the documents assigned to topic k excluding x_i, $\vec{N}_{.,i}$ and $\vec{N}_k^{(0)}$ are vectors with word counts in document x_i and in all the documents assigned to k in prior knowledge respectively. According to this equation, documents are likely to go into clusters which are bigger and give higher likelihood of the documents. When the Gibbs sampler converges, we obtain topic cluster assignments of all the documents. Different from DPMM inference process in which topics are removed when no documents is assigned to them, TSDPMM inference can retain prior topics all the time due to the initial number of documents $\vec{\alpha}^{(0)}$, making it able to track prior topics, as well as to detect new topics.

3 Experiments

We evaluate our proposed TSDPMM model for document clustering on 3 datasets where each cluster corresponds to a topic. We implement both DPMM and TSDPMM models — their source codes are available at `https://github.com/newsminer/DPMM_and_TSDPMM`.

3.1 Datasets

We collect machine learning conference NIPS datasets composed of paper titles and abstracts from 2012 to 2014 – each year includes 342, 360 and 411 documents respectively. They are named as NIPS-12, NIPS-13 and NIPS-14.

We also employ the standard *benchmark* news datasets, including 20 Newsgroups [3] and Reuters-21578. As news is often timely reported, we choose three continuous days with the largest number of documents in 20 Newsgroups (i.e. 11, 12 and 13 May) and Reuters-21578 (i.e. 3, 4 and 5 March) for our experiments. These datasets are

[3] http://people.csail.mit.edu/jrennie/20Newsgroups/

denoted as 20N-1, 20N-2, 20N-3 (including 103, 96, 106 documents) and Reu-1, Reu-2, Reu-3 (including 282, 249, 207 documents), respectively.

For all the datasets, we conduct the following preprocessing: (1) Convert letters into lowercase; (2) Remove non-Latin characters and stop words; (3) Remove words with *document frequency* < 2.

3.2 Experimental Setup

We take the standard DPMM as our baseline method and compare it with our proposed TSDPMM model using *different* prior knowledge obtained with different manners.

For NIPS datasets, we use two kinds of prior knowledge: one is the topics learned by DPMM from *previous* year's dataset; the other one is from an *external* resource "CFP" [4] (10 topics, same for each year). We name them as TSDPMM-P and TSDPMM-E respectively. As the topic descriptions in "CFP" are sparse, we repeat each topic description by ten times and then represent a topic with the words with word frequencies in its description text.

For both 20 Newsgroups and Reuters datasets, we use prior knowledge learned by DPMM from the previous day's dataset. Furthermore, to test if we can improve the results *continuously* by applying TSDPMM, every time when we model a new dataset, we incorporate prior topics learned by TSDPMM from previous day's dataset, similar to lifelong learning (Chen and Liu, 2014; Thrun, 1998). We call this model as TSDPMM-L.

Parameter Setting. Following a previous work (Vlachos et al., 2009), we set the hyper-parameters α=1, $\vec{\alpha}^{(0)}$=$\{1.0\}$, $\vec{\beta}$ =$\{1.0\}$. We run Gibbs sampler for 100 iterations and stop the iteration once the log-likelihood of the training data converges.

Evaluation. The widely used NMI (normalized mutual information) measure (Dom, 2002), has been employed to evaluate document clustering results. The higher a value of NMI, the better a clustering result is. However, NMI needs true class labels for documents, and can only be applied to our benchmark news datasets. For NIPS datasets without true labels, we use the measure of *perplexity*, as defined in (Blei et al., 2003), to test per-word likelihood of the datasets. The lower the perplexity, the better a model fits the data.

[4] https://nips.cc/Conferences/2014/CallForPapers

3.3 Results

Table 1 shows the average perplexity values of five runs of 3 models on NIPS datasets. It shows that both TSDPMM-P and TSDPMM-E, leveraging prior topics from previous learning and "CFP" significantly outperform DPMM. In addition, TSDPMM-E achieves lower performance than TSDPMM-P due to its lower quality of prior topics directly obtained from "CFP", compared to higher quality topics from past learning. We may improve "CFP" knowledge by extending it with related texts from search engines or Wikipedia using keywords in "CFP" in future work.

An insight of our clustering results on NIPS-14 dataset suggests that most prior topics in 2013 are covered again in 2014 (consistent topics), except a few missing topics such as "*lasso for Bayesian networks*". Additionally, some newly evolved topics in 2014, e.g. "*monte carlo particle filtering*" and "*nash games*", are successfully discovered by our proposed model.

Models	NIPS-12	NIPS-13	NIPS-14
DPMM	321.7	317.1	362.9
TSDPMM-P	**290.1**	**298.7**	**346.8**
TSDPMM-E	307.5	315.6	360.1

Table 1: Average perplexity of different models on NIPS.

Table 2 illustrates the average NMI values of five runs of DPMM, TSDPMM and TSDPMM-L on news datasets. The results show that TSDPMM using prior topics learnt by DPMM outperforms DPMM (on average **+5.8%**; p <0.025 with *t-test*). Additionally, TSDPMM-L, which continuously uses prior topics learnt by TSDPMM from previous dataset, further outperforms TSDPMM (on average **+3.2%**; p <0.025 with *t-test*). Note TSDPMM-L uses TSDPMM results of 20N-1 and Reu-1 as prior knowledge for the first time, so there are no TSDPMM-L results for the first days in Table 2 for 20N-1 and Reu-1 respectively.

3.4 Discussion

The experimental results across 3 datasets have demonstrated that our proposed models can improve DPMM model by incorporating prior topic knowledge, and the higher-quality knowledge will lead to better results. By applying our TSDPMM in a lifelong continuous learning framework, namely TSDPMM-L, can further improve

Models	20N-1	20N-2	20N-3	Reu-1	Reu-2	Reus-3
DPMM	0.610	0.537	0.590	0.509	0.647	0.653
TSDPMM	0.645	0.610	0.681	0.648	0.654	0.655
TSDPMM-L	—	**0.681**	**0.697**	—	**0.689**	**0.656**

Table 2: Average NMI of different models on news datasets.

text clustering due to the better prior topic knowledge obtained in the evolving environment.

4 Related Work

Our work is related to papers (Vlachos et al., 2008; Vlachos et al., 2009), which added supervision (*instance*-level must-links or cannot-links between documents) to the DPMM. (Ahmed and Xing, 2008) proposed recurrent Chinese Restaurant Process to incorporate previous documents with known topic clusters. However, our work is very different as we focus on how to incorporate latent *topic*-level prior knowledge. We model prior topics as known colors that have a certain probability proportional to $\alpha_k^{(0)}$ to be assigned to a document. In addition, our inference mechanism subsequently takes the prior knowledge into consideration for automatically assigning topics to documents.

Some existing studies such as (Ramage et al., 2009; Andrzejewski et al., 2009; Jagarlamudi et al., 2012; Andrzejewski et al., 2011) worked on incorporating prior lexical or domain knowledge into LDA. Different from all these work, we focus on the nonparametric model DPMM and propose to incorporate the prior topic knowledge obtained in multiple ways.

5 Conclusion

In this paper, we propose a novel problem of incorporating prior topics into DPMM model and address it through a simple yet principled seeded Pólya urn scheme. We show that the topic knowledge can be obtained in multiple ways. Experiments on document clustering across 3 datasets demonstrate our proposed model can effectively incorporate the prior topic knowledge and significantly enhance the standard DPMM for text clustering. In future work, we will study how to discover overlapping clusters, i.e., allowing one document to be grouped into multiple topic clusters. We will also explore how to incorporate prior knowledge about topic relations (such as causation and correlation) into topic modeling.

Acknowledgments

The work is supported by 973 Program (No. 2014CB340504), NSFC-ANR (No. 61261130588), Tsinghua University Initiative Scientific Research Program (No. 20131089256), Science and Technology Support Program (No. 2014BAK04B00), and THU-NUS NExT Co-Lab.

References

Amr Ahmed and Eric P Xing. 2008. Dynamic nonparametric mixture models and the recurrent chinese restaurant process: with applications to evolutionary clustering. In *SDM*, pages 219–230. SIAM.

David Andrzejewski, Xiaojin Zhu, and Mark Craven. 2009. Incorporating domain knowledge into topic modeling via dirichlet forest priors. In *Proceedings of the 26th Annual ICML*, pages 25–32. ACM.

David Andrzejewski, Xiaojin Zhu, Mark Craven, and Benjamin Recht. 2011. A framework for incorporating general domain knowledge into latent dirichlet allocation using first-order logic. In *Proceedings-IJCAI*, volume 22, page 1171.

Charles E Antoniak. 1974. Mixtures of dirichlet processes with applications to bayesian nonparametric problems. *The annals of statistics*, pages 1152–1174.

David Blackwell and James B MacQueen. 1973. Ferguson distributions via pólya urn schemes. *The annals of statistics*, pages 353–355.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research*, 3:993–1022.

Zhiyuan Chen and Bing Liu. 2014. Mining topics in documents: standing on the shoulders of big data. In *Proceedings of the 20th ACM SIGKDD international conference*, pages 1116–1125. ACM.

Byron E Dom. 2002. An information-theoretic external cluster-validity measure. In *Proceedings of the Eighteenth conference on Uncertainty in artificial intelligence*, pages 137–145. Morgan Kaufmann Publishers Inc.

Gregor Heinrich. 2009. Parameter estimation for text analysis. Technical report, vsonix GmbH and University of Leipzig.

Ruizhang Huang, Guan Yu, Zhaojun Wang, Jun Zhang, and Liangxing Shi. 2013. Dirichlet process mixture model for document clustering with feature partition. *Knowledge and Data Engineering,*, 25(8):1748–1759.

Jagadeesh Jagarlamudi, Hal Daumé III, and Raghavendra Udupa. 2012. Incorporating lexical priors into topic models. In *Proceedings of the 13th Conference of the European Chapter of the ACL*, pages 204–213. Association for Computational Linguistics.

Radford M Neal. 2000. Markov chain sampling methods for dirichlet process mixture models. *Journal of computational and graphical statistics*, 9(2):249–265.

Daniel Ramage, David Leo Wright Hall, Ramesh Nallapati, and Christopher D. Manning. 2009. Labeled LDA: A supervised topic model for credit attribution in multi-labeled corpora. In *Proceedings of the 2009 Conference on EMNLP*, pages 248–256.

Ananth Ranganathan. 2006. The dirichlet process mixture (dpm) model. Technical report, Citeseer.

Sebastian Thrun. 1998. Lifelong learning algorithms. In *Learning to learn*, pages 181–209. Springer.

Andreas Vlachos, Zoubin Ghahramani, and Anna Korhonen. 2008. Dirichlet process mixture models for verb clustering. In *Proceedings of the ICML workshop on Prior Knowledge for Text and Language*.

Andreas Vlachos, Anna Korhonen, and Zoubin Ghahramani. 2009. Unsupervised and constrained dirichlet process mixture models for verb clustering. In *Proceedings of the workshop on geometrical models of natural language semantics*, pages 74–82. Association for Computational Linguistics.

Chan Wang, Caixia Yuan, Xiaojie Wang, and Wenwei Xue. 2011. Dirichlet process mixture models based topic identification for short text streams. In *NLP-KE*, pages 80–87. IEEE.

Jianhua Yin and Jianyong Wang. 2014. A dirichlet multinomial mixture model-based approach for short text clustering. In *Proceedings of the 20th ACM SIGKDD*, pages 233–242. ACM.

Sentence Modeling with Gated Recursive Neural Network

Xinchi Chen, Xipeng Qiu*, Chenxi Zhu, Shiyu Wu, Xuanjing Huang
Shanghai Key Laboratory of Intelligent Information Processing, Fudan University
School of Computer Science, Fudan University
825 Zhangheng Road, Shanghai, China
{xinchichen13,xpqiu,czhu13,syu13,xjhuang}@fudan.edu.cn

Abstract

Recently, neural network based sentence modeling methods have achieved great progress. Among these methods, the recursive neural networks (RecNNs) can effectively model the combination of the words in sentence. However, RecNNs need a given external topological structure, like syntactic tree. In this paper, we propose a gated recursive neural network (GRNN) to model sentences, which employs a full binary tree (FBT) structure to control the combinations in recursive structure. By introducing two kinds of gates, our model can better model the complicated combinations of features. Experiments on three text classification datasets show the effectiveness of our model.

1 Introduction

Recently, neural network based sentence modeling approaches have been increasingly focused on for their ability to minimize the efforts in feature engineering, such as Neural Bag-of-Words (NBoW), Recurrent Neural Network (RNN) (Mikolov et al., 2010), Recursive Neural Network (RecNN) (Pollack, 1990; Socher et al., 2013b; Socher et al., 2012) and Convolutional Neural Network (CNN) (Kalchbrenner et al., 2014; Hu et al., 2014).

Among these methods, recursive neural networks (RecNNs) have shown their excellent abilities to model the word combinations in sentence. However, RecNNs require a pre-defined topological structure, like parse tree, to encode sentence, which limits the scope of its application. Cho et al. (2014) proposed the gated recursive convolutional neural network (grConv) by utilizing the directed acyclic graph (DAG) structure instead of parse tree to model sentences. However, DAG structure is relatively complicated. The number of the hidden neurons quadraticly increases with the length of sentences so that grConv cannot effectively deal with long sentences.

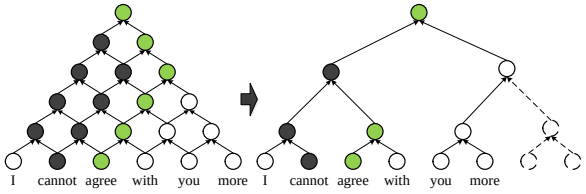


Figure 1: Example of Gated Recursive Neural Networks (GRNNs). Left is a GRNN using a directed acyclic graph (DAG) structure. Right is a GRNN using a full binary tree (FBT) structure. (The green nodes, gray nodes and white nodes illustrate the positive, negative and neutral sentiments respectively.)

Inspired by grConv, we propose a gated recursive neural network (GRNN) for sentence modeling. Different with grConv, we use the full binary tree (FBT) as the topological structure to recursively model the word combinations, as shown in Figure 1. The number of the hidden neurons linearly increases with the length of sentences. Another difference is that we introduce two kinds of gates, reset and update gates (Chung et al., 2014), to control the combinations in recursive structure. With these two gating mechanisms, our model can better model the complicated combinations of features and capture the long dependency interactions.

In our previous works, we have investigated several different topological structures (tree and directed acyclic graph) to recursively model the semantic composition from the bottom layer to the top layer, and applied them on Chinese word segmentation (Chen et al., 2015a) and dependency parsing (Chen et al., 2015b) tasks. However, these structures are not suitable for modeling sentences.

*Corresponding author.

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 793–798,
Lisbon, Portugal, 17-21 September 2015.

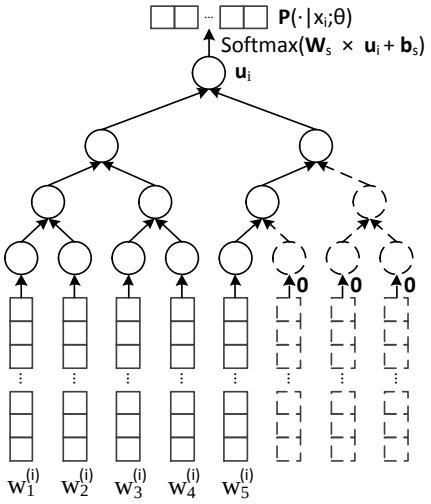


Figure 2: Architecture of Gated Recursive Neural Network (GRNN).

In this paper, we adopt the full binary tree as the topological structure to reduce the model complexity.

Experiments on the Stanford Sentiment Treebank dataset (Socher et al., 2013b) and the TREC questions dataset (Li and Roth, 2002) show the effectiveness of our approach.

2 Gated Recursive Neural Network

2.1 Architecture

The recursive neural network (RecNN) need a topological structure to model a sentence, such as a syntactic tree. In this paper, we use a full binary tree (FBT), as showing in Figure 2, to model the combinations of features for a given sentence.

In fact, the FBT structure can model the combinations of features by continuously mixing the information from the bottom layer to the top layer. Each neuron can be regarded as a complicated feature composition of its governed sub-sentence. When the children nodes combine into their parent node, the combination information of two children nodes is also merged and preserved by their parent node. As shown in Figure 2, we put all-zero padding vectors after the last word of the sentence until the length of $2^{\lceil log_2^n \rceil}$, where n is the length of the given sentence.

Inspired by the success of the gate mechanism of Chung et al. (2014), we further propose a gated recursive neural network (GRNN) by introducing two kinds of gates, namely "reset gate" and "update gate". Specifically, there are two reset gates, $\mathbf{r}_L$ and $\mathbf{r}_R$, partially reading the information from

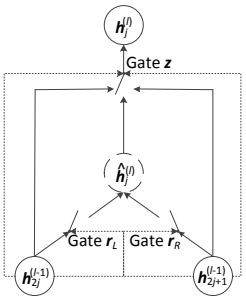


Figure 3: Our proposed gated recursive unit.

left child and right child respectively. And the update gates $\mathbf{z}_N$, $\mathbf{z}_L$ and $\mathbf{z}_R$ decide what to preserve when combining the children's information. Intuitively, these gates seem to decide how to update and exploit the combination information.

In the case of text classification, for each given sentence $x_i = w^{(i)}_{1:N(i)}$ and the corresponding class y_i, we first represent each word $w^{(i)}_j$ into its corresponding embedding $\mathbf{w}_{w^{(i)}_j} \in \mathbb{R}^d$, where $N(i)$ indicates the length of i-th sentence and d is dimensionality of word embeddings. Then, the embeddings are sent to the first layer of GRNN as inputs, whose outputs are recursively applied to upper layers until it outputs a single fixed-length vector. Next, we receive the class distribution $\mathrm{P}(\cdot|x_i;\theta)$ for the given sentence x_i by a softmax transformation of $\mathbf{u}_i$, where $\mathbf{u}_i$ is the top node of the network (a fixed length vectorial representation):

$$\mathrm{P}(\cdot|x_i;\theta) = \mathrm{softmax}(\mathbf{W}_s \times \mathbf{u}_i + \mathbf{b}_s), \qquad (1)$$

where $\mathbf{b}_s \in \mathbb{R}^{|T|}$, $\mathbf{W}_s \in \mathbb{R}^{|T| \times d}$. d is the dimensionality of the top node $\mathbf{u}_i$, which is same with the word embedding size and T represents the set of possible classes. θ represents the parameter set.

2.2 Gated Recursive Unit

GRNN consists of the minimal structures, gated recursive units, as showing in Figure 3.

By assuming that the length of sentence is n, we will have recursion layer $l \in [1, \lceil log_2^n \rceil + 1]$, where symbol $\lceil q \rceil$ indicates the minimal integer $q^* \geq q$. At each recursion layer l, the activation of the j-th ($j \in [0, 2^{\lceil log_2^n \rceil - l}$)) hidden node $\mathbf{h}^{(l)}_j \in \mathbb{R}^d$ is computed as

$$\mathbf{h}^{(l)}_j = \begin{cases} \mathbf{z}_N \odot \hat{\mathbf{h}}^l_j + \mathbf{z}_L \odot \mathbf{h}^{l-1}_{2j} + \mathbf{z}_R \odot \mathbf{h}^{l-1}_{2j+1}, & l > 1, \\ \text{corresponding word embedding}, & l = 1, \end{cases} \qquad (2)$$

where $\mathbf{z}_N$, $\mathbf{z}_L$ and $\mathbf{z}_R \in \mathbb{R}^d$ are update gates for new activation $\hat{\mathbf{h}}_j^l$, left child node $\mathbf{h}_{2j}^{l-1}$ and right child node $\mathbf{h}_{2j+1}^{l-1}$ respectively, and $\odot$ indicates element-wise multiplication.

The update gates can be formalized as:

$$\mathbf{z} = \begin{bmatrix} \mathbf{z}_N \\ \mathbf{z}_L \\ \mathbf{z}_R \end{bmatrix} = \begin{bmatrix} 1/Z \\ 1/Z \\ 1/Z \end{bmatrix} \odot \exp(\mathbf{U} \begin{bmatrix} \hat{\mathbf{h}}_j^l \\ \mathbf{h}_{2j}^{l-1} \\ \mathbf{h}_{2j+1}^{l-1} \end{bmatrix}), \quad (3)$$

where $\mathbf{U} \in \mathbb{R}^{3d \times 3d}$ is the coefficient of update gates, and $Z \in \mathbb{R}^d$ is the vector of the normalization coefficients,

$$Z_k = \sum_{i=1}^{3} [\exp(\mathbf{U} \begin{bmatrix} \hat{\mathbf{h}}_j^l \\ \mathbf{h}_{2j}^{l-1} \\ \mathbf{h}_{2j+1}^{l-1} \end{bmatrix})]_{d \times (i-1)+k}, \quad (4)$$

where $1 \leq k \leq d$.

The new activation $\hat{\mathbf{h}}_j^l$ is computed as:

$$\hat{\mathbf{h}}_j^l = \tanh(\mathbf{W}_{\hat{\mathbf{h}}} \begin{bmatrix} \mathbf{r}_L \odot \mathbf{h}_{2j}^{l-1} \\ \mathbf{r}_R \odot \mathbf{h}_{2j+1}^{l-1} \end{bmatrix}), \quad (5)$$

where $\mathbf{W}_{\hat{\mathbf{h}}} \in \mathbb{R}^{d \times 2d}$, $\mathbf{r}_L \in \mathbb{R}^d$, $\mathbf{r}_R \in \mathbb{R}^d$. $\mathbf{r}_L$ and $\mathbf{r}_R$ are the reset gates for left child node $\mathbf{h}_{2j}^{l-1}$ and right child node $\mathbf{h}_{2j+1}^{l-1}$ respectively, which can be formalized as:

$$\begin{bmatrix} \mathbf{r}_L \\ \mathbf{r}_R \end{bmatrix} = \sigma(\mathbf{G} \begin{bmatrix} \mathbf{h}_{2j}^{l-1} \\ \mathbf{h}_{2j+1}^{l-1} \end{bmatrix}), \quad (6)$$

where $\mathbf{G} \in \mathbb{R}^{2d \times 2d}$ is the coefficient of two reset gates and σ indicates the sigmoid function.

Intuiativly, the reset gates control how to select the output information of the left and right children, which result to the current new activation $\hat{\mathbf{h}}$. By the update gates, the activation of a parent neuron can be regarded as a choice among the the current new activation $\hat{\mathbf{h}}$, the left child, and the right child. This choice allows the overall structure to change adaptively with respect to the inputs.

This gate mechanism is effective to model the combinations of features.

2.3 Training

We use the Maximum Likelihood (ML) criterion to train our model. Given training set (x_i, y_i) and the parameter set of our model θ, the goal is to minimize the loss function:

$$J(\theta) = -\frac{1}{m} \sum_{i=1}^{m} \log \mathrm{P}(y_i|x_i;\theta) + \frac{\lambda}{2m}\|\theta\|_2^2, \quad (7)$$

Initial learning rate	$\alpha = 0.3$
Regularization	$\lambda = 10^{-4}$
Dropout rate on input layer	$p = 20\%$

Table 1: Hyper-parameter settings.

where m is number of training sentences.

Following (Socher et al., 2013a), we use the diagonal variant of AdaGrad (Duchi et al., 2011) with minibatchs to minimize the objective.

For parameter initialization, we use random initialization within (-0.01, 0.01) for all parameters except the word embeddings. We adopt the pre-trained English word embeddings from (Collobert et al., 2011) and fine-tune them during training.

3 Experiments

3.1 Datasets

To evaluate our approach, we test our model on three datasets:

- **SST-1** The movie reviews with five classes in the Stanford Sentiment Treebank[1] (Socher et al., 2013b): negative, somewhat negative, neutral, somewhat positive, positive.

- **SST-2** The movie reviews with binary classes in the Stanford Sentiment Treebank[1] (Socher et al., 2013b): negative, positive.

- **QC** The TREC questions dataset[2] (Li and Roth, 2002) involves six different question types.

3.2 Hyper-parameters

Table 1 lists the hyper-parameters of our model. In this paper, we also exploit dropout strategy (Srivastava et al., 2014) to avoid overfitting. In addition, we set the batch size to 20. We set word embedding size $d = 50$ on the TREC dataset and $d = 100$ on the Stanford Sentiment Treebank dataset.

3.3 Experiment Results

Table 2 shows the performance of our GRNN on three datasets.

[1] http://nlp.stanford.edu/sentiment

[2] http://cogcomp.cs.illinois.edu/Data/QA/QC/

Methods	SST-1	SST-2	QC
NBoW (Kalchbrenner et al., 2014)	42.4	80.5	88.2
PV (Le and Mikolov, 2014)	44.6*	82.7*	91.8*
CNN-non-static (Kim, 2014)	48.0	87.2	93.6
CNN-multichannel (Kim, 2014)	47.4	**88.1**	92.2
MaxTDNN (Collobert and Weston, 2008)	37.4	77.1	84.4
DCNN (Kalchbrenner et al., 2014)	**48.5**	86.8	93.0
RecNTN (Socher et al., 2013b)	45.7	85.4	-
RAE (Socher et al., 2011)	43.2	82.4	-
MV-RecNN (Socher et al., 2012)	44.4	82.9	-
AdaSent (Zhao et al., 2015)	-	-	92.4
GRNN (our approach)	47.5	85.5	**93.8**

Table 2: Performances of the different models. The result of PV is from our own implementation based on Gensim.

Competitor Models Neural Bag-of-Words (NBOW) model is a simple and intuitive method which ignores the word order. Paragraph Vector (PV) (Le and Mikolov, 2014) learns continuous distributed vector representations for pieces of texts, which can be regarded as a long term memory of sentences as opposed to short memory in recurrent neural network. Here, we use the popular open source implementation of PV in Gensim[1]. Methods in the third block are CNN based models. Kim (2014) reports 4 different CNN models using max-over-time pooling, where CNN-non-static and CNN-multichannel are more sophisticated. MaxTDNN sentence model is based on the architecture of the Time-Delay Neural Network (TDNN) (Waibel et al., 1989; Collobert and Weston, 2008). Dynamic convolutional neural network (DCNN) (Kalchbrenner et al., 2014) uses the dynamic k-max pooling operator as a non-linear sub-sampling function, in which the choice of k depends on the length of given sentence. Methods in the fourth block are RecNN based models. Recursive Neural Tensor Network (RecNTN) (Socher et al., 2013b) is an extension of plain RecNN, which also depends on a external syntactic structure. Recursive Autoencoder (RAE) (Socher et al., 2011) learns the representations of sentences by minimizing the reconstruction error. Matrix-Vector Recursive Neural Network (MV-RecNN) (Socher et al., 2012) is a extension of RecNN by assigning a vector and a matrix to every node in the parse tree. AdaSent (Zhao et al., 2015) adopts recursive neural network using DAG structure.

[1] https://github.com/piskvorky/gensim/

Moreover, the plain GRNN which does not incorporate the gate mechanism cannot outperform the GRNN model. Theoretically, the plain GRNN can be regarded as a special case of GRNN, whose parameters are constrained or truncated. As a result, GRNN is a more powerful model which outperforms the plain GRNN. Thus, we mainly focus on the GRNN model in this paper.

Result Discussion Generally, our model is better than the previous recursive neural network based models (RecNTN, RAE, MV-RecNN and AdaSent), which indicates our model can better model the combinations of features with the FBT and our gating mechanism, even without an external syntactic tree.

Although we just use the top layer outputs as the feature for classification, our model still outperforms AdaSent.

Compared with the CNN based methods (MaxTDNN, DCNN and CNNs), our model achieves the comparable performances with much fewer parameters. Although CNN based methods outperform our model on SST-1 and SST-2, the number of parameters[2] of GRNN ranges from 40K to 160K while the number of parameters is about 400K in CNN.

4 Related Work

Cho et al. (2014) proposed grConv to model sentences for machine translation. Unlike our model, grConv uses the DAG structure as the topological structure to model sentences. The number of the

[2] We only take parameters of network into account, leaving out word embeddings.

internal nodes is $n^2/2$, where n is the length of the sentence. Zhao et al. (2015) uses the same structure to model sentences (called AdaSent), and utilizes the information of internal nodes to model sentences for text classification. Unlike grConv and AdaSent, our model uses full binary tree as the topological structure. The number of the internal nodes is $2n$ in our model. Therefore, our model is more efficient for long sentences. In addition, we just use the top layer neurons for text classification.

Moreover, grConv and AdaSent only exploit one gating mechanism (update gate), which cannot sufficiently model the complicated feature combinations. Unlike them, our model incorporates two kind of gates and can better model the feature combinations.

Hu et al. (2014) also proposed a similar architecture for matching problems, but they employed the convolutional neural network which might be coarse in modeling the feature combinations.

5 Conclusion

In this paper, we propose a gated recursive neural network (GRNN) to recursively summarize the meaning of sentence. GRNN uses full binary tree as the recursive topological structure instead of an external syntactic tree. In addition, we introduce two kinds of gates to model the complicated combinations of features. In future work, we would like to investigate the other gating mechanisms for better modeling the feature combinations.

Acknowledgments

We would like to thank the anonymous reviewers for their valuable comments. This work was partially funded by the National Natural Science Foundation of China (61472088, 61473092), National High Technology Research and Development Program of China (2015AA015408), Shanghai Science and Technology Development Funds (14ZR1403200).

References

Xinchi Chen, Xipeng Qiu, Chenxi Zhu, and Xuanjing Huang. 2015a. Gated recursive neural network for Chinese word segmentation. In *Proceedings of Annual Meeting of the Association for Computational Linguistics*.

Xinchi Chen, Yaqian Zhou, Chenxi Zhu, Xipeng Qiu, and Xuanjing Huang. 2015b. Transition-based dependency parsing using two heterogeneous gated recursive neural networks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of ICML*.

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

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159.

Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. In *Advances in Neural Information Processing Systems*.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of ACL*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Quoc V. Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of ICML*.

Xin Li and Dan Roth. 2002. Learning question classifiers. In *Proceedings of the 19th International Conference on Computational Linguistics*, pages 556–562.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH*.

Jordan B Pollack. 1990. Recursive distributed representations. *Artificial Intelligence*, 46(1):77–105.

Richard Socher, Jeffrey Pennington, Eric H Huang, Andrew Y Ng, and Christopher D Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 151–161.

Richard Socher, Brody Huval, Christopher D Manning, and Andrew Y Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1201–1211.

Richard Socher, John Bauer, Christopher D Manning, and Andrew Y Ng. 2013a. Parsing with compositional vector grammars. In *In Proceedings of the ACL conference*. Citeseer.

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013b. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Alex Waibel, Toshiyuki Hanazawa, Geoffrey Hinton, Kiyohiro Shikano, and Kevin J Lang. 1989. Phoneme recognition using time-delay neural networks. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 37(3):328–339.

Han Zhao, Zhengdong Lu, and Pascal Poupart. 2015. Self-adaptive hierarchical sentence model. *arXiv preprint arXiv:1504.05070*.

Learning Timeline Difference for Text Categorization

Fumiyo Fukumoto
Graduate Faculty of
Interdisciplinary Research
Univ. of Yamanashi, Japan
fukumoto@yamanashi.ac.jp

Yoshimi Suzuki
Graduate Faculty of
Interdisciplinary Research
Univ. of Yamanashi, Japan
ysuzuki@yamanashi.ac.jp

Abstract

This paper addresses text categorization problem that training data may derive from a different time period from the test data. We present a learning framework which extends a boosting technique to learn accurate model for timeline adaptation. The results showed that the method was comparable to the current state-of-the-art biased-SVM method, especially the method is effective when the creation time period of the test data differs greatly from the training data.

1 Introduction

Text categorization supports and improves several tasks such as creating digital libraries, information retrieval, and even helping users to interact with search engines (Mourao et al., 2008). A growing number of machine learning (ML) techniques have been applied to the text categorization task (Xue et al., 2008; Gopal and Yang, 2010). Each document is represented using a vector of features/terms (Yang and Pedersen, 1997; Hassan et al., 2007). Then, the documents with category label are used to train classifiers. Once category models are trained, each test document is classified by using these models. A basic assumption in the categorization task is that the distributions of terms between training and test documents are identical. When the assumption does not hold, the classification accuracy is worse. However, it is often the case that the term distribution in the training data is different from that of the test data when the training data may drive from a different time period from the test data. Manual annotation of tagged new data is very expensive and time-consuming. The methodology for accurate classification of the new test data by making the maximum use of tagged old data is needed in learning techniques.

In this paper, we present a method for text categorization that minimizes the impact of temporal effects. Our approach extends a boosting technique to learn accurate model for timeline adaptation. We used two types of labeled training data: One is the same creation time period with the test data. Another is different creation time period with the test data. We call the former *same-period* training, and the latter *diff-period* training data. For the same-period training data, the learner shows the same behavior as the boosting. In contrast, for diff-period training data, once they are wrongly predicted by the learned model, these data would be useless to classify test data. We decreased the weights of these data by applying Gaussian function in order to weaken their impacts.

2 Related Work

The analysis of temporal aspects is a practical problem as well as the process of large-scale heterogeneous data since the World-Wide Web (WWW) is widely used by various sorts of people. It is widely studied in many text processing tasks. One attempt is concept or topic drift dealing with temporal effects (Klinkenberg and Joachims, 2000; Kleinberg, 2002; Lazarescu et al., 2004; Folino et al., 2007; Song et al., 2014). Wang *et al*. developed the continuous time dynamic topic model (cDTM) (Wang et al., 2008). He *et al*. proposed a method to find bursts, periods of elevated occurrence of events as a dynamic phenomenon instead of focusing on arrival rates (He and Parker, 2010). They used Moving Average Convergence/Divergence (MACD) histogram which was used in technical stock market analysis (Murphy, 1999) to detect bursts.

Another attempt is domain adaptation. The goal of this attempt is to develop learning algorithms that can be easily ported from one domain to another (Daumé III, 2007; Sparinnapakorn and Ku-

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 799–804,
Lisbon, Portugal, 17-21 September 2015.

bat, 2007; Glorot et al., 2011; Siao and Guo, 2013). Domain adaptation is particularly interesting in Natural Language Processing (NLP) because it is often the case that we have a collection of labeled data in one domain but truly desire a model that can work well for another domain. Lots of studies addressed domain adaptation in NLP tasks such as part-of-speech tagging (Siao and Guo, 2013), named-entity (Daumé III, 2007), and sentiment classification (Glorot et al., 2011) are presented. One approach to domain adaptation is to use transfer learning. The transfer learning is a learning technique that retains and applies the knowledge learned in one or more tasks to efficiently develop an effective hypothesis for a new task. The earliest discussion is done by ML community in a NIPS-95 workshop[1], and more recently, transfer learning techniques have been successfully applied in many applications. Blitzer *et al*. proposed a method for sentiment classification using structural correspondence learning that makes use of the unlabeled data from the target domain to extract some relevant features that may reduce the difference between the domains (Blitzer et al., 2006). Several authors have attempted to learn classifiers across domains using transfer learning in the text classification task (Raina et al., 2006; Dai et al., 2007; Sparinnapakorn and Kubat, 2007). Raina *et al*. proposed a transfer learning algorithm that constructs an informative Bayesian prior for a given text classification task (Raina et al., 2006). They reported that a 20 to 40% test error reduction over a commonly used prior in the binary text classification task. Dai *et al*. presented a method called TrAdaBoost which extends boosting-based learning algorithms (Dai et al., 2007). Their experimental results show that TrAdaBoost allows knowledge to be effectively transferred from the old data to the new one. All of these approaches aimed at utilizing a small amount of newly labeled data to leverage the old data to construct a high-quality classification model for the new data. However, the temporal effects are not explicitly incorporated into their models.

To our knowledge, there have been only a few previous work on temporal-based text categorization. Mourao *et al*. investigated the impact of temporal evolution of document collections on the document classification (Mourao et al., 2008). Salles *et al*. presented an approach to classify documents in scenarios where the method uses information about both the past and the future, and this information may change over time (Salles et al., 2010). They address the drawbacks of which instances to select by approximating the Temporal Weighting Function (TWF) using a mixture of two Gaussians. However, their method needs tagged training data across full temporal range of training documents to construct TWF.

There are three novel aspects in our method. Firstly, we propose a method for text categorization that minimizes the impact of temporal effects in a learning technique. Secondly, from manual annotation of data perspective, the method allows users to annotate only a limited number of newly training data. Finally, from the perspective of robustness, the method is automated, and can be applied easily to a new domain, or different languages, given sufficient old labeled documents.

3 Learning Timeline Difference

Our learning model, Timeline Adaptation by Boosting (TABoost) is based on AdaBoost (Freund and Schapire, 1997). AdaBoost aims to boost the accuracy of a weak learner by adjusting the weights of training instances and learn a classifier accordingly. The TABoost uses two types of training data, same-period and diff-period training data. The assumption is that the quantity of the same-period data is limited, while diff-period training data is abundant. The TABoost aims at utilizing the diff-period training data to make up the deficit of a small amount of the same-period to construct a high-quality classification model for the test data. Similar to the TrAdaBoost presented by (Dai et al., 2007), TABoost is the same behavior as boosting for the same-period training data. In contrast, once diff-period training instances are wrongly predicted, we assume that these instances do not contribute to the accurate test data classification, and the weights of these instances decrease in order to weaken their impacts. The difference between TrAdaBoost and TABoost is a weighting manner, *i.e.* TABoost is a continuous timeline model, and it weights these instances by applying Gaussian function in order to weaken their impacts. TABoost is illustrated in Figure 1.

The training data set Tr is partitioned into two labeled sets Tr_{dp}, and Tr_{sp}. Tr_{dp} in Figure 1

[1] http://socrates.acadiau.ca/courses/comp/dsilver/NIPS95_LTL/transfer.workshop.1995.html.

Input {
The diff-period data Tr_{dp}, the same-period data Tr_{sp}, and the maximum number of iterations N.
}

Output {
$h_f(x) = \sum_{t=1}^{N} \alpha_t h_t(x_i)$.
}

Initialization {
$\mathbf{w^1} = (w_1^1, \cdots, w_{n+m}^1)$.
}

TABoost {
For $t = 1, \cdots, N$
1. Set $\mathbf{P^t} = \mathbf{w^t} / (\sum_{k=1}^{n+m} w_i^t)$.
2. Train a weak learner on the combined training set Tr_{dp} and Tr_{sp} with the distribution $\mathbf{P^t}$, and create weak hypothesis h_t: $X \rightarrow \{-1, +1\}$
3. Calculate the error of h_t on the combined training set Tr_{dp} and Tr_{sp}:
$$\epsilon_t = \sum_{i=1}^{n+m} \frac{w_i^t \cdot |h_t(x_i) - c(x_i)|}{\sum_{i=1}^{n+m} w_i^t}.$$
4. Chose $\alpha_t = \frac{1}{2}\mathrm{ln}(\frac{1-\epsilon_t}{\epsilon_t})$
5. Update the new weight vector:

$$w_i^{t+1} = \begin{cases} w_i^t \exp^{F(\delta)\alpha_t |h_t(x_i) - c(x_i)|}, \\ \qquad 1 \leq i \leq n \\ w_i^t \exp^{-\alpha_t |h_t(x_i) - c(x_i)|}, \\ \qquad n+1 \leq i \leq n+m \end{cases}$$

}

Figure 1: Flow of the algorithm

shows the diff-period training data that $Tr_{dp} = \{(x_i^{dp}, c(x_i^{dp}))\}$, where $x_i^{dp} \in X_{dp}$ ($i = 1, \cdots, n$), and X_{dp} refers to the diff-period instance space. Similarly, Tr_{sp} represents the same-period training data that $Tr_{sp} = \{(x_i^{sp}, c(x_i^{sp}))\}$, where $x_i^{sp} \in X_{sp}$ ($i = 1, \cdots, m$), and X_{sp} refers to the same-period instance space. n and m are the number of documents in Tr_{dp} and Tr_{sp}, respectively. $c(x_i)$ returns a label for the input instance x_i. The combined training set $Tr = \{(x_i, c(x_i))\}$ is given by:

$$x_i = \begin{cases} x_i^{dp} & i = 1, \cdots, n \\ x_i^{sp} & i = n+1, \cdots, n+m \end{cases}$$

In each iteration round shown in Figure 1, if a diff-period training instance is wrongly predicted, the instance may be useless to classify test data correctly. We decrease its training weight to reduce the effect. To do this, we assume a standard lognormal distribution (Crow, 1988), *i.e.* $F(\delta) = \frac{1}{\sqrt{2\pi}}\exp(-\frac{\delta^2}{2})$. δ in $F(\delta)$ represents time difference between diff-period training and test data. For instance, if the training data is 1999, and test data is 2000, δ equals to 1. Similarly, if the training data is 2000, and test data is 1999, δ is -1. The greater the time difference value, the smaller the training weight. We fit the model according to the temporal range of the data. As shown in Figure 1, we decrease its training weight w_i^t to reduce its effect through multiplying its weight by $\exp^{F(\delta)\alpha_t |h_t(x_i) - c(x_i)|}$.

We used the Support Vector Machines (SVM) as a learner. We represented each training and test document as a vector, each dimension of a vector is a noun word appeared in the document, and each element of the dimension is a term frequency. We applied the algorithm shown in Figure 1. After several iterations, a learner model is created by linearly combining weak learners, and a test document is classified by using a learner.

4 Experiments

We evaluated our TABoost by using the Mainichi Japanese newspaper documents.

4.1 Experimental setup

We choose the Mainichi Japanese newspaper corpus from 1991 to 2012. The corpus consists of 2,883,623 documents organized into 16 categories. We selected 8 categories, "International(Int)", "Economy(Eco)", "Home", "Culture", "Reading", "Arts", "Sports", and "Local news(Local)", each of which has sufficient number of documents. All documents were tagged by using a morphological analyzer Chasen (Matsumoto et al., 2000) and selected noun words. The total number of documents assigned to these categories are 787,518. For each category within each year, we divided documents into three folds: 2% of documents are used as the same-period training data, 50% of documents are the diff-period training data, and the remains are used to test our classification method.[2]

[2]When the creation time period of the training data is the same as the test data, we used only the same-period training data.

Table 1: The error rates across categories

Cat	TAB_s	SVM	TrAdaB	b-SVM	TAB
Int	0.409	0.467	0.326	0.253	**0.329**
Eco	0.368	0.429	0.243	0.228	**0.208**
Home	0.475	0.649	0.312	0.460	**0.172**
Culture	0.468	0.848	0.440	0.559	**0.196**
Reading	0.358	0.520	0.298	0.357	**0.337**
Arts	0.402	0.684	0.330	0.588	**0.331**
Sports	0.226	0.212	0.107	0.075	**0.123**
Local	0.586	0.305	0.400	0.156	**0.303**
M-Avg	0.411	0.514	0.307	0.334	**0.257**

We used LIBLINEAR (Fan et al., 2008) as a basic learner in the experiments. We compared our method, TABoost with four baselines: (1) TABoost with the same-period training data only (TAB_s), (2) SVM, (3) TrAdaBoost (Dai et al., 2007), and (4) biased-SVM (Liu et al., 2003) by SVM-light (Joachims, 1998). TAB_s is the same behavior as boosting. TrAdaBoost (TrAdaB) is presented by (Dai et al., 2007). Biased-SVM (b-SVM) is known as the state-of-the-art SVMs method, and often used for comparison (Elkan and Noto, 2008). Similar to SVM, for biased-SVM, we used the first two folds as a training data, and classified test documents directly, *i.e.* we used closed data. We empirically selected values of two parameters, "c" (trade-off between training error and margin) and "j", *i.e.* cost (cost-factor, by which training errors on positive instances) that optimized result obtained by classification of test documents. Similar to (Liu et al., 2003), "c" is searched in steps of 0.02 from 0.01 to 0.61. "j" is searched in steps of 5 from 1 to 200. As a result, we set c and j to 0.01 and 10, respectively. To make comparisons fair, all five methods including our method are based on linear kernel. Throughout the experiments, the number of iterations is set to 100. We used error rate as an evaluation measure (Dai et al., 2007).

4.2 Results

Categorization results for 8 categories (48% of the test documents, *i.e.* 378,008 documents) are shown in Table 1. Each value in Table 1 shows macro-averaged error rate across 22 years. "M-Avg" refers to macro-averaged error rate across categories. The results obtained by biased-SVM show minimum error rate obtained by varying the parameters, "c" and "j".

As can be seen clearly from Table 1, the overall

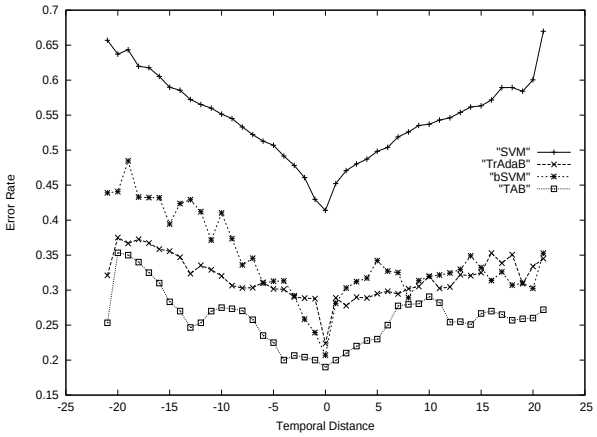


Figure 2: Performance against temporal distance

performance obtained by TAB was the best among the five methods. The macro average error rates with TrAdaB and TAB were lower to those obtained by b-SVM, although b-SVM in Table 1 was the result obtained by using the closed data. In contrast, SVM did not work well. This demonstrates that once the training data drive from a different time period from the test data, the distributions of terms between training and test documents are not identical. The results obtained by TAB_s were worse than those obtained by TrAdaBoost, b-SVM, and TAB. This shows that (i) the same-period training data we used is not sufficient to train a model alone, and (ii) TAB demonstrates a good transfer ability.

Figure 2 illustrates error rate against the temporal difference between diff-period training and test data. Both training and test data are the documents from 1991 to 2012. For instance, "10" of the x-axis in Figure 2 indicates that the test documents are created 10 years later than the training documents. We can see from Figure 2 that the result obtained by TAB was the best in all of the temporal distances. There are no significant differences among three methods, bSVM, TrAdaB, and TAB when the test and training data are the same time period. The performance of these methods including SVM drops when the creation time of the test data differs greatly from the diff-period training data. However, the performance of TAB was still better to those obtained by other methods. This demonstrates that the algorithm with continuous timeline model works well for categorization.

Figure 3 shows the error rate against the number of iterations. Each curve shows averaged error rate under time period between same-period and diff-

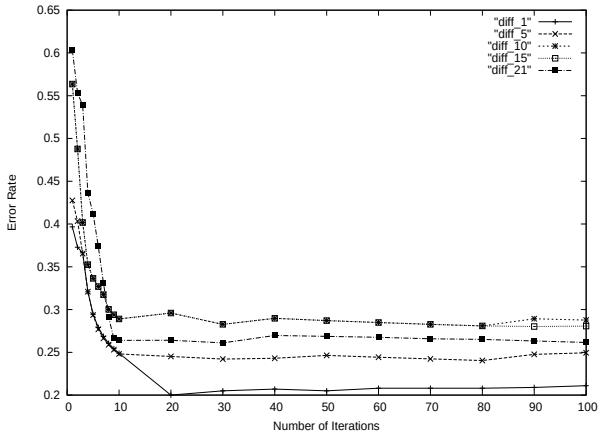


Figure 3: Iteration curves by temporal distance

period training data. For example, "diff_10" indicates that the difference time period between same and diff training data is $\pm$ 10 years. We can see from Figure 3 that all curves except for "diff_20" drop rapidly and converge around 10 iterations. "diff_20" converges around 20 iterations. This was the same behaviour as TrAdaBoost, *i.e.* TrAdaBoost converges around 20 iterations. The fast convergence is not particularly surprising because we used a small number of same-period (2%) and a large number of diff-period (50%) training data. It is necessary to examine how the ratio between same-period and diff-period training data affects overall performance for further quantitative evaluation, although a main contribution of TAB is in situations using by both of a small amount of labeled new data which is not sufficient to train a model alone, and a large amount of old data.

5 Conclusion

We have presented a method for text categorizaiton that minimizes the impact of temporal effects. The results using Japanese Mainichi Newspaper corpus show that it works well for categorization, especially when the creation time of the test data differs greatly from the training data. There are a number of interesting directions for future work. The rate of convergence of TAB ($O(\sqrt{\mathrm{In}n/N})$) is slow which can also be found in (Dai et al., 2007). Here, n is the number of training data, and N is the number of iterations. In the future, we will try to extend the framework to address this issue. We used Japanese newspaper documents in the experiments. For quantitative evaluation, we need to apply our mehtod to other data such as ACM-DL and a large, heterogeneous collection of web content in addition to the experiment to examine the performance agasint the ratio between same-period and diff-period training data.

References

J. Blitzer, R. McDonald, and F. Pereira. 2006. Domain Adaptation with Structural Correspondence Learning. In *Proc. of the Conference on Empirical Methods in Natural Language Processing*, pages 120–128.

S. K. Crow. 1988. Log-normal Distributions: Theory and Application. *NewYork: Dekker*.

W. Dai, Q. Yang, G.R. Xue, and Y. Yu. 2007. Boosting for Transfer Learning. In *Proc. of the 24th International Conference on Machine Learning*, pages 193–200.

H. Daumé III. 2007. Frustratingly Easy Domain Adaptation. In *Proc. of the 45th Annual Meeting of the Association of computational Linguistics*, pages 256–263.

C. Elkan and K. Noto. 2008. Learning Classifiers from Only Positive and Unlabeled Data. In *Proc. of the KDD'08*, pages 213–220.

F. E. Fan, K. W. Chang, C. J. Hsieh, X. R. Wang, and C. J. Lin. 2008. LIBLINEAR: A Library for Large Linear Classification. *Machine Learning*, 9:1871–1874.

G. Folino, C. Pizzuti, and G. Spezzano. 2007. An Adaptive Distributed Ensemble Approach to Mine Concept-drifting Data Streams. In *Proc. of the 19th IEEE International Conference on Tools with Artificial Intelligence*, pages 183–188.

Y. Freund and R. E. Schapire. 1997. A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. *Journal of Computer and System Sciences*, 55(1):119–139.

X. Glorot, A. Bordes, and Y. Bengio. 2011. Domain Adaptation for Large-Scale Sentiment Classification: A Deep Learning Approach. In *Proc. of the 28th International Conference on Machine Learning*, pages 97–110.

S. Gopal and Y. Yang. 2010. Multilabel Classification with Meta-level Features. In *Proc. of the 33rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 315–322.

S. Hassan, R. Mihalcea, and C. Nanea. 2007. Random-Walk Term Weighting for Improved Text Classification. In *Proc. of the IEEE International Conference on Semantic Computing*, pages 242–249.

D. He and D. S. Parker. 2010. Topic Dynamics: An Alternative Model of Bursts in Streams of Topics. In *Proc. of the 16th ACM SIGKDD Conference on Knowledge discovery and Data Mining*, pages 443–452.

T. Joachims. 1998. SVM Light Support Vector Machine. In *Dept. of Computer Science Cornell University*.

M. Kleinberg. 2002. Bursty and Hierarchical Structure in Streams. In *Proc. of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 91–101.

R. Klinkenberg and T. Joachims. 2000. Detecting Concept Drift with Support Vector Machines. In *Proc. of the 17th International Conference on Machine Learning*, pages 487–494.

M. M. Lazarescu, S. Venkatesh, and H. H. Bui. 2004. Using Multiple Windows to Track Concept Drift. *Intelligent Data Analysis*, 8(1):29–59.

B. Liu, Y. dai, X. Li, W. S. Lee, and P. S. Yu. 2003. Building Text Classifiers using Positive and Unlabeled Examples. In *Proc. of the ICDM'03*, pages 179–188.

Y. Matsumoto, A. Kitauchi, T. Yamashita, Y. Hirano, Y. Matsuda, K. Takaoka, and M. Asahara. 2000. *Japanese Morphological Analysis System Chasen Version 2.2.1*. In Naist Technical Report.

F. Mourao, L. Rocha, R. Araujo, T. Couto, M. Goncalves, and W. M. Jr. 2008. Understanding Temporal Aspects in Document Classification. In *Proc. of the 1st ACM International Conference on Web Search and Data Mining*, pages 159–169.

J. Murphy. 1999. *Technical Analysis of the Financial Markets*. Prentice Hall.

R. Raina, A. Y. Ng, and D. Koller. 2006. Constructing Informative Priors using Transfer Learning. In *Proc. of the 23rd International Conference on Machine Learning*, pages 713–720.

T. Salles, L. Rocha, and G. L. Pappa. 2010. Temporally-aware Algorithms for Document Classification. In *Proc. of the 33rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 307–314.

M. Siao and Y. Guo. 2013. Domain Adaptation for Sequence Labeling Tasks with a Probabilistic Language Adaptation Model. In *Proc. of the 30th International Conference on Machine Learning*, pages 293–301.

M. Song, G. E. Heo, and S. Y. Kim. 2014. Analyzing topic evolution in bioinformatics: Investigation of dynamics of the field with conference data in dblp. *Scientometrics*, 101(1):397–428.

K. Sparinnapakorn and M. Kubat. 2007. Combining Subclassifiers in Text Categorization: A DST-based Solution and a Case Study. In *Proc. of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 210–219.

C. Wang, D. Blei, and D. Heckerman. 2008. Continuous Time Dynamic Topic Models. In *Proc. of the 24th Conference on Uncertainty in Artificial Intelligence*, pages 579–586.

G. R. Xue, W. Dai, Q. Yang, and Y. Yu. 2008. Topic-bridged PLSA for Cross-Domain Text Classification. In *Proc. of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 627–634.

Y. Yang and J. O. Pedersen. 1997. A Comparative Study on Feature Selection in Text Categorization. In *Proc. of the 14th International Conference on Machine Learning*, pages 412–420.

Summarizing topical contents from PubMed documents using a thematic analysis

Sun Kim, Lana Yeganova and W. John Wilbur
National Center for Biotechnology Information
National Library of Medicine, National Institutes of Health
Bethesda, MD 20894, USA
{sun.kim,yeganova,wilbur}@ncbi.nlm.nih.gov

Abstract

Improving the search and browsing experience in PubMed® is a key component in helping users detect information of interest. In particular, when exploring a novel field, it is important to provide a comprehensive view for a specific subject. One solution for providing this panoramic picture is to find sub-topics from a set of documents. We propose a method that finds sub-topics that we refer to as themes and computes representative titles based on a set of documents in each theme. The method combines a thematic clustering algorithm and the Pool Adjacent Violators algorithm to induce significant themes. Then, for each theme, a title is computed using PubMed document titles and theme-dependent term scores. We tested our system on five disease sets from OMIM® and evaluated the results based on normalized point-wise mutual information and MeSH® terms. For both performance measures, the proposed approach outperformed LDA. The quality of theme titles were also evaluated by comparing them with manually created titles.

1 Introduction

PubMed[1], currently a collection of about 25 million bibliographic records, has grown exponentially in size. With the abundance and diversity of information in PubMed many queries retrieve thousands of documents making it difficult for users to browse the results and identify the information most relevant to their topic of interest. The query 'cystic fibrosis', for example, retrieves papers that discuss different aspects of the disease, including its clinical features, treatment options, diagnosis, etc. A possible solution to this problem is to automatically group the retrieved documents into meaningful thematic clusters or themes (these terms are used interchangeably). However, clustering alone does not solve the problem entirely, as a significant amount of human post-processing is required to infer the topic of the cluster.

There exists a vast collection of probabilistic clustering methods. One common problem among most of them is that different results are obtained depending on the cluster initialization, suggesting that some clusters are unstable or weak. However, there is no obvious way to effectively and efficiently evaluate the quality of clusters. In this paper, we combine EM-based thematic clustering (Kim and Wilbur, 2012) with the Pool Adjacent Violators (PAV) algorithm (Ayer et al., 1955; Wilbur et al., 2005). PAV is an isotonic regression algorithm which we use as a method for converting a score into a probability. Here, we show how PAV can be applied to evaluate the quality of clusters.

Another issue that motivated this research is that most existing algorithms produce clusters that are not self-descriptive. Presenting meaningful titles can significantly improve the user perception of clustering results. To that end, we utilize PubMed document titles and cluster-related term scores to automatically obtain a title for each theme. The method results in thematic clusters of documents with cluster titles.

Studies similar to our approach are ASI (Adaptive Subspace Iteration) (Li et al., 2004) and SKWIC (Simultaneous Keyword Identification and Clustering of text documents) (Frigui and Nasraoui, 2004). Both perform document clustering and cluster-dependent keyword identification simultaneously. SKWIC can only produce hard clustering, while ASI is computationally very expensive as it heavily depends on matrix operations. A study by Hammouda et al. (2005) sug-

[1] http://pubmed.gov

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 805–810,
Lisbon, Portugal, 17-21 September 2015.

gests automatic keyphrase extraction from a cluster of documents as a surrogate to providing a cluster title, but they treat document clustering and cluster-dependent keyword extraction as separate problems.

Topic modeling (Hofmann, 1999; Blei et al., 2003; Blei and Lafferty, 2005) is the most popular and an alternative approach that has a similar underlying goal of discovering hidden thematic structure of a document collection and organizing the collection according to the discovered topics. Topic models are based upon the idea that documents are mixtures of topics, where a topic is a probability distribution over words (Steyvers and Griffiths, 2007). However, topic modeling is not a document clustering scheme in nature. Although a list of keywords that represent a topic is available, the title of the cluster may not be evident.

2 Methods

We here describe the EM-based clustering algorithm, and show how PAV is incorporated with it to yield the PAV-EM thematic clustering technique. We further present a cluster summarization method to induce theme titles.

2.1 Theme definition

Let D be a document set and let T be the set of terms in D. Let R denote the relation between elements of T and D. tRd means $t \in d$. We define a theme as a subject that is described by non-empty sets $U \subseteq T$ and $V \subseteq D$, where all the elements of U have a high probability of occurring in all the element of V. An EM framework is used to extract subject terms for a theme (Wilbur, 2002). In addition to the observed data R, a theme is defined by the latent indicator variables z_d, $\{z_d\}_{d \in D}$. The parameters are

$$\Theta = U(\|U\| = n_U), \{p_t, q_t\}_{t \in U}, \{r_t\}_{t \in T}, \quad (1)$$

where n_U is the size of the set U. For any $t \in U$, p_t is the probability that for any $d \in V$, tRd. q_t is the probability that for any $d \in D - V$, tRd. For any $t \in T$, r_t is the probability that for any $d \in D$, tRd. Assuming all relations tRd are independent of each other, the goal is to obtain the highest probabilities

$$p(R, \{z_d\}|\Theta) = p(R|\{z_d\}, \Theta)p(\{z_d\}|\Theta). \quad (2)$$

E-step (expectation step) evaluates the expectation of the logarithm of Eqn. 2. M-step (maximization step) maximizes this expectation over the parameters Θ. For each term, t, we define a quantity α_t which is the difference between the contribution coming from t depending on whether $u_t = 1$ or $u_t = 0$. The maximization is completed by choosing the n_U largest α_t's and setting $u_t = 1$ for each of them and $u_t = 0$ for all others. Details of this theme extraction scheme can be found in Wilbur (2002).

Algorithm 1 PAV-EM algorithm

Let D be the dataset, where $d \in D$.
Give a value for the parameter q.

Set $X = \emptyset$.
for $i \leftarrow 1, n$ **do**
  Create q random clusters.
  Run the theme clustering algorithm.
  For each cluster C and d with pz_d^C,
    $X \leftarrow X \cup \{< pz_d^C, 1, 1 >\}$[2] if $d \in C$,
    $X \leftarrow X \cup \{< pz_d^C, 1, 0 >\}$ if $d \notin C$.
Obtain the PAV function, $PAV(pz_d^C)$, over X.

Set $S = \emptyset$, where S is the output cluster set.
repeat
  Create q random clusters for $\{d|d \notin \cup S\}$.
  Run the theme clustering algorithm.
  Select any cluster C, where
    $C' = \{d|d \in C, PAV(pz_d^C) > 0.9\}$
    satisfies $|C'| > 10$.
    $S \leftarrow S \cup \{C'\}$.
until no more changes in S.

2.2 PAV-EM thematic clustering

In thematic clustering, a document is assigned to a theme that has the highest probability to the document (Kim and Wilbur, 2012). Although this approach shows a reasonable performance for theme-based document clustering, the dynamic nature of random initialization and multiple subjects described in a document may create many weak themes. Moreover, there is no clear guideline to distinguish strong and weak themes. Thus, we here propose a method that extracts strong themes more effectively. In the EM-based theme extraction scheme, the log odds score pz_d^C indicates the extent to which a document d is coupled with a specific theme C. If a cluster in-

[2]The second and the third arguments in the bracket are the weight and the probability estimate of the data, respectively.

cludes a reasonable number of documents that have high pz_d^Cs, it indicates that the cluster represents a strong theme. Therefore, we can obtain strong themes by collecting these clusters.

Let the probability $p(score)$ be a monotonically non-decreasing function of $score$. The PAV algorithm (Ayer et al., 1955; Wilbur et al., 2005) is a regression method to derive from the data that monotonically non-decreasing estimate of $p(score)$ which assigns maximal likelihood to the data. For our approach, $score = pz_d^C$.

Algorithm 1 shows the theme clustering process using the PAV algorithm. For the given dateset D and the initial number of clusters q, theme clustering is performed n times, and an isotonic regression function is learned by applying the PAV algorithm. Note that q is an initial guess for the number of clusters and it is not guaranteed to remain the same in the output set. For our experiments, we set $q = 50$ and $n = 100$. After the PAV algorithm is applied, theme clustering is performed. At each iteration, we select any cluster in which there are more than 10 documents with PAV scores higher than 0.9. Unselected documents are re-used for clustering in the next iteration. This procedure is repeated until there are no more changes in the selected cluster set S.

2.3 Theme summarization

After obtaining themes (document clusters and their subject terms), we summarize each theme by choosing a text segment from PubMed document titles. A title should cover as many subject terms as possible, but also it should be well-formed, i.e. be descriptive enough and humanly understandable. To achieve this goal, we first extract all possible candidates from document titles as follows:

(i) Extract all possible candidates as n-grams, where $n = 1, ..., 20$. Noun phrases are treated as units and must be totally inside or outside a candidate.

(ii) Check POS tags for starting and ending words in a candidate. Starting with a conjunction, verb, preposition and symbol is not allowed. Ending with a conjunction, verb, preposition, symbol, determiner, adjective or certain pronouns is not allowed.

(iii) Discard any candidates that start or end with '-' or '.'. The candidates including certain characters such as '/', ';', ':' are also removed.

(iv) Check grammatical dependency relations. We discard candidates for which the head word of a preposition does not appear in the same candidate as the proposition. Also, we validate the case, 'between A and B', so that A and B are not separated.

Next, for each candidate, a score is calculated by

$$score(cand_i) = \log \frac{\prod_{t \in U}(tf_t \alpha_t)}{\prod_{t \notin U} tf_t}, \qquad (3)$$

where tf_t is the term frequency of the term t. However, an ideal title should have enough words to be descriptive, hence we subtract $(len(cand_i) - 5)^2$ from $score(cand_i)$, where $len(cand_i)$ is the number of words in $cand_i$, and choose the top score as a title.

3 Experimental Results

We applied our method to the five disease sets, "cystic fibrosis", "deafness", "DiGeorge syndrome", "autism" and "hypertrophic cardiomyopathy" from OMIM[3]. These sets consist of 3000, 3000, 956, 2917 and 1997 PubMed documents, respectively, and are available at `http://www.ncbi.nlm.nih.gov/CBBresearch/Wilbur/IRET/PAVEM`.

For evaluating PAV-EM and comparing with the topic modeling method, latent Dirichlet allocation (LDA) (Blei et al., 2003), both approaches were performed 10 times for each disease set and scores were averaged over all runs. Mallet[4] was used to run LDA. The same tokenization was applied to LDA and PAV-EM. The number of topics given for LDA was 50 and the recommended optimization parameter was used for producing LDA topics.

Table 1 presents average runtimes[5] for LDA and PAV-EM. LDA and PAV-EM spent 15.2 and 13.3 seconds on average for processing the smallest set, "DiGeorge syndrome". However, in larger sets, e.g. "autism", it took 46.9 and 31.3 seconds for LDA and PAV-EM, respectively. We also ran another implementation[6] of LDA, which was 30 times slower than Mallet. While PAV-EM and

[3] `http://www.ncbi.nlm.nih.gov/omim`

[4] `http://mallet.cs.umass.edu`

[5] Both methods were tested on a single linux server. The processing times reported do not include the preprocessing stages done by Mallet and our implementation.

[6] `http://www.cs.princeton.edu/~blei/lda-c`

Dataset	LDA	PAV-EM
Set 1	25.7	18.4
Set 2	36.5	24.7
Set 3	15.2	13.3
Set 4	46.9	31.3
Set 5	30.3	19.2

Table 1: Average runtimes for LDA and PAV-EM in seconds. Sets 1, 2, 3, 4 and 5 are "cystic fibrosis", "deafness", "DiGeorge syndrome", "autism" and "hypertrophic cardiomyopathy", respectively.

Method	Topic terms	
	Top 5	Top 10
LDA	2.8906	10.9760
PAV-EM	4.0322	14.6213

Table 2: NMPI scores for LDA and PAV-EM.

LDA can be implemented in parallel computation[7], this indicates that PAV-EM may be more efficient to obtain themes for a larger set of PubMed documents.

The PAV-EM algorithm automatically learns themes from unlabeled PubMed documents, hence the performance measures that are used in supervised learning cannot be applied to our setup. Recent studies have shown more interest in topic coherence measures (Chang et al., 2009; Newman et al., 2010; Mimno et al., 2011), which capture the semantic interpretability of topics based on subject terms. Table 2 shows the topic coherence scores measured by normalized point-wise mutual information (NPMI). For both top 5 and top 10 subject terms, PAV-EM achieves better NMPI scores than LDA. NPMI is known to be strongly correlated with human ratings (Aletras and Stevenson, 2013; Röder et al., 2015) and is defined by

$$\text{NPMI} = \sum_{i=2}^{N} \sum_{j=1}^{i-1} \frac{\log \frac{p(t_i,t_j)+\epsilon}{p(t_i)p(t_j)}}{-\log\left(p(t_i,t_j)+\epsilon\right)}, \qquad (4)$$

where $p(t_i, t_j)$ is the fraction of documents containing both terms t_i and t_j, and N indicates the number of top subject terms. $\epsilon = \frac{1}{D}$ is the smoothing factor, where D is the size of the dataset.

MeSH (Medical Subject Headings) is a controlled vocabulary for indexing and searching biomedical literature (Lowe and Barnett, 1994).

[7]A parallel implementation of LDA appears in Wang et al. (2009)

MeSH	Method	Prec.	Recall	F1
Top 1	LDA	0.4529	0.3827	0.4125
	PAV-EM	0.3842	0.5303	0.4427
Top 3	LDA	0.3935	0.3931	0.3925
	PAV-EM	0.3388	0.5239	0.4086

Table 3: Classification performance based on top significant MeSH terms appearing in themes.

MeSH terms assigned to an article are often used to indicate the topics of the article, thus these terms can be used to identify how well documents are grouped by topics. In each cluster, p-values of MeSH terms are calculated using the hypergeometric distribution (Kim and Wilbur, 2001), and the top N significant MeSH terms are used to calculate precision, recall and F1. Table 3 compares PAV-EM with LDA[8] for the MeSH term-based performance. In the table, PAV-EM provides higher recall and F1 for top 1 and top 3 MeSH terms. Higher recall has an advantage in our task because the theme summarization process uses a consensus among PubMed documents to reach a theme title.

The next experiment was performed to compare machine generated titles with manually labeled titles. Although human judgements are subjective, it is not uncommon to collect human judgements for evaluating topic modeling methods (Mei et al., 2007; Chang et al., 2009; Xie and Xing, 2013). To validate the performance of the theme summarization approach, we first chose 500 documents from each disease set, and produced themes and titles. For each topic, five strongest themes were chosen, and they were shown to three human annotators with extracted subject terms. Table 4 shows an example of the proposed approach and the manual annotation for the "hypertrophic cardiomyopathy" set. Among 25 themes, our approach correctly identified 21 theme titles. We assumed that a machine-generated title was correct if it included any of manually annotated titles.

4 Conclusion

This study was inspired by an EM-based thematic clustering approach. In this probabilistic framework, theme terms are iteratively selected and documents are assigned to a most likely theme. The number of themes is dynamically adjusted

[8]For LDA, each document was assigned to the highest scoring topic.

Proposed approach	Annotator 1	Annotator 2	Annotator 3
cardiac myosin binding protein c	myosin binding protein c	cardiac myosin binding protein c	cardiac myosin binding protein c
ptpn11 mutations in leopard syndrome	ptpn11 mutations in leopard syndrome	ptpn11 mutations in leopard syndrome	ptpn11 mutations in leopard syndrome
cytochrome c oxidase	cytochrome c oxidase	mitochondrial cytochrome-c-oxidase deficiency	mitochondrial cytochrome c oxidase deficiency
friedreich ataxia and diabetes mellitus	friedreich ataxia	friedreich ataxia	friedreich ataxia
hepatitis c virus infection	hepatitis c virus	role of hepatitis c virus in cardiomyopathies	hepatitis c virus infection

Table 4: Comparison of the titles generated from the proposed approach and manual annotation for the "hypertrophic cardiomyopathy" set.

by probabilistic evidence from documents. The PAV algorithm is utilized to measure the quality of themes. After themes are identified, subject term weights and PubMed document titles are used to form humanly understandable titles. The experimental results show that our approach provides a useful overview of a set of documents. In addition, the method may allow for a new way of browsing by semantically clustered documents as well as searching with context-based query suggestions.

Acknowledgments

The authors would like to thank Donald C. Comeau and Rezarta Islamaj Doğan for their contribution to the manual evaluation. This research was supported by the Intramural Research Program of the NIH, National Library of Medicine.

References

N. Aletras and M. Stevenson. 2013. Evaluating topic coherence using distributional semantics. In *Proc. International Conference on Computational Semantics (IWCS 2013)*, pages 13–22, March.

M. Ayer, H. D. Brunk, G. M. Ewing, W. T. Reid, and E. Silverman. 1955. An empirical distribution function for sampling with incomplete information. *The Annals of Mathematical Statistics*, 26(4):641–647.

D. Blei and J. Lafferty. 2005. Correlated topic models. In *Proc. Advances in Neural Information Processing Systems (NIPS 2005)*, pages 147–154, December.

D. M. Blei, A. Y. Ng, and M. I. Jordan. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

J. Chang, J. Boyd-Graber, S. Gerrish, C. Wang, and D. M. Blei. 2009. Reading tea leaves: How humans interpret topic models. In *Proc. Advances in Neural Information Processing Systems (NIPS 2009)*, pages 288–296, December.

H. Frigui and O. Nasraoui. 2004. *Simultaneous Clustering and Dynamic Keyword Weighting for Text Documents*. Springer, New York, USA.

K. M. Hammouda, D. N. Matute, and M. S. Kamel. 2005. CorePhrase: keyphrase extraction for document clustering. In *Proc. International Conference on Machine Learning and Data Mining*, pages 265–274, July.

T. Hofmann. 1999. Probabilistic latent semantic indexing. In *Proc. Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 50–57, August.

W. Kim and W. J. Wilbur. 2001. Corpus-based statistical screening for content-bearing terms. *Journal of the American Society for Information Science and Technology*, 52(3):247–259.

S. Kim and W. John Wilbur. 2012. Thematic clustering of text documents using an EM-based approach. *Journal of Biomedical Semantics*, 3(Suppl 3):S6.

T. Li, S. Ma, and M. Ogihara. 2004. Document clustering via adaptive subspace iteration. In *Proc. Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 218–225, July.

H. J. Lowe and G. O. Barnett. 1994. Understanding and using the medical subject headings (MeSH) vocabulary to perform literature searches. *The Journal of the American Medical Association*, 271(14):1103–1108.

Q. Mei, X. Shen, and C. Zhai. 2007. Automatic labeling of multinomial topic models. In *Proc. ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2007)*, pages 490–499, August.

D. Mimno, H. M. Wallach, E. Talley, M. Leenders, and A. McCallum. 2011. Optimizing semantic coherence in topic models. In *Proc. Conference on Empirical Methods in Natural Language Processing (EMNLP 2011)*, pages 262–272, July.

D. Newman, J. H. Lau, K. Grieser, and T. Baldwin. 2010. Automatic evaluation of topic coherence.

In *Proc. Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL 2010)*, pages 100–108, June.

M. Röder, A. Both, and A. Hinneburg. 2015. Exploring the space of topic coherence measures. In *Proc. ACM International Conference on Web Search and Data Mining (WSDM 2015)*, pages 399–408, February.

M. Steyvers and T. Griffiths. 2007. *Probabilistic Topic Models*. Erlbaum, Hillsdale, NJ, USA.

Y. Wang, H. Bai, M. Stanton, W.-Y. Chen, and E. Y. Chang. 2009. PLDA: Parallel latent Dirichlet allocation for large-scale applications. In *Proc. International Conference on Algorithmic Aspects in Information and Management (AAIM 2009)*, pages 301–314, June.

W. John Wilbur, L. Yeganova, and W. Kim. 2005. The synergy between PAV and AdaBoost. *Machine Learning*, 61(1-3):71–103.

W. John Wilbur. 2002. A thematic analysis of the AIDS literature. In *Proc. Pacific Symposium on Biocomputing*, pages 386–397, January.

P. Xie and E. Xing. 2013. Integrating document clustering and topic modeling. In *Proc. Conference on Uncertainty in Artificial Intelligence (UAI 2013)*, pages 694–703, July.

Recognizing Biographical Sections in Wikipedia

Alessio Palmero Aprosio
Fondazione Bruno Kessler
Via Sommarive, 18
38123 Trento, Italy
aprosio@fbk.eu

Sara Tonelli
Fondazione Bruno Kessler
Via Sommarive, 18
38123 Trento, Italy
satonelli@fbk.eu

Abstract

Wikipedia is the largest collection of encyclopedic data ever written in the history of humanity. Thanks to its coverage and its availability in machine-readable format, it has become a primary resource for large-scale research in historical and cultural studies. In this work, we focus on the subset of pages describing persons, and we investigate the task of recognizing biographical sections from them: given a person's page, we identify the list of sections where information about her/his life is present. We model this as a sequence classification problem, and propose a supervised setting, in which the training data are acquired automatically. Besides, we show that six simple features extracted only from the section titles are very informative and yield good results well above a strong baseline.

1 Introduction

In the last years, several projects have started to address the mechanisms behind cultural development, borrowing techniques and algorithms from computer science and natural language processing to serve historical investigation. Efforts such as BiographyNet[1], Pantheon[2] or the Austrian Prosopographical Information System[3] prove an increasing interest in automatically extracting biographical descriptions from large amounts of data and combining them in a more general picture, taking advantage of the availability of such descriptions on the web. Wikipedia has been the main source of information for research in this direction despite its many biases, for instance its well-known English, Western and gender bias (Wikipedia Contributors, 2014). In fact, Wikipedia coverage both in terms of pages and in terms of languages, as well as the structured information that can be leveraged through DBpedia, has made it the primary resource for large-scale analyses on biographies. However, the lack of a consistent template for describing persons' lives led to the creation of a plethora of page types, where biographical information is displayed in diverse ways.

Based on a random sample of 100 persons' pages, we noticed that only 20% of them includes a section called *Biography* or *Life*, typically containing a set of subsections describing the main periods in a person's life from birth to death (see for instance `https://en.wikipedia.org/wiki/Leonard_Bernstein`). The other pages in our sample do not follow a pre-defined pattern and present the person's biography in one or several sections at the same level of the other ones (see for instance `https://en.wikipedia.org/wiki/Judy_Holliday`, with the *Filmography* and *Discography* sections at the same level of *Early Life* and *Career*). Given this high variability, it is very difficult to extract all and only those sections that describe a biography, and that build all together in sequence the description of a person's life. This depends also on the different types of non-biographical sections available, which in the case of prominent persons typically include main themes, reception, style, influences, legacy, work titles, etc. (see for instance *Will to power*, *Eternal return*, *Perspectivism*, *Critique on mass culture* in `https://en.wikipedia.org/wiki/Friedrich_Nietzsche`).

In this work, we present a simple methodology that, given a person's page in Wikipedia, recog-

[1] `http://www.biographynet.nl`

[2] `http://pantheon.media.mit.edu/treemap/country_exports/IQ/all/-4000/2010/H15/pantheon`

[3] `http://www.oeaw.ac.at/acdh/de/node/188`

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 811–816,
Lisbon, Portugal, 17-21 September 2015.

nizes all sections that deal with his/her life even if no *Biography* section is present. The problem is modeled as a classification task using Conditional Random Fields, which are particularly suitable for our study because the biographical sections tend to follow a chronological order and present typical sequential patterns (for instance, the section *Early Life* is often followed by *Early Career*). While a simple token-based baseline is very difficult to beat when the task is performed at section level (i.e. deciding whether a section is biographical or not), our method performs best when the evaluation is performed at *page* level, recognizing *all* sections that describe a person's life. This is crucial if the task under investigation is meant as a preliminary step towards the automatic extraction of all events that compose a person's biography.

2 Related Work

To our knowledge, this is the first attempt to extract biographical sections from Wikipedia. Other past works focused on the recognition of biographical sentences (Biadsy et al., 2008; Zhou et al., 2004; Biryukov et al., 2005). However the two tasks have different goals: in our case, we aim at extracting all biographical sections, so that all events of a person's life from birth to death are present. The other approach, instead, is used to generate biography summaries, which was a task of the DUC2004 evaluation exercise[4]. Besides, while approaches for sentence selection look for textual features such as typical unigrams or bigrams that characterize biographical descriptions (Filatova and Prager, 2005), we adopt a much simpler approach by considering only section titles.

Other works focused on the analysis of typical events in selected articles from Wikipedia biographies by looking for a particular list of predefined events (Bamman and Smith, 2014). Our approach may complement such works by introducing a preprocessing step that extracts all and only the sections describing the biographies, upon which event extraction experiments can be performed. This would increase both the precision and the recall of the extracted information.

3 Experimental Setup

In this section we detail the data used for our experiments and the classification task.

[4] http://duc.nist.gov/duc2004/

3.1 Data set

Since our goal is to distinguish between biographical and non biographical sections, we focus only on Wikipedia pages describing persons. We derive our development, training and test data from the Pantheon data set (Yu et al., 2015), freely available for download.[5] The data set includes a list of 11,340 notable individuals with the link to their Wikipedia page in multiple languages, plus a number of additional information such as date and place of birth, category and language editions, which we do not consider for our study. Only the persons whose Wikipedia page is translated in at least 25 languages are included in Pantheon, as a proxy of prominent world personalities.

For each person in the list, we download the corresponding Wikipedia page in English and preprocess it using TheWikiMachine library[6]. Overall, we collect 11,075 pages, while 265 pages could not be retrieved because of problems with the links (mainly redirection links). We randomly select 100 pages as development set, 500 pages for test and the remaining 10,475 for building the training set.

For each page in the development and test set, we ask an annotator to assign a yes/no label to each section, to mark if it describes part of the person's life or not. We involve also a second annotator to label manually the development set (100 pages containing 834 sections). We compute Cohen's Kappa, which corresponds to 0.88. As a rule of thumb, this is considered an almost perfect agreement, which shows the clear-cut difference between biographical and non-biographical sections. The annotators use the Wikipedia original page to decide whether a section is biographical or not, therefore they can see both title and content of sections.

For the training set, we devise a novel methodology to acquire it completely automatically. We first extract from our collection of 10,475 pages the subset of pages containing a section called *Life* or *Biography*, which amount to 2,547. We consider such pages as a gold standard, since the presence of *Life* or *Biography* shows that their editors paid attention to the structure of the page, distinguishing between what belonged to the person's biography and what not. Therefore, all subsec-

[5] http://thedata.harvard.edu/dvn/dv/pantheon

[6] https://bitbucket.org/fbk/twm-lib

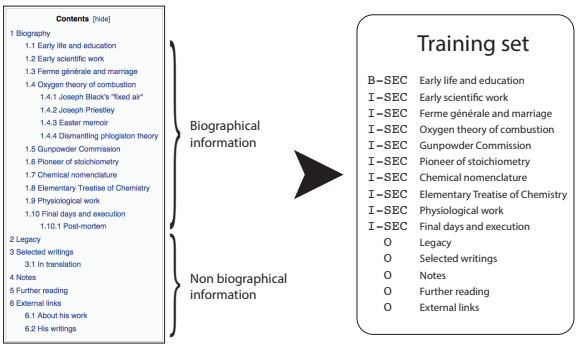


Figure 1: Extraction of training instances, from the Wikipedia page of Antoine Lavoisier (`https://en.wikipedia.org/wiki/Antoine_Lavoisier`). Sections are annotated in IOB2 format.

tions of *Life* or *Biography* are included as positive examples in the training set, because we assume that they all have biographical content. Instead, the remaining sections in the same pages represent our negative examples (see Figure 1). For the negative cases, we do not consider subsections because this level of detail is not needed for the classification, i.e. non-biographical sections contain only non-biographical subsections, and the classification of the first level is enough to propagate the corresponding label to the lower levels. Overall, the training set includes 2,547 sequences of sections (22,499 sections in total: 6,861 positive, 15,638 negative).

3.2 Classification experiment

We cast the problem as a supervised learning classification task, with the goal to label sequences of Wikipedia sections as describing a person's biography or not. As discussed in the introduction, we use Conditional Random Fields, since they are particularly suitable for sequence labelling (Lafferty et al., 2001). We use the implementation provided by CRFsuite[7] (Okazaki, 2007) for both training and classification tasks. The algorithm parameters are tuned on the development set.

In order to compare our approach with a non-sequential one, we perform the classification task also using Support Vector Machines (Vapnik, 1998). We use YAMCHA[8], a tool developed for chunking tasks, in which SVMs are easily combined with different context window sizes and dynamic features (Kudo and Matsumoto, 2001).

Since this work is only a preliminary step towards the automatic identification and extraction of biographical information from Wikipedia, we first experiment with the simplest approach. Therefore, we consider a small set of shallow features extracted *only from section titles*, and we ignore the content of the sections. Our six features are: the whole title, the tokens (lowercased), the bigrams, the first token of the section title, the first bigram, the position of the section with respect to the other sections in the same page (first, last, inside). We also use a sliding window of size 1 (both with CRF and SVM), so that the features extracted from the previous and the next sections are also considered to classify the current one. We experimented with window sizes > 1 on the development set, but they led to worse results.

Other features we implemented include the last word of the section, and a binary feature indicating whether the section title contains a year which is included between the date of birth and date of death of the person of interest. However, both pieces of information led to a performance drop on the development set, so we did not include them in the final feature set.

3.3 Baseline

To assess the performance of our system, we compare it with a baseline approach considering only the most frequent words in section titles. As shown in the evaluation (Section 4), this is indeed a very strong baseline. We identify the four most frequent tokens included in section titles, i.e. "Biography", "Life", "Death" and "Career", and we extract all sections that contain at least one of them (ignoring upper/lowercase). Finally, we consider as a positive example the smallest sequence of consecutive sections containing all of them. For instance, giving the sequence "Early life", "Career", "Presidency", "Death", "Legacy", "Awards", and "References", the first four sections are selected by the baseline as biographical sections: even if the "Presidency" word is not included in the tokens set, it is added as a consequence of its inclusion between "Early life" and "Death", both containing a token from the most frequent set.

[7] `http://www.chokkan.org/software/crfsuite/`

[8] `http://chasen.org/~taku/software/yamcha/`

4 Evaluation

We evaluate our system based on two different metrics, accounting for both exact and partial matches.

- In the *Exact* setting, a true positive is scored when *all* and *only* those sections with biographical information in a Wikipedia page are extracted. This measure is useful to understand how often it is possible to extract the *complete* and *exact* biographical text concerning a person.

- The *Intersection* measure, instead, assigns a score between 0 and 1 for every predicted sequence of sections based on how much it overlaps with the gold standard sequence (Johansson and Moschitti, 2013).

Evaluation results are reported in Table 1. The CRF-based approach outperforms the baseline in both configurations, with the highest improvement in the exact setting (+0.111 F1). Compared with the classification performance obtained with SVMs, CRFs yield better results only in the exact setting, while the intersection-based performance does not show substantial differences. In general, CRFs achieves a better precision but a lower recall than SVMs. If we look at the average length (in sections) of the false positive sequences, it is 3.72 for SVMs and 2.71 for CRFs. This difference confirms the behaviour of the SVM-based approach, which tends to overestimate the amount of biographical sections that should be tagged in sequence.

The results in Table 1 show also that a simple baseline relying on the four most frequent tokens in section titles achieves surprisingly good results, especially with the intersection-based metrics. This means that this basic approach tends to recognize correctly at least some of the sections describing a person's biography, because they present some recurrent patterns in their titles. In general, we observe that section titles alone are good indicators of their content, also without the need of more complex features. Although Wikipedia editors are free to name the sections and decide how to arrange them, there are some patterns that can be easily recognized automatically, especially by means of CRF.

	P	R	F1
CRFsuite			
Exact	0.694	0.662	0.677
Intersection	0.933	0.863	0.897
Yamcha SVM			
Exact	0.605	0.630	0.617
Intersection	0.857	0.942	0.898
Baseline			
Exact	0.584	0.548	0.566
Intersection	0.882	0.809	0.844

Table 1: Classification results using the *Exact* and *Intersection* settings

5 Discussion

We manually inspected the output of the classifier to identify possible issues. Apart from single classification mistakes, mainly due to unusual section titles that do not appear in the training set (such as "Anathematization" in Pope Honorius I page)[9], we found that some wrong classifications depended on specific types of persons in our data set. In particular, the classifier tends to assign a positive label to sections in the pages of mythological characters, even if they cannot have a biography because they did not exist. For instance, in the page of Apollo[10] there are sections entitled "Birth", "Youth", "Consorts and Children", which led the classifier to label them as biographical. Although mythological characters were included in the original Pantheon data set, we believe that they should be discarded, for instance by filtering them out a priori based on their Wikipedia categories.

Other false positives were found in pages of historical characters whose life was uncertain and was transmitted by others. For instance, the page of the geographer Pytheas[11] reports what the Roman author Pliny told about him, as well as what was said by other sources. These sections are very similar to those found in biographies, and are labelled as such.

We also performed additional experiments in order to investigate the impact of the size of the training data on the classification task. In particular, we extended our training data by creating new training sets based on 25,000, 50,000, 75,000 and 100,000 Wikipedia pages. These were obtained by

[9] https://en.wikipedia.org/wiki/Pope_Honorius_I

[10] https://en.wikipedia.org/wiki/Apollo

[11] https://en.wikipedia.org/wiki/Pytheas

ranking all pages with the PERSONDATA metadata according to the number of languages in which they are available, and then looking for the *Biography* and *Life* sections in the n top-ranked ones. Our evaluation shows that increasing the training size does not lead to a better performance, with an improvement of 0.01 with 100,000 pages over the results in Table 1 at the cost of a significant drop in processing speed.

6 Conclusions and Future Work

In this work we presented a simple yet effective approach to extract sequences of biographical sections from Wikipedia persons' pages. We model this task as a sequence classification problem using CRF and show that the section title alone conveys enough information to achieve a good classification result both in a supervised setting and with a rule-based baseline. Our contribution is threefold: *i)* we introduce the novel task of annotating the sequence of all sections describing a person's biography. This can be used as a preliminary step towards the extraction of all events characterizing a person's life; *ii)* we shed light on the regularities in Wikipedia persons' pages. Although Wikipedia is seen as a resource lacking consistency, with a flawed structure, our results show that at least persons' pages often present recurring patterns that are consistent across different biographies. The fact that only the most prominent figures have been included in the Pantheon data set is only a partial explanation of the good quality of such pages, because the English pages in our data set are often a reduced version of more extensive and edited pages in other languages. Finally, *iii)* we present an original approach to automatically acquire training data, using the pages with a *Biography* or *Life* section as gold data.

In the future, we plan to compare our approach based on section titles with more sophisticated approaches considering also the sections' content, to assess whether the latter improves over our simple methodology. Besides, we will build upon the outcome of this study by extracting the event sequence in a person's life starting from the complete biographies retrieved from Wikipedia.

The ongoing work is available as an open source project on GitHub[12] and is released under the GPLv3 license. In the project wiki one can find the dataset, the gold annotation and all the material needed to replicate the experiments.

[12] https://github.com/dkmfbk/biographies

Acknowledgments

The research leading to this paper was partially supported by the European Union's 7th Framework Programme via the NewsReader Project (ICT-316404). We would like to thank Naoaki Okazaki for the useful insight into CRFSuite.

References

David Bamman and Noah Smith. 2014. Unsupervised Discovery of Biographical Structure from Text. *Transactions of the Association for Computational Linguistics*, 2:363–376.

Fadi Biadsy, Julia Hirschberg, and Elena Filatova. 2008. An unsupervised approach to biography production using Wikipedia. In *In Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics (ACL-2008)*.

Maria Biryukov, Roxana Angheluta, and Marie-Francine Moens. 2005. Multidocument question answering text summarization using topic signatures. *Journal of Digital Information Management*, 3(1):27–33.

Elena Filatova and John M. Prager. 2005. Tell me what you do and i'll tell you what you are: Learning occupation-related activities for biographies. In *HLT/EMNLP 2005, Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference, 6-8 October 2005, Vancouver, British Columbia, Canada*.

Richard Johansson and Alessandro Moschitti. 2013. Relational features in fine-grained opinion analysis. *Computational Linguistics*, 39(3):473–509.

Taku Kudo and Yuji Matsumoto. 2001. Chunking with support vector machines. In *Second Meeting of the North American Chapter of the Association for Computational Linguistics*.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Naoaki Okazaki. 2007. CRFsuite: a fast implementation of Conditional Random Fields (CRFs). `http://www.chokkan.org/software/crfsuite/`.

Vladimir N. Vapnik. 1998. *Statistical Learning Theory*. Wiley-Interscience.

Wikipedia Contributors. 2014. Wikipedia: Systematic bias. https://en.wikipedia.org/wiki/Wikipedia:Systemic_bias. Accessed: 2015-06-14.

Amy Zhao Yu, Shahar Ronen, Kevin Zeng Hu, Tiffany Lu, and César A. Hidalgo. 2015. Pantheon: A Dataset for the Study of Global Cultural Production. *CoRR*, abs/1502.07310.

Liang Zhou, Miruna Ticrea, and Eduard H. Hovy. 2004. Multi-document biography summarization. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing , EMNLP 2004, A meeting of SIGDAT, a Special Interest Group of the ACL, held in conjunction with ACL 2004, 25-26 July 2004, Barcelona, Spain*, pages 434–441.

Learn to Solve Algebra Word Problems Using Quadratic Programming

Lipu Zhou, Shuaixiang Dai, and Liwei Chen
Baidu Inc., Beijing, China
zhoulipu@outlook.com, {daishuaixiang, chenliwei}@baidu.com

Abstract

This paper presents a new algorithm to automatically solve algebra word problems. Our algorithm solves a word problem via analyzing a hypothesis space containing all possible equation systems generated by assigning the numbers in the word problem into a set of equation system templates extracted from the training data. To obtain a robust decision surface, we train a log-linear model to make the margin between the correct assignments and the false ones as large as possible. This results in a quadratic programming (QP) problem which can be efficiently solved. Experimental results show that our algorithm achieves 79.7% accuracy, about 10% higher than the state-of-the-art baseline (Kushman et al., 2014).

1 Introduction

An algebra word problem describes a mathematical problem which can be typically modeled by an equation system, as demonstrated in Figure 1. Seeking to automatically solve word problems is a classical AI problem (Bobrow, 1964). The word problem solver is traditionally created by the rule-based approach (Lev et al., 2004; Mukherjee and Garain, 2008; Matsuzaki et al., 2013). Recently, using machine learning techniques to construct the solver has become a new trend (Kushman et al., 2014; Hosseini et al., 2014; Amnueypornsakul and Bhat, 2014; Roy et al., 2015). This is based on the fact that word problems derived from the same mathematical problem share some common semantic and syntactic features due to the same underlying logic. Our method follows this trend.[1]

To solve a word problem, our algorithm analyzes all the possible ways to assign the numbers in the word problem to a set of equation system templates. Kushman et al. (2014) also consider filling the equation system templates to generate the candidate equations. But Kushman's template contains number slots (e.g. n_1, n_2, n_3, n_4 in Figure 1) and unknown slots (e.g. u_1^1, u_1^2, u_2^1, u_2^2 in Figure 1). They separately consider assigning nouns into the unknown slots and numbers into the number slots, as demonstrated in Figure 1. As filling the unknown slots is closely related to the number slots assignment, we only consider assigning the number slots, and design effective features to describe the relationship between numbers and unknowns. This scheme significantly reduces the hypothesis space, as illustrated in Figure 1, which benefits the learning and inference processes.

	Our method	Kushman's method
Word problem	An **amusement park** sells **2 kinds** of **tickets**. **Tickets** for **children** cost **$ 1.50**. **Adult tickets** cost **$ 4**. On a certain **day**, **278 people** entered the **park**. On that same **day** the **admission fees** collected totaled **$ 792.** How many **children** were admitted on that **day**? How many **adults** were admitted?	
Template	$u_1+u_2-n_1=0$ $n_2\times u_1+n_3\times u_2-n_4=0$	$u_1^1+u_2^1-n_1=0$ $n_2\times u_1^2+n_3\times u_2^2-n_4=0$
Assignment	2, 1.5, 4, 278, 792 ⇓ n_1, n_2, n_3, n_4	2, 1.5, 4, 278, 792, *nouns* ⇓ $n_1,n_2,n_3,n_4,u_1^1,u_2^1,u_1^2,u_2^2$
Possible Assignment	$5\times4\times3\times2=120$	$17^4\times5\times4\times3\times2=10022520$

Figure 1: Comparison between our algorithm and (Kushman et al., 2014). Nouns are boldfaced.

We use a log-linear model to describe the template selection and number assignment. To learn the model parameters of such problem, maximizing the log-likelihood objective is generally adopted (Kwiatkowski et al., 2010; Kushman et al., 2014). The key difficulty of this method is that calculating the gradient of the objective function needs to sum over exponentially many samples. Thus, it is essential to approximate the gradient. For instance, Kushman et al. (2014) use

[1] Our code is available at http://pan.baidu.com/s/1dD336Sx

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 817–822,
Lisbon, Portugal, 17-21 September 2015.

beam search to approximately calculate the gradient. This method can not exploit all the training samples. Thus the resulting model may be suboptimal. Motivated by the work (Taskar et al., 2005; Li, 2014), we adopt the max-margin objective. This results in a QP problem and opens the way toward an efficient learning algorithm (Koller and Friedman, 2009).

We evaluate our algorithm on the benchmark dataset provided by (Kushman et al., 2014). The experimental results show that our algorithm significantly outperforms the state-of-the-art baseline (Kushman et al., 2014).

2 Problem Formulation

Our word problem solver is constructed by training a log-linear model to find the correct mapping from a word problem to an equation.

Notations: Let $\mathcal{X}$ denote the set of training word problems, and $\mathcal{T}$ denote the set of equation system templates abstracted from $\mathcal{X}$ as (Kushman et al., 2014). x_i is the *i*-th word problem in $\mathcal{X}$. Assume T_j is the *j*-th equation system template in $\mathcal{T}$, and $\mathcal{N}_{T_j} = \left\{n^1_{T_j}, n^2_{T_j}, \cdots, n^m_{T_j}\right\}$ is the set of number slots of T_j, where m represents the size of $\mathcal{N}_{T_j}$. Denote the numbers in x_i by $\mathcal{N}_{x_i} = \left\{n^1_{x_i}, n^2_{x_i}, \cdots, n^l_{x_i}\right\}$, where l represents the size of $\mathcal{N}_{x_i}$. Assuming $l \geq m$, we further define π_{ijk} a sequence of m numbers chosen from $\mathcal{N}_{x_i}$ without repetition. Given π_{ijk}, we can map T_j to an equation system e_{ijk} by filling the number slots $\mathcal{N}_{T_j}$ of T_j sequently with the numbers in π_{ijk}. Solving e_{ijk}, we can obtain the corresponding solution s_{ijk}. To simplify the notation, we define $y_{ijk} = (T_j, \pi_{ijk}, e_{ijk}, s_{ijk})$ the k-th derivation give x_i and T_j, and let $\mathcal{Y}_i$ denote the set of all possible y_{ijk} given x_i and $\mathcal{T}$. Therefore, to correctly solve x_i is to find the correct $y_{ijk} \in \mathcal{Y}_i$.

Probabilistic Model: As (Kushman et al., 2014), we use the log-linear model to define the probability of $y_{ijk} \in \mathcal{Y}_i$ given x_i:

$$p(y_{ijk}|x_i;\theta) = \frac{e^{\theta\cdot\phi\left(x_i,y_{ijk}\right)}}{\sum\limits_{y'_{ijk}\in\mathcal{Y}_i} e^{\theta\cdot\phi\left(x_i,y'_{ijk}\right)}} \quad (1)$$

where θ is the parameter vector of the model, and $\phi\left(x_i, y_{ijk}\right)$ denotes the feature function. We adopt the max-margin objective (Vapnik, 2013) to directly learn the decision boundary for the correct derivations and the false ones.

3 Learning and Inference

3.1 Learning

Using (1), we obtain the difference between the log-probability of a correct derivation $y^c_{ijk} \in \mathcal{Y}_i$ and a false one $y^f_{ijl} \in \mathcal{Y}_i$ as:

$$\begin{aligned}&\ln P\left(y^c_{ijk}|x_i;\theta\right) - \ln P\left(y^f_{ijl}|x_i;\theta\right)\\ =&\theta\cdot\left(\phi\left(x_i, y^c_{ijk}\right) - \phi\left(x_i, y^f_{ijl}\right)\right)\end{aligned} \quad (2)$$

Note that the subtraction in (2) cancels the denominator of (1) which contains extensive computation. To decrease the generalization error of the learned model, we would like the minimal gap between the correct derivations and the false ones as large as possible. In practice, we may not find a decision hyperplane to perfectly separate the correct and the false derivations. Generally, this can be solved by introducing a slack variable $\xi_{ijkl} \geq 0$ (Bishop, 2006) for each constraint derived from (2). Define $\varphi\left(x_i, y^c_{ijk}, y^f_{ijl}\right) = \phi\left(x_i, y^c_{ijk}\right) - \phi\left(x_i, y^f_{ijl}\right)$. For $\forall\, x_i \in \mathcal{X}$, the resulting optimization problem is:

$$\begin{aligned}&\arg\min \frac{1}{2}\|\theta\|_2 + C\sum_{i,j,k,l}\xi_{ijkl}\\ &s.t.\ \theta\cdot\varphi\left(x_i, y^c_{ijk}, y^f_{ijl}\right) \geq 1 - \xi_{ijkl},\ \xi_{ijkl} \geq 0\end{aligned} \quad (3)$$

The parameter C is used to balance the slack variable penalty and the margin. This is a QP problem and has been well studied (Platt, 1999; Fan et al., 2008).

According to the Karush-Kuhn-Tucker (KKT) condition, only a part of the constraints is active for the solution of (3) (Bishop, 2006). This leads to an efficient learning algorithm called constraint generation (Koller and Friedman, 2009; Felzenszwalb et al., 2010). Specifically, an initial model is trained by a randomly selected subset of the constraints. Next this model is used to check the constraints and at most N false deviations that are erroneously classified by this model are collected for each word problem. These constraints are then added to train a new model. This process repeats until converges. Our experimental results show that this process converges fast.

3.2 Inference

When we obtain the model parameter θ, the inference can be performed by finding the maximum

Single slot features
Relation between numbers and the question sentence.
Position of a number w.r.t a comparative word.
Context of a number.
Is one or two?
Is a multiplier?
Is between 0 and 1?
Slot pair features
Relation between two numbers.
Context similarity between two numbers.
Does there exist coreference relationship?
Are two numbers both multipliers?
Are two numbers in the same sentence or continuous sentences?
Information of raw path and dependency path between two numbers
One number is larger than another.
Solution features
Is integer solution?
Is positive solution?
Is between 0 and 1?

Table 1: Features used in our algorithm.

value of (1). This can be simplified by computing

$$\arg\max_{y_{ijk} \in \mathcal{Y}_i} \theta \cdot \phi(x_i, y_{ijk}) \quad (4)$$

As we only consider assigning the number slots of the templates in $\mathcal{T}$, generally, the size of the possible assignments per word problem is bearable, as shown in the Table 2. Thus we simply evaluate all the $y_{ijk} \in \mathcal{Y}_i$. The one with the largest score is considered as the solution of x_i.

4 Features

A feature vector $\phi(x_i, y_{ijk})$ is calculated for each word problem x_i and derivation y_{ijk} pair. As Kushman (2014), a feature is associated with a signature related to the template of y_{ijk}. We extract three kinds of features, i.e., single slot features, slot pair features and solution features. Unless otherwise stated, single slot and slot pair features are associated with the slot and slot pair signature of the equation system template, respectively, and solution features are generated for the signature of the equation system template. Table 1 lists the features used in our algorithm. The detailed description is as follows.

4.1 Single Slot Features

To reduce the search space, we only consider the assignment of the number slots of the template. It seems that our algorithm will lose the information about the unknown. But such information can be recovered by the features that include the information of the question sentence. Specifically, we associate a number with all the nouns in the same sentence sorted by the length of the dependence path between them. For instance, [$, tickets, children] is the sorted noun list for 1.5 in Figure 1. Assume the n-th noun of the nouns associated to a given number is the first noun that appears in the question sentence. We quantify the relationship between a number and a queried entity by the reciprocal of n. For instance, in Figure 1, "children" appears in the question sentence, and it is the third noun associated to 1.5. So the value of this feature is $1/3$. A larger value of this feature means a number more likely relates to the queried entity. The maximum value of this feature is 1. Thus we introduce a feature to indicate whether this special case occurs. We also use a feature to indicate whether a number appears in the question sentence.

The comparative meaning is sensitive to both the comparative words and the position of a number relative to them. For example, "one number is 3 less than twice another" is different to "one number is 3 more than twice another", but equal to "twice a number is 3 more than another". To account for this, we use the comparative words coupled with the position of a number relative to them as features.

On the other hand, we use the lemma, part of speech (POS) tag and the dependence type related to the word within a widow [-5, +5] around a number as features. Besides, if the POS tag or the named entity tag of a number is not labeled as a general number, we also import these tags together with the first noun and the dependence type related to the number as features.

Additionally, the numbers 1 and 2 are usually used to indicate the number of variables, such as "the sum of two numbers". To capture such usage, we use a feature to denote whether a number is one or two as (Kushman et al., 2014). Since such usage appears in various kinds of word problems, this feature does not contain the slot signature. We also generate features to indicate whether a number belongs to (0, 1), and whether it is a multiplier, such as twice, triple.

4.2 Slot Pair Features

Assume n_1 and n_2 are two numbers in a word problem. Suppose NP_1 and NP_2 are the lists of nouns associated to n_1 and n_2 (described in section 4.1), respectively. We evaluate the relationship $r(n_1, n_2)$ between n_1 and n_2 by:

$$\max_{\substack{noun_1^i \in NP_1, \\ noun_2^j \in NP_2 \\ s.t.\ noun_1^i = noun_2^j}} \left(\frac{2}{ord\left(noun_1^i\right) + ord\left(noun_2^j\right)} \right)$$

where $ord(\cdot)$ denotes the index of a noun in $NP_i\,(i = 1, 2)$, starting from 1. A larger $r(n_1, n_2)$ means n_1 and n_2 are more related. The maximum value of $r(n_1, n_2)$ is 1, which occurs when the first nouns of NP_1 and NP_2 are equal. We use a feature to indicate whether $r(n_1, n_2)$ is 1. This feature helps to import some basic rules of the arithmetic operation, e.g., the units of summands should be the same.

If two slots are symmetric in a template (e.g., n_2 and n_3 in Figure 1), the contexts around both numbers are generally similar. Assume CT_1 and CT_2 are two sets of certain tags within a window around n_1 and n_2, respectively. Then we calculate the contextual similarity between n_1 and n_2 by:

$$sim(ST_1, ST_2) = \frac{|ST_1 \cap ST_2|}{|ST_1 \cup ST_2|}$$

In this paper, the tags include the lemma, POS tag and dependence type, and the window size is 5.

Besides, we exploit features to denote whether there exists coreference relationship between any elements of the sentences where n_1 and n_2 locate, and whether two numbers are both multipliers. Finally, according to (Kushman et al., 2014), we generate features related to the raw path and dependence path between two numbers, and use the numeric relation between them as a feature to import some basic arithmetic rules, such as the positive summands are smaller than their sum. We also include features to indicate whether two numbers are in the same sentence or continuous sentences.

4.3 Solution Features

Many word problems are math problems about the real life. This background leads the solutions of many word problems have some special numerical properties, such as the positive and integer properties used by (Kushman et al., 2014). To capture such fact, we introduce a set of features to describe the solution properties.

5 Experiments

Dataset: The dataset used in our experiment is provided by (Kushman et al., 2014). Equivalent equation systme templates are automatically merged. The word problems are parsed by (Manning et al., 2014). The version of the parser is the same as (Kushman et al., 2014). The performance of our algorithm is evaluated by comparing each number of the correct answer with the calculated one, regardless of the ordering. We report the average accuracy of 5-fold cross-validation.

Learning: We use liblinear (Fan et al., 2008) to solve the QP problem. The parameter C in (3) is set to 0.01 in all the following experiments. We randomly select 300 false derivations of each word problem to form the initial training set. We add at most 300 false derivations for each word problem during the constraint generation step, and use 5-fold cross-validation to avoid overfitting. We stop iterating when the cross-validation error becomes worse or the training error converges or none new constraints are generated.

Supervision Level: We consider the learning with two different levels of supervision. In the first case, the learning is conducted by providing the equation and the correct answer of every training sample. In the second case, the correct answer is available for every training sample but without the equation. Instead, all the templates are given, but the correspondence between the template and the training sample is not available. During learning, the algorithm should evaluate every derivation of each template to find the true one.

Results: Table 2 lists the learning statistics for our algorithm and (Kushman et al., 2014). We can observe that the number of possible alignments per word problem of our algorithm is much smaller than (Kushman et al., 2014). However, the number of all the false alignments is still 80K. Using the constraint generation algorithm (Koller and Friedman, 2009), only 9K false alignments are used in the quadratic programming. We trained our model on a Intel i5-3210M CUP and 4G RAM laptop. Kushman's algorithm (2014) needs much more memory than our algorithm and can not run on a general laptop. Therefore, we tested their algorithm on a workstation with Intel E5-2620 CPU and 128G memory. As shown in Table 2, their algorithm takes more time than our algorithm.

Table 3 lists the accuracy of our algorithm and Kushman's algorithm (2014). It is clear that our

Mean negative samples	80K
Mean negative samples used in learning	9K
Mean time for feature extraction	22m
Mean training time	7.3m
Mean feature extraction and training time of (Kushman et al., 2014)	83m
# Alignments per problem of (Kushman et al., 2014)	4M
# Alignments per problem of our algorithm	1.9K

Table 2: Learning statistics.

Algorithm	**Accuracy**
Our algorithm fully supervised	**79.7%**
Our algorithm weakly supervised	72.3%
Kushman's algorithm (2014) fully supervised	68.7%

Table 3: Algorithm comparison.

Feature Ablation	**Accuracy**
Without single slot features	70.4%
Without slot pair features	69.3%
Without solution features	71.8%

Table 4: Ablation study for fully supervised data.

algorithm obtains better result. The result of the weakly supervised data is worse than the fully supervised one. But this result is still higher than Kushman's fully supervised result.

Table 4 gives the results of our algorithm with different feature ablations. We can find that all the features are helpful to get the correct solution and none of them dramatically surpasses the others.

Discuss: Although our algorithm gives a better result than (Kushman et al., 2014), there still exist two main problems that need to be further investigated, as demonstrated in Table 5. The first problem is caused by our feature for semantic representation. Our current lexicalized feature can not generalize well for the unseen words. For example, it is hard for our algorithm to relate the word "forfeits" to "minus", if it does not appear in the training corpus. The second problem is caused by the fact that our algorithm only considers the single noun as the entity of a word problem. Thus when the entity is a complicated noun phrase, our algorithm may fail.

Problem	**Example**
Lexicalized features can not generalize well for unseen words.	A woman is paid 20 dollars for each day she works and **forfeits** a 5 dollars for each day she is idle. At the end of 25 days she nets 450 dollars. How many days did she work?
Can not deal with complicated noun phrases.	**The probability that San Francisco plays in the next super bowl** is nine times **the probability that they do not play in the next super bowl**. **The probability that San Francisco plays in the next super bowl** plus **the probability that they do not play** is 1. What is **the probability that San Francisco plays in the next super bowl?**

Table 5: The problems of our algorithm.

6 Conclusion and Future work

In this paper, we present a new algorithm to learn to solve algebra word problems. To reduce the possible derivations, we only consider filling the number slots of the equation system templates, and design effective features to describe the relationship between numbers and unknowns. Additionally, we use the max-margin objective to train the log-linear model. This results in a QP problem that can be efficiently solved via the constraint generation algorithm. Experimental results show that our algorithm significantly outperforms the state-of-the-art baseline (Kushman et al., 2014).

Our future work will focus on studying the performance of applying nonlinear kernel function to the QP problem (3), and using the word embedding vector (Bengio et al., 2003; Mikolov et al., 2013) to replace current lexicalized features. Besides, we would like to compare our algorithm with the algorithms designed for specific word problems, such as (Hosseini et al., 2014).

7 Acknowledgments

This work is supported by the National Basic Research Program of China (973 program No. 2014CB340505). We would like to thank Hua Wu and the anonymous reviewers for their helpful comments that improved the work considerably.

References

Bussaba Amnueypornsakul and Suma Bhat. 2014. Machine-guided solution to mathematical word problems.

Yoshua Bengio, Rjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, 3(6):1137–1155.

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

Daniel G Bobrow. 1964. Natural language input for a computer problem solving system.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. Liblinear: A library for large linear classification. *The Journal of Machine Learning Research*, 9:1871–1874.

Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan. 2010. Object detection with discriminatively trained part-based models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(9):1627–1645.

Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. 2014. Learning to solve arithmetic word problems with verb categorization. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 523–533.

Daphne Koller and Nir Friedman. 2009. *Probabilistic graphical models: principles and techniques*. MIT press.

Nate Kushman, Yoav Artzi, Luke Zettlemoyer, and Regina Barzilay. 2014. Learning to automatically solve algebra word problems. *ACL (1)*, pages 271–281.

Tom Kwiatkowski, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2010. Inducing probabilistic ccg grammars from logical form with higher-order unification. In *Proceedings of the 2010 conference on empirical methods in natural language processing*, pages 1223–1233. Association for Computational Linguistics.

Iddo Lev, Bill MacCartney, Christopher D Manning, and Roger Levy. 2004. Solving logic puzzles: From robust processing to precise semantics. In *Proceedings of the 2nd Workshop on Text Meaning and Interpretation*, pages 9–16. Association for Computational Linguistics.

Hang Li. 2014. Learning to rank for information retrieval and natural language processing. *Synthesis Lectures on Human Language Technologies*, 7(3):1–121.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60.

Takuya Matsuzaki, Hidenao Iwane, Hirokazu Anai, and Noriko Arai. 2013. The complexity of math problems–linguistic, or computational. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 73–81.

Tomas Mikolov, Wen Tau Yih, and Geoffrey Zweig. 2013. Linguistic regularities in continuous space-word representations. *In HLT-NAACL*.

Anirban Mukherjee and Utpal Garain. 2008. A review of methods for automatic understanding of natural language mathematical problems. *Artificial Intelligence Review*, 29(2):93–122.

John C. Platt. 1999. Fast training of support vector machines using sequential minimal optimization. *In B. Scho04lkopf, C. Burges and A. Smola (Eds.), Advances in kernel methods - Support vector learning*.

Subhro Roy, Tim Vieira, and Dan Roth. 2015. Reasoning about quantities in natural language. *Transactions of the Association for Computational Linguistics*, 3:1–13.

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

Vladimir Vapnik. 2013. *The nature of statistical learning theory*. Springer Science & Business Media.

An Unsupervised Method for Discovering Lexical Variations in Roman Urdu Informal Text

Abdul Rafae, ‡Abdul Qayyum, ‡Muhammad Moeenuddin,
Asim Karim, †Hassan Sajjad and ‡Faisal Kamiran,
†Qatar Computing Research Institute, Hamad Bin Khalifa University
Lahore University of Management Sciences, ‡Information Technology University

Abstract

We present an unsupervised method to find lexical variations in Roman Urdu informal text. Our method includes a phonetic algorithm *UrduPhone*, a feature-based similarity function, and a clustering algorithm *Lex-C*. UrduPhone encodes roman Urdu strings to their phonetic equivalent representations. This produces an initial grouping of different spelling variations of a word. The similarity function incorporates word features and their context. Lex-C is a variant of k-medoids clustering algorithm that group lexical variations. It incorporates a similarity threshold to balance the number of clusters and their maximum similarity. We test our system on two datasets of SMS and blogs and show an f-measure gain of up to 12% from baseline systems.

1 Introduction

Urdu is the national language of Pakistan and one of the official languages of India. It is written in Perso-Arabic script. However in social media and short text messages (SMS), a large proportion of Urdu speakers use roman script (i.e., the English alphabet) for writing, called *Roman Urdu*.

Roman Urdu lacks standard lexicon and usually many spelling variations exist for a given word, e.g., the word *zindagi* [life] is also written as *zindagee*, *zindagy*, *zaindagee* and *zndagi*. Specifically, the following normalization issues arise: (1) differently spelled words (see example above), (2) identically spelled words that are lexically different (e.g., *bahar* can be used for both [outside] and [spring], and (3) spellings that match words in English (e.g, *had* [limit] for the English word 'had'). These inconsistencies cause a problem of data sparsity in basic natural language processing tasks such as Urdu word segmentation (Durrani and Hussain, 2010), part of speech tagging (Sajjad and Schmid, 2009), spell checking (Naseem and Hussain, 2007), machine translation (Durrani et al., 2010), etc.

In this paper, we propose an unsupervised feature-based method that tackles above mentioned challenges in discovering lexical variations in Roman Urdu. We exploit phonetic and string similarity based features and incorporate contextual features via top-k previous and next words' features. For phonetic information, we develop an encoding scheme for Roman Urdu, *UrduPhone*, motivated from Soundex. Compared to other available phonetic-based schemes that are mostly limited to English sounds only, UrduPhone maps Roman Urdu homophones effectively. Unlike previous work on short text normalization (see Section 2), we do not have information about standard word forms in the dataset. The problem becomes more challenging as every word in the corpus is a candidate of every other word. We present a variant of the k-medoids clustering algorithm that forms clusters in which every word has at least a specified minimum similarity with the cluster's centroidal word. We conduct experiments on two Roman Urdu datasets: an SMS dataset and a blog dataset and evaluate performance using a gold standard. Our method shows an f-measure gain of up to 12% compared to baseline methods. The dataset and code are made available to the research community.

2 Previous Work

Normalization of short text messages and tweets has been in focus (Sproat et al., 2001; Wei et al., 2011; Clark and Araki, 2011; Roy et al., 2013; Chrupala, 2014; Kaufmann and Kalita, 2010; Sidarenka et al., 2013; Ling et al., 2013; Desai and Narvekar, 2015; Pinto et al., 2012). However, most of the work is limited to English or to other

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 823–828,
Lisbon, Portugal, 17-21 September 2015.

resource-rich languages. In this paper, we focus on Roman Urdu, an under-resourced language, that does not have any gold standard corpus with standard word forms. Therefore, we are restricted to the task of finding lexical variations in informal text. This is a rather more challenging problem since in this case every word is a possible variation of every other word in the corpus.

Researchers have used phonetic, string, and contextual knowledge to find lexical variations in informal text.[1] Pinto et al. (2012; Han et al. (2012; Zhang et al. (2015) used phonetic-based methods to find lexical variations. Han et al. (2012) also used word similarity and word context to enhance performance. Wang and Ng (2013) used normalization operations e.g., missing word recovery and punctuation correction to improve normalization process. Irvine et al. (2012) used manually prepared training data to build an automatic normalization system. Contractor et al. (2010) used string edit distance to find candidate lexical variations. Yang and Eisenstein (2013) used an unsupervised approach with log linear model and sequential Monte Carlo approximation.

We propose an unsupervised method to find lexical variations. It uses string edit distance like Contractor et al. (2010), Sound-based encoding like Pinto et al. (2012) and context like Han et al. (2012) combined in a discriminative framework. However, in contrast, it does not use any corpus of standard word forms to find lexical variations.

3 Our Method

The lexical variations of a lexical entry usually have high phonetic, string-based, and contextual similarity. We integrate a phonetic-based encoding scheme, UrduPhone, a feature-based similarity function, and a clustering algorithm, Lex-C.

3.1 UrduPhone

Several sound-based encoding schemes for words have been proposed in literature such as *Soundex* (Knuth, 1973; Hall and Dowling, 1980), *NYSIIS* (Taft, 1970), *Metaphone* (Philips, 1990), Caverphone (Wang, 2009) and Double Metaphone.[2] These schemes encode words based on their sound which in turn serves as grouping words of similar sounds (lexical variations) to one code. However, most of the schemes are designed for English and European languages and are limited when apply to other family of languages like Urdu.

In this work, we propose a phonetic encoding scheme, *UrduPhone*, tailored for Roman Urdu. The scheme is derived from the Soundex algorithm. It groups consonants on the basis of common homophones in Urdu and English. It is different from Soundex in two particular ways:[3] Firstly, UrduPhone generates encoding of length six compared to length four in Soundex. This enables UrduPhone to avoid mapping different forms of a root word to same code. For example, *muskurana* [smiling] and *mshuraht* [smile] encode to one form *MSKR* in Soundex but in UrduPhone, they have different encoding which are *MSKRN*, *MSKRHT* respectively. Secondly, we introduce consonant groups which are mapped differently in Soundex. We do this by analyzing Urdu alphabets that map to a single roman form e.g. words *samar* [reward], *sabar* [patience] and *saib* [apple], all start with different Urdu alphabets that have identical roman representation: *s*. In UrduPhone, we map all such cases to a single form.[4]

3.2 Similarity Function

The similarity between two words w_i and w_j is computed by the following similarity function:

$$S(w_i, w_j) = \frac{\sum_{f=1}^{F} \alpha^{(f)} \times \sigma_{ij}^{(f)}}{\sum_{f=1}^{F} \alpha^{(f)}}$$

Here, $\alpha^{(f)} > 0$ is the weight given to feature f, $\sigma_{ij}^{(f)} \in [0, 1]$ is the similarity contribution made by feature f, and F is the total number of features. In the absence of additional information, and as used in the experiments in this work, all weights can be taken equal to one. The similarity function returns a value in the interval [0, 1] with larger values signifying higher similarity.

We use two types of features in our method: word features and contextual features. Word features can be based on phonetics and/or string similarity. The phonetic similarity between words w_i and w_j is 1 (i.e., $\sigma_{ij} = 1$) if both words have the same UrduPhone ID or encoding; otherwise, their

[1] Spell correction is also considered as a variant of text normalization (Damerau, 1964; Tahira, 2004; Fossati and Di Eugenio, 2007). Here, we limit ourselves to the previous work on short text normalization.

[2] http://en.wikipedia.org/wiki/Metaphone

[3] Due to limited space, we limit the description of UrduPhone to its comparison with Soundex.

[4] A complete table of UrduPhone mappings is provided in the supplementary material.

similarity is zero. The string similarity between words w_i and w_j is defined as follows:

$$\sigma_{ij} = \frac{lcs(w_i, w_j)}{\min[len(w_i), len(w_j)] \times edist(w_i, w_j)}$$

Here, $lcs(w_i, w_j)$ is the length of the longest common subsequence in words w_i and w_j and $len(w_i)$ is the length of word w_i. $edist(w_i, w_j)$ returns the edit distance between words except when the edit distance is 0, in which case it returns 1.

Contextual features include top-k frequently occurring previous and next words' features. Let $a_1^i, a_2^i, \ldots, a_5^i$ and $a_1^j, a_2^j, \ldots, a_5^j$ be the word IDs for the top-5 frequently occurring words preceding word w_i and w_j, respectively. Then, the similarity between words is given by (Hassan et al., 2009)

$$\sigma_{ij} = \frac{\sum_{k=1}^{5} \rho_k}{\sum_{k=1}^{5} k}$$

Here, ρ_k is zero if a_k^i does not have a match in a_*^j (i.e., in the context of word w_j); otherwise, $\rho_k = 5 - \max[k, l] - 1$ where $a_k^i = a_l^j$ and l is the highest rank (smallest integer) at which a previous match had not occurred. Instead of word IDs in a_i's, UrduPhone IDs or string similarity based cluster IDs can be used to reduce sparsity and improve matches among similar words.

3.3 Lex-C: Clustering Algorithm

We develop a new clustering algorithm, called Lex-C, for discovering lexical variations in informal text. This algorithm is a modified version of the k-medoids algorithm (Han, 2005). It incorporates an assignment similarity threshold, $t > 0$, for controlling the number of clusters and their similarity. In particular, it ensures that all words in a cluster have a similarity greater than or equal to this threshold. It is important to note that the poular k-means algorithm is known to be effective for numeric datasets only which is not true in our case, and it cannot utilize our specialized similarity function for lexical variation discovery.

Specifically, Lex-C starts from an initial clustering based on UrduPhone or string similarity. It finds the centroidal word, w_c^k, for cluster k as the word with which the sum of similarities of all other words in the cluster is a maximum. Then, each non-centroidal word is assigned to the cluster k if $S(w_i, w_c^k)$ is a maximum among all clusters *and* $S(w_i, W_c^k) \geq t$. If the latter condition is not satisfied (i.e., $S(w_i, w_c^k) < t$) then instead of assigning word w_i to cluster k, it starts a new cluster. These two steps are repeated until convergence.

4 Experimental Evaluation

We empirically evaluate UrduPhone and our complete method involving Lex-C separately on two real-world datasets. Performance is reported with B-Cubed precision, recall, and f-measure (Bagga and Baldwin, 1998; Hassan et al., 2015) on a gold standard dataset. These performance measures are based on element-wise comparisons between predicted and actual clusters that are then aggregated over all elements in the clustering. This avoided the issue of 100% precision with low recall (all words belong to separate clusters) and 100% recall with low precision (all words belong to one cluster).

4.1 Dataset and Gold Standard

The first dataset, Web dataset, is scraped from Roman Urdu websites on news[5], poetry[6], SMS[7] and blog[8]. The second dataset, SMS dataset, is obtained from chopaal, an internet based group SMS service[9]. For evaluation, we use a manually annotated database of Roman Urdu variations (Khan and Karim, 2012). Table 1 shows statistics of the datasets in comparison with the gold standard.

Dataset	Web	SMS
Unique words	22,044	28,908
Overlap with Gold Standard	12,600	13,087
UrduPhone IDs	3,952	3,599

Table 1: Datasets and gold standard statistics. Overlap with gold standard = number of words appearing in gold standard; UrduPhone IDs = number of distinct UrduPhone encodings.

4.2 UrduPhone Evaluation

We compare UrduPhone with Soundex and its variants.[10] These algorithms are used to group words based on their encoding and then evaluated against the gold standard. Table 2 shows

[5] http://www.shashca.com,http://stepforwardpak.com/

[6] https://hadi763.wordpress.com/

[7] http://www.replysms.com/

[8] http://roman.urdu.co/

[9] http://chopaal.org

[10] We use NLTK-Trainer's phonetic library `http://bit.ly/1OJGL9Q`

the results on Web dataset. UrduPhone outperforms Soundex, Caverphone, and Metaphone while Nysiis's f-measure is comparable to that of UrduPhone. We observe that Nysiis produces a large number of single word clusters (out of 6,943 clusters produced 5,159 have only one word). This gives high precision but recall is low. UrduPhone produces fewer clusters (and fewer one word clusters) with high recall.

Algorithm	Pre	Rec	Fme
Soundex	0.30	0.97	0.46
Metaphone	0.49	0.80	0.64
Caverphone	0.31	0.92	0.46
Nysiis	0.63	0.69	0.66
UrduPhone	0.51	0.94	**0.67**

Table 2: Comparison of UrduPhone with other algorithms on Web dataset

4.3 Lex-C Evaluation

We compared Lex-C with k-means and EM clustering algorithms. With both of these algorithms we used the same feature set (i.e., word features, phonetic features, and contextual features), However, their performance lagged the performance of our approach. The primary reason for this is that our feature space is not continuous while k-means and EM algorithms work best for continuous feature spaces.

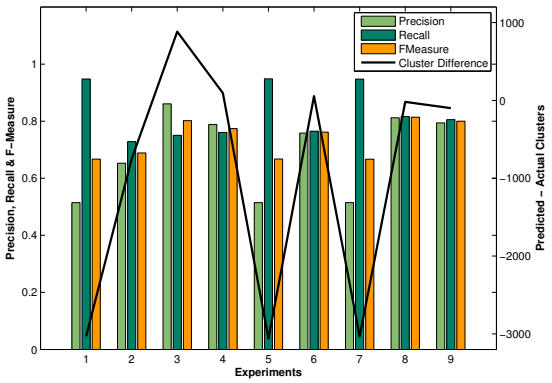


Figure 1: Performance results for Web dataset

4.4 Performance of our Method

We conduct extensive experiments to evaluate the performance of our method. We vary initial clusterings (UrduPhone encoding or string similarity based clustering); evaluate various combinations of phonetic, string, and contextual features; and consider different previous/next top-5 words' features (word ID, cluster ID, and/or UrduPhone ID). Table 3 gives details of each experiment setting.

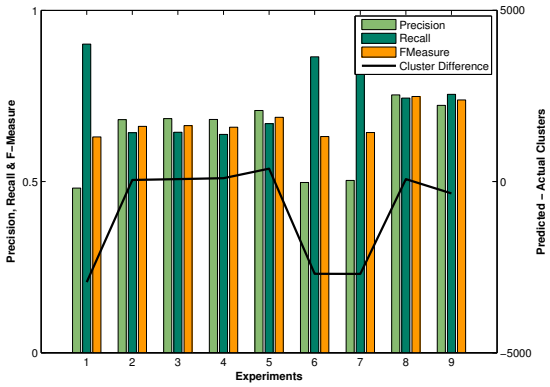


Figure 2: Performance results for SMS dataset

ID	Initial	Word	Context-1	Context-2
1	–	UPhone	–	–
2	–	String	–	–
3	String	String	Word ID	–
4	String	String	Cluster ID	–
5	String	UPhone	Cluster ID	–
6	UPhone	UPhone	Cluster ID	–
7	UPhone	UPhone	Word ID	–
8	UPhone	UPhone	UPhone ID	Word ID
9	UPhone	UPhone	UPhone ID	Cluster ID

Table 3: Details of experiments' settings where Initial is initial clustering based on String or UrduPhone (UPhone)

Figures 1 and 2 show results of selected experiments for Web and SMS datasets respectively. The x-axis shows the experiment (Exp.) IDs while the left y-axis gives the precision, recall, and f-measure and the right y-axis shows the difference between the number of predicted and actual clusters. Exp. 1 and 2 are baselines corresponding to UrduPhone encoding (UPhone ID) and string similarity based word clustering (Cluster ID) respectively. The remaining experiments have different initial clustering, word features, and up to two contextual features. In these results, the similarity threshold t is selected such that the number of discovered clusters is as close as possible to the number of actual clusters in the gold standard for each dataset. This is done to make the results comparable across different settings.

Compared to baselines, our method shows a gain of up to 12% and 8% in Web and SMS datasets respectively. The best performances are obtained when UrduPhone is used as a feature and UrduPhone IDs are used to define the context (Exp. 8 and 9). In particular, when both Urdu-

Phone IDs and word IDs/cluster IDs are used for contextual information (i.e, with two sets of top-5 previous and next words' features) the f-measure is consistently high.

We analyzed the performance of Exp. 8 (best settings for Web dataset) with varying t and showed it in Figure 3. It is observed that the value of t controls the number of clusters smoothly, and precision increases with the number of clusters and f-measure reaches a peak when number of clusters is close to that in the gold standard.

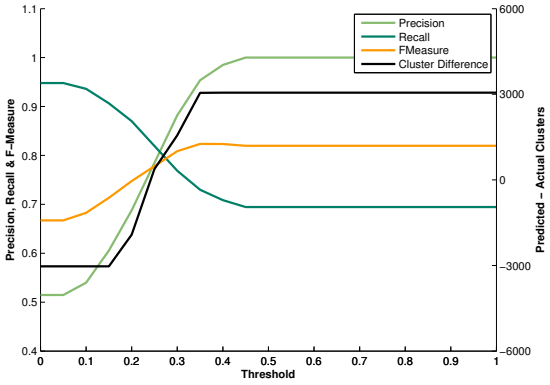


Figure 3: Effect of varying threshold t on Web dataset (experiment 8)

5 Conclusion

We proposed an unsupervised method for finding lexical variations in Roman Urdu. We presented a phonetic encoding scheme UrduPhone for Roman Urdu, and developed a feature-based clustering algorithm Lex-C. Our experiments are evaluated on a manually developed gold standard. The results confirmed that our method outperforms baseline methods. We made the datasets and algorithm code available to the research community.

References

Amit Bagga and Breck Baldwin. 1998. Algorithms for scoring coreference chains. In *In The First International Conference on Language Resources and Evaluation Workshop on Linguistics Coreference*, pages 563–566.

Grzegorz Chrupala. 2014. Normalizing tweets with edit scripts and recurrent neural embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 2: Short Papers*, pages 680–686.

Eleanor Clark and Kenji Araki. 2011. Text normalization in social media: Progress, problems and applications for a pre-processing system of casual english. *Procedia-Social and Behavioral Sciences*, 27:2–11.

Danish Contractor, Tanveer A Faruquie, and L Venkata Subramaniam. 2010. Unsupervised cleansing of noisy text. In *Proceedings of the 23rd International Conference on ComputationalLinguistics: Posters*, pages 189–196. Association for Computational Linguistics.

Fred J. Damerau. 1964. A technique for computer detection and correction of spelling errors. *Commun. ACM*, 7(3), March.

Neelmay Desai and Meera Narvekar. 2015. Normalization of noisy text data. *Procedia Computer Science*, 45:127–132.

Nadir Durrani and Sarmad Hussain. 2010. Urdu word segmentation. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 528–536, Los Angeles, California, June. Association for Computational Linguistics.

Nadir Durrani, Hassan Sajjad, Alexander Fraser, and Helmut Schmid. 2010. Hindi-to-urdu machine translation through transliteration. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 465–474, Uppsala, Sweden, July. Association for Computational Linguistics.

Davide Fossati and Barbara Di Eugenio. 2007. A mixed trigrams approach for context sensitive spell checking. In *Computational Linguistics and Intelligent Text Processing*, pages 623–633. Springer.

Patrick A. V. Hall and Geoff R. Dowling. 1980. Approximate string matching. *ACM Comput. Surv.*, 12(4):381–402.

Bo Han, Paul Cook, and Timothy Baldwin. 2012. Automatically constructing a normalisation dictionary for microblogs. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 421–432. Association for Computational Linguistics.

Jiawei Han. 2005. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Malik Tahir Hassan, Khurum Nazir Junejo, and Asim Karim. 2009. Learning and predicting key web navigation patterns using bayesian models. In *Computational Science and Its Applications–ICCSA*, pages 877–887. Springer.

Malik Tahir Hassan, Asim Karim, Jeong-Bae Kim, and Moongu Jeon. 2015. Cdim: Document clustering by discrimination information maximization. *Information Sciences*.

Ann Irvine, Jonathan Weese, and Chris Callison-Burch. 2012. Processing informal, romanized pakistani text messages. In *Proceedings of the Second Workshop on Language in Social Media*, LSM '12, pages 75–78. Association for Computational Linguistics.

Max Kaufmann and Jugal Kalita. 2010. Syntactic normalization of twitter messages. In *International Conference on Natural Language Processing, Kharagpur, India*.

Osama Khan and Asim Karim. 2012. A rule-based model for normalization of sms text. In *IEEE 24th International Conference on Tools with Artificial Intelligence, ICTAI 2012, Athens, Greece, November 7-9, 2012*, pages 634–641.

Donald E Knuth. 1973. *The Art of Computer Programming: Volume 3, Sorting and Searching*. Addison-Wesley.

Wang Ling, Chris Dyer, Alan W Black, and Isabel Trancoso. 2013. Paraphrasing 4 microblog normalization. In *EMNLP*, pages 73–84.

Tahira Naseem and Sarmad Hussain. 2007. A novel approach for ranking spelling error corrections for urdu. *Language Resources and Evaluation*, 41(2):117–128.

Lawrence Philips. 1990. Hanging on the metaphone. *Computer Language Magazine*, 7(12):39–44, December.

David Pinto, Darnes Vilariño Ayala, Yuridiana Alemán, Helena Gómez-Adorno, Nahun Loya, and Héctor Jiménez-Salazar. 2012. The soundex phonetic algorithm revisited for SMS text representation. In *Text, Speech and Dialogue - 15th International Conference, TSD 2012, Brno, Czech Republic, September 3-7, 2012. Proceedings*, pages 47–55.

Sudipta Roy, Sourish Dhar, Saprativa Bhattacharjee, and Anirban Das. 2013. A lexicon based algorithm for noisy text normalization as pre processing for sentiment analysis. *International Journal of Research in Engineering and Technology*, 02.

Hassan Sajjad and Helmut Schmid. 2009. Tagging urdu text with parts of speech: A tagger comparison. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics (EACL-09)*, pages 692–700, Athens, Greece.

Uladzimir Sidarenka, Tatjana Scheffler, and Manfred Stede. 2013. Rule-based normalization of german twitter messages. In *Proc. of the GSCL Workshop Verarbeitung und Annotation von Sprachdaten aus Genres internetbasierter Kommunikation*.

Richard Sproat, Alan W. Black, Stanley F. Chen, Shankar Kumar, Mari Ostendorf, and Christopher Richards. 2001. Normalization of non-standard words. *Computer Speech & Language*, 15(3):287–333.

R. Taft. 1970. Name search techniques. *Special report. Bureau of Systems Development,New York State Identification and Intelligence System*.

Naseem Tahira. 2004. A hybrid approach for Urdu spell checking.

Pidong Wang and Hwee Tou Ng. 2013. A beam-search decoder for normalization of social media text with application to machine translation. In *HLT-NAACL*, pages 471–481.

John Wang, editor. 2009. *Encyclopedia of Data Warehousing and Mining, Second Edition (4 Volumes)*. IGI Global.

Zhongyu Wei, Lanjun Zhou, Binyang Li, Kam-Fai Wong, Wei Gao, and Kam-Fai Wong. 2011. Exploring tweets normalization and query time sensitivity for twitter search. In *TREC*.

Yi Yang and Jacob Eisenstein. 2013. A log-linear model for unsupervised text normalization. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 61–72.

Xin Zhang, Jiaying Song, Yu He, and Guohong Fu. 2015. Normalization of homophonic words in chinese microblogs. In *Intelligent Computation in Big Data Era*, pages 177–187. Springer.

Component-Enhanced Chinese Character Embeddings

Yanran Li[1], **Wenjie Li**[1], **Fei Sun**[2], and **Sujian Li**[3]
[1]Department of Computing, The Hong Kong Polytechnic University, Hong Kong
[2]Institute of Computing Technology, Chinese Academy of Sciences, China
[3]Key Laboratory of Computational Linguistics, Peking University, MOE, China
{csyli, cswjli}@comp.polyu.edu.hk, ofey.sunfei@gmail.com,
lisujian@pku.edu.cn

Abstract

Distributed word representations are very useful for capturing semantic information and have been successfully applied in a variety of NLP tasks, especially on English. In this work, we innovatively develop two component-enhanced Chinese character embedding models and their bigram extensions. Distinguished from English word embeddings, our models explore the compositions of Chinese characters, which often serve as semantic indictors inherently. The evaluations on both word similarity and text classification demonstrate the effectiveness of our models.

1 Introduction

Due to its advantage over traditional one-hot representation, distributed word representation has demonstrated its benefit for semantic representation in various NLP tasks. Among the existing approaches (Huang et al, 2012; Levy and Goldberg, 2014; Yang and Eisenstein, 2015), the continuous bag-of-words model (CBOW) and the continuous skip-gram model (SkipGram) remain the most popular ones that one can use to build word embeddings efficiently (Mikolov et al, 2013a; Mikolov et al, 2013b). These two models learn the distributed representation of a word based on its context. The context defined by the window of surrounding words may unavoidably include certain less semantically-relevant words and/or miss the words with important and relevant meanings (Levy and Goldberg, 2014).

To overcome this shortcoming, a line of research deploys the order information of the words in the contexts by either deriving the contexts using dependency relations where the target word participates (Levy and Goldberg, 2014; Yu and Dredze, 2014; Bansal et al, 2014) or directly keeping the order features (Ling et al, 2015). As to another line, Luong et al (2013) captures morphological composition by using neural networks and Qiu et al (2014) introduces the morphological knowledge as both additional input representation and auxiliary supervision to the neural network framework. While most previous work focuses on English, there is a little work on Chinese. Zhang et al (2013) extracts the syntactical morphemes and Cheng et al (2014) incorporates the POS tags and dependency relations. Basically, the work in Chinese follows the same ideas as in English.

Distinguished from English, Chinese characters are logograms, of which over 80% are phono-semantic compounds, with a semantic component giving a broad category of meaning and a phonetic component suggesting the sound[1]. For example, the semantic component 亻 (*human*) of the Chinese character 他 (*he*) provides the meaning connected with human. In fact, the components of most Chinese characters inherently bring with certain levels of semantics *regardless of* the contexts. Being aware that the components of Chinese characters are finer grained semantic units, then an important question arises before slipping to the applications of word embeddings—would it be better to learn the semantic representations from the character components in Chinese?

We approach this question from both the practical and the cognitive points of view. In practice, we expect the representations to be optimized for good generalization. As analyzed before, the components are more generic unit *inside* Chinese characters that provides semantics. Such *inherent* information somehow alleviates the shortcoming of the *external* contexts. From the cognitive point of view, it has been found that the knowledge of semantic components significantly corre-

[1]http://en.wikipedia.org/wiki/Radical_(Chinese_characters)

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 829–834,
Lisbon, Portugal, 17-21 September 2015.

late to Chinese word reading and sentence comprehension (Ho et al, 2003).

These evidences inspire us to explore novel Chinese character embedding models. Different from word embeddings, character embeddings relate Chinese characters that occur in similar contexts with their component information. Chinese characters convey the meanings from their components, and beyond that, the meanings of most Chinese words also take roots in their composite characters. For example, the meaning of the Chinese word 摇篮 (*cradle*) can be interpreted in terms of its composite characters 摇 (*sway*) and 篮 (*basket*). Considering this, we further extend character embeddings from uni-gram models to bi-gram models.

At the core of our work is the exploration of Chinese semantic representations from a novel character-based perspective. Our proposed Chinese character embeddings incorporate the finer-grained semantics from the components of characters and in turn enrich the representations inherently in addition to utilizing the external contexts. The evaluations on both intrinsic word similarity and extrinsic text classification demonstrate the effectiveness and potentials of the new models.

2 Component-Enhanced Character Embeddings

Chinese characters are often composed of smaller and primitive **components** called radicals or radical-like components, which serve as the most basic units for building character meanings. Dating back to the 2nd century AD, the Han dynasty scholar Shen XU organizes his etymological dictionary *shuō wén jiě zì* (*word and expression*) by selecting 540 recurring graphic components that he called bù (means "categories"). Bù is nearly the same as what we call **radicals** today[2]. Most radicals are common semantic components. Over time, some original radicals evolve into radical-like components. Nowadays, a Chinese character often contains exactly one radical (rarely has two) and several other radical-like components. In what follows, we refer to as components both radicals and radical-like components.

Distinguished from English, these composite components are unique and inherent features inside Chinese characters. A lot of times, they allow us to assumingly understand or infer the meanings of characters without any context. In other words, the component-level features inherently bring with additional information that benefits semantic representations of characters. For example, we know that the characters 你 (*you*), 他 (*he*), 伙 (*companion*), 侣 (*companion*), and 们 (*people*) all have the meanings related to human because of their shared component 亻 (*human*), a variant of the Chinese character 人 (*human*). This kind of component information is intrinsically different from the contexts deriving by dependency relations and POS tags. It motivates us to investigate the component-enhanced Chinese character embedding models. While Sun et al (2014) utilizes radical information in a supervised fashion, we build our models in a holistic unsupervised and bottom-up way.

It is important to note the variation of a radical inside a character. There are two types of variations. The main type is position-related. For example, the radical of the Chinese character 水 (*water*) is itself, but it becomes 氵 as the radical of 池 (*pool*). The original radicals are stretched or squeezed so that they can fit into the general Chinese character shape of a square. The second variation type emerges along with the history of character simplification when traditional characters are converted into simplified characters. For instance, 食 (*eat*) is written as 飠 when it forms as a part of some traditional characters, but is written as 饣 in simplified characters. To cope with these variations and recover the semantics, we match all the radical variants back into their original forms. We extract all the components to build a **component list** for each Chinese character. With the assumption that a character's radical often bring more important semantics than the rest[3], we regard the radical of a character as the first component in its component list.

Let a sequence of characters $D = \{z_1, \ldots, z_N\}$ denotes a corpus of N characters over the character vocabulary V. And $z, c, e, K, T, M, |V|$ denote the Chinese character, the context character, the component list, the corresponding embedding dimension, the context window size, the number of components taken into account for each character, and the vocabulary size, respectively. We develop two component-enhanced character embedding models, namely *char*CBOW

[2] http://en.wikipedia.org/wiki/Radical_(Chinese_characters)

[3] Inside a character, its radical often serves as the semantic-component while its other radical-like components may be phonetics.

and *char*SkipGram.

*char*CBOW follows the original continuous bag-of-words model (CBOW) proposed by (Mikolov et al, 2013a). We predict the central character z_i conditioned on a $2(M+1)TK$-dimensional vector that is the concatenation of the remaining character-level contexts $(c_{i-T}, \ldots, c_{i-1}, c_{i+1}, \ldots, c_{i+T})$ and the components in their component lists. More formally, we wish to maximize the log likelihood of all the characters as follows,

$$L = \sum_{z_i^n \in D} \log p(z_i|h_i),$$

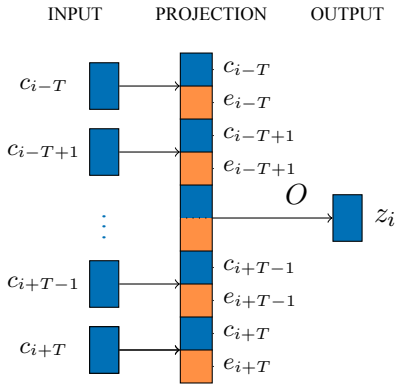
$$h_i = \text{cat}(c_{i-T}, e_{i-T}, \ldots, c_{i+T}, e_{i+T})$$

where h_i denotes the concatenation of the component-enhanced contexts. We make prediction using a $2KT(M+1)|V|$-dimensional matrix $\mathbf{O}$. Different from the original CBOW model, the extra parameter introduced in the matrix $\mathbf{O}$ allows us to maintain the relative order of the components and treat the radical differently from the rest components.

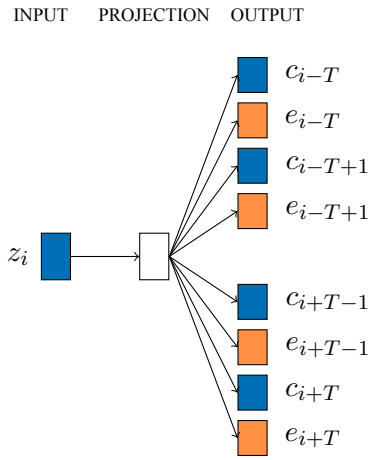
The development of *char*SkipGram is straightforward. We derive the component-enhanced contexts as $(\langle c_{i-T}, e_{i-T}\rangle, \ldots, \langle c_{i+T}, e_{i+T}\rangle)$ based on the central character z_i. The sum of log probabilities given z_i is maximized:

$$L = \sum_{z_i \in D} \sum_{\substack{j=-T \\ j\neq 0}}^{T} \Big(\log p(c_{j+i}|z_i) + \log p(e_{j+i}|z_i)\Big)$$

Figure 1 illustrates the two component-enhanced character embedding models. It is easy to extend *char*CBOW and *char*SkipGram to their corresponding bi-character extensions. Denote the z_i, c_i and e_i in *char*CBOW and *char*SkipGram as uni-character z_{ui}, c_{ui} and e_{ui}, the bi-character extensions are the models fed by bi-character formed z_{bi}, c_{bi} and e_{bi}.



(a) *char*CBOW



(b) *char*SkipGram

Figure 1: Illustrations of two component-enhanced character embedding models.

3 Evaluations

We examine the quality of the proposed two Chinese character embedding models as well as their corresponding extensions on both intrinsic word similarity evaluation and extrinsic text classification evaluation.

Word Similarity. As the widely used public word similarity datasets like WS-353 (Finkelstein et al, 2001), RG-65 (Rubsenstein and Goodenough, 1965) are built for English embeddings, we start from developing appropriate Chinese synonym sets. Two candidate choices are Chinese dictionaries HowNet (Dong and Dong, 2006) and HIT-CIR's Extended Tongyici Cilin (denoted as E-TC)[4]. As HowNet contains less modern words, such as 谷歌 (*Google*), we select E-TC as our benchmark for word similarity evaluation.

Text Classification. We use Tencent news titles as our text classification dataset[5]. A total of 8,826 titles of four categories (*society, entertainment, healthcare*, and *military*) are extracted. The lengths of titles range from 10 to 20 words. We train ℓ_2-regularized logistic regression classifiers using the `LIBLINEAR` package (Fan et al, 2008) with the learned embeddings.

To build the component-enhanced character embeddings, we employ the GB2312 character set

[4] `http://ir.hit.edu.cn/demo/ltp/Sharing_Plan.htm`

[5] `http://www.datatang.com/data/44341`

Table 1: Word Similarity Results of Embedding Models

Model	Spearman's rank correlation (%)											
	A	B	C	D	E	F	G	H	I	J	K	L
CBOW	33.2	25.2	32.2	27.8	36.5	37.6	43.2	40.2	37.3	39.5	44.2	40.4
SkipGram	**35.9**	**26.7**	33.8	**29.9**	36.6	40.2	45.3	44.3	39.0	41.2	46.9	43.0
*char*CBOW	34.0	23.2	**34.1**	26.7	**37.8**	**49.2**	**48.1**	**44.5**	**40.2**	**42.0**	**48.0**	**43.2**
*char*SkipGram	33.8	22.6	33.1	25.2	37.2	47.5	48.0	43.0	38.8	40.9	46.5	41.8
CBOW-bi	37.0	27.8	34.2	29.2	38.1	43.2	50.3	48.2	43.5	46.3	50.9	45.2
SkipGram-bi	**38.2**	**29.0**	34.0	29.4	38.9	44.9	50.2	49.3	**45.6**	48.4	51.3	47.4
*char*CBOW-bi	36.0	25.3	**36.8**	**31.2**	**40.2**	**54.3**	**55.7**	**49.7**	45.3	**48.9**	**53.2**	**47.7**
*char*SkipGram-bi	35.7	24.6	33.4	30.5	39.7	53.3	53.9	48.2	33.2	47.1	52.0	45.7

Table 2: Text Classification Results of Embedding Models

Model	Society			Entertainment			Healthcare			Military		
	P	R	F	P	R	F	P	R	F	P	R	F
CBOW-bi	43.0	28.0	33.9	48.2	32.7	39.0	47.6	29.5	36.4	57.6	40.8	47.8
SkipGram-bi	47.2	31.1	37.5	49.8	34.0	40.4	48.4	32.7	39.0	58.8	42.3	49.2
*char*CBOW-bi	**57.4**	**37.4**	**45.2**	**62.2**	**42.0**	**50.1**	**59.2**	**45.3**	**51.3**	**70.3**	**51.0**	**59.1**
*char*SkipGram-bi	50.3	34.6	41.0	57.6	34.5	43.2	57.3	42.5	48.8	67.8	48.3	56.4
CBOW-combine	46.2	29.0	35.6	50.3	35.0	41.3	51.0	33.6	40.5	62.2	45.7	52.7
SkipGram-combine	50.9	34.6	41.2	51.4	37.2	43.2	52.1	35.6	42.3	62.1	49.0	54.8
*char*CBOW-combine	**62.2**	**39.8**	**48.5**	**66.7**	**46.6**	**54.9**	**62.2**	**50.2**	**55.6**	**74.4**	**53.8**	**62.4**
*char*SkipGram-combine	54.4	38.2	44.9	59.2	36.5	45.2	62.0	47.9	54.0	73.4	53.5	61.9

and extract all their component lists. It is easy to obtain the first components (*i.e.*, the radicals), as they are readily available in the *online Xinhua Dictionary*[6]. For the rest radical-like components, we extract them by matching the patterns like "从 (*from*)+X" in the Xinhua dictionary. Such a pattern indicates that a character has a component of X. We also enrich the component lists by matching the pattern "X is only seen" in *Hong Kong Computer Chinese Basic Component Reference*[7] .

It is observed that nearly 65% Chinese characters have only one component (their radicals), and 95% Chinese characters have two components (including their radicals). Thus, we decide to maintain up to two extracted components to build the character embeddings according to the frequency of their occurrences. To cope with the radical variation problem, we transform 24 radical variants to their origins, such as 亻 to 人 (*human*), 扌 to 手 (*hand*), 氵 to 水 (*water*) and 辶 to 足 (*foot*). The complete list of the transformations is provided in Appendix for easy reference.

We adopt Chinese Wikipedia Dump[8] to train our models as well as the original CBOW and SkipGram, implemented in the `Word2Vec` tool[9] for comparison. The corpus in total contains 232,894 articles. In preprocessing, we remove pure digit words and non-Chinese characters, and ignore the words less than 10 occurrences during training. We set the context window size T as 2 and use 5 negative samples experimentally. All the embedding dimensions K are set to 50.

In the word similarity evaluation, we compute the Spearman's rank correlation (Myers and Well, 1995) between the similarity scores based on the learned embedding models and the E-TC similarity scores computed by following Tian and Zhao (2010). The bi-character embeddings are concatenation of the composite character embeddings. For the text classification evaluation, we average the composite single character embeddings for each bi-gram. And each bi-gram overlaps with the previous one. The titles are represented by averaging

[6] `http://xh.5156edu.com/`

[7] `http://www.ogcio.gov.hk/tc/business/tech_promotion/ccli/cliac/glyphs_guidelines.htm`

[8] `http://download.wikipedia.com/zhwiki/`

[9] `https://code.google.com/p/word2vec/`

the embeddings of their composite grams[10].

Table 1 presents the word similarity evaluation results of the eight embedding models mentioned above, where A–L denote the twelve categories in E-TC. The first four rows are the results with the uni-character inputs, and the last four rows correspond to the bi-character embeddings results.

We can see that both CBOW and CBOW-bi perform worse than the corresponding SkipGram and SkipGram-bi. This result is consistent with the finding in the previous work (Pennington et al, 2014; Levy and Goldberg, 2014; Levy et al, 2015). To some extent, CBOW and its extension CBOW-bi are the most different among the eight (the first four models in Table 1 and the first four models in Table 2). They tie together the characters in each context window by representing the context vector as the sum of their characters' vectors. Although they have a potential of deriving better representations (Levy et al, 2015), they lose some particular information from each unit of input in the average operations.

Although the performance on twelve different categories varies, in overall *char*CBOW, *char*SkipGram and their extensions consistently better correlate to E-TC. It provides the evidence that the component information in Chinese characters is of significance. Clearly, the bi-character models achieve higher rank correlations. These results are not surprised. As a matter of fact, a majority of Chinese words are compounds of two characters. Thus, in many cases two characters together is equivalent to a Chinese word. Considering the superiority of the bi-character models, we only apply them in the text classification evaluations.

The results shown in the first four rows of Table 2 are similar to those in the word similarity evaluation. Please notice the significant improvement of *char*CBOW and *char*CBOW-bi. We conjecture this as a hint of the importance of the order information, which is introduced by the extra parameter in the output matrixes. Their better performances verify our assumption that the radicals are more important than non-radicals. This is also attributed to the benefit from the order of the characters in the contexts.

Actually, we also conduct an additional experiment to combine the uni-gram and the bi-gram embeddings for text classification and notice in average about 8.4% of gain over the bi-gram embeddings alone. The detailed results are presented in the last four rows of Table 2.

[10]We do not compare the uni-formed characters with bi-formed compound characters. The word pairs that cannot be found in the vocabulary are removed.

4 Conclusions and Future Work

In this paper, we propose two component-enhanced Chinese character embedding models and their extensions to explore both the internal compositions and the external contexts of Chinese characters. Experimental results demonstrate their benefits in learning rich semantic representations. For the future work, we plan to devise embedding models based together on the composition of component-character and of character-word. The two types of compositions will serve in a coordinate fashion for the distributional representations.

Acknowledgements

The work described in this paper was supported by the grants from the Research Grants Council of Hong Kong (PolyU 5202/12E and PolyU 152094/14E), the grants from the National Natural Science Foundation of China (61272291 and 61273278) and a PolyU internal grant (4-BCB5).

Appendix

As mentioned in Section 3, we present the complete list of transformations of the variant and original forms of 24 radicals. The *meaning* columns provide the corresponding meanings of the components in the left.

transform	meaning	transform	meaning
艹 → 艸	grass	扌 → 手	hand
亻 → 人	human	氵 → 水	water
刂 → 刀	knife	車 → 车	vehicle
犭 → 犬	dog	攵 → 攴	hit
灬 → 火	fire	纟 → 糸	silk
钅 → 金	gold	耂 → 老	old
麥 → 麦	wheat	牜 → 牛	cattle
饣 → 食	eat	飠 → 食	eat
礻 → 示	memory	忄 → 心	heart
罒 → 网	nest	王 → 玉	jade
讠 → 言	speak	衤 → 衣	cloth
月 → 肉	body	辶 → 辵	walk

References

Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2014. Tailoring continuous word representations for dependency parsing. In *Proc. of ACL*.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, et al. 2003. A neural probabilistic language model. *The Journal of Machine Learning Research*, 3: 1137-1155.

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

Fei Cheng, Kevin Duh, Yuji Matsumoto. 2014. Parsing Chinese Synthetic Words with a Character-based Dependency Model. *LREC*.

Ronan Collobert, Jason Weston, L´eon Bottou, et al. 2011. Natural language processing (almost) from scratch. *JMLR*, 12.

Zhendong Dong and Qiang Dong. 2006. HowNet and the Computation of Meaning. *World Scientific Publishing Co. Pte. Ltd., Singapore*.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, et al. 2008. LIBLINEAR: A library for large linear classification. *The Journal of Machine Learning Research*, 9: 1871-1874.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, et al. 2001. Placing search in context: the concept revisited. In *Proc. of WWW*.

Connie Suk-Han Ho, Ting-Ting Ng, and Wing-Kin Ng. 2003. A "radical" approach to reading development in Chinese: The role of semantic radicals and phonetic radicals. In *Journal of Literacy Research*, 35(3), 849-878.

Eric H. Huang, Richard Socher, Christopher D. Manning, and Andrew Y. Ng. 2012. Improving Word Representations via Global Context and Multiple Word Prototypes. In *Proc. of ACL*.

Lingpeng Kong, Nathan Schneider, Swabha Swayamdipta, et al. 2014. A dependency parser for tweets. In *Proc. of EMNLP*, pages 1001–1012, Doha, Qatar, October.

Remi Lebret, Jo ´el Legrand, and Ronan Collobert. 2013. Is deep learning really necessary for word embeddings? In *Proc. of NIPS*.

Omer Levy and Yoav Goldberg. 2014. Dependency-based word embeddings. In *Proc. of ACL*.

Omer Levy, Yoav Goldberg, And Ido Dagan. 2015. Improving Distributional Similarity with Lessons Learned from Word Embeddings. In *Proc. of TACL*.

Shujie Liu, Nan Yang, Mu Li, and Ming Zhou. 2014. A recursive recurrent neural network for statistical machine translation. In *Proc. of ACL*, pages 1491–1500.

Minh-Thang Luong, Richard Socher, and Christopher D. Manning. 2013. Better Word Representations with Recursive Neural Networks for Morphology. In *Proc. of CoNLL*.

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

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013b. Distributed representations of words and phrases and their composition-ality. In *Advances in Neural Information Processing Systems*. pages 3111-3119.

Jerome L. Myers and Arnold D. Well. 1995. *Research Design & Statistical Analysis*. Routledge.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global Vectors for Word Representation. In *Proc. of EMNLP*.

Siyu Qiu, Qing Cui, Jiang Bian, and et al. 2014. Co-learning of Word Representations and Morpheme Representations. In *Proc. of COLING*.

Herbert Rubenstein and John B. Goodenough. 1965. Contextual correlates of synonymy. *Commun. ACM*, 8(10):627–633, October.

Yaming Sun, Lei Lin, Duyu Tang, et al. 2014. Radical-Enhanced Chinese Character Embedding. *CoRR* abs/1404.4714.

Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112.

Duyu Tang, Furu Wei, Nan Yang, et al. 2014. Learning sentiment-specific word embedding for twitter sentiment classification. In *Proc. of ACL*.

Jiu-le Tian and Wei Zhao. 2010. Words Similarity Algorithm Based on Tongyici Cilin in Semantic Web Adaptive Learning System.

Wang Ling, Chris Dyer, Alan Black, and Isabel Trancoso. 2015. Two/too simple adaptations of word2vec for syntax problems. In *Proc. of NAACL*, Denver, CO.

Jason Weston, Antoine Bordes, Oksana Yakhnenko, and Nicolas Usunier. 2013. Connecting language and knowledge bases with embedding models for relation extraction. In *Proc. of Computation and Language*.

Yi Yang and Jacob Eisenstein. 2015. Unsupervised multi-domain adaptation with feature embeddings. In *Proc. of NAACL-HIT*.

Mo Yu and Mark Dredze. 2014. Improving lexical embeddings with semantic knowledge. In *Proc. of ACL*.

Meishan Zhang, Yue Zhang, Wan Xiang Che, and et al. 2013. Chinese parsing exploiting characters. In *Proc. of ACL*.

Multi-label Text Categorization with Joint Learning Predictions-as-Features Method

Li Li[1] **Baobao Chang**[1] **Shi Zhao**[2] **Lei Sha**[1] **Xu Sun**[1] **Houfeng Wang**[1]
Key Laboratory of Computational Linguistics(Peking University), Ministry of Education, China[1]
Key Laboratory on Machine Perception(Peking University), Ministry of Education, China[2]
{li.l, chbb, shalei, z.s, xusun, wanghf}@pku.edu.cn

Abstract

Multi-label text categorization is a type of text categorization, where each document is assigned to one or more categories. Recently, a series of methods have been developed, which train a classifier for each label, organize the classifiers in a partially ordered structure and take predictions produced by the former classifiers as the latter classifiers' features. These predictions-as-features style methods model high order label dependencies and obtain high performance. Nevertheless, the predictions-as-features methods suffer a drawback. When training a classifier for one label, the predictions-as-features methods can model dependencies between former labels and the current label, but they can't model dependencies between the current label and the latter labels. To address this problem, we propose a novel joint learning algorithm that allows the feedbacks to be propagated from the classifiers for latter labels to the classifier for the current label. We conduct experiments using real-world textual data sets, and these experiments illustrate the predictions-as-features models trained by our algorithm outperform the original models.

1 Introduction

The multi-label text categorization is a type of text categorization, where each document is assigned to one or more categories simultaneously. The multi-label setting is common and useful in the real world. For example, in the news categorization task, a newspaper article concerning global warming can be classified into two categories simultaneously, namely environment and science. For another example, in the task of classifying music lyrics into emotions, a song's lyrics can deliver happiness and excitement simultaneously. The research about the multi-label text categorization attracts increasing attention (Srivastava and Zane-Ulman, 2005; Katakis et al., 2008; Rubin et al., 2012; Nam et al., 2013; Li et al., 2014).

Recently, a series of predictions-as-features style methods have been developed, which train a classifier for each label, organize the classifiers in a partially ordered structure and take predictions produced by the former classifiers as the latter classifiers' features. These predictions-as-features style methods model high order label dependencies (Zhang and Zhang, 2010) and obtain high performance. *Classifier chain* (CC) (Read et al., 2011) and *multi-label Learning by Exploiting lAbel Dependency* (Lead) (Zhang and Zhang, 2010) are two famous predictions-as-features methods. CC organizes classifiers along a chain and LEAD organizes classifiers in a Bayesian network. Besides, there are other works on extending the predictions-as-features methods (Zaragoza et al., 2011; Gonçalves et al., 2013; Sucar et al., 2014). In this paper, we focus on the predictions-as-features style methods.

The previous works of the predictions-as-features methods focus on learning the partially ordered structure. They neglect a drawback. When training a classifier for one label, predictions-as-features methods can model dependencies between former labels and the current label, but they can't model dependencies between the current label and the latter labels. Consider the case of three labels. We organize classifiers in a partially ordered structure shown in figure 1. When training the classifier for the second label, the feature (the bold lines in figure) consists of the origin feature and the prediction for the first label. The information about the third label can't be incorporated. It means that we only model the dependencies between the first label and the sec-

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 835–839,
Lisbon, Portugal, 17-21 September 2015.

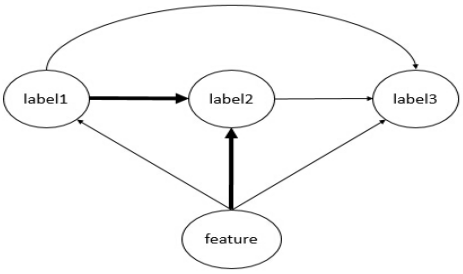


Figure 1: When training the classifier for the second label, the feature (the bold lines) consists of only the origin feature and the prediction for the first label. In this time, it is impossible to model the dependencies between the second label and the third label.

ond label and that the dependencies between the second label and the third label is missing.

To address this problem, we propose a novel joint learning algorithm that allows the feedbacks to be propagated from the classifiers for latter labels to the classifier for the current label, so that the information about the latter labels can be incorporated. It means that the proposed method can model, not only the dependencies between former labels and current label as the usual predictions-as-features methods, but also the dependencies between current label and latter labels. With not missing dependencies. Hence, the proposed method will improve the performance. Our experiments illustrate the models trained by our algorithm outperform the original models. You can find the code of this paper online [1].

The rest of this paper is organized as follows. Section 2 presents the proposed method. We conduct experiments to demonstrate the effectiveness of the proposed method in section 3. Section 4 concludes this paper.

2 Joint Learning Algorithm

2.1 Preliminaries

Let $\mathcal{X}$ denote the document feature space, and $\mathcal{Y} = \{0,1\}^m$ denote label space with m labels. A document instance $\boldsymbol{x} \in \mathcal{X}$ is associated with a label vector $\boldsymbol{y} = (y_1, y_2, ..., y_m)$, where $y_i = 1$ denotes the document has the i-th label and 0 otherwise. The goal of multi-label learning is to learn a function $\mathbf{h} : \mathcal{X} \rightarrow \mathcal{Y}$. In general, $\mathbf{h}$ consists of m functions, one for a label, i.e., $\mathbf{h}(\boldsymbol{x}) = [\mathbf{h}_1(\boldsymbol{x}), \mathbf{h}_2(\boldsymbol{x}), ..., \mathbf{h}_m(\boldsymbol{x})]$.

In the predictions-as-features methods, the classifiers are organized in a partially ordered structure and take predictions produced by the former classifiers as features. We can describe the classifier in the predictions-as-features method as follows.

$$\mathbf{h}_j : \boldsymbol{x}, h_{k\in \boldsymbol{pa_j}}(\boldsymbol{x}) \rightarrow y_j \tag{1}$$

where $\boldsymbol{pa_j}$ denotes the set of parents of the j-th classifiers in the partially ordered structure.

2.2 Architecture and Loss

In this subsection, we introduce architecture and loss function of our joint learning algorithm. As a motivating example, we employ logistic regression as the base classifier in the predictions-as-features methods. The classification function is the sigmoid function, as shown in Eq.(2).

$$\begin{aligned} p_j &= \mathbf{h}_j(\boldsymbol{x}, p_{k\in \boldsymbol{pa_j}}) \\ &= \frac{exp([\boldsymbol{x}, p_{k\in \boldsymbol{pa_j}}]^T \boldsymbol{W_j})}{1 + exp([\boldsymbol{x}, p_{k\in \boldsymbol{pa_j}}]^T \boldsymbol{W_j})} \end{aligned} \tag{2}$$

where p_j denotes the probability the document has the j-th label, $\boldsymbol{W_j}$ denotes the weight vector of the j-th model and $[\boldsymbol{x}, p_{k\in \boldsymbol{pa_j}}]$ denotes the feature vector x extended with predictions $[p_{k\in \boldsymbol{pa_j}}]$ produced by the former classifiers.

The joint algorithm learns classifiers in the partially ordered structure jointly by minimizing a global loss function. We use the sum of negative log likelihood losses of all classifiers as the global loss function.

$$\begin{aligned} \mathcal{L}(\boldsymbol{y}, \boldsymbol{h}(\boldsymbol{x})) &= \sum_{j=1}^{m} \ell(p_j, y_j) \\ &= -\sum_{j=1}^{m} (y_j log(p_j) + (1 - y_j) log(1 - p_j)) \end{aligned} \tag{3}$$

The joint algorithm minimizes this global loss function, as Eq.(4) shows.

$$\boldsymbol{h^*} = \underset{\boldsymbol{h}}{\operatorname{argmin}}\, \mathcal{L}(\boldsymbol{y}, \boldsymbol{h}(\boldsymbol{x})) \tag{4}$$

Minimizing this global loss function is *inequivalent* to minimizing the loss function of each base classifier separately, since minimizing the global

[1] https://github.com/rustle1314/Joint_Learning_Predictions_as_Features_for_Multi_Label_Classification

loss function results in feedbacks from latter classifiers. In the predictions-as-features methods, the weights of the k-th classifier are the factors of not only the k-th classifier but also the latter classifiers. Consequently, when minimizing the global loss function, the weights of the k-th classifier are updated according to not only the loss of the k-th classifier but also the losses of the latter classifiers. In other words, feedbacks are propagated from the latter classifiers to the k-th classifier.

The predictions-as-features models trained by our proposed joint learning algorithm can model the dependencies between former labels and current label, since they take predictions by the former classifiers to extend the latter classifiers' features, as the usual predictions-as-features methods do. Besides, they can also model the dependencies between current label and latter labels due to the feedbacks incorporated by the joint learning algorithm.

Here, we employ logistic regression as the motivating example. If we want to employ other classification models, we use other classification function and other loss function. For example, if we want to employ L2 SVM as base classifiers, we resort to the linear classification function and the L2 hinge loss function.

We employ the Back propagation Through Structure (BTS) (Goller and Kuchler, 1996) to minimize the global loss function. In BTS, parent node is computed with its child nodes at the forward pass stage; child node receives gradient as the sum of derivatives from all its parents.

3 Experiments

3.1 Datasets

We perform experiments on four real world data sets: 1) the first data set is Slashdot (Read et al., 2011). The Slashdot data set is concerned about predicting multiple labels given science and technology news titles and partial blurbs mined from Slashdot.org. 2) the second data set is Medical (Pestian et al., 2007). This data set involves the assignment of ICD-9-CM codes to radiology reports. 3) The third data set is Enron. The enron data set is a subset of the Enron Email Dataset, as labelled by the UC Berkeley Enron Email Analysis Project[2]. It is concerned about classifying emails into some categories. 4) the fourth data set is Tmc2007 (Srivastava and Zane-Ulman, 2005). It is concerned about safety report categorization, which is to label aviation safety reports with respect to what types of problems they describe.

dataset	n	d	m
slashdot	3782	1079	22
medical	978	1449	45
enron	1702	1001	53
tmc2007	28596	500	22

Table 2: Multi-label data sets and associated statistics.

Table 2 shows these multi-label data sets and associated statistics. n denotes the size of the entire data set, d denotes the number of the bag-of-words features, m denotes the number of labels. These data sets are available online [3].

3.2 Evaluation Metrics

We use three common used evaluation metrics. The Hamming loss is defined as the percentage of the wrong labels to the total number of labels.

$$Hammingloss = \frac{1}{m}|\mathbf{h}(\boldsymbol{x})\Delta\boldsymbol{y}| \tag{5}$$

where Δ denotes the symmetric difference of two sets, equivalent to XOR operator in Boolean logic.

The multi-label 0/1 loss is the exact match measure as it requires any predicted set of labels $\mathbf{h}(\boldsymbol{x})$ to match the true set of labels S *exactly*. The 0/1 loss is defined as follows:

$$0/1loss = I(\mathbf{h}(\boldsymbol{x}) \neq \boldsymbol{y}) \tag{6}$$

Let p_j and r_j denote the precision and recall for the j-th label. The macro-averaged F score is a harmonic mean between precision and recall, defined as follows:

$$Fscore = \frac{1}{m}\sum_{i=j}^{m}\frac{2 * p_j * r_j}{p_j + r_j} \tag{7}$$

3.3 Method Setup

In this paper, we focus on the predictions-as-features style methods, and use CC and LEAD as the baselines. Our methods are JCC and JLEAD. JCC(JLEAD) is CC(LEAD) trained by our joint algorithm and we compare JCC(JLEAD) to CC(LEAD) respectively. Put it another way, CC/LEAD provide the partial order structure of

[2] http://bailando.sims.berkeley.edu/enron_email.html

[3] http://mulan.sourceforge.net/datasets.html and http://mlkd.csd.auth.gr/multilabel.html

Dataset	BR	CC	LEAD	JCC	JLEAD
hamming loss (lower is better)					
slashdot	0.046 ± 0.002	0.043 ± 0.001	$0.045 \pm 0.001\circ$	0.043 ± 0.001	0.043 ± 0.001
medical	0.013 ± 0.001	$0.013 \pm 0.001\bullet$	$0.012 \pm 0.000\circ$	0.011 ± 0.000	0.010 ± 0.001
enron	0.052 ± 0.001	$0.053 \pm 0.002\bullet$	$0.052 \pm 0.001\circ$	0.049 ± 0.001	0.049 ± 0.001
tmc2007	0.063 ± 0.002	0.058 ± 0.001	0.058 ± 0.001	0.057 ± 0.001	0.057 ± 0.001
0/1 loss (lower is better)					
slashdot	0.645 ± 0.013	$0.637 \pm 0.015\bullet$	$0.631 \pm 0.017\circ$	0.610 ± 0.014	0.614 ± 0.011
medical	0.398 ± 0.034	$0.377 \pm 0.032\bullet$	$0.379 \pm 0.033\circ$	0.353 ± 0.030	0.345 ± 0.030
enron	0.856 ± 0.016	0.848 ± 0.017	0.853 ± 0.017	0.848 ± 0.018	0.850 ± 0.017
tmc2007	0.698 ± 0.004	0.686 ± 0.006	0.689 ± 0.009	0.684 ± 0.006	0.681 ± 0.006
F score (higher is better)					
slashdot	0.345 ± 0.016	$0.354 \pm 0.015\bullet$	$0.364 \pm 0.015\circ$	0.385 ± 0.017	0.383 ± 0.017
medical	0.403 ± 0.012	$0.416 \pm 0.013\bullet$	$0.426 \pm 0.011\circ$	0.444 ± 0.009	0.446 ± 0.013
enron	0.222 ± 0.014	0.224 ± 0.019	0.225 ± 0.018	0.223 ± 0.017	0.222 ± 0.015
tmc2007	0.524 ± 0.007	$0.531 \pm 0.009\bullet$	$0.508 \pm 0.017\circ$	0.547 ± 0.007	0.546 ± 0.006

Table 1: Performance (mean$\pm$std.) of each approach in terms of different evaluation metrics. $\bullet/\circ$ indicates whether JCC/JLEAD is statistically superior to CC/LEAD respectively (pairwise t-test at 5% significance level).

classifiers, and train these classifiers one by one. JCC/LEAD train classifiers jointly in the partial order structure provided by CC/LEAD.

For LEAD and JLEAD, we use the Banjo (Bayesian ANalysis with Java Objects) (Smith et al., 2006) package as the Bayesian structure learning tool. Besides, we also perform experiments with Binary Relevance (BR), which is the baseline for the predictions-as-features methods. BR trains a classifier for a label independently, which doesn't model dependencies. The base classifier of all of them is set to logistic regression without regularization. Experiments are performed in ten-fold cross validation with pairwise t-test at 5% significance level.

3.4 Performance

We reports the detailed results in terms of different evaluation metrics on different data sets in table 1. As shown in this figures, CC and LEAD outperform BR, which shows the values of the prediction-as-features methods. JCC and JLEAD wins over CC and LEAD respectively, which shows the values of the proposed joint learning algorithm.

The improvements are much smaller on the Enron data set than other data sets. In fact, BR, the original prediction-as-features methods and our proposed methods share similar performance on the Enron data set. The reason may be that the label dependencies in the Enron dataset is weak. The label dependencies weakness can be validated by the fact that the modeling-correlation CC and LEAD can't obtain much higher performance than the not-modeling-correlation BR. Due to the weak label dependencies, the modeling-correlation-better JCC(JLEAD) can't obtain much higher performance than CC(LEAD).

Criteria	JCC against CC	JLEAD against LEAD
hamming loss	**2/2/0**	**3/1/0**
0/1 loss	**2/2/0**	**2/2/0**
F-score	**3/1/0**	**3/1/0**
Total	**7/5/0**	**8/4/0**

Table 3: The win/tie/loss results for the joint learning algorithm against the original predictions-as-features methods in terms of different evaluation metrics (pairwise t-test at 5% significance level).

We summarize the detailed results into Table 3. JCC is significantly superior to CC in 7/12 cases, tie in 5/12 cases, inferior in zero case. JLEAD is significantly superior to LEAD in 8/12 cases, tie in 4/12 cases, inferior in zero case. The results indicates that our proposed joint algorithm can improve the performance of the predictions-as-features methods.

3.5 Time

The training time (mean) of each approach is showed detailed in table 4. First, we find the training time is related to the number of labels. The training time on the Tmc2007 dataset (28596 instances, 500 features and 22 labels) is less than that on the Enron dataset (1702 instances, 1001 features and 53 labels). This is very easy to understand. We train more classifiers with respect to more labels, which leads to more training time. Second, LEAD/JLEAD have slightly less training time than CC/JCC. The Bayesian network struc-

Dataset	CC	JCC	LEAD	JLEAD
slashdot	63.85	85.63	52.17	73.85
medical	134.11	142.51	115.33	128.78
enron	234.28	257.89	196.87	218.95
tmc2007	153.70	169.52	145.80	158.56

Table 4: The average training time (in seconds) of each approach

ture learning tool limits that a node has five parent nodes at most. Hence, the partially order structure of LEAD/JLEAD is much simpler. Third, the training time of the joint algorithm is slightly more than that of the original methods. Some time is spent on back-propagating feedbacks from latter classifiers.

4 Conclusion

The multi-label text categorization is a common and useful text categorization. Recently, a series of predictions-as-features style methods have been developed, which model high order label dependencies and obtain high performance. The predictions-as-features methods suffer from the drawback that they methods can't model dependencies between current label and the latter labels. To address this problem, we propose a novel joint learning algorithm that allows the feedbacks to be propagated from the latter classifiers to the current classifier. Our experiments illustrate the models trained by our algorithm outperform the original models.

5 Acknowledge

We sincerely thank all the anonymous reviewers for their valuable comments, which have helped to improve this paper greatly. Our work is supported by National High Technology Research and Development Program of China (863 Program) (No. 2015AA015402), National Natural Science Foundation of China(No.61370117 & No.61433015)and Major National Social Science Fund of China(No.12 & ZD227).

References

Christoph Goller and Andreas Kuchler. 1996. Learning task-dependent distributed representations by backpropagation through structure. In *Neural Networks, 1996., IEEE International Conference on*, volume 1, pages 347–352. IEEE.

Eduardo Corrêa Gonçalves, Alexandre Plastino, and Alex A Freitas. 2013. A genetic algorithm for optimizing the label ordering in multi-label classifier chains. In *Tools with Artificial Intelligence (ICTAI), 2013 IEEE 25th International Conference on*, pages 469–476. IEEE.

Ioannis Katakis, Grigorios Tsoumakas, and Ioannis Vlahavas. 2008. Multilabel text classification for automated tag suggestion. In *Proceedings of the ECML/PKDD*.

Li Li, Longkai Zhang, and Houfeng Wang. 2014. Muli-label text categorization with hidden components. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1816–1821, Doha, Qatar, October. Association for Computational Linguistics.

Jinseok Nam, Jungi Kim, Iryna Gurevych, and Johannes Fürnkranz. 2013. Large-scale multi-label text classification-revisiting neural networks. *arXiv preprint arXiv:1312.5419*.

John P Pestian, Christopher Brew, Paweł Matykiewicz, DJ Hovermale, Neil Johnson, K Bretonnel Cohen, and Włodzisław Duch. 2007. A shared task involving multi-label classification of clinical free text. In *Proceedings of the Workshop on BioNLP 2007: Biological, Translational, and Clinical Language Processing*, pages 97–104. Association for Computational Linguistics.

Jesse Read, Bernhard Pfahringer, Geoff Holmes, and Eibe Frank. 2011. Classifier chains for multi-label classification. *Machine learning*, 85(3):333–359.

Timothy N Rubin, America Chambers, Padhraic Smyth, and Mark Steyvers. 2012. Statistical topic models for multi-label document classification. *Machine Learning*, 88(1-2):157–208.

V Anne Smith, Jing Yu, Tom V Smulders, Alexander J Hartemink, and Erich D Jarvis. 2006. Computational inference of neural information flow networks. *PLoS computational biology*, 2(11):e161.

Ashok N Srivastava and Brett Zane-Ulman. 2005. Discovering recurring anomalies in text reports regarding complex space systems. In *Aerospace Conference, 2005 IEEE*, pages 3853–3862. IEEE.

L Enrique Sucar, Concha Bielza, Eduardo F Morales, Pablo Hernandez-Leal, Julio H Zaragoza, and Pedro Larrañaga. 2014. Multi-label classification with bayesian network-based chain classifiers. *Pattern Recognition Letters*, 41:14–22.

Julio H Zaragoza, Luis Enrique Sucar, Eduardo F Morales, Concha Bielza, and Pedro Larranaga. 2011. Bayesian chain classifiers for multidimensional classification. In *IJCAI*, volume 11, pages 2192–2197. Citeseer.

Min-Ling Zhang and Kun Zhang. 2010. Multi-label learning by exploiting label dependency. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 999–1008. ACM.

A Framework for Comparing Groups of Documents

Arun S. Maiya
Institute for Defense Analyses — Alexandria, VA, USA
amaiya@ida.org

Abstract

We present a general framework for comparing multiple groups of documents. A bipartite graph model is proposed where document groups are represented as one node set and the comparison criteria are represented as the other node set. Using this model, we present basic algorithms to extract insights into similarities and differences among the document groups. Finally, we demonstrate the versatility of our framework through an analysis of NSF funding programs for basic research.

1 Introduction and Motivation

Given multiple sets (or groups) of documents, it is often necessary to *compare* the groups to identify similarities and differences along different dimensions. In this work, we present a general framework to perform such comparisons for extraction of important insights. Indeed, many real-world tasks can be framed as a problem of comparing two or more *groups* of documents. Here, we provide two motivating examples.

1. Program Reviews. To better direct research efforts, funding organizations such as the National Science Foundation (NSF), the National Institutes of Health (NIH), and the Department of Defense (DoD), are often in the position of reviewing research programs via their artifacts (*e.g.*, grant abstracts, published papers, and other research descriptions). Such reviews might involve identifying overlaps across different programs, which may indicate a duplication of effort. It may also involve the identification of unique, emerging, or diminishing topics. A "document group" here could be defined either as a particular research program that funds many organizations, the totality of funded research conducted by a specific organization, or all research associated with a particular time period (*e.g.*, fiscal year). In all cases, the objective is to draw comparisons *between* groups by comparing the document sets associated with them.

2. Intelligence. In the areas of defense and intelligence, document sets are sometimes obtained from different sources or entities. For instance, the U.S. Armed Forces sometimes seize documents during raids of terrorist strongholds.[1] Similarities between two document sets (each captured from a different source) can potentially be used to infer a non-obvious association between the sources.

Of course, there are numerous additional examples across many domains (*e.g.*, comparing different news sources, comparing the reviews for several products, etc.). Given the abundance of real-world applications as illustrated above, it is surprising, then, that there are no existing general-purpose approaches for drawing such comparisons. While there is some previous work on the comparison of document sets (referred to as *comparative text mining*), these existing approaches lack the generality to be widely applicable across different use case scenarios with different comparison criteria. Moreover, much of the work in the area focuses largely on the summarization of shared or unshared topics among document groups (*e.g.*, Wan et al. (2011), Huang et al. (2011), Campr and Ježek (2013), Wang et al. (2012), Zhai et al. (2004)). That is, the problem of drawing *multi-faceted* comparisons among the groups themselves is not typically addressed. This, then, motivates our development of a *general-purpose* model for comparisons of document sets along arbitrary dimensions. We use this model for the identification of similarities, differences, trends, and anomalies among large *groups* of documents. We begin by

[1] http://en.wikipedia.org/wiki/Document_Exploitation_(DOCEX)

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 840–845,
Lisbon, Portugal, 17-21 September 2015.

formally describing our model.

2 Our Formal Model for Comparing Document Groups

As input, we are given several groups of documents, and our task is to compare them. We now formally define these document groups and the criteria used to compare them. Let $D = \{d_1, d_2, \ldots, d_N\}$ be a document collection comprising the totality of documents under consideration, where N is the size. Let D^P be a partition of D representing the document groups.

Definition 1 *A document group is a subset $D_i^P \in D^P$ (where index $i \in \{1 \ldots |D^P|\}$).*

Each document group in D^P, for instance, might represent articles associated with either a particular organization (*e.g.,* university), a research funding source (*e.g.,* NSF or DARPA program), or a time period (*e.g.,* a fiscal year). Document groups are compared using *comparison criteria*, D^C, a family of subsets of D.

Definition 2 *A comparison criterion is a subset $D_i^C \in D^C$ (where index $i \in \{1 \ldots |D^C|\}$).*

Intuitively, each subset of D^C represents a set of documents sharing some attribute. Our model allows great flexibility in how D^C is defined. For instance, D^C might be defined by the named entities mentioned within documents (*e.g.,* each subset contains documents that mention a particular person or organization of interest). For the present work, we define D^C by topics discovered using latent Dirichlet allocation or LDA (Blei et al., 2003).

LDA Topics as Comparison Criteria. Probabilistic topic modeling algorithms like LDA discover latent themes (*i.e.,* topics) in document collections. By using these discovered topics as the comparison criteria, we can compare arbitrary groups of documents by the themes and subject areas comprising them. Let K be the number of topics or themes in D. Each document in D is composed of a sequence of words: $d_i = \langle s_{i1}, s_{i2}, \ldots, s_{iN_i} \rangle$, where N_i is the number of words in d_i and $i \in \{1 \ldots N\}$. $V = \bigcup_{i=1}^{N} f(d_i)$ is the vocabulary of D, where $f(\cdot)$ takes a sequence of elements and returns a set. LDA takes K and D (including its components such as V) as input and produces two matrices as output, one of which is θ. The matrix $\theta \in \mathbb{R}^{N \times K}$ is the document-topic distribution matrix and shows the distribution of topics within each document. Each row of the matrix represents a probability distribution. D^C is constructed using K subsets of documents, each of which represent a set of documents pertaining largely to the same topic. That is, for $t \in \{1 \ldots K\}$ and $i \in \{1 \ldots N\}$, each subset $D_t^C \in D^C$ is comprised of all documents d_i where $t = \text{argmax}_x \theta_{ix}$.[2] Having defined the document groups D^P and the comparison criteria D^C, we now construct a bipartite graph model used to perform comparisons.

A Bipartite Graph Model. Our objective is to compare the *document groups* in D^P based on D^C. We do so by representing D^P and D^C as a weighted bipartite graph, $G = (P, C, E, w)$, where P and C are disjoint sets of nodes, E is the edge set, and $w : E \rightarrow \mathbb{Z}^+$ are the edge weights. Each subset of D^P is represented as a node in P, and each subset of D^C is represented as a node in C. Let $\alpha : P \rightarrow D^P$ and $\beta : C \rightarrow D^C$ be functions that map nodes to the document subsets that they represent. Then, the edge set E is $\{(u, v) \mid u \in P, v \in C, \alpha(u) \cap \beta(v) \neq \emptyset\}$, and the edge weight for any two nodes $u \in P$ and $v \in C$ is $w((u, v)) = |\alpha(u) \cap \beta(v)|$. Concisely, each weighted edge in G between a document group (in P) and a topic (in C) represents the number of documents shared among the two sets. Figure 1 shows a toy illustration of the model. Each node in P is shown in black and represents a subset of D^P (*i.e.,* a document group). Each node in C is shown in gray and represents a subset of D^C (*i.e.,* a document cluster pertaining primarily to the same topic). Each edge represents the intersection of the two subsets it connects. In the next section, we will describe basic algorithms on such bipartite graphs capable of yielding important insights into the similarities and differences among document groups.

3 Basic Algorithms Using the Model

We focus on three basic operations in this work.

Node Entropy. Let $\vec{w}$ be a vector of weights for all edges incident to some node $v \in E$. The *entropy* H of v is: $\text{H}(v) = -\sum_i p_i \log_{|\vec{w}|}(p_i)$, where $p_i = \frac{w_i}{\sum_j w_j}$ and $i, j \in \{1 \ldots |\vec{w}|\}$. A similar formulation was employed in Eagle et al. (2010). Intuitively, if $v \in P$, $H(v)$ measures the extent to which the document group is concentrated around

[2] D^C is also a partition of D, when defined in this way.

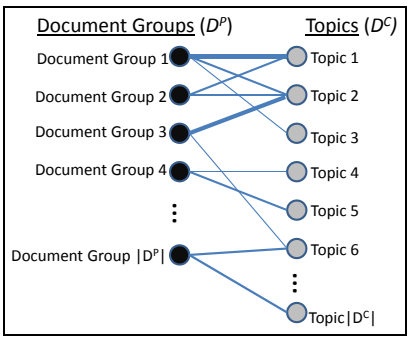


Figure 1: **[Toy Illustration of Bipartite Graph Model.]** Each black node (*i.e.*, node $\in P$) represents a document group. Each gray node (*i.e.*, node $\in C$) represents a cluster of documents pertaining primarily to the same topic.

a small number of topics (lower values of $H(v)$ mean more concentrated). Similarly, if $v \in C$, it is the extent to which a topic is concentrated around a small number of document groups.

Node Similarity. Given a graph G, there are many ways to measure the similarity of two nodes based on their connections. Such measures can be used to infer similarity (and dissimilarity) among document groups. However, existing methods are not well-suited for the task of document group comparison. The well-known SimRank algorithm (Jeh and Widom, 2002) ignores edge weights, and neither SimRank nor its extension, SimRank++ (Antonellis et al., 2008), scale to larger graphs. SimRank++ and ASCOS (Chen and Giles, 2013) do incorporate edge weights but in ways that are not appropriate for document group comparisons. For instance, both SimRank++ and ASCOS incorporate magnitude in the similarity computation. Consider the case where document groups are defined as research labs. ASCOS and SimRank++ will measure large research labs and small research labs as less similar when in fact they may publish nearly identical lines of research. Finally, under these existing methods, document groups sharing zero topics in common could still be considered similar, which is undesirable here. For these reasons, we formulate similarity as follows. Let $N^G(\cdot)$ be a function that returns the neighbors of a given node in G. Given two nodes $u, v \in P$, let $L^{u,v} = N^G(u) \cup N^G(v)$ and let $x : I \rightarrow L^{u,v}$ be the indexing function for $L^{u,v}$.[3] We construct two vectors, $\vec{a}$ and $\vec{b}$, where $a_k = w(u, x(k))$, $b_k = w(v, x(k))$, and $k \in I$. Each vector is essentially a sequence of weights for edges between $u, v \in P$ and each node in $L^{u,v}$. Similarity of two nodes is measured using the cosine similarity of their corresponding sequences, $\frac{\vec{a}\cdot\vec{b}}{\|\vec{a}\|\|\vec{b}\|}$, which we compute using a function $sim(\cdot,\cdot)$. Thus, document groups are considered more similar when they have similar sets of topics in similar proportions. As we will show later, this simple solution, based on item-based collaborative filtering (Sarwar et al., 2001), is surprisingly effective at inferring similarity among document groups in G.

Node Clusters. Identifying clusters of related nodes in the bipartite graph G can show how document groups form larger classes. However, we find that G is typically fairly dense. For these reasons, partitioning of the one-mode projection of G and other standard bipartite graph clustering techniques (*e.g.*, Dhillion (2001) and Sun et al. (2009)) are rendered less effective. We instead employ a different tack and exploit the node similarities computed earlier. We transform G into a new weighted graph $G^P = (P, E^P, w^{sim})$ where $E^P = \{(u, v) \mid u, v \in P, sim(u, v) > \xi\}$, ξ is a pre-defined threshold, and w^{sim} is the edge weight function (*i.e.*, $w^{sim} = sim$). Thus, G^P is the similarity graph of document groups. $\xi = 0.5$ was used as the threshold for our analyses. To find clusters in G^P, we employ the Louvain algorithm, a heuristic method based on modularity optimization (Blondel et al., 2008). Modularity measures the fraction of edges falling within clusters as compared to the expected fraction if edges were distributed evenly in the graph (Newman, 2006). The algorithm initially assigns each node to its own cluster. At each iteration, in a local and greedy fashion, nodes are re-assigned to clusters with which they achieve the highest modularity.

4 Example Analysis: NSF Grants

As a realistic and informative case study, we utilize our model to characterize funding programs of the National Science Foundation (NSF). This corpus consists of 132,372 grant abstracts describing awards for basic research and other support funded by the NSF between the years 1990 and 2002 (Bache and Lichman, 2013).[4] Each award is associated with both a program element (*i.e.*, funding source) and a date. We define *document*

[3] I is the index set of $L^{u,v}$.

[4] Data for years 1989 and 2003 in this publicly available corpus were partially missing and omitted in some analyses.

groups in two ways: by program element and by calendar year. For comparison criteria, we used topics discovered with the MALLET implementation of LDA (McCallum, 2002) using $K = 400$ as the number of topics and 200 as the number of iterations. All other parameters were left as defaults. The NSF corpus possesses unique properties that lend themselves to experimental evaluation. For instance, program elements are not only associated with specific sets of research topics but are named based on the content of the program. This provides a measure of ground truth against which we can validate our model. We structure our analyses around specific questions, which now follow.

Which NSF programs are focused on specific areas and which are not? When defining *document groups* as program elements (*i.e.,* each NSF program is a node in P), node entropy can be used to answer this question. Table 1 shows examples of program elements most and least associated with specific topics, as measured by entropy. For example, the program *1311 Linguistics* (low entropy) is largely focused on a single *linguistics* topic (labeled by LDA with words such as "language," "languages," and "linguistic"). By contrast, the *Australia* program (high entropy) was designed to support US-Australia cooperative research across many fields, as correctly inferred by our model.

Low Entropy Program Elements	
Program	**Primary LDA Topic**
1311 Linguistics	language languages linguistic
4091 Network Infrastructure	network connection internet
High Entropy Program Elements	
Program	**Primary LDA Topic**
5912 Australia	(many topics & disciplines)
9130 Research in Minority Instit.	(many topics & disciplines)

Table 1: **[Examples of High/Low Entropy Programs.]**

Which research areas are growing/emerging? When defining *document groups* as calendar years (instead of program elements), low entropy nodes in C are topics concentrated around certain years. Concentrations in later years indicate growth. The LDA-discovered topic *nanotechnology* is among the lowest entropy topics (*i.e.,* an outlier topic with respect to entropy). As shown in Figure 2, the number of *nanotechnology* grants drastically increased in proportion through 2002. This result is consistent with history, as the National Nanotechnology Initiative was proposed in the late 1990s to promote nanotechnology R&D.[5] One could also measure such trends using budget allocations by incorporating the award amounts into the edge weights of G.

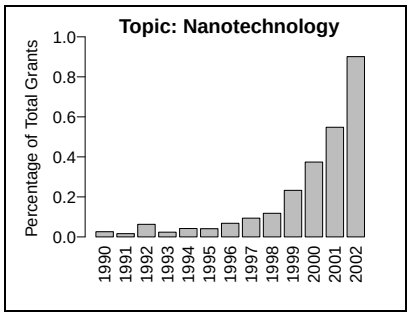


Figure 2: **[Uptrend in Nanotechnology.]** Our model correctly identifies the surge in nanotechnology R&D beginning in the late 1990s.

Given an NSF program, to which other programs is it most similar? As described in Section 3, when each node in P represents an NSF program, our model can easily identify the programs most similar to a given program. For instance, Table 2 shows the top three most similar programs to both the *Theoretical Physics* and *Ecology* programs. Results agree with intuition. For each NSF program, we identified the top n most similar programs ranked by our $sim(\cdot,\cdot)$ function, where $n \in \{3, 6, 9\}$. These programs were manually judged for relatedness, and the Mean Average Precision (MAP), a standard performance metric for ranking tasks in information retrieval, was computed. We were unsuccessful in evaluating alternative weighted similarity measures mentioned in Section 3 due to their aforementioned issues with scalability and the size of the NSF dataset. (For instance, the implementations of ASCOS (Antonellis et al., 2008) and SimRank (Jeh and Widom, 2002) that we considered are available here.[6]) Recall that our $sim(\cdot,\cdot)$ function is based on measuring the cosine similarity between two weight vectors, $\vec{a}$ and $\vec{b}$, generated from our bipartite graph model. As a baseline for comparison, we evaluated two additional similarity implementations using these weight vectors. The first measures the similarity between weight vectors using weighted Jaccard similarity, which is $\frac{\sum_k \min(a_k,b_k)}{\sum_k \max(a_k,b_k)}$ (denoted as *Wtd. Jaccard*). The second measure is implemented by taking the Spearman's rank correlation coefficient of $\vec{a}$ and $\vec{b}$ (de-

[5] http://en.wikipedia.org/wiki/National_Nanotechnology_Initiative

[6] https://github.com/hhchen1105/networkx_addon

noted as *Rank*). Figure 3 shows the Mean Average Precision (MAP) for each method and each value of n. With the exception of the difference between *Cosine* and *Wtd. Jaccard* for MAP@3, all other performance differentials were statistically significant, based on a one-way ANOVA and post-hoc Tukey HSD at a 5% significance level. This, then, provides some validation for our choice.

1245 Theoretical Physics	**1182 Ecology**
1286 Elementary Particle Theory	1128 Ecological Studies
1287 Mathematical Physics	1196 Environmental Biology
1284 Atomic Theory	1195 Ecological Research

Table 2: **[Similarity Queries.]** Three most similar programs to the *Theoretical Physics* and *Ecology* programs.

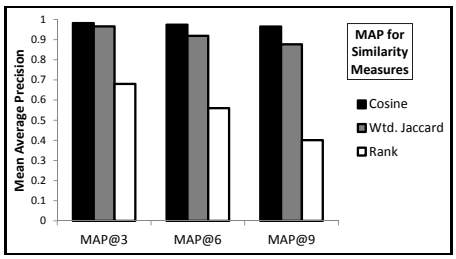


Figure 3: **[Mean Average Precision (MAP).]** Cosine similarity outperforms alternative approaches.

How do NSF programs join together to form larger program categories? As mentioned, by using the similarity graph G^P constructed from G, clusters of related NSF programs can be discovered. Figure 4, for instance, shows a discovered cluster of NSF programs all related to the field of neuroscience. Each NSF program (*i.e.*, node) is composed of many documents.

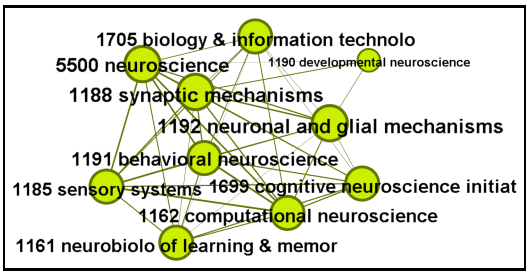


Figure 4: **[Neuroscience Programs.]** A discovered cluster of program elements all related to *neuroscience*.

Which pairs of grants are the most similar in the research they describe? Although the focus of this paper is on drawing comparisons among *groups* of documents, it is often necessary to draw comparisons among *individual* documents, as well. For instance, one may wish to identify pairs of grants from *different* programs describing highly similar lines of research. One common approach to this is to measure the similarity among low-dimensional representations of documents returned by LDA (Blei et al., 2003). We employ the Hellinger distance metric for this. Unfortunately, identifying the set of *most* similar document pairs in this way can be computationally expensive, as the number of pairwise comparisons scales quadratically with the size of the corpus. To address this, our bipartite graph model can be exploited as a *blocking* heuristic using either the document groups or the comparison criteria. In the latter case, one can limit the pairwise comparisons to only those documents that reside in the same subset of D^C. For the former case, *node similarity* can be used. Instead of comparing each document with every other document, we can limit the comparisons to only those document groups of interest that are deemed similar by our model. As an illustrative example, the program *1271 Computational Mathematics* and the program *2865 Numeric, Symbolic, and Geometric Computation* are inferred as being highly similar. Between these groups, the following two grants are easily identified as being the most similar with a Hellinger similarity score of 0.73 (only titles are shown due to space constraints):

- **Grant #1: Analyses of Structured Computational Problems and Parallel Iterative Algorithms**
  (Discusses parallel iterative methods for solutions to large sparse/dense systems of linear equations.)
- **Grant #2: Sparse Matrix Algorithms on Distributed Memory Multiprocessors**

As can be seen, despite some differences in terminology, the two lines of research are related, as matrices (studied in Grant #2) are used to compactly represent and work with systems of linear equations (studied in Grant #1).

5 Conclusion

We have presented a bipartite graph model for drawing comparisons among large *groups* of documents. We showed how basic algorithms using the model can identify trends and anomalies among the document groups. As an example analysis, we demonstrated how our model can be used to better characterize and evaluate NSF research programs. For future work, we plan on employing alternative comparison criteria in our model such as those derived from named entity recognition and paraphrase detection.

References

Ioannis Antonellis, Hector G. Molina, and Chi C. Chang. 2008. Simrank++: Query Rewriting Through Link Analysis of the Click Graph. *Proc. VLDB Endow.*, 1(1):408–421, August.

K. Bache and M. Lichman. 2013. UCI machine learning repository.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet Allocation. *J. Mach. Learn. Res.*, 3(4-5):993–1022, March.

Vincent D. Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. 2008. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008+, July.

Michal Campr and Karel Ježek. 2013. Topic Models for Comparative Summarization. In Ivan Habernal and Václav Matoušek, editors, *Text, Speech, and Dialogue*, volume 8082 of *Lecture Notes in Computer Science*, pages 568–574. Springer Berlin Heidelberg.

Hung H. Chen and C. Lee Giles. 2013. ASCOS: An Asymmetric Network Structure COntext Similarity Measure. In *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, ASONAM '13, pages 442–449, New York, NY, USA. ACM.

Inderjit S. Dhillion. 2001. Co-clustering Documents and Words Using Bipartite Spectral GraphPartitioning. Technical report, Austin, TX, USA.

Nathan Eagle, Michael Macy, and Rob Claxton. 2010. Network diversity and economic development. *Science*, 328(5981):1029–1031, May.

Xiaojiang Huang, Xiaojun Wan, and Jianguo Xiao. 2011. Comparative News Summarization Using Linear Programming. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers - Volume 2*, HLT '11, pages 648–653, Stroudsburg, PA, USA. Association for Computational Linguistics.

Glen Jeh and Jennifer Widom. 2002. SimRank: a measure of structural-context similarity. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '02, pages 538–543, New York, NY, USA. ACM.

Andrew K. McCallum. 2002. MALLET: A Machine Learning for Language Toolkit.

M. E. J. Newman. 2006. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, 103(23):8577–8582, June.

Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based Collaborative Filtering Recommendation Algorithms. In *Proceedings of the 10th International Conference on World Wide Web*, WWW '01, pages 285–295, New York, NY, USA. ACM.

Yizhou Sun, Yintao Yu, and Jiawei Han. 2009. Ranking-based Clustering of Heterogeneous Information Networks with Star Network Schema. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, pages 797–806, New York, NY, USA. ACM.

Xiaojun Wan, Houping Jia, Shanshan Huang, and Jianguo Xiao. 2011. Summarizing the Differences in Multilingual News. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '11, pages 735–744, New York, NY, USA. ACM.

Dingding Wang, Shenghuo Zhu, Tao Li, and Yihong Gong. 2012. Comparative Document Summarization via Discriminative Sentence Selection. *ACM Trans. Knowl. Discov. Data*, 6(3), October.

ChengXiang Zhai, Atulya Velivelli, and Bei Yu. 2004. A Cross-collection Mixture Model for Comparative Text Mining. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, pages 743–748, New York, NY, USA. ACM.

$C3EL$: A Joint Model for Cross-Document Co-Reference Resolution and Entity Linking

Sourav Dutta
Max-Planck Institute for Informatics
Saarbrücken, Germany
sdutta@mpi-inf.mpg.de

Gerhard Weikum
Max-Planck Institute for Informatics
Saarbrücken, Germany
weikum@mpi-inf.mpg.de

Abstract

Cross-document co-reference resolution (CCR) computes equivalence classes over textual mentions denoting the same entity in a document corpus. Named-entity linking (NEL) disambiguates mentions onto entities present in a knowledge base (KB) or maps them to $null$ if not present in the KB. Traditionally, CCR and NEL have been addressed separately. However, such approaches miss out on the mutual synergies if CCR and NEL were performed jointly.

This paper proposes $C3EL$, an unsupervised framework combining CCR and NEL for jointly tackling both problems. $C3EL$ incorporates results from the CCR stage into NEL, and vice versa: additional global context obtained from CCR improves the feature space and performance of NEL, while NEL in turn provides distant KB features for already disambiguated mentions to improve CCR. The CCR and NEL steps are interleaved in an iterative algorithm that focuses on the highest-confidence still unresolved mentions in each iteration. Experimental results on two different corpora, news-centric and web-centric, demonstrate significant gains over state-of-the-art baselines for both CCR and NEL.

1 Introduction

With the advent of large knowledge bases (KB) like DBpedia, YAGO, Freebase, and others, entities (people, places, organizations, etc.) along with their attributes and relationships form the basis of smart applications like search, analytics, recommendations, question answering, and more. The major task that arises in both the KB construction process and the entity-centric applications involves precise *recognition*, *resolution*, and *linking* of named entities distributed across web pages, news articles, and social media.

Named Entity Recognition (NER) deals with the identification of entity *mentions* in a text and their classification into coarse-grained semantic types (person, location, etc.) (Finkel et al., 2005; Nadeau & Sekine, 2007; Ratinov & Roth, 2009). This involves segmentation of token sequences to obtain mention boundaries, and mapping relevant token spans to pre-defined entity categories. For example, NER on the text `Einstein won the Nobel Prize` identifies the mentions "Einstein" and "Nobel Prize" and marks them as *person* and *misc* type, respectively.

Named Entity Linking (NEL)[1] involves the *disambiguation* of textual mentions, based on context and semantic information, and their mapping to proper *entities in a KB* (Bunescu & Paşca, 2006; Cucerzan, 2007; Milne & Witten, 2008; Hoffart et al., 2011; Ratinov et al., 2011; Cornolti et al., 2013). For example, in the above text, the mention "Einstein" is linked to the physicist *Albert Einstein*.

Entity *Co-reference Resolution* (CR) (Haghighi & Klein, 2010; Ng, 2010; Lee et al., 2013) is essentially a clustering task to identify mentions (and anaphoras) within a document referring to the same entity, thus computing equivalence classes or *mention groups*. For example, mentions `Albert Einstein` and `Nobel laureate Einstein` both refer to the same entity *German physicist Albert Einstein*, but are different from the mention `Hans Albert Einstein`.

When CR is extended to an entire text corpus, in order to generate equivalence classes of co-referring mentions across documents, the task is known as *Cross-document Co-reference Resolution* (CCR) (Bagga & Baldwin, 1998; Culotta et al., 2007; Singh et al., 2011; Dutta & Weikum, 2015). Note that CCR is not the same as merely concatenating all documents in the corpus and utilizing existing CR methods. The linguistic diversity across documents and high computational cost for huge numbers of mentions in the corpus would typically make such a CR-based simulation perform poorly. Neither CR nor CCR links mention groups to corresponding KB entities. Thus, they represent both in-KB entities and out-of-KB entities (e.g., long-tail or emerging entities that do not have a Wikipedia article) in the same way.

[1] *Named Entity Disambiguation (NED)* and "Wikification" are often used to denote the same task. The latter may be more broadly used, though, to include the disambiguation of common nouns and phrases onto concepts, whereas NED restricts itself to noun phrases that denote individual entities.

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 846–856,
Lisbon, Portugal, 17-21 September 2015.

State-of-the-Art and its Limitations: Established CR methods rely on rule-based methods or supervised learning techniques on syntactic paths between mentions, semantic compatibility, and other linguistic features (Haghighi & Klein, 2009), with additional use of distant features from KBs (Lee et al., 2013). Modern cluster-ranking (Rahman & Ng, 2011) and multi-sieve methods (Ratinov & Roth, 2012) involve incremental expansion of mention groups by considering semantic types and Wikipedia categories. CCR methods utilize transitivity-aware clustering techniques (Singh et al., 2011), by considering mention-mention similarities (Bagga & Baldwin, 1998) along with features extracted from external KBs (Dutta & Weikum, 2015).

NEL methods often harness the semantic similarity between mentions and entities and also among candidate entities for different mentions (in Wikipedia or other KBs) for contextualization and coherence disambiguation (Hoffart et al., 2011; Milne & Witten, 2008; Kulkarni et al., 2009; Ratinov et al., 2011). However, in the absence of CR mention groups, NEL has limited context and is bound to miss out on certain kinds of difficult cases.

Although NER, CR, CCR and NEL involve closely related tasks and their tighter integration has been shown to be promising (Chen & Roth, 2013; Zheng et al., 2013), they have mostly been explored in isolation. Recently, several *joint models* have been proposed for `CR-NER` (Haghighi & Klein, 2010; Singh et al., 2013), `CR-NEL` (Hajishirzi et al., 2013), and `NER-CR-NEL` (Durrett & Klein, 2014). However, to the best of our knowledge, no method exists for jointly handling CCR and NEL on large text corpora.

1.1 Approach and Contributions

This paper proposes the novel $C3EL$ *(Cross-doCument Co-reference resolution and Entity Linking)* framework for jointly modeling cross-document co-reference resolution (CCR) and linkage of mention groups to entities in a knowledge base (NEL).

Example: To illustrate the potential synergies between CCR and NEL, consider the 3 documents in Figure 1 containing 9 mentions (on the left) with candidate entities from a KB (on the right). CCR alone would likely miss the co-reference relation between `Logan` (Doc 1) and its alias `Wolverine` (Doc 2), leaving NEL with the difficult task of disambiguating "Logan" in a document with sparse and highly ambiguous context (Doc 1). On the other hand, NEL alone would likely map `Australia` (Doc 3) to the country (not the movie) and could easily choose the wrong link for mention "Hugh". Moreover, the presence of `Ava Eliot` as an out-of-KB mention complicates the task.

However, if we could more freely interleave

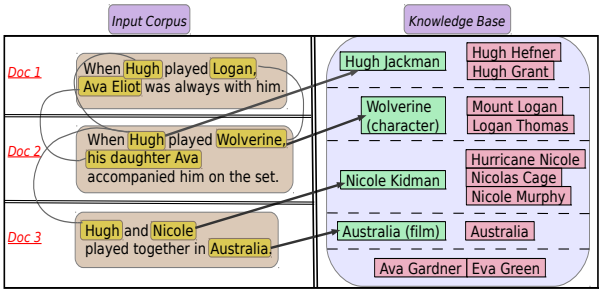


Figure 1: Joint CCR-NEL Example *(Green KB entries connected via arrows denote the correct entity linkage for the mention co-reference groups; while the red ones represent alternative incorrect candidates with similar surface forms)*

CCR and NEL and could iterate them several times, we would be in a much stronger position. An initial NEL step for the easiest mention, namely "Wolverine", maps it to the character of X-Men movies. This indicates that the three "Hugh" mentions could all be the same actor, and are thus easily merged into a co-reference group using CCR. We now have enough cues for NEL to choose the right entity for the "Hugh" mention group, which in turn enables the proper mapping of "Australia" to the movie. Finally, it becomes clear that mentions "Ava Eliot" and "his daughter Ava" should be merged into the same group and represented as an out-of-KB entity mapped to $null$.

The above example clearly demonstrates that interleaving CCR and NEL is highly beneficial. However, appropriate choices for the ordering of CCR and NEL steps are usually not obvious at all. The proposed $C3EL$ algorithm solves this problem: automatically determining an efficient interleaving of CCR and NEL.

Approach: $C3EL$ iteratively aggregates intermediate information obtained from alternating steps of CCR and NEL, thus forming a *feedback loop* for propagating mention features and entity knowledge. Intuitively, co-referring mentions obtained via CCR generate global context for improved NEL performance, while mentions linked to KB entities (by NEL) provide distant semantic features with additional cues for CCR. $C3EL$ couples several building blocks like unsupervised hierarchical clustering, *context summaries* for mentions and distant KB features for entities, drawing inspiration from the CCR-only method of (Dutta & Weikum, 2015). Mention linking to the KB (NEL) is performed using distant knowledge and co-occurring mentions.

In a nutshell, the major contributions of this paper are:

- the $C3EL$ framework for joint computation of cross-document co-reference resolution (CCR) and entity linking to a KB (NEL), based on propagating information across iterative CCR and NEL steps;
- techniques for considering co-occurring mentions in *context summaries* and for harnessing

context-based keywords for *link validation* in NEL, improving accuracy on out-of-KB entities;

- an experimental evaluation with two different corpora, one based on news articles and one based on web pages, demonstrating substantial gains for both CCR and NEL over state-of-the-art methods.

2 $C3EL$: Joint CCR-NEL Framework

Given an input corpus C of n documents, $C = \{D_1, D_2, \cdots, D_n\}$ with entity mentions $EM = \{m_{11}, m_{12}, \cdots, m_{21}, m_{22}, \cdots\}$ ($m_{ij} \in D_i$), $C3EL$ aims to *jointly compute*:

- CCR: an equivalence relation over EM with equivalence classes E_i, such that $E_i \cap_{i \neq j} E_j = \emptyset$ and $\cup_i E_i = EM$, and
- NEL: linking each of the classes E_i to entities present in a KB or map it to $null$ if there is no proper entity in the KB.

To this end, $C3EL$ consists of 3 algorithmic stages: **(i)** Pre-Processing, **(ii)** Interleaved NEL and CCR, and **(iii)** Finalization.

2.1 Pre-Processing Stage

HTML pages in the input corpus C are transformed into plain text using standard tools like `jsoup.org`. Recognition and markup of mentions are performed using the Stanford CoreNLP toolkit (`nlp.stanford.edu`), and a coarse-grained lexical type for each mention (e.g., person, location, organization, etc.) is obtained from the Stanford NER Tagger (Finkel et al., 2005). The multi-pass sieve algorithm for single-document CR (Raghunathan et al., 2010; Lee et al., 2011; Lee et al., 2013) then computes mention co-reference chains per document, and a *head mention* is chosen for each of the mention groups (chains). The head mention is typically represented by the most explicit denotation of the entity (e.g., person's full name with title, location name with country, etc.).

For each of the mention groups M_i, C3EL then constructs a *context summary* using:

- **Sentences** – all sentences in the document that contain mentions of group M_i; and
- **Co-occurrence** – all sentences for other mention groups that contain mentions co-occurring in any of the sentences of M_i (as obtained above).

Formally, for each mention group M_i, let $S(M_i) = \{sentence(m_j) \mid m_j \in M_i\}$ represent the set of extracted sentences, where $sentence(m_j)$ denotes the sentences in which mention m_j occurs. Also, let the co-occurring mention set of M_i be $Co(M_i) = \{m' \mid m' \in S(M_i) \wedge m' \notin M_i\}$. The context summary of M_i is defined as:

$$\mathbb{CS}(M_i) = S(M_i) \cup \left(\bigcup_{m' \in Co(M_i)} S(m') \right)$$

The context summaries intentionally do not include any distant KB features for mentions. The intuition is to minimize potential noise from overly speculative mappings to the KB at this initial stage.

2.2 Interleaved NEL & CCR Approach

After the preliminary CR step on each document and the construction of context summaries, $C3EL$ now performs an initial NEL step for each of the mention groups M_i, using the extracted sentences $S(M_i)$ as inputs to NEL. It obtains the best matching entity, the confidence of the match, and its corresponding Wikipedia page. Off-the-shelf NEL software (like WikipediaMiner or Illinois-Wikifier) is used for mention-entity mapping based on prior popularity of the named-entities (from the KB) and textual similarity between $S(M_i)$ (context of the mention group) and the entity descriptions in KB.

For each M_i, the entity link obtained (from NEL) is then "validated" using a similarity measure between features from the *context summary*, $\mathbb{CS}(M_i)$ (including co-occurring mentions) and distant KB labels – forming the *link validation* procedure of $C3EL$. This explicit use of co-occurring mentions' ($Co(M_i)$) contexts helps to better identify out-of-KB entities compared to direct full-fledged NEL using the entire input text (shown in Section 3). Also the use of NEL on $S(M_i)$ alone, makes $C3EL$ "light-weighted".

The mappings between the mention groups and KB entries are then classified, on the basis of the NEL confidence scores, into *Strong Evidence* (SE), *Weak Evidence* (WE), and *No Evidence* (NE) classes. For mention groups placed in SE, the KB features (obtained previously) are appended to their context summaries, while mentions strongly linked to same KB entities are considered to be co-referring and hence grouped together (performing implicit CCR).

Considering our example (Figure 1), we now outline the iterative steps of $C3EL$ interleaving NEL & CCR.

1. During **Iteration 1**, $C3EL$ performs:

- **NEL:** The initial NEL step maps the unambiguous mentions, `Wolverine` to the X-Men movie character and `Australia` to the country, with high confidence. However, link validation fails for "Australia" as there is very low similarity between the mention context features (e.g., Hugh, Wolverine, etc.) and the distant KB labels extracted from its Wikipedia page (e.g., Commonwealth, population, etc.); thus the link is dropped and the mention is added to NE. So only the mention "Wolverine" is added to the

SE class and enriched with KB features (e.g., alias Logan).

On the other hand, the 3 "Hugh" mentions exhibit low NEL confidence due to the high ambiguity of this first name and are therefore classified into WE. The remaining mentions have extremely low NEL confidence (due to sparse contextual information) and are added to NE.

- **CCR:** The WE and NE classes are fed separately to the CCR procedure. Based on the *context summary* similarities between mentions, $C3EL$ performs hierarchical clustering to group together the "Hugh" mentions (in the WE class) and creates a co-referring mention group with the individual mentions' context summaries concatenated. This merging of summaries grows and strengthens captured contexts, which propagates across documents. This concludes the first iteration of $C3EL$.

2. The above results are provided to the **second Iteration**:

- **NEL:** The *context summary* of the "Hugh" mention group in WE now provides definitive cues to correctly map it to the actor Hugh Jackman with high confidence, thus placing it in the SE class.
- **CCR:** The ensuing CCR step groups together "Ava Eliot" and "Ava" (in NE) using co-occurrence context of the co-referring Hugh mentions.

3. Subsequent NEL iterations (on WE and NE) identify "Ava" as an out-of-KB entity and correctly links "Australia" to the movie using CCR-generated mention-group contexts and link validation. CCR finally groups together "Logan" with "Wolverine" based on context similarity with distant KB features. This process of alternating CCR and NEL is repeated until all mention groups are strongly connected to KB entities (placed in SE), or no changes are made anymore.

The NEL and CCR procedures are performed separately on the different mention types (like PER, LOC, etc.), since different mention types rarely co-refer. We next present the internal working details of the NEL and CCR stages of $C3EL$.

2.2.1 Named-Entity Linking (NEL) Stage

In its NEL procedure, $C3EL$ disambiguates mentions to entities in the YAGO knowledge base (`yago-knowledge.org`). We perform NEL on the sentences ($S(M_i)$) of a mention group, using named-entity popularity statistics and context, to obtain the best matching entity, its confidence score, and the corresponding Wikipedia page (from $sameAs$ link in YAGO). Assume a mention group M_i to be mapped to an entity e_i with a confidence score of $\phi(M_i, e_i)$.

A. Link Validation: For each mention group (e.g., Hugh), we extract *distant KB labels* such as semantic types or categories (e.g., actor), title (e.g., Golden Globe winner), alias (e.g., Wolverine), location, and gender (for person) from the Wikipedia page infoboxes. The similarity of these features to keywords obtained from the context summary $\mathbb{CS}(M_i)$ is computed using IR-style term frequencies within a document (tf) and inverse document frequencies within the corpus (idf). We utilize the bag-of-words model based $tf \times idf$-weighted cosine similarity measure. If the similarity score is above a threshold, τ, the NEL result is accepted, otherwise it is discarded – thus avoiding noisy linkage of sparse mentions to prominent KB entries. This subtle introduction of *controlled distant supervision* within the $C3EL$ framework enables efficient detection of out-of-KB mentions.

B. Classification: To sift out well-known and long-tail entities from new ones, and prevent "noisy" interactions among the contexts of in-KB and out-of-KB mentions (with similar surface forms), mention groups M_i (linked to e_i with score $\phi(M_i, e_i)$) are classified into 3 classes by 2 threshold parameters, δ_s and δ_w, as:

- *Strong Evidence* (SE): For $\phi(M_i, e_i) \geq \delta_s$, mention group M_i exhibits high linkage confidence with e_i and is placed in SE. If two or more mentions in SE are independently mapped to the same KB entity, they co-refer transitively and are hence grouped together with their context summaries merged (implicit CCR). Distant KB features for mentions in SE are extracted and appended to $\mathbb{CS}(M_i)$, providing additional cues for later steps.
- *Weak Evidence* (WE): Mention groups with $\delta_w \leq \phi(M_i, e_i) < \delta_s$ are placed in this class. They mostly represent long-tail in-KB entities (sparsely represented in KB) with limited semantic information (for detection) but might also be new/emerging entities absent from KB.
- *No Evidence* (NE): $\phi(M_i, e_i) < \delta_w$ represents mentions groups that have been mapped to $null$ (or have near-zero match confidence) or have failed link validation during the NEL procedure. These entities are most likely to be out-of-KB and are allocated to this class.

2.2.2 Cross-Document CR (CCR) Stage

The CCR stage of $C3EL$ adopts the sampling-based hierarchical clustering approach of (Dutta & Weikum, 2015), to obtain co-referring mention clusters.

A. Similarity Measure: To infer whether two mention groups represent the same entity, the similarity between the context summaries are computed based on (i) *tf-idf*-weighted bag-of-words cosine distance, and (ii) partial-match scores of multi-word keyphrases in bounded text windows (Taneva et al., 2011). The context summaries (with stopwords removed) are re-interpreted as, (i) bag of words, and (ii) bag of keyphrases, to extract fea-

ture vectors for similarity computation. Finally, the mixture model of *bag-of-words* (BoW) and *keyphrases* (KP) of (Dutta & Weikum, 2015) is used to assign feature weights using *tf-idf* measure.

B. Hierarchical Clustering: s mention groups are uniformly randomly sampled and their similarities to the other groups (using context summary) are computed. A similarity-weighted graph with the mention groups as nodes and edge weights representing mention-mention similarities is constructed. Bisection-based hierarchical *balanced min-edge-cut graph partitioning* (Buluc et al., 2013) is performed, using the $METIS$ software (Karypis & Kumar, 1999)[2], to partition non-coreferent mentions groups. The *Bayesian Information Criterion* (BIC) (Schwarz, 1978; Hourdakis et al., 2010), a Bayesian variant of Minimum Description Length (Grünwald, 2007), is used as the cluster split stopping criterion, and the context summaries within each final cluster are merged.

CCR aims to process heterogeneous corpora that go beyond a single domain and style, such as Web collections.

2.3 Finalization Stage

For the remaining mention groups in WE, we finally perform threshold based disambiguation of mention clusters using the context summaries. For each mention group $M_i \in WE$, we compute (1) its context summary similarities (as in Section 2.2.2) to all other mention groups M_j in SE by also using distance features from the weakly linked KB entities, and (2) textual overlap between the mention group representatives. M_i is concatenated with the best matching entity M_k (in SE) if the similarity score is above a threshold θ; else M_i is marked as an out-of-KB entity (mapped to $null$) and is placed in the NE class. This helps in reducing propagated CR errors like erroneous mention boundary detection (in NER), omissions in co-reference chain, etc. (leading to "phantom" out-of-KB entities).

The obtained mention groups represent the final equivalence classes of co-referring mentions across documents – capturing both in-KB entities (with links to the KB) in the SE class and out-of-KB entities (mapped to $null$) in the NE class.

3 Experimental Evaluation

In this section, we empirically study the performance of $C3EL$ against various state-of-the-art methods. We analyze the individual gains in CCR and NEL due to the joint modeling.

Datasets: We use the following 2 publicly available corpora:

- **EventCorefBank (ECB) corpus**[3] (Bejan & Harabagiu, 2010)**:** contains 482 news and Web articles (classified into 43 topics) with a total of 5447 mentions corresponding to 1068 distinct named-entities. Entity co-reference annotations (across documents within each topic cluster) were provided by (Lee et al., 2012), and we performed manual examination of the annotations for KB linking of the entities to Wikipedia entries, if present; thus providing ground truth for both CCR and NEL.
- **ClueWeb2009 FACC1 dataset**[4] (Gabrilovich et al., 2013)**:** provides machine automated entity-linkage annotations of the *ClueWeb09* corpus (ca. 1 Billion crawled Web pages) with Freebase entries[5]. The corpus contains many topical domains and highly diverse documents from news, movie reviews, people home pages to blogs and other social media posts. We randomly select $500K$ documents containing 4.64 Million mentions associated with 1.29 Million distinct entities to form our corpus. For NEL ground-truth construction, we link the entities to their Wikipedia pages (using Freebase's "on the web" property). Since no explicit annotations of inter-document entity co-references exists, we consider two mentions (in different documents) to co-refer if they are linked with the same Freebase entity.

Evaluation: To assess the output quality of $C3EL$ we use the following established metrics:

- B^3 $F1$ **score** (Bagga & Baldwin, 1998): measures the F1 score as the harmonic mean of average *precision* and *recall* computed over all mention groups in the final equivalence classes. Precision (for a mention group) represents the ratio of the number of correctly reported co-references (or linking) to the actual number; while recall computes the fraction of the gold-standard annotations correctly identified.
- ϕ_3 $-$ $CEAF$ **score** (Luo, 2005): provides an alternate F1 score computed as in the B^3 measure; but calculates precision and recall of mention groups using the best 1-to-1 mapping (i.e., mapping with maximum mention overlap) between the resultant equivalence classes and those in the ground truth. Normalization with the number of mentions for each of the resultant classes yields the ϕ_4-CEAF score.

We consider only the 3 most notable mention types: person (PER), location (LOC), and organization (ORG) – accounting for 99.7% of entities present in the ECB corpus and 96.3% of our ClueWeb09 corpus. All experiments were conducted on a 4 core Intel i5 2.50 GHz processor with 8GB RAM running Ubuntu 12.04 LTS.

3.1 Parameter Tuning & Sensitivity Study

Validation of entity linkage to KB and their subsequent classification into *confidence classes* (as de-

[2] glaros.dtc.umn.edu/gkhome/metis/metis/overview

[3] faulty.washington.edu/bejan/data/ECB1.0.tar.gz

[4] lemurproject.org/clueweb09/FACC1

[5] Human analysis of a subset of the annotations generated revealed a precision of $80 - 85\%$ (Gabrilovich et al., 2013)

Approach	P	R	B^3	ϕ_3	ϕ_4
EECR	74.9	55.5	63.7	-	33.7
CROCS	73.11	75.28	74.18	67.35	-
C3EL	79.52	82.91	81.18	73.89	53.3

Table 2: CCR performance (%) comparison on ECB

scribed in Section 2) during the NEL step of $C3EL$ are based on 3 parameters: confidence thresholds (δ_s and δ_w) and validation threshold (τ); the values of which can be tuned based on *cross-validation* approach with *train* and *test* data subsets. Using the "gold annotations" of the train-set (30% of total data), parameter values providing the best precision score are individually learnt using *line search* with small step size.

In our experimental setup, we systematically vary the parameter values and observe its effects on $C3EL$ for the training data. With increase in δ_s, the number of mentions mapped to the *Strong Evidence* (SE) class decreases. This in turn limits the influx of external KB features, thus degrading CCR performance as observed in Table 1(a). While for low values of δ_s, even weak mention links are placed in SE, leading to a decrease in precision due to noisy KB feature inclusion. On the other hand, a high δ_w value increases the number of mentions in the NE class, while low values tends to accumulate mentions in the WE class. This adversely affects the detection of out-of-KB entities due to noise from other co-occurring similar KB mentions (refer Table 1(b)) during clustering in CCR step.

The effect of τ on $C3EL$ has be shown in Table 1(c). Similar to the behavior induced by δ_s, we observe that a high τ limits entity linking and possible KB feature inclusion, while an extremely low value (near to zero) allows for noisy feature incorporation – both situations leading to lowered CCR efficiency. However, since τ prevents gross mis-alignment of mentions to KB entities, a wide range of small value ($0.1 - 0.35$) is seen to provide comparable performance.

Hence, for our remaining experimental study we set $\delta_s = 0.11$ and $\delta_w = 0.06$ (as in (Hoffart et al., 2014)), while τ is set to 0.1, and threshold for the finalization stage $\theta = 2 \times \delta_s = 0.22$.

3.2 CCR Performance Results

We initially benchmark the performance improvement in cross-document co-reference resolution (CCR) procedure by $C3EL$ against two competing approaches:
(1) state-of-the art sampling based hierarchical clustering method, $CROCS$ (Dutta & Weikum, 2015); and
(2) iterative joint entity-event CCR, $EECR$ (Lee et al., 2012).

Table 2 tabulates the results obtained on the ECB dataset. We observe C3EL to decisively outperform both the existing methods, providing a B^3 F1 improvement of around 7% over $CROCS$ and 17% over EECR. We further attain around 6% $\phi_3 - CEAF$ score enhancement over CROCS, and a significant 20% improved $\phi_4 - CEAF$ score compared to EECR.

Approach	P (%)	R (%)	B^3 (%)	ϕ_3 (%)
$CROCS_G$	79.9	83.33	81.58	74.11
$C3EL_G$	84.74	89.9	87.24	80.5

Table 3: CCR results on ECB

Type	Approach	P (%)	R (%)	B^3 (%)
PER	$CROCS_G$	71.8	74.15	72.96
	$C3EL_G$	84.85	82.73	83.78
LOC	$CROCS_G$	78.23	85.41	81.66
	$C3EL_G$	81.41	94.31	87.29
ORG	$CROCS_G$	85.73	87.89	86.8
	$C3EL_G$	88.52	91.82	90.14

Table 4: CCR results on ECB for mention types

A. Gold Results: Errors introduced during the preprocessing stage of C3EL (e.g., mention omission, tag mis-classification, intra-document CR errors, etc., by the Stanford CoreNLP toolkit) propagate to subsequent computing stages and adversely impacts the overall system performance. To provide an unbiased viewpoint of the actual performance of $C3EL$, we manually provided "exact" mentions, mention tags, and intra-document CR mention chains for the ECB corpus; thereby obtaining *gold performance results*. From Table 3 we observe a 6% F1 points improvement (for both B^3 & CEAF-ϕ_3) in $C3EL$ compared to $CROCS$.

B. Mention Categorization: Person mention type (PER) provides the greatest challenge for CCR systems (compared to other types like LOC, ORG, etc.) due to associated nicknames, titles, and varied surface forms (abbreviations, spellings, etc.). We thus evaluate the CCR performance of $C3EL$ (and compare it with CROCS) on the ECB data, with "exact" input mentions, for the different mention categories. Table 4 validates that our joint modeling provides better global information cues, reporting a B^3 F1 score enhancement of around 11% over CROCS for PER mentions; along with improved results for the other mention types as well.

C. Large Data: To study the robustness of $C3EL$ and the effects of large datasets on CCR, we performed evaluations on the ClueWeb09-FACC1 dataset. Similar to the ECB dataset, $C3EL$ exhibits a B^3 F1 score improvement of nearly 10% and a ϕ_3-CEAF F1 improvement of 12% over $CROCS$ (refer Table 5).

The above experimental results showcase that a combined approach helps overcome challenges faced in CCR by entity linkage and corresponding distant KB feature extraction; improving the overall accuracy.

3.3 Named-Entity Linking (NEL) Results

We now benchmark the performance of named-entity linking (NEL) procedure for $C3EL$ against the state-of-the-art open-source AIDA software (`github.com/yago-naga/aida`). We separately inspect the precision of mention linking for *prominent entities* (in-KB) as well as *new/emerging* (out-of-KB) entities, and characterize the links as

Datasets	δ_s (B^3 F1)					δ_w (P)					τ (B^3 F1)				
	0.01	*0.05*	*0.10*	*0.15*	*0.20*	*0.01*	*0.02*	*0.04*	*0.06*	*0.08*	*0.03*	*0.10*	*0.20*	*0.35*	*0.50*
ECB	79.3	82.2	84.2	83.5	81.0	73.1	75.3	77.3	78.7	78.4	76.9	81.2	81.2	81.1	79.2
ClueWeb	70.1	77.2	81.5	81.0	78.7	78.2	81.1	83.6	85.1	85.1	70.3	79.1	78.2	78.8	76.4
	(a)					(b)					(c)				

Table 1: $C3EL$ performance (a) in CCR with δ_s, (b) in out-of-KB NEL with δ_w, and (c) in CCR with τ

Approach	P (%)	R (%)	B^3 (%)	ϕ_3 (%)
CROCS	68.66	70.96	69.79	62.85
C3EL	75.76	81.42	78.49	74.13

Table 5: CCR results on ClueWeb09-FACC1

Approach	Within-KB			Out-of-KB		Overall
	$\mathbb{C}$	$\mathbb{I}$	$\mathbb{U}$	$\mathbb{C}$	$\mathbb{I}$	**P** (%)
AIDA	86.5	13.5	0.0	63.9	36.1	83.4
C3EL	85.4	14.4	0.2	79.0	21.0	84.9

Table 6: NEL performance (%) comparison on ECB

Correct ($\mathbb{C}$), *Incorrect* ($\mathbb{I}$), or *Unlinked* ($\mathbb{U}$). The results on the ECB corpus are reported in Table 6. $C3EL$ attains comparable performance ($\sim$ 85% precision) to that of AIDA for well-known entity-mentions present in KB; albeit with a few mentions remaining unlinked due to our cautious *link validation* (using τ) approach. However, the use of τ reduces aggressive KB linking to provide a significant 15% improvement (over AIDA) in precise detection of new/emerging entities absent in KB. Overall, an 1.5% precision gain is observed by the joint formulation.

A. Large Data: The diverse nature of the web-scale ClueWeb09 dataset clearly portrays the performance gains in NEL procedure due to CCR generated information integration. For entities present in the KB, we observe an accuracy improvement of 0.5% over AIDA (refer Table 7). Similar to that of the ECB data, $C3EL$ attains a significant $\sim$ 14% improvement in the detection of new/emerging entities not represented in KB. For the 1 million mentions, $C3EL$ provides around 4% overall performance improvements.

Using a bootstrap re-sampling t-test (as in (Durrett & Klein, 2014)), we observed high statistical significance ($p < 0.01$) for Out-of-KB and Overall NEL, whereas the difference for Within-KB NEL is not statistically significant. Coping with Out-of-KB entities is essential for joint CCR+NEL, and an improved NEL performance using propagated information from CCR using semantics along with link validation enables highly efficient detection of new or emerging entities.

3.4 Comparison with Joint Models

Traditional CR methods fail to cope with the heterogeneity of mentions and contexts across multiple documents, and some form of clustering or joint reasoning over all mentions is thus mandatory. These methods have quadratic or cubic (sometimes even exponential) complexity, and hence running CR+NEL on a concatenated super-document works only for small corpora, and would be prohibitively expensive for large corpora, even in offline processing mode (Singh et al., 2011).

However, to study the behavior of existing CR-NEL joint models under "small" CCR environments, we compare $C3EL$ with:
(1) multi-sieve based $NECo$ (Hajishirzi et al., 2013)[6]; and
(2) conditional random field based BER (Durrett & Klein, 2014) [7].

Approach	Within-KB			Out-of-KB		Overall
	$\mathbb{C}$	$\mathbb{I}$	$\mathbb{U}$	$\mathbb{C}$	$\mathbb{I}$	**P** (%)
AIDA	88.5	10.6	1.0	69.6	30.4	84.6
C3EL	89.0	9.8	1.2	83.7	16.3	88.1

Table 7: NEL results (%) on ClueWeb09-FACC1 (statistical significance $p < 0.01$ for Out-of-KB entities)

Three topic clusters from the ECB corpus with 3, 4, and 5 articles respectively were selected, and the documents within each cluster were merged to form 3 "super-articles" (one per topic), forming a simulated CR setting. $NECo$ and BER were then used to perform CR and NEL on these 3 articles, and the results compared with that obtained by $C3EL$ on the original documents. We repeatedly sample 12 articles across 3 topic clusters, and execute the approaches to report the micro-averaged results across 5 independent runs.

From Table 8(a) we observe that the algorithms exhibit comparable co-reference resolution performance; thus validating propagation of global semantics in $C3EL$ due to the joint formulation. However, such CR methods using multi-sieves and CRF do not scale beyond few documents (upon concatenation), and require at least $4\times$ more runtime compared to $C3EL$. Hence, CCR cannot be efficiently tackled by simply employing CR methods on a "super-document".

However, harnessing of non-local mention features (via CCR) and efficient detection of new mentions using link validation enables $C3EL$ to achieve a gain of around 5% in NEL compared to others (see Table 8(b)). For both procedures, we observed statistically significant improvements of C3EL over BER and NECo with $p < 0.05$, using the bootstrap re-sampling t-test.

To further study the effect of larger corpus, we sampled 25 documents (with co-referring mentions) from the ClueWeb09 dataset and performed analysis among the algorithms. As previously, we observed significant computational complexity for traditional CR methods when applied to CCR setting making them far slower ($6-7\times$) than $C3EL$. Table 9 reports the CCR and NEL averaged results obtained across 5 independent runs. We attained comparable performance in CCR with around 3% improvement in NEL. All the algorithms are seen to achieve high NEL results due to the large presence of well-known (in-KB) entities.

[6] cs.washington.edu/research-projects/nlp/neco
[7] nlp.cs.berkeley.edu/projects/entity.shtml

Approach	P (%)	R (%)	B^3 (%)
NECo	87.77	82.09	84.84
BER	88.30	86.53	87.41
C3EL	87.54	88.11	87.82

(a)

Approach	$\mathbb{C}$ (%)	$\mathbb{I}$ (%)	$\mathbb{U}$ (%)
NECo	89.13	10.87	0.0
BER	89.89	10.11	0.0
C3EL	93.2	4.61	2.19

(b)

Table 8: Joint "Simulated" results on ECB subset for (a) CCR, and (b) NEL (statistical significance $p < 0.05$)

Approach	P (%)	R (%)	B^3 (%)
NECo	81.14	79.65	80.39
BER	84.36	83.01	83.68
C3EL	83.52	85.56	84.53

(a)

Approach	$\mathbb{C}$ (%)	$\mathbb{I}$ (%)	$\mathbb{U}$ (%)
NECo	94.71	5.29	0.0
BER	95.27	4.73	0.0
C3EL	98.23	1.5	0.27

(b)

Table 9: Joint "Simulated" results on ClueWeb09 subset for (a) CCR, and (b) NEL

3.5 Algorithmic Baseline Study

We explore the performance of variants of $C3EL$ (on both corpora) ablating various system components (see Table 10). Explicitly, we consider:

- **Co-occurring Mentions:** Removal of cooccurrence mentions context from the *context summaries* constructed, reduces semantic information and adversely affects both NEL and CCR procedures. We thus observe a sharp decrease in CCR performance and also a degradation in entity linking.
- **Link Validation:** Filtering of mention linking to KB entities using link validation step (with threshold τ) in $C3EL$ enables corroboration of mention context keywords with the linked entity features. This leads to enhanced detection of new or emerging entities by reducing induction of noise during the CCR phase. Removal of this process permits aggressive entity linking and introduces noise, affecting new/emerging entity detection. We observe (from Table 10) nearly 20% reduction of precision (on both datasets) in identification of out-of-KB entity-mentions compared to $C3EL$.
- **NEL Categorization:** The differentiation of mentions (into classes) confidently mapped to KB entity reduces the collusion of "strong" linked mentions with other "noisy" mention contexts. This reduces incorrect grouping of different mentions with similar surface forms, contexts, etc., thereby improving precision of the CCR process. Use of a single NEL classification approach is observed to degrade CCR results, which in turn increases spurious entity linkage, decreasing NEL efficiency (Table 10).
- **Distant KB features:** As observed in (Baker, 2012; Zheng et al., 2013), extracted external KB features provide global and enhanced information cues promoting CR. We similarly observe CCR to attain the lowest F1 scores (compared to other baselines) when KB features are ignored. This in turn affects the linking of (some) well-known entities due to reduced context, leading to incorrect or low confidence NEL. Since no feature inclusion is performed for out-of-KB mentions, no effect is observed.

We observe that a joint formulation encompassing multiple information sources (along with noise filtering) enables mutually enhanced CCR and NEL within the proposed iterative feedback based framework, $C3EL$.

4 Related Work

Co-reference Resolution (CR): Traditional intra-document CR methods involve syntactic and semantic feature combination for identifying the best antecedent (preceding name or phrase) for a mention. CR methods employ rules or supervised learning techniques based on linguistic features such as syntactic paths and mention distances to assess semantic compatibility (Haghighi & Klein, 2009; Raghunathan et al., 2010; Rahman & Ng, 2011), while syntactic features are derived by deep parsing of sentences and noun group parsing. Semantic features from background knowledge resources like encyclopedia were used in (Daumé & Marcu, 2005; Ponzetto & Strube, 2006; Ng, 2007). The use of Wikipedia and structured knowledge bases (such as YAGO) to obtain mention-type relation and fine-grained mention attributes was explored by (Haghighi & Klein, 2009; Rahman & Ng, 2011). An overview of CR methods is given in (Ng, 2010).

Recent methods involve the use of multi-phase sieve, applying a cascade of rules for narrowing down the antecedent candidates for a mention (Raghunathan et al., 2010). Cluster ranking functions have also been proposed (Rahman & Ng, 2011; Zheng et al., 2013) to extend this paradigm for incrementally expanding and merging mention groups with preceding candidate clusters using relatedness features (Ratinov & Roth, 2012) and distant knowledge inclusion (Durrett & Klein, 2013). Person name disambiguation, a specific variation of CR, dealing with only person names, titles, nicknames, and other surface form variations was introduced in (Chen & Martin, 2007).

Distant Knowledge Labels: For obtaining semantic features, additional knowledge resources such as Wikipedia, YAGO, and FrameNet have been considered (Rahman & Ng, 2011; Baker, 2012). CR methods with confidence-thresholds were proposed in (Ratinov & Roth, 2012; Lee et al., 2013), and (Zheng et al., 2013) generalized these tech-

Baseline	ECB Dataset								ClueWeb09-FACC1 Dataset							
	CCR result			NEL results					CCR result			NEL results				
				Within-KB			*Out-of-KB*					*Within-KB*			*Out-of-KB*	
	P	R	B^3	$\mathbb{C}$	$\mathbb{I}$	$\mathbb{U}$	$\mathbb{C}$	$\mathbb{I}$	P	R	B^3	$\mathbb{C}$	$\mathbb{I}$	$\mathbb{U}$	$\mathbb{C}$	$\mathbb{I}$
Ignored Mention Co-occurrence	72.5	74.4	73.4	80.2	19.6	0.2	74.4	25.6	69.3	72.2	70.7	83.8	14.6	1.6	80.6	19.4
Link Validation (τ) ignored	79.0	81.4	80.2	**85.5**	14.5	**0.0**	62.8	37.2	74.8	81.0	77.8	**88.9**	**10.1**	**1.0**	69.8	30.2
Removed NEL Classification	73.2	80.7	76.8	83.9	15.9	0.2	76.1	23.9	70.1	77.6	73.6	86.1	12.3	1.6	79.5	20.5
Distant KB feature dropped	68.9	73.1	70.9	82.8	17.0	0.2	**79.0**	**21.0**	66.4	72.9	69.5	85.4	13.0	1.6	**83.7**	**16.3**
C3EL (Complete)	**79.5**	**82.9**	**81.18**	85.4	**14.4**	0.2	**79.0**	**21.0**	**75.8**	**81.4**	**78.5**	88.3	**10.1**	1.6	**83.7**	**16.3**

Table 10: CCR and NEL results (%) of C3EL for different baseline variations

niques by ranking the matching entities for distant labeling. However, such prior methods utilize distance labels of the current mention and considers all matching mentions making the procedure expensive. On the other hand, we extract distant features for the strongly matching (best) candidate only, reducing the performance overhead.

Cross-Document CR (CCR): Early approaches towards CCR involved the use contextual information from input documents for IR-style similarity measures (e.g., tf×idf score, KL divergence, etc.) over textual features (Bagga & Baldwin, 1998; Gooi & Allan, 2004). Probabilistic graphical models jointly learning the mappings of mentions to equivalent classes (co-referring mentions) using features similar to local CR techniques were studied in (Culotta et al., 2007; Singh et al., 2010; Singh et al., 2011), A clustering approach coupled with statistical learning of parameters was studied in (Baron & Freedman, 2008). However, such methods fail to cope with large corpora, and hence a "light-weight" streaming variant of CCR was introduced by (Rao et al., 2010).

Co-occurring mentions context have been harnessed for disambiguating person names for CR in (Mann & Yarowsky, 2003; Niu et al., 2004; Chen & Martin, 2007; Baron & Freedman, 2008). However, these methods do not use KB and depend on information extraction (IE) methods, witnessing substantial noise due to IE quality variance. A CCR framework combining co-occurring mention context with distant KB features embedded in an active hierarchical clustering procedure (Dutta & Weikum, 2015) was recently shown to perform efficiently, and provides inspiration for parts of our proposed $C3EL$ approach.

Named Entity Linking (NEL): Named entity resolution and linking stems from SemTag (Dill et al., 2003), and similar frameworks like GLOW, WikipediaMiner, AIDA, and others (Milne & Witten, 2008; Ratinov et al., 2011). A collection of entity disambiguation models was presented in (Kulkarni et al., 2009). Other NEL approaches utilize the notion of semantic similarity of entities to corresponding Wikipedia pages (Milne & Witten, 2008), while co-referent mention graph construction modeling mention co-occurrences and context similarity from outgoing hyperlinks in Wikipedia was used by (Hoffart et al., 2011). An integer linear programming (ILP) formulation also based on Wikipedia page similarities was presented in (Ratinov et al., 2011). However, none of these methods involve the incorporation of CR results for NEL. The first study on the benefits of CR for NEL was by (Ratinov & Roth, 2012); but a joint model was not proposed, instead attributes from Wikipedia categories were used as features. An overview and evaluation of different NEL methods has been given by (Hachey et al., 2013).

Joint Models: Jointly solving CR for entities and events utilizing cluster construction based on feature semantic dependencies was devised in (Lee et al., 2012). The use of CR as a pre-processing step for subsequent NEL procedure using an ILP formulation was proposed by (Chen & Roth, 2013). Recently, (Hajishirzi et al., 2013) proposed a joint model for CR and NEL using the Stanford multi-pass cluster update CR system with automatic linking of mentions to Wikipedia. An integrated belief propagation-based framework for CR, NER, and relation extraction was developed in (Singh et al., 2013). Subsequently, the model was enhanced by the use of structured conditional random fields, to solve CR, NER, and NEL in combination (Durrett & Klein, 2014). Other works involving joint formulation of NER and NEL use uncertainty of mention boundaries along with segmentation information extracted from Wikipedia (Sil & Yates, 2013). However, to the best of our knowledge, this work provides the first approach to jointly tackle CCR and NEL across documents in an entire corpus.

5 Conclusions

This paper presented the novel $C3EL$ framework for joint computation of cross-document coreference resolution (CCR) and named-entity linking (NEL). Our approach utilizes: (1) *context summaries* including co-occurring mention groups allowing for global context and feature propagation, and (2) *link validation* for NEL using distant KB features. This is embedded in an interleaved CCR and NEL model allowing for global semantics and feature propagation. The iterative approach enables information feedback between CCR (provides corpus-wide cues) and NEL (providing distant KB features). Experimental results on news and web data demonstrate improved performance of both CCR and NEL compared to prior methods.

References

Amit Bagga and Breck Baldwin. Entity-Based Cross-Document Coreferencing Using the Vector Space Model. *In COLING-ACL* 1998, pages 79–85.

Collin F. Baker. FrameNet, Current Collaborations and Future Goals. *LREC* 2012, 46(2):269–286.

Alex Baron and Marjorie Freedman. Who is Who and What is What: Experiments in Cross-Document Co-Reference. *In EMNLP* 2008, pages 274–283.

Cosmin A. Bejan and Sanda Harabagiu. Unsupervised Event Coreference Resolution with Rich Linguistic Features. *In ACL* 2010, pages 1412–1422.

Aydin Buluc, Henning Meyerhenke, Ilya Safro, Peter Sanders, and Christian Schulz. Recent Advances in Graph Partitioning. Karlsruhe Institute of Technology, Technical Report 2013.

Razvan Bunescu and Marius Paşca. Using Encyclopedic Knowledge for Named Entity Disambiguation. *In EACL* 2006, pages 9–16.

Ying Chen and James Martin. Towards Robust Unsupervised Personal Name Disambiguation. *In EMNLP* 2007, pages 190–198.

Xiao Cheng and Dan Roth. Relational Inference for Wikification. *In EMNLP* 2013, pages 1787–1796.

Marco Cornolti, Paolo Ferragina, and Massimiliano Ciaramita. A Framework for Benchmarking Entity-Annotation Systems. *In WWW* 2013, pages 249–260.

Silviu Cucerzan. Large-Scale Named Entity Disambiguation Based on Wikipedia Data. *In EMNLP-CoNLL* 2007, pages 708–716.

Aron Culotta, Michael L. Wick, and Andrew McCallum. First-Order Probabilistic Models for Coreference Resolution. *In HLT-NAACL* 2007, pages 81–87.

Hal Daumé III, Daniel Marcu. A large-scale exploration of effective global features for a joint entity detection and tracking model. *In HLT-EMNLP* 2005, pages 97–104.

Stephan Dill, Nadav Eiron, David Gibson, Daniel Gruhl, Ramanathan V. Guha, Aanat Jhingran, Tapas Kanungo, Sridhar Rajagopalan, Andrew Tomkins, John A. Tomlin, and Jason Y. Zien. SemTag and Seeker: bootstrapping the semantic web via automated semantic annotation. *In WWW* 2003, pages 178–186.

Greg Durrett and Dan Klein. Easy victories and uphill battles in coreference resolution. *In EMNLP* 2013, pages 1971–1982.

Greg Durrett and Dan Klein. A Joint Model for Entity Analysis: Coreference, Typing, and Linking. *TACL* 2014, 2:477–490.

Sourav Dutta and Gerhard Weikum. Cross-Document Co-Reference Resolution using Sample-Based Clustering with Knowledge Enrichment. *TACL* 2015, 3:15–28.

Jenny R. Finkel, Trond Grenager, and Christopher D. Manning. Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling. *In ACL* 2005, pages 363–370.

Jenny R. Finkel and Christopher D. Manning. Joint Parsing and Named Entity Recognition. *In EMNLP-CoNLL* 2009, pages 326–334.

Evgeniy Gabrilovich, Michael Ringgaard, and Amarnag Subramanya. FACC1: Freebase annotation of ClueWeb corpora, Version 1 (Format version 1, Correction level 0). 2013.

Chung H. Gooi and James Allan. Cross-Document Coreference on a Large Scale Corpus. *In HLT-NAACL* 2004, pages 9–16.

Peter D. Grünwald. *The Minimum Description Length Principle*. MIT University Press, 2007.

Ben Hachey, Will Radford, Joel Nothman, Matthew Honnibal, and James R. Curran. Evaluating entity linking with Wikipedia. *Artificial Intelligence Journal* 2013, 194:130–150.

Aria Haghighi and Dan Klein. Simple Coreference Resolution with Rich Syntactic and Semantic Features. *In EMNLP* 2009, pages 1152–1161.

Aria Haghighi and Dan Klein. Coreference Resolution in a Modular, Entity-Centered Model. *In HLT-NAACL* 2010, pages 385–393.

Hannaneh Hajishirzi, Leila Zilles, Daniel S. Weld, and Luke Zettlemoyer. Joint Coreference Resolution and Named-Entity Linking with Multi-pass Sieves. *In EMNLP* 2013, pages 289–299.

Johannes Hoffart, Mohamed A. Yosef, Ilaria Bordino, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. Robust Disambiguation of Named Entities in Text. *In EMNLP* 2011, pages 782–792.

Johannes Hoffart, Yasemin Altun, and Gerhard Weikum. Discovering Emerging Entities with Ambiguous Names. *In WWW* 2014, pages 385–396.

Nikos Hourdakis, Michalis Argyriou, Euripides G. M. Petrakis, and Evangelos E. Milios. Hierarchical Clustering in Medical Document Collections: the BIC-Means Method. *Journal of Digital Information Management* 2010, 8(2):71–77.

George Karypis and Vipin Kumar. A Fast and Highly Quality Multilevel Scheme for Partitioning Irregular Graphs. *Journal on Scientific Computing* 1999, 20(1):359–392.

Akshay Krishnamurty, Sivaraman Balakrishnan, Min Xu, and Aarti Singh. Efficient Active Algorithms for Hierarchical Clustering. *In ICML* 2012, pages 887–894.

Sayali Kulkarni, Amit Singh, Ganesh Ramakrishnan, and Soumen Chakrabarti. Collective annotation of Wikipedia entities in Web text. *In KDD* 2009, pages 457–466.

Heeyoung Lee, Yves Peirsman, Angel Chang, Nathanael Chambers, Mihai Surdeanu, and Dan Jurafsky. Stanford's Multi-Pass Sieve Coreference Resolution System at the CoNLL-2011 Shared Task. *In CoNLL* 2011, pages 28–34.

Heeyoung Lee, Marta Recasens, Angel Chang, Mihai Surdeanu, and Dan Jurafsky. Joint Entity and Event Coreference Resolution across Documents. *In EMNLP* 2012, pages 489–500.

Heeyoung Lee, Angel Chang, Yves Peirsman, Nathanael Chambers, Mihai Surdeanu, and Dan Jurafsky. Deterministic Coreference Resolution based on Entity-centric, Precision-ranked Rules. *Computational Linguistics* 2013, 39(4): 885–916.

Xiaoqiang Luo. On Coreference Resolution Performance Metrics. *In EMNLP* 2005, pages 25–32.

Gideon S. Mann and David Yarowsky: Unsupervised Personal Name Disambiguation. *In CoNLL* 2003, pages 33–40.

David Milne and Ian H. Witten. Learning to Link with Wikipedia. *In CIKM* 2008, pages 509–518.

David Nadeau and Satoshi Sekine. A survey of named entity recognition and classification. *Lingvisticae Investigationes* 2007, 30(1):3–26.

Vincent Ng. Shallow semantics for coreference resolution. *In IJCAI* 2007, pages 1689–1694.

Vincent Ng. Supervised Noun Phrase Coreference Research: The First Fifteen Years. *In ACL* 2010, pages 1396–1411.

Cheng Niu, Wei Li, and Rohini K. Srihari. Weakly Supervised Learning for Cross-document Person Name Disambiguation Supported by Information Extraction. *In ACL* 2004, article 597.

Simone P. Ponzetto and Michael Strube. Exploiting semantic role labeling, WordNet and Wikipedia for coreference resolution. *In HLT-NAACL* 2006, pages 192–199.

Karthik Raghunathan, Heeyoung Lee, Sudarshan Rangarajan, Nathanael Chambers, Mihai Surdeanu, Dan Jurafsky, and Christopher Manning. A Multi-Pass Sieve for Coreference Resolution. *In EMNLP* 2010, pages 492–501.

Altaf Rahman and Vincent Ng. Coreference Resolution with World Knowledge. *In ACL* 2011, pages 814–824.

Altaf Rahman and Vincent Ng. Ensemble-Based Coreference Resolution. *In IJCAI* 2011, pages 1884–1889.

Delip Rao, Paul McNamee, and Mark Dredze. Streaming Cross Document Entity Coreference Resolution. *In COLING* 2010, pages 1050–1058.

Lev A. Ratinov and Dan Roth. Design Challenges and Misconceptions in Named Entity Recognition. *In CoNLL* 2009, pages 147–155.

Lev A. Ratinov, Dan Roth, Doug Downey, and Mike Anderson. Local and Global Algorithms for Disambiguation to Wikipedia. *In ACL* 2011, pages 1375–1384.

Lev A. Ratinov and Dan Roth. Learning-based Multi-Sieve Co-reference Resolution with Knowledge. *In EMNLP-CoNLL* 2012, pages 1234–1244.

Gideon E. Schwarz. Estimating the Dimension of a Model. *Annals of Statistics* 1978, 6(2):461–464.

Avirup Sil and Alexander Yates. Re-ranking for Joint Named-Entity Recognition and Linking. *In CIKM* 2013, pages 2369–2374.

Sameer Singh, Michael L. Wick, and Andrew McCallum. Distantly Labeling Data for Large Scale Cross-Document Coreference. CoRR abs/1005.4298, 2010.

Sameer Singh, Amarnag Subramanya, Fernando Pereira, and Andrew McCallum. Large-Scale Cross-Document Coreference Using Distributed Inference and Hierarchical Models. *In ACL* 2011, pages 793–803.

Sameer Singh, Sebastian Reidel, Brian Martin, Jiaping Zheng, and Andrew McCallum. Joint Inference of Entities, Relations, and Coreference. *In Workshop of AKBC* 2013, pages 1–6.

Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. YAGO: a Core of Semantic Knowledge. *In WWW* 2007, pages 697–706.

Bilyana Taneva, Mouna Kacimi, and Gerhard Weikum. Finding Images of Difficult Entities in the Long Tail. *In CIKM* 2011, pages 189–194.

Mohamed A. Yosef, Johannes Hoffart, Marc Spaniol, and Gerhard Weikum. AIDA: An Online Tool for Accurate Disambiguation of Named Entities in Text and Tables. *In VLDB* 2011, 4(12):1450–1453.

Jiaping Zheng, Luke Vilnis, Sameer Singh, Jinho D. Choi, and Andrew McCallum. Dynamic knowledge-base alignment for coreference resolution. *In CoNLL* 2013, pages 153–162.

Joint Mention Extraction and Classification with Mention Hypergraphs

Wei Lu
Singapore University
of Technology and Design
luwei@sutd.edu.sg

Dan Roth
University of Illinois
at Urbana-Champaign
danr@illinois.edu

Abstract

We present a novel model for the task of joint mention extraction and classification. Unlike existing approaches, our model is able to effectively capture overlapping mentions with unbounded lengths. The model is highly scalable, with a time complexity that is linear in the number of words in the input sentence and linear in the number of possible mention classes. Our model can be extended to additionally capture mention heads explicitly in a joint manner under the same time complexity. We demonstrate the effectiveness of our model through extensive experiments on standard datasets.

1 Introduction

One of the essential goals in natural language processing (NLP) is the development of effective systems that can capture the underlying semantics conveyed by human languages. An important step towards such a goal is the development of practical systems that can efficiently extract useful shallow semantic information such as entities and at the same time identify their semantic classes (*e.g.*, person, organization, etc).

Such a task is often known as named entity recognition and classification (NERC), one of the standard tasks in information extraction (IE). While such a task focuses on the extraction and classification of entities in the texts which are named, recently researchers also showed interest in a closely related task – *mention extraction and classification/typing*. Unlike a named entity, a mention is typically defined as a reference to an entity in natural language text that can be either named, nominal or pronominal (Florian et al., 2004). The task of mention detection and tracking has received substantial attention, largely due to its important role in conducting several downstream tasks, such as *relation extraction* (Mintz et al., 2009), *entity linking* (Guo et al., 2013), and *coreference resolution* (Chang et al., 2013).

While most existing work on named entity recognition and mention extraction and classification have been effective, there remain several key limitations associated with existing models. In fact, one can view these problems as instances of the more general problem of *semantic tagging* – the task of assigning appropriate semantic tags to certain text spans for a given input sentence. Unlike part-of-speech (POS) tagging, which has been extensively studied in the past few decades by the community, such a semantic tagging task presents several additional new challenges. First, a mention can consist of multiple words, so its length can be arbitrarily long. Second, the mentions can overlap with one another. Popular models used for POS tagging, such as linear-chain conditional random fields (Lafferty et al., 2001) or semi-Markov conditional random fields (Sarawagi and Cohen, 2004) have difficulties coping with these issues. While approaches on addressing these issues exist, current algorithms typically suffer from high time complexity (Finkel and Manning, 2009) and are therefore difficult to scale to large datasets. On the other hand, the problem of designing efficient and scalable models for mention extraction and classification from natural language texts becomes increasingly important in this era where a large volume of textual data is becoming available on the Web every day – users need systems which are able to scale to extremely large datasets to support efficient semantic analysis for timely decison-making.

In this paper, we tackle the above-mentioned issue by introducing a novel model for joint mention extraction and classification. We make the following major contributions in this work:

- We propose a model that is able to effectively

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 857–867,
Lisbon, Portugal, 17-21 September 2015.

handle overlapping mentions with unbounded lengths.

- The learning and inference algorithms of our proposed model have a time complexity that is linear in the number of words in the input sentence and also linear in the number of possible semantic classes/types, making our model scalable to extremely large datasets.
- Our model can additionally capture mentions' head information in a joint manner under the same time complexity.

Our system and code are available for download from `http://statnlp.org/research/ie/`.

2 Related Work

Existing work has been largely focused on the task of named entity recognition and classification (NERC). The survey of (Nadeau and Sekine, 2007) is a comprehensive study of this topic.

Most prior work took a supervised learning approach. Zhou and Su (2002) presented a system for recognizing named entities using an HMM-based approach. Florian et al. (2003) presented a system for named entity recognition by combining different classifiers. McDonald and Pereira (2005) used conditional random fields for extracting gene and protein mentions from biomedical texts. Ratinov and Roth (2009) presented a systematic analysis over several issues related to the design of a named entity recognition and classification system where issues such as chunk representations and the choice of inference algorithms were discussed. Researchers also looked into semi-supervised and unsupervised approaches for such a task (Cucchiarelli and Velardi, 2001; Etzioni et al., 2005). Additional efforts on addressing the NERC problem under a multilingual or cross lingual setting also exist (Florian et al., 2004; Che et al., 2013; Wang et al., 2013).

As pointed out by Finkel and Manning (2009), named entities are often nested. This fact was often ignored by the community largely due to technical reasons. They therefore proposed to use a constituency parser with a $O(n^3)$ time complexity (n is the number of words in the input sentence) to handle nested entities, and showed its effectiveness across several datasets. Alex et al. (2007) also presented several approaches by building models on top of linear-chain conditional random fields for recognizing nested entities in biomedical texts. Hoffmann et al. (2011) looked into a separate issue, which is to identify overlapping *relations* amongst entities.

Named entity recognition and classification still remains a popular topic in the field of statistical natural language processing. Ritter et al. (2011) looked into recognizing entities from social media data that involves informal and potentially noisy texts. Pasupat and Liang (2014) looked into the issue of zero-shot entity extraction from Web pages with natural language queries where minimal supervision was used. Neelakantan and Collins (2014) looked into the problem of automatically constructing dictionaries with minimal supervision for improved named entity extraction. Li and Ji (2014) presented an approach to perform the task of extraction of mentions and their relations in a joint and incremental manner.

3 Approach

3.1 Mentions and Their Combinations

Typically, a mention that appears in a natural language sentence consists of a contiguous sequence of natural language words. Consider a sentence that consists of n words where each word is indexed with its position in the sentence. A mention m can be uniquely represented with a tuple $\langle b_m, e_m, \tau \rangle$, where b_m and e_m are the indices of the first and last word of the mention, respectively, and τ is its semantic class (type).

We can see that for a given sentence consisting of n words, there are altogether $tn(n+1)/2$ possible different mention candidates, where t is the total number of possible mention types. Now, for each such candidate in the given sentence, it can be either a mention, or not a mention. This leads to a total number of $2^{tn(n+1)/2}$ possible mention combinations. This number is prohibitively large even for small values of n and t, which prevents us from exhaustively enumerating all of them during learning and inference.

One approach to performing inference over such a large space is to introduce compact representations that are able to encode exponentially many mentions that would enable tractable inference algorithms to be employed. We discuss in the next section our novel mention hypergraph representation proposed for such a purpose.

3.2 Mention Hypergraphs

Central to our approach is the introduction of the novel *mention hypergraphs* that allow us to

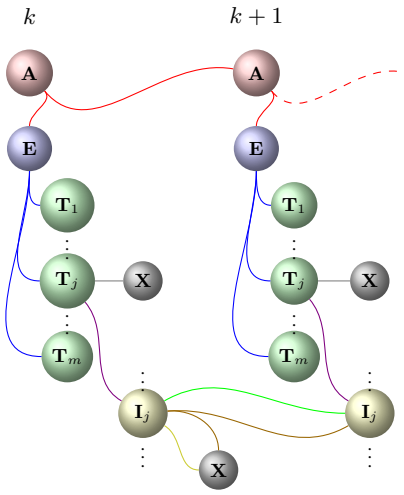


Figure 1: The (partial) hypergraph for representing all possible combinations of mention occurrences. Links that belong to the same hyperedge are highlighted with the same color, and different hyperedges are highlighted with different colors, *e.g.*, the green link that connects two **I** nodes forms a single hyperedge, while the two brown links that connect two **I** nodes and one **X** node form a separate single hyperedge.

compactly represent exponentially many possible combinations of potentially overlapping, length-unbounded mentions of different types.

A *hypergraph* is a generalization of a conventional graph, whose edges (*a.k.a. hyperedges*) can connect two or more nodes. In this work, we consider a special class of hypergraphs, where each hyperedge consists of a designated parent node and an *ordered* list of child nodes. Hypergraphs have also been used in other fields, such as syntactic parsing (Klein and Manning, 2001), semantic parsing (Lu, 2015) and machine translation (Cmejrek et al., 2013).

Our mention hypergraphs consist of five types of nodes which are used to compactly represent many mentions of different semantic types and boundaries, namely, **A** nodes, **E** nodes, **T** nodes, **I** nodes, and **X** nodes. A partial mention hypergraph is depicted in Figure 1. We describe the definition of each type of nodes next.

- **A nodes.** These nodes are used to sequentially arrange mentions with different left boundaries. Specifically, each **A** node at position k (the k-th word), or $\mathbf{A}^k$, is used to compactly represent all such mentions in the sentence whose left boundaries are *exactly at or strictly after* k.
- **E nodes.** The node $\mathbf{E}^k$ is used to compactly represent all possible mentions (possibly of length zero) whose left boundaries are *exactly at* the current position k.
- **T nodes.** The node $\mathbf{T}^k_j$ is used to compactly represent all mentions (possibly of length zero) whose left boundaries are *exactly at* position k, and *have the mention type* j.
- **I nodes.** The node $\mathbf{I}^k_j$ is used to compactly represent all *incomplete* mentions which contain the current word at position k as part of the mention, and have the mention type j.
- **X nodes.** These are the "terminal" nodes indicating the completion of a path. No additional node will be attached to such nodes as a child.

There are also various hyperedges that connect different nodes in the mention hypergraph. We use $\langle \alpha \leftarrow \beta_1, \ldots, \beta_n \rangle$ to denote a hyperedge which connects a parent node α and child nodes $\beta_1, \ldots, \beta_n$. Each hyperedge essentially provides one possible way of re-expressing the semantics conveyed by the parent node using the child nodes. For example, as shown in Figure 1, the hyperedge connecting the parent node $\mathbf{A}^k$ and the child nodes $\mathbf{E}^k, \mathbf{A}^{k+1}$ explains the fact that any mention covered by $\mathbf{A}^k$ either has a left boundary that is "*exactly at* k" ($\mathbf{E}^k$), or "*exactly at* or *strictly after* $k+1$" ($\mathbf{A}^{k+1}$).

Similarly, for each **I** node, there exist 3 hyperedges that connect it to other child nodes. The top hyperedge (in green) encodes the fact that the current word appears in the middle of a mention; the bottom hyperedge (in yellow) encodes the fact that the current word appears in a mention as the last word; the middle hyperedge (in brown) encodes the fact that both cases can occur at the same time (*i.e.*, the current word belongs to multiple overlapping mentions of the same type). We have the following theorem:

Theorem 3.1 *Any combination of mentions in a sentence can be represented with exactly one sub-hypergraph of the complete mention hypergraph.*

Proof For each mention, there exists a unique path in the mention hypergraph to represent it. For any combination of mentions, there exist unique paths in the mention hypergraph to represent such a combination. These paths altogether form a unique sub-hypergraph of the original hypergraph.

For example, consider the following sentence: "*he also talked with the egyptian president* ." This sentence contains three mentions. The first is "*he*" with type `PER`, the second is "*the egyptian president*" with type `PER`, and the third mention is "*egyptian*" with type `GPE`. Figure 2 gives the sub-hypergraph structure showing how these mentions

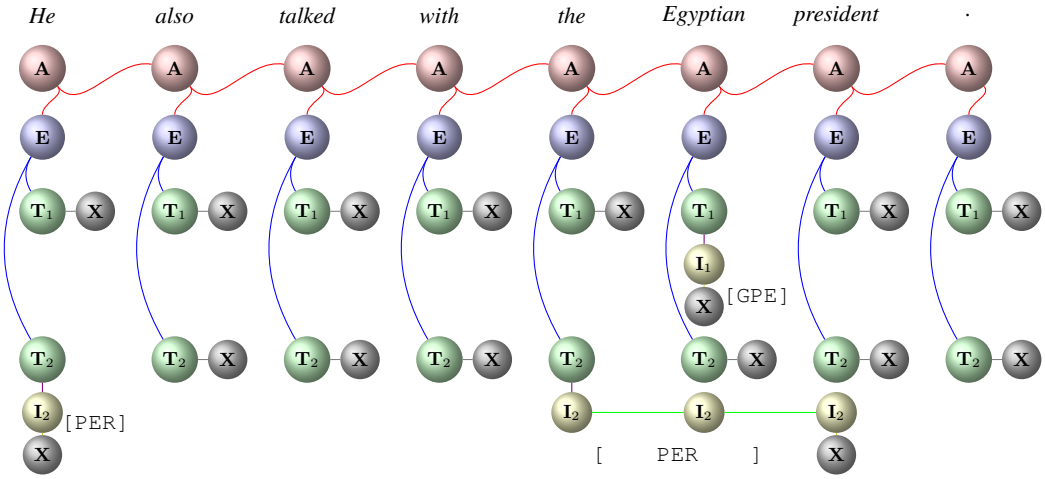


Figure 2: An example sub-hypergraph structure for jointly representing all the three mentions that appear in the sentence "*He also talked with the Egyptian president* ." For simplicity and the ease of illustration, we assume there are only two possible mention types: PER and GPE.

are jointly represented. The mention hypergraph defined over the input sentence contains exponentially many such sub-hypergraph structures.

We note that the converse of Theorem 3.1 is not true. In certain cases, it is possible for two different overlapping mention combinations to share the same mention hypergraph.

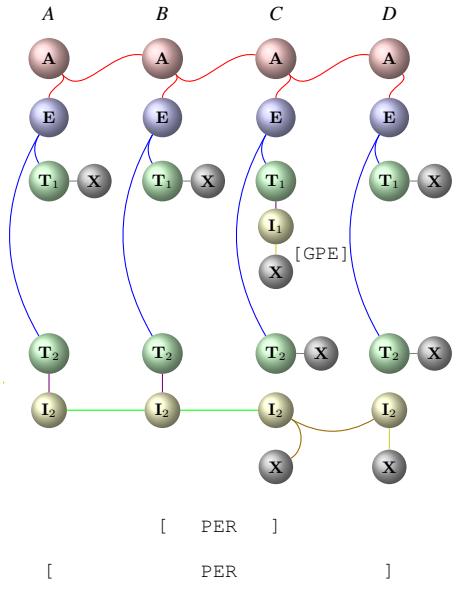


Figure 3: An example illustrating the converse of Theorem 3.1 is not true.

For example, consider a toy example sentence *A B C D* shown in Figure 3, both *B C* and *A B C D* are mentions of the same type PER (*i.e.* one is strictly contained by the other. We call such combinations type-I combinations). The above sub-hypergraph shows how to encode such a combination. However, if both *A B C* and *B C D* are mentions of the same type PER (*i.e.*, two mentions overlap but no one is contained by the other. We call such combinations type-II combinations), such a combination shares the same representation as the above sub-hypergraph. Note that such an ambiguity happens only when two overlapping mentions have the same type, and one mention is strictly contained by the other and their boundaries are all different. In practice, however, we found that in the two datasets that we used for evaluations, if two mentions overlap with one another, they almost always form a type-I combination, and type-II combinations are very rare. Empirically, as we will see later in our experiments, our model is effective in handling overlapping mentions.

3.3 Log-Linear Modeling

Following the conditional random fields (Lafferty et al., 2001), we adopted a log-linear approach for such a joint mention extraction and typing task. Specifically, for a given input sentence $\mathbf{x}$, the probability of predicting a possible output $\mathbf{y}$ (a mention sub-hypergraph that represents a particular combination of mentions) is given as follows:

$$p(\mathbf{y}|\mathbf{x}) = \frac{\exp(\mathbf{w}^T\mathbf{f}(\mathbf{x},\mathbf{y}))}{\sum_{\mathbf{y}'}\exp(\mathbf{w}^T\mathbf{f}(\mathbf{x},\mathbf{y}'))} \quad (1)$$

where $\mathbf{f}(\mathbf{x},\mathbf{y})$ is the feature vector defined over the input-output pair $(\mathbf{x},\mathbf{y})$, and the weight vector $\mathbf{w}$ gives the parameters of the model.

Our objective is to minimize the regularized negative joint log-likelihood of the dataset:

$$\mathcal{L}(\mathbf{w}) = \sum_i \log \sum_{\mathbf{y}'} \exp(\mathbf{w}^T\mathbf{f}(\mathbf{x_i},\mathbf{y}')) - \sum_i \mathbf{w}^T\mathbf{f}(\mathbf{x_i},\mathbf{y_i}) + \lambda\mathbf{w}^T\mathbf{w} \quad (2)$$

where $(\mathbf{x_i}, \mathbf{y_i})$ refers to the i-th training instance, and the last term is a L_2 regularization term with λ being a positive scalar (fixed to 0.01 in this work).

The gradient of the above objective function is:

$$\frac{\partial \mathcal{L}(\mathbf{w})}{\partial w_k} = \sum_i \mathbf{E}_{p(\mathbf{y}'|\mathbf{x_i})}[f_k(\mathbf{x_i}, \mathbf{y}')] - \sum_i f_k(\mathbf{x_i}, \mathbf{y_i}) + 2\lambda w_k \quad (3)$$

where w_k is the weight of the k-th feature f_k.

We note that unlike many recent latent-variable approaches to structured prediction (Petrov and Klein, 2007; Blunsom et al., 2008), we are able to represent each of our outputs $\mathbf{y}$ with a single fully-observed structure. Thus, our objective function essentially defines a standard regularized softmax regression model, and is therefore convex (Boyd and Vandenberghe, 2004), where a global optimum can be found.

The objective function defined in Equation 2 can be optimized with standard gradient-based methods. We used L-BFGS (Liu and Nocedal, 1989) as our optimization method.

3.4 Algorithms

In order to solve the optimization problem described above, one needs to compute the values of the gradient scores in Equation 3. Computation of the second and third terms in this equation is straightforward. The first term in Equation 3 involves the computation of an expectation of feature values over all possible mention combinations for a given input sentence. Following classic dynamic programming algorithms used in graphical models, we develop analogous efficient dynamic programming algorithms that work on hypergraphs and generalize the conventional forward-backward/inside-outside algorithm to efficiently compute such values.

Time Complexity At each time step k, we need to compute scores for m $\mathbf{I}$ nodes, m $\mathbf{T}$ nodes, 1 $\mathbf{E}$ node, and 1 $\mathbf{A}$ node. Hence, the overall time complexity for our algorithm is in $O(mn)$ (assuming computation of the feature scores at each node involves a constant time), where m is the total number of possible mention types, and n is the total number of words in the given sentence. [1]

[1] Note that the time complexity for the linear chain CRF is in $O(m^2 n)$ due to their first-order assumption.

3.5 Features

The features that we use are inspired by the work of (Carreras et al., 2002). Specifically, we consider the following features defined over the inputs:

- Words (and POS tags, if available) that appear around the current word (with position information), with a window of size 3.
- Word n-grams (and POS n-grams, if available) that contain the current word (with position information), for $n = 2, 3, 4$.
- Bag of words around the current word, with a window of size 5.
- Word pattern features [2].

Note that these are the indicator functions defined over the inputs. The final set of features are defined over $(\mathbf{x}, \mathbf{y})$ tuples, which is obtained as a cross-product between the above indicator functions and the following indicator function:

- The type of the node (such as $\mathbf{T}$ or $\mathbf{I}$).

In addition, we also introduce the following feature defined over the output structure only:

- The number of such hyperedges that exactly connect one $\mathbf{T}$ node and one $\mathbf{I}$ node.

We call this feature *mention penalty*. This feature learns a global preference of the number of mentions that should appear in any input sentence.

3.6 Joint Modeling of Mention Heads

One additional assumption for the mention extraction and typing task is that each mention comes with a *head*. A head is strictly a substring of the mention and provides important information about the mention. It is possible to extend our model to support joint modeling of mention heads, while still maintaining the same time complexity.

Due to space limitations, we could only give a relatively brief description of this extension in this section. The idea is to replace the $\mathbf{I}$ nodes with three different types of nodes, namely $\mathbf{I}_j$–$\mathbf{B}$ nodes (used to represent words that appear within a mention of type j and *before* its head), $\mathbf{I}_j$–$\mathbf{W}$ nodes (used to represent words that appear *within* the head of a mention of type j), and $\mathbf{I}_j$–$\mathbf{A}$ nodes (used to represent words that appear within a mention of type j and *after* its head). The hyperedges also need to be established accordingly in order to properly model all possible mention and head

[2] *all-caps, all-digits, all-alphanumeric, contains-digits, contains-dots, contains-hyphen, initial-caps, lonely-initial, punctuaion-mark, roman-number, single-character, URL.*

	ACE2004			ACE2005		
	TRAIN	DEV	TEST	TRAIN	DEV	TEST
Documents	356	41	46	370	43	51
Sentences	6,799	829	879	7,336	958	1,047
with o.l. mentions	2,683 (39%)	293 (35%)	373 (42%)	2,683 (37%)	340 (35%)	330 (32%)
Mentions	22,207	2,511	3,031	24,687	3,217	3,027
length > 6	1,439 (6%)	179 (7%)	199 (7%)	1,343 (5%)	148 (5%)	160 (6%)
max length	57	35	43	49	30	27

Table 1: Corpora statistics for the ACE2004 and ACE2005 datasets.

	ACE2004						ACE2005					
	DEV			TEST			DEV			TEST		
	P	R	F	P	R	F	P	R	F	P	R	F
CRF (BIO)	69.6	42.8	53.0	70.0	40.3	51.2	69.8	45.3	55.0	67.6	43.7	53.1
CRF (BILOU)	70.7	42.6	53.1	71.8	40.8	52.1	71.1	45.5	55.5	69.5	44.5	54.2
CRF (CC)	77.9	49.1	60.2	78.4	46.4	58.3	76.8	52.0	62.0	74.8	49.1	59.3
Semi-CRF (c=6)	75.4	44.1	55.6	76.1	41.4	53.6	75.3	48.5	59.0	72.8	45.0	55.6
Semi-CRF (c=∞)	66.8	44.8	53.7	66.7	42.0	51.5	69.7	48.9	57.5	67.5	46.1	54.8
MH	**79.6**	50.0	61.4	**79.2**	46.8	58.9	**79.3**	50.6	61.8	**76.9**	47.7	58.9
MH (F)	70.0	**59.2**	**63.8**	70.0	**56.9**	**62.8**	67.5	**61.8**	**64.5**	66.3	**59.2**	**62.5**

Table 2: Results on ACE2004 and ACE2005. The last two rows give the results of this work.

combinations. Since in such a new hypergraph, at each time step, only a constant number (2) of additional nodes are involved, the time complexity for learning and inference with such a model remains the same, which is in $O(mn)$.

3.7 Optimization of F measure

One standard evaluation metric for named entity recognition is the F (F_1) measure. In our task, the F measure is defined as the harmonic mean of the precision (P) and recall (R) scores, where precision is the ratio between the number of correctly predicted mentions and the total number of predicted mentions, and recall is the ratio between the number of correctly predicted mentions and the total number of gold mentions. We will also adopt these metrics in our evaluations later. Unfortunately, the model only optimizes its objective function defined in Equation 2, which is the negative (regularized) joint log-likelihood. Previous work showed it was possible to optimize the F measure in a log-linear model (Suzuki et al., 2006). Culotta and McCallum (2004) also proposed a method for optimizing information extraction performance based on confidence estimation. Their work is based on linear-chain CRF and estimate the confidence of extracted fields based on marginal probabilities. The technique is not directly applicable to our task where a hypergraph representation is used to encode overlapping mentions. In this work, we used a very simple and intuitive technique for optimizing the F measure. The idea is to further tune the weight of a single parameter – *mention penalty* based on the development set, after the training process completes. This is based on the observation that by increasing the value of the mention penalty, we are essentially forcing our model to predict more mentions. Therefore the recall is a monotonic function with respect to the mention penalty. Based on this fact, we use a simple search algorithm with a fixed step size (we set it to 0.01) to determine the optimal value of the modified mention penalty so that the F measure of the development set is optimized.

4 Experiments

In this section, we present empirical evaluations. Our main experiments were conducted on the standard ACE2004 and ACE2005 datasets which contain overlapping mentions. Two additional experiments on the GENIA and CONLL2003 dataset were also conducted .

4.1 Results on ACE

Our primary experiments were conducted based on the English portion of the ACE2004 dataset[3] and the ACE2005 dataset[4]. Following previous work, for ACE2004, we considered all documents from *arabic_treebank, bnews, chinese_treebank*, and *nwire*, and for ACE2005, we considered all documents from *bc, bn, nw*, and *wl* . We randomly split the documents for each dataset into three portions: 80% for training, 10% for development, and the remaining 10% for evaluations. The statistics of the datasets are summarized in Table 1[5]. We

[3]https://catalog.ldc.upenn.edu/LDC2005T09

[4]https://catalog.ldc.upenn.edu/LDC2006T06

[5]Exact train/dev/test splits information can be found on http://statnlp.org/research/ie/.

can observe that overlapping mentions are common – over 30% of the sentences contain overlapping mentions (see row 3 of the table). Mentions can also be very long – over 5% of the mentions consist of more than 6 words, and the longest mention consists of 57 words.

We compared our system's performance with those of several baseline approaches. We first built two simple baseline approaches based on sequence labelling models using the conditional random fields (CRFs). Such approaches can not handle overlapping mentions. To train such models, whenever two mentions overlap with one another in the training set, we remove the mention that is shorter in length. Following (Ratinov and Roth, 2009), we considered the BIO (Begin, Inside, Outside) approach and the BILOU (Begin, Inside, Last, Outside, Unit) approach for designing the output labels. Results show the BILOU approach yields better results. Similar observations were reported in Ratinov and Roth (2009).

In the work of (Alex et al., 2007), the authors proposed several approaches for building models to handle nested named entities in biomedical texts. Their best results were obtained from a *cascaded* approach where they built one model for each named entity class. Outputs from one model can then served as the inputs to the next model for predicting the named entity class of a different type. One fundamental limitation of such an approach is that it being unable to handle overlapping mentions of the same type. Nevertheless, this approach worked very well on both datasets. The results are shown in the row of "CRF (CC)".[6]

Another class of models that is often used in information extraction are the semi-Markov conditional random fields (semi-CRFs) (Sarawagi and Cohen, 2004). Semi-CRF models are able to capture the non-Markovian properties of mentions. However, they are unable to handle nested or overlapping mentions. We thus used the same method as discussed above to exclude certain mentions for training. Such semi-CRF models typically assume there is a length restriction for the mentions – each mention can consist of up to c words – in order to scale linearly. When such a restriction is lifted, the time complexity of such models becomes quadratic in the number of words in the input sentence. We train two models: one with a length restriction, where $c = 6$, and the other without a length restriction ($c = \infty$). For features defined over the inputs, besides the Markovian features described in Sec 3.5, we also used the surface forms of complete mention spans as features. The results of these two models are reported in the fourth and fifth row of Table 2, respectively. Interestingly, imposing the length restriction appears to be helpful for precision, and as a result it makes a positive contribution towards the final F measure.

[6]For all such linear chain CRF-related experiments, we used the CRF++ toolkit (https://code.google.com/p/crfpp/) with L-BFGS, which gives us the most competitive results over several different CRF implementations (see: http://www.chokkan.org/software/crfsuite/benchmark.html).

	7 TYPES		14 TYPES		28 TYPES	
	#f	w/s	#f	w/s	#f	w/s
CRF	3.6M	1219	13.6M	305	51.9M	76
MH	4.2M	**1532**	8.4M	**733**	16.9M	**430**

Table 3: The decoding time and the number of features change as we increase the number of possible types. (#f: number of features created (in millions). w/s: number of words processed per second.) Experiments are conducted on the ACE 2004 dataset.

Our basic model (**MH**: *mention hypergraph*) that optimizes the negative joint log likelihood is able to obtain the best precision across these two datasets. When the model is further augmented with the F measure optimization step described in Sec 3.7 (**MH** (F)) it consistently yields the best results in terms of both recall score and F measure across these two datasets.

4.1.1 Running Time

We also conducted controlled experiments to report the actual execution time of our model and make a comparison with the linear-chain CRF model (BILOU approach). The experiments are all conducted on the ACE2004 dataset on the same machine. To make a proper comparison here, we implemented the linear-chain CRF model using Java (the same language is used when implementing our model), and employed the same data structures for creating features as well as the same learning and inference routines used by our mention hypergraph model.

To understand how the features and speed change as we increase the number of mention types (*i.e.,* semantic types), we also conducted experiments where we increase the number of possible mention types. Specifically, we created subtypes from each original type annotated in the dataset. For example, we randomly replaced the type "GPE" by sub-types "GPE1" or "GPE2" in the dataset. This gave us 14 different mention types. Similarly, we could randomly replace the type "GPE" by sub-types "GPE1" – "GPE4", re-

	ACE2004						ACE2005					
	DEV			TEST			DEV			TEST		
	P	R	F	P	R	F	P	R	F	P	R	F
CRF (CC-S)	57.0	35.9	44.1	52.7	31.2	39.2	56.5	38.3	45.6	54.2	35.5	42.9
CRF (CC-F)	51.5	32.5	39.9	47.4	28.0	35.2	53.3	36.1	43.1	51.3	33.6	40.6
CRF (CC-L)	64.4	40.6	49.9	61.6	36.4	45.8	66.6	45.1	53.8	65.3	42.8	51.7
CRF (CC-CC)	63.6	40.5	49.5	60.8	36.1	45.3	65.9	44.8	53.3	64.2	42.0	50.7
MH (L)	66.7	41.9	51.5	64.7	38.2	48.1	70.5	45.0	55.0	69.2	43.0	53.0
MH (Joint)	**78.6**	47.4	59.1	**79.0**	44.1	56.6	**79.2**	48.7	60.3	**70.1**	45.1	54.8
MH (Joint F)	73.9	**52.4**	**61.3**	74.4	**50.0**	**59.8**	70.2	**57.4**	**63.2**	63.4	**53.8**	**58.3**

Table 4: Results on joint mention boundary, type, and head prediction on ACE2004 and ACE2005.

sulting in 28 different mention types in total. Our purpose of doing so is to understand how the models behave when the number of possible mention types becomes large. We found that training on the entire training set of ACE2004 using the linear-chain CRF model with a large number of mention types was very expensive due to the extremely large number of features involved. We instead trained the models on the development set and presented decoding time on the test set.

Table 3 shows the results. We empirically captured the relationship between the speed of each system (average number of words processed per second) and the number of mention types. Specifically, we found that as we linearly increased the number of mention types, for the linear-chain CRF model, the number of features grew quadratically and the speed dropped quadratically, whereas for our model, the number of features grew linearly and the speed dropped linearly. This indicates that our model is more scalable to large, practical datasets with a large number of fine-grained mention types.

4.1.2 Joint Modeling of Heads

We also conducted experiments on these two datasets for the task of joint modeling of mention boundaries, types and heads. We used the same training and tuning methodology for optimizing the F measure. In such experiments, we adopted a very strict evaluation criterion: a predicted mention is regarded as correct iff and only if its boundaries, type and head all exactly match those of the gold standard.

We compared our system's results with those of several baseline approaches based on CRF where the cascaded BILOU approach described above was always used. Specifically, we considered approaches that always regarded the complete span (CC-S), the first word (CC-F), and the last word (CC-L) as the predicted mention's head, respectively. We also considered a cascaded approach (CC-CC) where we first predicted mentions, and then predicted their heads by following a similar approach used for predicting overlapping mentions discussed above. The first four rows of Table 4 give the results of these baseline approaches. We can observe that always predicting the last word as the head gives the best performance. Inspired by this, we performed a simple approach by training a model presented in the previous section without considering head information. When making predictions, we always regarded the last word of each predicted mention as its head. The results for such an approach are given in the fifth row of Table 4. The sixth row shows the results obtained by optimizing our model's objective function. The last row gives the results obtained by tuning the mention penalty based on the development set. As seen, our joint models significantly outperformed all those baseline approaches. We are not aware of any prior work in the literature that performs joint modeling of mention boundaries, types, and heads.

4.2 Additional Experiments

We also additionally evaluated on the GENIA dataset (v3.02) whose focus was on biomedical related named entity recognition and classification, where the entities may overlap with one another. Furthermore, to see how our model works on datasets where mentions do not overlap with one another, we also conducted evaluations on the standard CONLL2003 NER dataset.

4.2.1 Results on GENIA

We followed the description of Finkel and Manning (2009) to set up our experiments on the GENIA dataset. Specifically, we used the first 90% of the sentences as the training data and the remaining 10% as the evaluation data. We also adhered to the paper's prescription of collapsing all *DNA* subtypes into *DNA*; *RNA* subtypes into *RNA*; and all *protein* subtypes into *protein*. We kept *cell line* and *cell type*, and removed all other entities.

To optimize the F measure, we further split the

	P	R	F
Semi-CRF	76.2	61.7	68.2
F & M (2009)	75.4	65.9	70.3
MH (F)	72.5	65.2	68.7

Table 5: Results on the GENIA dataset

training set into two portions. We trained a model using the first 90% of the training data, and used the remaining 10% for development. For features, no POS and no bag-of-words features are used.

We compared our model's performance with that of a model based on a constituency parser proposed by (Finkel and Manning, 2009), as well as the semi-CRF model reported there. The results are shown in Table 5. Our model yields a better F measure than the semi-CRF model, but gives a lower performance than the model of (Finkel and Manning, 2009). We note that, however, these results are not directly comparable. Specifically, both of these two previous models relied on an additional 200 million words from PubMed abstracts to learn word clusters as additional features, which we do not have access to.

One distinctive advantage of our model is the efficiency and scalability. The model of (Finkel and Manning, 2009) had a time complexity that is cubic in the number of words in the input sentence. In contrast, our model scales linearly as the length of the input sentence increases. [7]

4.2.2 Results on CONLL2003

To understand how well our model works on datasets where mentions or entities do not overlap with one another, we conducted additional experiments on the standard dataset used in the CONLL 2003 shared task (Tjong Kim Sang and De Meulder, 2003), where the named entities strictly do not overlap with one another. We compared our system's performance against that of a baseline version of the state-of-the-art Illinois NER system (Ratinov and Roth, 2009). Their system performed sequential prediction over the input words and adopted the BILOU approach. Their full model also incorporates external knowledge resources (*e.g.*, gazetteers and word class).

In order to make a proper comparison with the baseline version of their model, besides the general features we mentioned earlier, we also followed (Ratinov and Roth, 2009) in incorporating word's prefixes and suffixes (of length up to 5) as features, and normalized words referring to months, dates and numbers. Table 6 shows that our system gives an F measure that is comparable to that of the baseline version of their system, where no external resources are used.

	DEV			TEST		
	P	R	F	P	R	F
Illinois (b)	-	-	89.3	-	-	83.7
MH	94.7	83.0	88.5	91.4	76.5	83.3
MH (F)	91.4	86.7	89.2	87.3	80.7	83.8

Table 6: Results on the CONLL2003. Illinois (b): baseline version of (Ratinov and Roth, 2009).

This additional experiment showed that while our model is designed for handling more realistic scenarios where mentions can overlap, it yields a performance competitive to a state-of-the-art system which only handles datasets with non-overlapping mentions.

5 Conclusions

In this work, we have introduced a novel model for the task of joint modeling of mention boundaries, types, as well as their heads. Unlike many previous research efforts for mention extraction and classification, our novel *mention hypergraph* representations for compactly representing exponentially many possible mentions enables a mention's boundaries, type and head information to be jointly learned in a single framework. The model scales linearly with respect to the number of words in the input sentence, and performs exact learning where a unique global optimum can be found. Empirically, we have demonstrated the effectiveness of such a model across several standard datasets.

Future work include explorations of efficient algorithms for other information extraction tasks, such as joint mention and relation extraction (Li and Ji, 2014) and event extraction (Li et al., 2013). Our system and code can be downloaded from `http://statnlp.org/research/ie/`.

Acknowledgements

We would like to thank Kian Ming A. Chai, Hai Leong Chieu and the three anonymous reviewers for their comments on this work. This work is supported by Temasek Lab of Singapore University of Technology and Design project IGDSS1403011 and IGDST1403013, and is partly supported by DARPA (under agreement number FA8750-13-2-0008).

[7] In our experiments, for this dataset our model tagged over 5,000 words/second. In (Finkel and Manning, 2009), the authors mentioned that their model tagged about 38 words/second, and the semi-CRF model tagged about 45 words/second. However, we note these numbers are not directly comparable due to the advancement of CPU speed.

References

Beatrice Alex, Barry Haddow, and Claire Grover. 2007. Recognising nested named entities in biomedical text. In *BioNLP*, pages 65–72. Association for Computational Linguistics.

Phil Blunsom, Trevor Cohn, and Miles Osborne. 2008. A Discriminative Latent Variable Model for Statistical Machine Translation. In *ACL*, pages 200–208.

Stephen Boyd and Lieven Vandenberghe. 2004. *Convex optimization*. Cambridge university press.

Xavier Carreras, Lluis Marquez, and Lluís Padró. 2002. Named entity extraction using adaboost. *CONLL*, pages 167–170.

Kai-Wei Chang, Rajhans Samdani, and Dan Roth. 2013. A Constrained Latent Variable Model for Coreference Resolution. In *EMNLP*, pages 601–612.

Wanxiang Che, Mengqiu Wang, Christopher D Manning, and Ting Liu. 2013. Named Entity Recognition with Bilingual Constraints. In *NAACL-HLT*, pages 52–62.

Martin Cmejrek, Haitao Mi, and Bowen Zhou. 2013. Flexible and efficient hypergraph interactions for joint hierarchical and forest-to-string decoding. In *EMNLP*, pages 545–555.

Alessandro Cucchiarelli and Paola Velardi. 2001. Unsupervised named entity recognition using syntactic and semantic contextual evidence. *Computational Linguistics*, 27(1):123–131.

Aron Culotta and Andrew McCallum. 2004. Confidence estimation for information extraction. In *HLT-NAACL*, pages 109–112.

Oren Etzioni, Michael Cafarella, Doug Downey, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S Weld, and Alexander Yates. 2005. Unsupervised named-entity extraction from the web: An experimental study. *Artificial Intelligence*, 165(1):91–134.

Jenny Rose Finkel and Christopher D Manning. 2009. Nested named entity recognition. In *EMNLP*, pages 141–150. Association for Computational Linguistics.

Radu Florian, Abe Ittycheriah, Hongyan Jing, and Tong Zhang. 2003. Named entity recognition through classifier combination. In *CONLL*, pages 168–171. Association for Computational Linguistics.

Radu Florian, Hany Hassan, Abraham Ittycheriah, Hongyan Jing, Nanda Kambhatla, Xiaoqiang Luo, H Nicolov, and Salim Roukos. 2004. A statistical model for multilingual entity detection and tracking. In *HLT-NAACL*, pages 1–8. DTIC Document.

Stephen Guo, Ming-Wei Chang, and Emre Kiciman. 2013. To Link or Not to Link? A Study on End-to-End Tweet Entity Linking. In *NAACL-HLT*, pages 1020–1030.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *ACL*, pages 541–550. Association for Computational Linguistics.

Dan Klein and Christopher D. Manning. 2001. Parsing and hypergraphs. In *IWPT*, pages 123–134.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*.

Qi Li and Heng Ji. 2014. Incremental Joint Extraction of Entity Mentions and Relations. In *ACL*, pages 402–412.

Qi Li, Heng Ji, and Liang Huang. 2013. Joint Event Extraction via Structured Prediction with Global Features. In *ACL*, pages 73–82.

Dong C Liu and Jorge Nocedal. 1989. On the limited memory BFGS method for large scale optimization. *Mathematical programming*, 45(1-3):503–528.

Wei Lu. 2015. Constrained semantic forests for improved discriminative semantic parsing. In *ACL*.

Ryan McDonald and Fernando Pereira. 2005. Identifying gene and protein mentions in text using conditional random fields. *BMC bioinformatics*, 6(S6).

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

David Nadeau and Satoshi Sekine. 2007. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26.

Arvind Neelakantan and Michael Collins. 2014. Learning Dictionaries for Named Entity Recognition using Minimal Supervision. In *EACL*, pages 452–461.

Panupong Pasupat and Percy Liang. 2014. Zero-shot Entity Extraction from Web Pages. In *ACL*, pages 391–401.

Slav Petrov and Dan Klein. 2007. Discriminative log-linear grammars with latent variables. In *NIPS*, pages 1153–1160.

Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *CONLL*, pages 147–155. Association for Computational Linguistics.

Alan Ritter, Sam Clark, Mausam, and Oren Etzioni. 2011. Named entity recognition in tweets: an experimental study. In *EMNLP*, pages 1524–1534. Association for Computational Linguistics.

Sunita Sarawagi and William W Cohen. 2004. Semi-markov conditional random fields for information extraction. In *NIPS*, pages 1185–1192.

Jun Suzuki, Erik McDermott, and Hideki Isozaki. 2006. Training conditional random fields with multivariate evaluation measures. In *COLING/AC*, pages 217–224.

Erik F Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *CONLL*, pages 142–147. Association for Computational Linguistics.

Mengqiu Wang, Wanxiang Che, and Christopher D Manning. 2013. Joint Word Alignment and Bilingual Named Entity Recognition Using Dual Decomposition. In *ACL*, pages 1073–1082.

GuoDong Zhou and Jian Su. 2002. Named entity recognition using an hmm-based chunk tagger. In *ACL*, pages 473–480.

FINET: Context-Aware Fine-Grained Named Entity Typing

Luciano Del Corro* **Abdalghani Abujabal*** **Rainer Gemulla†** **Gerhard Weikum***
* Max-Planck-Institut für Informatik, Saarbrücken, Germany
{delcorro, abujabal, weikum}@mpi-inf.mpg.de
† Universität Mannheim, Mannheim, Germany
rgemulla@uni-mannheim.de

Abstract

We propose FINET, a system for detecting the types of named entities in short inputs—such as sentences or tweets—with respect to WordNet's super fine-grained type system. FINET generates candidate types using a sequence of multiple extractors, ranging from explicitly mentioned types to implicit types, and subsequently selects the most appropriate using ideas from word-sense disambiguation. FINET combats data scarcity and noise from existing systems: It does not rely on supervision in its extractors and generates training data for type selection from WordNet and other resources. FINET supports the most fine-grained type system so far, including types with no annotated training data. Our experiments indicate that FINET outperforms state-of-the-art methods in terms of recall, precision, and granularity of extracted types.

1 Introduction

Named entity typing (NET) is the task of detecting the type(s) of a named entity in context. For instance, given "John plays guitar on the stage", our goal is to infer that "John" is a *guitarist* or a *musician* and a *person*. We propose FINET, a system for detecting the types of named entities in short inputs—such as sentences or tweets—with respect to WordNet's super fine-grained type system (16k types of organizations, persons and locations).

Named entity typing is a fundamental building block for many natural-language processing tasks. NET is at the heart of information extraction methods for finding types for entities in a knowledge base[1] (KB) (Mitchell et al., 2015). Likewise, NET aids named entity disambiguation by reducing the candidate space for a given entity mention. Entity types are an important resource for entity-based retrieval or aggregation tasks, such as semantic search (Hoffart et al., 2014) or question answering (Yahya et al., 2013). Finally, type information helps to increase the semantic content of syntactic patterns (Nakashole et al., 2012) or in open information extraction (Lin et al., 2012).

The extraction of explicit types has been studied in the literature, most prominently in the context of taxonomy induction (Snow et al., 2006). Explicit types occur, for example, in phrases such as "Steinmeier, the German Foreign Minister, [...]" or "Foreign Minister Steinmeier." These explicit types are often extracted via patterns, such as the well-known Hearst patterns (Hearst, 1992), and subsequently integrated into a taxonomy. Pattern-based methods often have high precision but low recall: Types are usually mentioned when a named entity is introduced or expected to be unknown to readers, but often are not explicitly stated. The NET problem differs from taxonomy induction in that (1) the type system is prespecified, (2) types are disambiguated, and (3) types are associated with each occurrence of named entity in context.

Our FINET system makes use of explicit type extractions whenever possible. But even when types are not explicitly mentioned, sentences may give clues to the correct type. These clues range from almost explicit to highly implicit. For example, in "John plays soccer", the type *soccer player* is almost explicit. The sentence "Pavano never even made it to the mound," however, only implicitly indicates that "Pavano" is a *baseball player*. A

[1] In this paper, we refer to WordNet as a type system and to a collection of entities and their types as a KB.

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 868–878,
Lisbon, Portugal, 17-21 September 2015.

key challenge in NET is to extract such implicit, context-aware types to improve recall.

One way to extract implicit types is by training a supervised extractor on labeled data, in which each entity is annotated with appropriate types. The key problem of this approach is that training data is scarce; this scarcity is amplified for fine-grained type systems. To address this problem, many existing systems generate training data by exploiting KBs as a resource (Yosef et al., 2012). A popular approach is to train an extractor on a corpus of sentences (e.g., on Wikipedia), in which each named entity is associated with all its types in a KB. The key problem with such an approach is that the so-obtained type information is oblivious to the context in which the entity was mentioned. For example, in the sentences "Klitschko is known for his powerful punches" and "Klitschko is the Mayor of Kiew," "Klitschko" will be associated with all its types, e.g., *boxer*, *politician* and *mayor*. As a consequence, the labels in the training data can be misleading and negatively affect the extractors. Moreover, such learned extractors are often biased towards prominent types but perform poorly on infrequent types, and they are generally problematic when types are correlated (e.g., most *presidents* are also *graduates* and *authors*).

FINET addresses the above problems by first generating a set of type candidates using multiple extractors and then selecting the most appropriate type(s). To generate candidates, we make use of a sequence of extractors that range from explicit to highly implicit. Implicit extractors are only used when more explicit extractors fail to produce a good type. Our extractors are based on patterns, mention text, and verbal phrases. To additionally extract highly implicit types, we use word vectors (Mikolov et al., 2013) trained on a large unlabeled corpus to determine the types of *similar* entities that appear in *similar* contexts. This extractor is comparable to KB methods discussed above, but is unsupervised, and takes as candidates the types frequent within the related entities and contexts.

After type candidates have been generated, the final step of FINET selects the types that best fit the context. In this step, we leverage previous work on word sense disambiguation (WSD) and resources such as WordNet glosses, and, if available, manually annotated training data.

FINET leverages ideas from existing systems and extends them by (1) handling short inputs (2) supporting a very fine-grained type hierarchy, and (3) producing types that focus on the entity context. Existing systems are unable to extract more than a couple of hundred types. Hyena (Yosef et al., 2012), the system with the most fine-grained type system so far, focuses only on a subset of 505 types from WordNet. Hyena lacks important types such as *president* or *businessman*, and includes *soccer player* but not *tennis player*. Instead of restricting types, FINET operates on the the entire WordNet hierarchy with more than 16k types for persons, organizations, and locations.

We evaluated FINET on a number of real-world datasets. Our results indicate that FINET significantly outperforms previous methods.

Extractor	Stopping Condition
Pattern-based (final)	Always stop
Pattern-based (non-final)	KB-lookup
Mention-based	KB-lookup
Verb-based	KB-lookup
Corpus-based	$\geqslant 50\%$ of score in $\leqslant 10$ types

Table 1: Extractors and their stopping conditions

2 Candidate Generation

In this phase, we collect candidate types for each entity. We first preprocess the input (Sec. 2.1) and then apply a (i) pattern-based extractor (Sec. 2.2), (ii) a mention-based extractor (Sec. 2.4), (iii) a verb-based extractor (Sec. 2.5), and (iv) a corpus-based extractor (Sec. 2.6). The extractors are ordered by decreasing degree of explicitness.

Each extractor has a *stopping condition*, which we check whenever the extractor produced at least one type. When the stopping condition is met, we directly proceed to the type selection phase. The reasoning behind this approach is to bias FINET towards the most explicit types. When the condition is not met, we enrich the set of candidate types of the extractor with their hypernyms; we expect types to be overly specific and want to allow the selection phase to be able to select a more general type. We then also run subsequent extractors. Tab. 1 displays a summary of the extractors and stopping conditions.

All so-found type candidates are passed to the candidate selection phase (Sec. 3).

2.1 Preprocessing

Preprocessing consists of 5 steps: (i) dependency parsing (Socher et al., 2013); (ii) co-reference (Recasens et al., 2013); (iii) named entity recogni-

tion (NER) (Finkel et al., 2005) with the detection of coarse-grained types (i.e., *person*, *organization*, *location*); (iv) clause identification (Del Corro and Gemulla, 2013); (v) word and multi-word expression recognition (Del Corro et al., 2014).

FINET restricts its candidates to the hyponmys of the coarse-grained (CG) type of the NER system. Named entities with the same CG type in a coordinating relation (e.g., "Messi and Ronaldo are soccer players") and identical mentions share the candidate set; the latter is reasonable in short input.

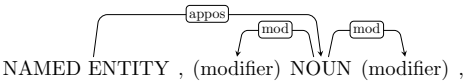
FINET extractors operate either on the sentence or the clause level. A clause is a part of a sentence that expresses a statement and is thus a suitable unit for automatic text processing (Del Corro and Gemulla, 2013). Finally, we identify multi-word explicit type mentions such as *Prime Minister* or *Secretary of Housing and Urban Development* (Del Corro et al., 2014).

2.2 Pattern-based extractor

Our pattern-based extractor targets explicit type mentions. These are commonly used to introduce entities when they first appear ("US President Barack Obama") or when their mention does not refer to the most prominent entity ("Barack Obama, father of the US President"). Following previous work (Hearst, 1992), we use a set of patterns to look for expressions that may refer named entity types. We refer to those expressions as *lexical types* (e.g., "father"). Once lexical types have been identified, we collect as candidate types the WordNet synsets to which they refer (e.g., ⟨*father*-1⟩, ..., ⟨*father*-8⟩, the eight senses of "father").

Our extractor makes use of both syntactic patterns, which operate on the dependency parse, and regular expression patterns, which operate on the text. Syntactic patterns are preferable in that they do not rely on continuous chunks of text and can skip non-relevant information. However, mistakes in the dependency parse may lower recall. To cope with this, we additionally include regular expressions for some syntactic patterns. Fig. 1 shows an example of a syntactic pattern and a related regular-expression pattern. Both produce lexical type "president" from "Barack Obama, president of the US," but only the syntactic pattern applies to "Barack Obama, the current US president."

Tab. 2 gives an overview of our patterns. Most of them also have a symmetric version (e.g., "The president, Barack Obama" and "Barack Obama,



(a) A syntactic pattern

NAMED_ENTITY , NOUN

(b) A regular expression pattern

Figure 1: Patterns capturing appositions

Pattern	Example
Final patterns	
Hearst I	{Presidents} such as [Obama] (and) [Bush]
Hearst II	{Presidents} like [Obama] (and) [Bush]
Hearst III	Obama (and) other {presidents}
Hearst IV	{Presidents} including [Obama] (and) [Bush]
Apposition	[Obama], (the) {president}
Copular	[Obama] is (the) {president}
Noun modifier	{President} [Barack Obama]
Among	[Joe Biden] among (other) {vice presidents}
Enough	[Messi] is enough (of) a {player}
As	[Messi] as {player}
Non-final patterns	
Location	{City} of [London]
Poss. + transf.	[Shakespeare]'s {productions}
by-prep + transf.	{productions} by [Shakespeare]

Table 2: Patterns for explicit type extraction

the president"), which is not displayed. We divide our patterns into final and non-final. Final patterns generally have high precision and extract the lexical type exactly as it occurs. When a final pattern produces a lexical type, we add the corresponding types to the candidate set and go directly to the type selection phase, i.e., we do not consider any other extractor. For non-final patterns, however, we expect erroneous extractions and proceed differently; we perform a KB lookup for all lexical types. The KB lookup (described in the next section) both prunes and expands the candidate set using a KB, and acts as a stopping condition. (FINET can also be run without using KB lookups.)

We treat a pattern as non-final if it may or may not denote a lexical type (e.g. "the president of Argentina" vs. "the city of Buenos Aires") or if a verb-noun transformation is required to obtain the lexical type (e.g. "Shakespeare's productions" to "producer"). To perform the transformations, we use WordNet's *derivationally related forms*, which connect semantically related morphological variations of verbs and nouns. For instance, the noun "production" is connected to the verb "produce," which in turn is connected to the noun "producer". These variations can be exploited to extract explicit types. We treat such transforma-

tions as non-final because we may make mistakes. Moreover, WordNet is highly incomplete in terms of derivational forms. For instance, we may not be able to reach all the possible senses for "producer" from "production," and from "works" in "works by Schumann", we cannot reach "musician" or "artist," but we reach "worker" and there is no synset of "worker" that specifically denotes an artist.

2.3 Exploiting a knowledge base

Most of our extractors (optionally) leverage a knowledge base to (1) prune candidate types, (2) find additional candidates, and (3) decide whether to consider subsequent extractors. To do so, we extract from a KB a repository of (entity mention, type)-pairs.[2] We use the KB conservatively, i.e., we consider KB evidence only if the extracted types match the ones in the KB.

The KB is leveraged via a *KB lookup*. Each KB lookup takes as input an entity mention e and a set T of candidate types found by the extractor (lexical or disambiguated). We first replace each lexical type in T by the set of the corresponding types from WordNet. Afterwards, for each type $t \in T$, we check whether there is one or more matching types (e, t_{KB}) in the KB for the given mention. Type t_{KB} matches t if it is either identical, a hypernym, or a hyponym of t. For each match, the KB lookup outputs t. If t_{KB} is a hyponym of t, we additionally output t_{KB}, that is, a type more specific than the one found by the extractor. We leave the decision of whether t or t_{KB} is a more suitable type to the type selection phase. For example, for e = "Messi" and $t = \langle$*player*-1$\rangle$, we output $\langle$*player*-1$\rangle$ and $\langle$*soccer_player*-1$\rangle$ (a hyponym).

The KB lookup is *successful* if it outputs at least one type. Then we add the resulting types to the candidate set and go directly to the type selection phase. A KB lookup *fails* if no matching type was found. We then proceed differently: if a KB lookup fails, we add the complete set T to the candidate set and continue to the next extractor.

As mentioned above, FINET can also be run without any KB lookups; the corresponding extractors then do not have a stopping condition. In Sec. 4, we experimented with both variants and found that KB lookups generally helped.

[2]We used Yago2 (Hoffart et al., 2013). Our repository contained roughly 9M entity mentions and 20M pairs.

2.4 Mention-based extractor

Our second extractor aims to extract type candidates from the entity mention. This is particularly effective for organizations, which often contain the lexical type in their name (e.g., "Johnson & Wales University" or "Republican House").

Given an entity mention, we check if any of the words or expressions in the name corresponds to a lexical type in WordNet. If so, we consider the corresponding types as potential candidates. For instance, for "Imperial College London", we extract "college" and obtain types $\langle$*college*-1$\rangle$ and $\langle$*college*-2$\rangle$ (both matching the CG type) from WordNet. We then perform a KB lookup.

We extend the above procedure for entities tagged as *location*. Since the set of (named-entity) locations is quite static and known, we assume that the KB contains all locations and their possible types (our experiments strengthened this assumption). If a mention of a location (e.g., "Berlin") occurs in the repository, we add all the corresponding types from the repository to the candidate set (e.g., $\langle$*city*-1$\rangle$) and move to the type selection phase.

2.5 Verb-based extractor

Verbs have been widely exploited to determine the types or roles of its arguments: A verb sense imposes a restriction on the type of its arguments (Quirk et al., 1985; Levin, 1993; Hanks, 1996; Baker et al., 1998; Palmer et al., 2005; Kipper et al., 2008). For instance, from "Ted Kennedy was elected to Congress," we infer that "Ted Kennedy" is a *person* who can be elected. Corresponding types include $\langle$*representative*-1$\rangle$, $\langle$*representative*-2$\rangle$, or $\langle$*politician*-1$\rangle$. Our verb-based extractor leverages this insight to extract types. The extractor operates at the clause level.

A simple way to infer lexical types for entities acting as subjects or objects of a clause is *nominalization*, i.e., the transformation of the verb into *deverbal nouns* (e.g., "play" into "player"). To exploit it, we apply a set of morphological transformations to the verb (Quirk et al., 1985), which depend on the grammatical function of the entity, i.e., subject or object. If the entity mention acts as a subject, we try adding the suffixes "-er,", "-or," and "-ant" to the verb's lemma. If the mention acts as an object, we use suffixes "-ee" and "-ed". To obtain candidate types, we again use derivationally related forms (DER). We consider as potential candidates all types referred to by one of the deverbal nouns and connected to a sense of the

verb via DER. For instance, given "Messi plays in Barcelona," we collect for "Messi" all the senses of "player" that are connected to some sense of "play" (⟨*player*-1⟩, ⟨*musician*-1⟩ and ⟨*actor*-1⟩).

We also explore WordNet in a way that is not restricted to morphological variations of the verb. For instance, in "John committed a crime," "commit" is a synonym of "perpetrate," which in turn can be varied to "perpetrator". We consider the morphological variations of all synonyms of the verb. Moreover, if the named entity is the subject of the clause, and if the clause contains a direct object, we form a new lexical type by adding the direct object as a noun modifier of the deverbal noun (e.g., From "Messi plays soccer", we form "soccer player"). If it exists in WordNet, we consider the respective types as potential candidates as well.

A more indirect way of exploiting the verb-type semantic concordance is via a corpus of frequent (verb, type)-pairs, where the type refers to possible types of the verb's subject or object. As stated above, the set of argument types compatible with a verb is limited. For instance, "treat" is usually followed by ⟨*condition*-1⟩, ⟨*disease*-1⟩, or ⟨*patient*-1⟩. FINET, uses the corpora of Flati and Navigli (2013) and Del Corro et al. (2014). Given a verb and an entity, we search for frequent candidate types (depending on whether the entity acts as a subject or object). For example, from "Messi was treated in the hospital," we obtain ⟨*patient*-1⟩.

Once potential candidates have been collected, we perform a KB lookup to decide how to proceed.

2.6 Corpus-based extractor

Our final extractor leverages a large unlabeled corpus to find entities that co-occur in similar contexts. It is based on the distributional hypothesis (Sahlgren, 2008): similar entities tend to occur in similar contexts. For example, "Messi" and "Cristiano Ronaldo" may both be mentioned in the context of soccer. Thus entity mentions similar to "Messi" in a sport context are likely to include soccer players. Our extractor is related to semi-supervised KB methods in that it propagates types of named entity mentions that may appear in a similar context. It differs in that it is unsupervised, does not require manually or automatically generated training data, and in the way context is modeled and candidates are generated.

Our corpus-based extractor makes use of word vectors (Rumelhart et al., 1988) trained on a large unlabeled corpus. A word vector is a semantic representation of a phrase and represents the semantic context in which the phrase occurs. Phrases that are semantically related, and thus appear in similar contexts, are close to each other in the word vector space. For instance, if "Messi" and "Cristiano Ronaldo" tend to co-occur with a similar sets of words, their word vectors are close. We also may expect "Arnold Schwarzenegger" to be close to both actors and politicians, since it occurs in both contexts. In our work, we use word2vec (Mikolov et al., 2013), which provides a model trained on Google News to predict related words or phrases for a *query* specified as a set of phrases. Given an integer k, word2vec outputs the set of k phrases that are most similar to the query.

Our corpus-based extractor uses (1) the input sentence to construct a set of relevant queries and (2) the word2vec results and a KB. To construct a query for a given entity mention, we focus on the part of the sentence directly related to the entity. This relevant part consists of the clause in which the entity occurs and the subordinate clauses that do not contain another entity. Since word2vec is most effective when queries are short, we construct a set of small queries, each consisting of the named entity mention and some context information. We construct a query for each noun phrase (of length at most 2) and for each other entity mention. If the named entity occurs as subject or object, we also take the corresponding verb and the head of the object or subject. For example, the queries for "Maradona expects to win in South Africa" are {"Maradona", "South Africa"} and {"Maradona", "expect", "win"}.

For each query, we retrieve the 100 most related phrases with their similarity score and union the results. We filter them using our KB and retain only those phrases that correspond to entity mentions (with the correct CG types). We then enrich each mention by the set of their possible types from the KB. Here we exclude widespread but irrelevant implicit types such as ⟨*male*-1⟩, ⟨*female*-1⟩, ⟨*adult*-1⟩, ⟨*commoner*-1⟩, ⟨*friend*-1⟩, or ⟨*alumnus*-1⟩. We also include the types corresponding to the entity mention (with score 1). If there is sufficient evidence that some of the so-obtained types are most prominent, we take these types as candidates. In our example, query {"Maradona" "South Africa"}, all of the top-15 persons (e.g., "Diego Maradona", "Carlos Alberto Parreira", "Dunga", "Beckenbauer") share type ⟨*coach*-1⟩; a strong indication that Maradona may

also be of type ⟨*coach*-1⟩. To select prominent types we traverse the results until we collect 50% of the total score. We take all so-collected types as candidates. If no more than 10 different types were added this way, we directly go to the type selection phase.

3 Type Selection

The type selection phase selects the most appropriate type from the set of candidates of a given named entity. We use techniques from WSD, but adapt them to our setting. WSD aims to disambiguate a word or phrase (e.g., a noun) with respect to a type system as WordNet; e.g., from "player" to ⟨*player*-1⟩. The main difference between WSD and our type selection is that our goal is to decide between a set of types for an entity mention; e.g., from "Messi" to ⟨*soccer_player*-1⟩. Our type selection step can be used as-is; it is not trained on any domain- or corpus-specific data.

3.1 Obtaining context

All WSD systems take a set of candidate types and contextual information as input. The key challenge lies in the construction of candidate types (Sec. 2) as well as context. For each entity, we consider *entity-oblivious context* (from the input sentence) as well as *entity-specific context* (using lexical expansions).

We take all words in the sentence as entity-oblivious context. To construct entity-specific context, we make use of lexical expansions, which have been successfully applied in WSD (Miller et al., 2012). Its goal is to enrich contextual information to boost disambiguation. In our case, it also helps to differentiate between multiple entities in a sentence. We build the entity-specific context using word vectors. As in the corpus-based extractor, we construct a set of queries for the entity but in this case we take as context all so-obtained words that do *not* correspond to a named entity. For instance, the entity-specific context for the entity mention "Maradona" for query "Maradona South_Africa" is: "coach", "cup", "striker", "midfielder", and "captain". The full context for "Maradona" in "Maradona expects to win in South Africa" additionally includes the entity-oblivious context "expects", "win", "South Africa".

3.2 Selecting types

WSD systems fall into two classes: unsupervised, which rely on background knowledge such as WordNet (Ponzetto and Navigli, 2010), and supervised, which require training data (Zhong and Ng, 2010). Here we take a combination, i.e., we leverage WordNet and manually annotated data.

We train a Naive Bayes classifier to select the most appropriate type given its context. We represent context by a bag of lemmatized words. This allows us to automatically generate training data from WordNet (and use manually labeled data). Since WordNet provides information for each of the relevant types, this approach combats the data sparsity that arises with supervised systems. The context for each individual WordNet type consists of all words appearing in the type's gloss and the glosses of its neighbors (Banerjee and Pedersen, 2003). We also include for each type the neighbors from Ponzetto and Navigli (2010) and the corresponding verbs from Del Corro and Gemulla (2013). Finally, we add all words in sentences containing the type in SemCor[3] (Landes et al., 1998) and Ontonotes 5.0 (Hovy et al., 2006).

We trained one classifier per CG type. To train the classifier, we create a single training point for each corresponding WordNet type and use the type's context as features. To map the CG types from our NER system to WordNet, we considered as persons all descendants of ⟨*person*-1⟩, ⟨*imaginary being*-1⟩, ⟨*characterization*-3⟩, and ⟨*operator*-2⟩ (10584 in total); as locations all descendants of ⟨*location*-1⟩, ⟨*way*-1⟩, and ⟨*landmass*-1⟩ (3681 in total); and as organizations all descendants of ⟨*organization*-1⟩ and ⟨*social group*-1⟩ (1968 in total). This approach of handling CG types suffers to some extent from WordNet's incompleteness, esp. with respect to persons and organizations. For instance, phrase "sponsored by Coca-Cola" implies that "Coca-Cola" is a "sponsor," but in WordNet, only persons can be sponsors. Nevertheless, this approach worked well in our experiments.

4 Experiments

We conducted an experimental study on multiple real-word datasets to compare FINET with two state-of-the-art approaches. FINET is used as-is; it does not require training or tuning for any specific dataset. All datasets, detected types, labels, and our source code are publicly available.[4]

[3] http://web.eecs.umich.edu/~mihalcea/downloads.html

[4] http://dws.informatik.uni-mannheim.de/en/resources/software/finet/

4.1 Experimental Setup

Methods. *Hyena (Yosef et al., 2012)* is a representative supervised method that uses a hierarchical classifier. Its features include the words in the named entity mention, in sentence and paragraph, and POS tags. It performs basic co-reference resolution and marks entity mentions connected to a type in the KB using a binary feature. Similar to Ling and Weld (2012), Hyena is trained on Wikipedia entities, each being annotated with its corresponding WordNet types from YAGO. Hyena's type system is restricted to 505 WordNet types with top categories ⟨*artifact*-1⟩, ⟨*event*-1⟩, ⟨*person*-1⟩, ⟨*location*-1⟩, and ⟨*organization*-1⟩. Hyena outperformed a number of previous systems (Fleischman and Hovy, 2002; Rahman and Ng, 2010; Ling and Weld, 2012). We used Hyena via its web service (Yosef et al., 2013).

Pearl (Nakashole et al., 2013) is a semi-supervised system that leverages a repository of 300k relational patterns (Nakashole et al., 2012). Subjects and objects of each pattern carry type information. Pearl types named entity mentions by the most likely type according to its pattern database. Pearl's type system is based on around 200 "interesting" WordNet types. We ran Pearl in its *hard* setting, which performed best.

FINET. We ran FINET in two configurations: (1) with KB lookup, (2) without the KB lookup. This allows us to estimate the extent to which referring to a KB helps. Note that the corpus-based extractor makes use of the KB in both settings.

Datasets. We used three different datasets representing real-world use cases. We created two datasets, New York Times and Twitter, and sampled a subset of the CoNLL data, which provides gold annotations for CG types. We did not consider datasets such as FIGER (Ling and Weld, 2012) or BBN (Weischedel and Brunstein, 2005) because they are not suitable for very fine-grained typing.

New York Times consists of 500 random sentences from the New York Times corpus (Sandhaus, 2008), year 2007; we selected only sentences that contained at least one named entity according to the Stanford CoreNLP 4.4.1 tool.

CoNLL. We sampled 500 sentences from CoNLL (Tjong Kim Sang and De Meulder, 2003), a collection of newswires with manually annotated entities with CG types labels. We directly used the annotations in our evaluation. The sentences tend to be short and sometimes non-verbal (e.g., "Jim Grabb (U.S.) vs. Sandon Stolle (Australia)"). Most entities are prominent and likely to be found in our KB (and the one of existing methods).

Twitter. We collected the first 100 tweets with named entities retrieved by the Twitter API.

Type system. FINET's type system consists of more than 16k types with top categories persons, locations and organizations. We used the mapping between these top categories and WordNet described in Sec. 3.2. Hyena and Pearl use 505 and 200 WordNet types, resp., which is significantly smaller. To compare the performance across different granularities, we classified each type as coarse-grained (CG), fine-grained (FG) or super fine-grained (SFG). The CG types were ⟨*artifact*-1⟩, ⟨*event*-1⟩, ⟨*person*-1⟩, ⟨*location*-1⟩ and ⟨*organization*-1⟩. The FG types were those included in Pearl. All remaining types were considered SFG.

Labeling. All extractions by all systems were independently evaluated by two labelers. We adopted a pessimistic view, i.e., we treat an extraction as correct only if it was labeled correct by both labelers. The Cohen's kappa measure ranged 0.54–0.86, indicating a substantial agreement.

4.2 Results

Description of Tab. 3. Our results are summarized in Tab. 3. When a method did not produce a type of the considered granularity but a more fine-grained type, we selected its closest hypernym. For each configuration, the table shows the number of named entities for which types have been extracted, the total number of extracted types (more than one distinct type per named entity for some methods), the total number of correct types, and the precision (P). The number of named entities for which types have been found and the total number of correct extractions can be seen as a loose measure of recall. It is difficult to estimate recall directly for FG and SFG types since some entities may be associated with either no or multiple such types. To gain more insight, we show the number of correct distinct types, and the average depth (shortest path from ⟨*entity*-1⟩ in WordNet) for both correct FG and correct SFG types. Finally, we list the Cohen's kappa inter-annotator agreement measure for each method.

System	Coarse-Grained (CG)			Fine-Grained (FG)			Super Fine-Grained (SFG)			Distinct	Avg. Depth		Cohen's
	Enti-ties	Total types	Correct types *(P)*	Enti-ties	Total types	Correct types *(P)*	Enti-ties	Total types	Correct types *(P)*	types	FG	SFG	kappa
New York Times (500 sentences)													
FINET	992	992	872 *(87.90)*	616	631	457 *(72.42)*	319	329	233 *(70.82)*	191	5.96	7.25	0.60
FINET (w/o KB l.)	992	992	872 *(87.90)*	598	613	436 *(71.13)*	294	304	204 *(67.11)*	174	5.98	7.18	0.58
Hyena	895	1076	779 *(72.40)*	770	1847	522 *(28.26)*	518	775	160 *(20.65)*	127	5.79	6.98	0.74
Pearl (hard)	15	15	5 *(33.33)*	2	2	0	–	–	– *(–)*	1	–	–	0.54
CoNLL (500 sentences)													
FINET	1355	1355	1355 *(1.0)*	1074	1086	876 *(80.66)*	668	679	510 *(75.11)*	136	6.09	7.38	0.62
FINET (w/o KB l.)	1355	1355	1355 *(1.0)*	1075	1087	869 *(79.94)*	661	672	498 *(66.13)*	134	6.06	7.35	0.62
Hyena	1162	1172	1172 *(1.0)*	1064	2218	1329 *(59.92)*	719	944	268 *(28.39)*	103	5.89	6.57	0.69
Pearl (hard)	18	18	18 *(1.0)*	8	11	5 *(45.45)*	–	–	– *(–)*	7	5.6	–	0.74
Twitter (100 tweets)													
FINET	135	135	123 *(91.11)*	103	104	69 *(66.35)*	54	54	33 *(61.11)*	40	6.25	7.64	0.58
FINET (w/o KB l.)	135	135	123 *(91.11)*	104	105	65 *(61.90)*	56	56	30 *(53.57)*	40	6.14	7.6	0.55
Hyena	125	146	105 *(71.91)*	117	280	75 *(26.79)*	91	129	21 *(16.28)*	42	6.11	6.19	0.67
Pearl (hard)	10	10	5 *(50.00)*	3	4	1 *(25.00)*	–	–	– *(–)*	3	6	–	0.86

Table 3: Summary of results

Discussion. First note that Pearl extracted significantly fewer types than any other system. Pearl does not support SFG. For CG and FG, we conjecture that its database, which was generated from Wikipedia, did not reflect the syntactic structure of the sentences in our datasets. This finding strengthens the case for the use of heterogeneous sources in semi-supervised methods.

Hyena performed better than Pearl and in many cases extracted the largest number of types. Hyena tended to extract multiple types per named entity and mostly at least one FG type. This more recall-oriented approach, and its context-unaware use of supervision, significantly reduced its precision.

FINET had significantly higher precision across all settings, especially for SFG types (almost three times more than Hyena). One reason for this is that FINET is conservative: We provide more than one type per named entity only if the types were explicit. In all other cases, our type selection phase produced only a single type. FINET extracted the largest number of correct SFG types on each dataset. Hyena extracted more FG types, but with a significantly lower precision. The average depth of correct FG and SFG types in FINET was higher than that of Pearl and Hyena. FINET also tended to use more distinct correct types (191 in NYT vs. 127 for Hyena). Again, this more fine-grained typing stemmed from FINET's use of multiple extractors, many of which do not rely on supervision.

Note that FINET also has higher precision for CG. As stated, FINET makes use of the Stanford NER to extract CG types (except in CoNLL, were we used the manual labels), and respects these types for its FG and SFG extractions. Hyena has lower precision for CG types because it sometimes outputs multiple CG types for a single named entity. To ensure a fair comparison, for CoNLL we indirectly used the gold labels for Pearl and Hyena by discarding all types with an incorrect CG type.

FINET's extractors. Tab. 4 shows individual influence of each of FINET's extractors in the NYT dataset with KB lookups. The table shows the number of entities typed by each extractor and the precision of the resulting types after type selection. The mention-based extractor was the most precise and also fired most often, mainly due to locations. The pattern-based extractor also had a good precision and tended to fire often. The first three extractors, the more explicit ones, generated more than half of the extracted types; this indicates that explicit type extractors are important. There were also a substantial fraction of implicit types, covered by the corpus-based extractor. The verb-based extractor had the lowest precision, mostly because of the noisiness and incompleteness of its underlying resources (such as the (verb,type)-repository). We expect overall precision to increase if this extractor is removed. However, this would hinder FINET to infer types from verbs. Instead, we believe a better direction is to improve the underlying resources.

Error analysis. One source of error for FINET were incorrect CG labels. When CG labels were correct (by Stanford NER), the precision of FINET for FG types increased to more than 70% for all datasets. When FG labels were correct, the pre-

Last used extractor	Entities	P
Pattern-based	180	71.11
Mention-based	219	82.65
Verb-based	47	48.94
Corpus-based	205	64.39

Table 4: Per-extractor performance NYT

cision of SFG labels exceeded 90%.

Incompleteness of and noise in our underlying resources also affected precision. For example, some types in WordNet have missing hypernyms, which reduced recall; e.g., sponsor in WordNet is a person but cannot be an organization. WordNet is also biased towards US types (e.g., *supreme court* only refers to the US institution). Our repositories of verbs and their argument types are incomplete and noisy as well. Finally, errors in the KB affected both KB lookups and our corpus-based extractor. One example of such errors are temporal discrepancies; e.g., a person who used to be a $\langle$*player*-1$\rangle$ may now be a $\langle$*coach*-1$\rangle$. The KB types are also noisy, e.g., many soccer players in Yago2 are typed as $\langle$*football_player*-1$\rangle$ and the United Nations is typed as a $\langle$*nation*-1$\rangle$.

Finally, the type selection phase of FINET introduced mistakes (i.e., even when the correct type was a candidate, type selection failed to select it). This is especially visible in the verb-based extractor, which may produce a large number of candidates and thus makes type selection difficult.

Hyena mainly suffered from the general problems of supervised systems. For instance, since $\langle$*graduate*-1$\rangle$ or $\langle$*region*-1$\rangle$ are highly frequent in the KB, many persons (locations) were incorrectly typed as $\langle$*graduate*-1$\rangle$ ($\langle$*region*-1$\rangle$). Errors in the KB also propagate in supervised system, which may lead to "contradictory" types (i.e., an entity being typed as both $\langle$*person*-1$\rangle$ and $\langle$*location*-1$\rangle$).

5 Related Work

The NET problem is related to taxonomy induction (Snow et al., 2006; Wu et al., 2012; Shi et al., 2010; Velardi et al., 2013) and KB construction (Lee et al., 2013; Paulheim and Bizer, 2014; Mitchell et al., 2015), although the goals are different. Taxonomy induction aims to produce or extend a taxonomy of types, whereas KB construction methods aim to find new types for the entities present in a KB. In both cases, this is done by reasoning over a large corpus. In contrast, we are interested in typing each named entity mention individually using an existing type system. FINET draws from ideas used in taxonomy induction or KB construction. Existing systems are either based on patterns or the distributional hypothesis; these two approaches are discussed and compared in (Shi et al., 2010). In FINET, we make use of patterns (such as the ones of Hearst (1992)) in most of our extractors and of the distributional hypothesis in our corpus-based extractor.

Yahya et al. (2014) developed a semi-supervised method to extract facts such as "president"("Barack Obama", "US"), in which the relation acts as a type. FINET differs in that it supports implicit types and produces disambiguated types.

A number of NET systems have been proposed which make use of a predefined type hierarchy. Lin et al. (2012) proposes a semi-supervised system that uses relational patterns to propagate type information from a KB to entity mentions. Similarly, the subsequent Pearl system (Nakashole et al., 2013) is based on a corpus of typed relation patterns. An alternative approach is taken by supervised methods, which train classifiers based on linguistic features (Fleischman and Hovy, 2002; Rahman and Ng, 2010; Ling and Weld, 2012). Both Yosef et al. (2013) and Ling and Weld (2012) use Wikipedia and a KB to generate automatic training data. FINET is less reliant on a KB or training data than the above methods, which improves both precision (no bias against KB types) and recall (more fine-grained types supported).

Our type selection phase is based on WSD (Navigli, 2012), a classification task where words or phrases are disambiguated against senses from some external resource such as WordNet. Supervised WSD systems (Dang and Palmer, 2005; Dligach and Palmer, 2008; Chen and Palmer, 2009; Zhong and Ng, 2010) use a classifier to assign such senses, mostly relying on manually annotated data. KB methods (Agirre and Soroa, 2009; Ponzetto and Navigli, 2010; Miller et al., 2012; Agirre et al., 2014; Del Corro et al., 2014) use of a background KB instead.

6 Conclusion

We presented FINET, a system for fine-grained typing of named entities in context. FINET generates candidates using multiple extractors, ranging from explicitly mentioned to implicit types, and subsequently selects the most appropriate. Our experimental study indicates that FINET has significantly better performance than previous methods.

References

Eneko Agirre and Aitor Soroa. 2009. Personalizing pagerank for word sense disambiguation. In *Proceedings of EACL*, pages 33–41.

Eneko Agirre, Oier Lopez de Lacalle, and Aitor Soroa. 2014. Random walks for knowledge-based word sense disambiguation. *Computational Linguistics*, 40(1):57–84.

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

Satanjeev Banerjee and Ted Pedersen. 2003. Extended gloss overlaps as a measure of semantic relatedness. In *Proceedings of IJCAI*, pages 805–810.

Jinying Chen and Martha Palmer. 2009. Improving english verb sense disambiguation performance with linguistically motivated features and clear sense distinction boundaries. *Language Resources and Evaluation*, 43(2):181–208.

Hoa Trang Dang and Martha Palmer. 2005. The role of semantic roles in disambiguating verb senses. In *Proceedings of ACL*, pages 42–49.

Luciano Del Corro and Rainer Gemulla. 2013. Clausie: clause-based open information extraction. In *Proceedings of WWW*, pages 355–366.

Luciano Del Corro, Rainer Gemulla, and Gerhard Weikum. 2014. Werdy: Recognition and disambiguation of verbs and verb phrases with syntactic and semantic pruning. In *Proceedings of EMNLP*, pages 374–385.

Dmitriy Dligach and Martha Palmer. 2008. Improving verb sense disambiguation with automatically retrieved semantic knowledge. In *Proceedings of ICSC*, pages 182–189.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of ACL*, pages 363–370.

Tiziano Flati and Roberto Navigli. 2013. Spred: Large-scale harvesting of semantic predicates. In *Proceedings of ACL*, pages 1222–1232.

Michael Fleischman and Eduard Hovy. 2002. Fine grained classification of named entities. In *Proceedings of COLING*, pages 1–7.

Patrick Hanks. 1996. Contextual dependency and lexical sets. *International Journal of Corpus Linguistics*, 1(1):75–98.

Marti A. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of COLING*, pages 539–545.

Johannes Hoffart, Fabian M. Suchanek, Klaus Berberich, and Gerhard Weikum. 2013. Yago2: A spatially and temporally enhanced knowledge base from wikipedia. *Artif. Intell.*, 194:28–61.

Johannes Hoffart, Dragan Milchevski, and Gerhard Weikum. 2014. Stics: Searching with strings, things, and cats. In *Proceedings of SIGIR (demo)*, SIGIR '14, pages 1247–1248.

Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. Ontonotes: The 90 In *Proceedings of HLT-NAACL (Companion Volume)*, pages 57–60.

Karin Kipper, Anna Korhonen, Neville Ryant, and Martha Palmer. 2008. A large-scale classification of English verbs. *Language Resources and Evaluation*, 42(1):21–40.

Shari Landes, Claudia Leacock, and Randee I. Tengi, 1998. *Building Semantic Concordances*. MIT Press.

Taesung Lee, Zhongyuan Wang, Haixun Wang, and Seung-won Hwang. 2013. Attribute extraction and scoring: A probabilistic approach. In *Proceedings of ICDE*, pages 194–205.

Beth Levin. 1993. *English Verb Classes and Alternations: A Preliminary Investigation*. University of Chicago Press.

Thomas Lin, Mausam, and Oren Etzioni. 2012. No noun phrase left behind: Detecting and typing unlinkable entities. In *Proceedings of EMNLP*, pages 893–903.

Xiao Ling and Daniel S. Weld. 2012. Fine-grained entity recognition. In *In Proceedings of AAAI*.

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

Tristan Miller, Chris Biemann, Torsten Zesch, and Iryna Gurevych. 2012. Using distributional similarity for lexical expansion in knowledge-based word sense disambiguation. In *Proceedings of COLING*, pages 1781–1796.

Tom Mitchell, William Cohen, Estevam Hruscha, Partha Talukdar, Justin Betteridge, Andrew Carlson, Bhavana Dalvi, Matthew Gardner, Bryan Kisiel, Jayant Krishnamurthy, Ni Lao, Kathryn Mazaitis, Thahir Mohammad, Ndapa Nakashole, Emmanouil Platanios, Alan Ritter, Mehdi Samadi, Burr Settles, Richard Wang, Derry Wijaya, Abhinav Gupta, Xinlei Chen, Abulhair Saparov, Malcolm Greaves, and Joel Welling. 2015. Never-ending learning. In *Proceedings of AAAI*, pages 2302–2310.

Ndapandula Nakashole, Gerhard Weikum, and Fabian Suchanek. 2012. Patty: A taxonomy of relational patterns with semantic types. In *Proceedings of EMNLP*, pages 1135–1145.

Ndapandula Nakashole, Tomasz Tylenda, and Gerhard Weikum. 2013. Fine-grained semantic typing of emerging entities. In *Proceedings of ACL*, pages 1488–1497.

Roberto Navigli. 2012. A quick tour of word sense disambiguation, induction and related approaches. In *Proceedings of SOFSEM*, pages 115–129.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106.

Heiko Paulheim and Christian Bizer. 2014. Improving the quality of linked data using statistical distributions. *IJSWIS*, 10(2):63–86.

Simone Paolo Ponzetto and Roberto Navigli. 2010. Knowledge-rich word sense disambiguation rivaling supervised systems. In *Proceedings of ACL*, pages 1522–1531.

Randolph Quirk, Sidney Greenbaum, Geoffrey Leech, and Jan Svartvik. 1985. *A Comprehensive Grammar of the English Language*. Longman.

Altaf Rahman and Vincent Ng. 2010. Inducing fine-grained semantic classes via hierarchical and collective classification. In *Proceedings of COLING*, pages 931–939.

Marta Recasens, Marie C. de Marneffe, and Christopher Potts. 2013. The Life and Death of Discourse Entities: Identifying Singleton Mentions. In *Proceedings of HLT-NAACL*, pages 627–633.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. 1988. Neurocomputing: Foundations of research. chapter Learning Representations by Back-propagating Errors, pages 696–699.

Magnus Sahlgren. 2008. The distributional hypothesis. *Italian Journal of Linguistics*, 20(1):33–54.

Evan Sandhaus. 2008. The New York Times Annotated Corpus. *Linguistic Data Consortium, Philadelphia*, 6(12).

Shuming Shi, Huibin Zhang, Xiaojie Yuan, and Ji-Rong Wen. 2010. Corpus-based semantic class mining: Distributional vs. pattern-based approaches. In *Proceedings of COLING*, pages 993–1001.

Rion Snow, Daniel Jurafsky, and Andrew Y. Ng. 2006. Semantic taxonomy induction from heterogenous evidence. In *Proceedings of COLING-ACL*, pages 801–808.

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

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of HLT-NAACL*, pages 142–147.

Paola Velardi, Stefano Faralli, and Roberto Navigli. 2013. Ontolearn reloaded: A graph-based algorithm for taxonomy induction. *Computational Linguistics*, 39(3):665–707.

Ralph Weischedel and Ada Brunstein. 2005. BBN Pronoun Coreference and Entity Type Corpus. Technical report, Linguistic Data Consortium.

Wentao Wu, Hongsong Li, Haixun Wang, and Kenny Q. Zhu. 2012. Probase: A probabilistic taxonomy for text understanding. In *Proceedings of SIGMOD*, pages 481–492.

Mohamed Yahya, Klaus Berberich, Shady Elbassuoni, and Gerhard Weikum. 2013. Robust question answering over the web of linked data. In *Proceedings of CIKM*, pages 1107–1116.

Mohamed Yahya, Steven Euijong Whang, Rahul Gupta, and Alon Halevy. 2014. Renoun: Fact extraction for nominal attributes. In *Proceedings of EMNLP*, pages 325–335.

Mohamed Amir Yosef, Sandro Bauer, Johannes Hoffart, Marc Spaniol, and Gerhard Weikum. 2012. HYENA: Hierarchical Type Classification for Entity Names. In *Proceedings of COLING*, pages 1361–1370.

Mohamed Amir Yosef, Sandro Bauer, Johannes Hoffart, Marc Spaniol, and Gerhard Weikum. 2013. HYENA-live: Fine-Grained Online Entity Type Classification from Natural-language Text. In *Proceedings of ACL*, pages 133–138.

Zhi Zhong and Hwee Tou Ng. 2010. It makes sense: A wide-coverage word sense disambiguation system for free text. In *Proceedings of ACL: System Demonstrations*, pages 78–83.

Joint Named Entity Recognition and Disambiguation

Gang Luo[1], **Xiaojiang Huang**[2], **Chin-Yew Lin**[2], **Zaiqing Nie**[2]
[1]Microsoft, California, USA
[2]Microsoft Research, Beijing, China
{gluo, xiaojih, cyl, znie}@microsoft.com

Abstract

Extracting named entities in text and linking extracted names to a given knowledge base are fundamental tasks in applications for text understanding. Existing systems typically run a named entity recognition (NER) model to extract entity names first, then run an entity linking model to link extracted names to a knowledge base. NER and linking models are usually trained separately, and the mutual dependency between the two tasks is ignored. We propose JERL, Joint Entity Recognition and Linking, to jointly model NER and linking tasks and capture the mutual dependency between them. It allows the information from each task to improve the performance of the other. To the best of our knowledge, JERL is the first model to jointly optimize NER and linking tasks together completely. In experiments on the CoNLL'03/AIDA data set, JERL outperforms state-of-art NER and linking systems, and we find improvements of 0.4% absolute F_1 for NER on CoNLL'03, and 0.36% absolute precision@1 for linking on AIDA.

1 Introduction

In applications of complex Natural Language Processing tasks, such as automatic knowledge base construction, entity summarization, and question answering systems, it is essential to first have high quality systems for lower level tasks, such as part-of-speech (POS) tagging, chunking, named entity recognition (NER), entity linking, and parsing among others. These lower level tasks are usually decoupled and optimized separately to keep the system tractable. The disadvantage of the decoupled approach is that each lower level task is not aware of other tasks and thus not able to leverage information provided by others to improve performance. What is more, there is no guarantee that their outputs will be consistent.

This paper addresses the problem by building a joint model for Entity Recognition and Disambiguation (ERD). The goal of ERD is to extract named entities in text and link extracted names to a knowledge base, usually Wikipedia or Freebase. ERD is closely related to NER and linking tasks. NER aims to identify named entities in text and classify mentions into predefined categories such as persons, organizations, locations, etc. Given a mention and context as input, entity linking connects the mention to a referent entity in a knowledge base.

Existing ERD systems typically run a NER to extract entity mentions first, then run an entity linking model to link mentions to a knowledge base. Such a decoupled approach makes the system tractable, and both NER and linking models can be optimized separately. The disadvantages are also obvious: 1) errors caused by NER will be propagated to linking and are not recoverable 2) NER can not benefit from information available used in entity linking; 3) NER and linking may create inconsistent outputs.

We argue that there is strong mutual dependency between NER and linking tasks. Consider the following two examples:

---

1. The New York Times (NYT) is an American daily newspaper.
2. Clinton plans to have more news conferences in 2nd term. WASHINGTON 1996-12-06

Example 1 is the first sentence from the Wikipedia article about "The New York Times". It is reasonable but incorrect for NER to identify "New York Times" without "The" as a named entity, while entity linking has no trouble connecting "The New York Times" to the correct entity.

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 879–888,
Lisbon, Portugal, 17-21 September 2015.

Example 2 is a news title where our NER classifies "WASHINGTON" as a location, since a location followed by a date is a frequent pattern in news articles it learned, while the entity linking prefers linking this mention to the U.S. president "George Washington" since another president's name "Clinton" is mentioned in the context. Both the entity boundaries and entity types predicted by NER are correlated to the knowledge of entities linked by entity linking. Modeling such mutual dependency is helpful in resolving inconsistency and improving performance for both NER and linking.

We propose JERL, Joint Entity Recognition and Linking, to jointly model NER and linking tasks and capture the mutual dependency between them. It allows the information from each task to improve the performance of the other. If NER is highly confident on its outputs of entity boundaries and types, it will encourage entity linking to link an entity which is consistent with NER's outputs, and vice versa. In other words, JERL is able to model how consistent NER and linking's outputs are, and predict coherent outputs. According to our experiments, this approach does improve the end to end performance. To the best of our knowledge, JERL is the first model to jointly optimize NER and linking tasks together completely .

Sil (2013) also proposes jointly conducting NER and linking tasks. They leverage existing NER/chunking systems and Freebase to over generate mention candidates and leave the linking algorithm to make final decisions, which is a reranking model. Their model captures the dependency between entity linking decisions and mention boundary decisions with impressive results. The difference between our model and theirs is that our model jointly models NER and linking tasks from the training phrase, while their model is a combined one which depends on an existing state-of-art NER system. Our model is more powerful in capturing mutual dependency by considering entity type and confidences information, while in their model the confidence of outputs is lost in the linking phrase. Furthermore, in our model NER can naturally benefit from entity linking's decision since both decisions are made together, while in their model, it is not clear how the linking decision can help the NER decision in return.

Joint optimization is costly. It increases the problem complexity, is usually inefficient, and requires the careful consideration of features of multiple tasks and mutual dependency, making proper assumptions and approximations to enable tractable training and inference. However, we believe that joint optimization is a promising direction for improving performance for NLP tasks since it is closer to how human beings process text information. Experiment result indicates that our joint model does a better job at both NER and linking tasks than separate models with the same features, and outperforms state-of-art systems on a widely used data set. We found improvements of 0.4% absolute F_1 for NER on CoNLL'03 and 0.36% absolute precision@1 for linking on AIDA. NER is a widely studied problem, and we believe our improvement is significant.

The contributions of this paper are as follows:
1. We identify the mutual dependency between NER and linking tasks, and argue that NER and linking should be conducted together to improve the end to end performance.
2. We propose the first completely joint NER and linking model, JERL, to train and inference the two tasks together. Efficient training and inference algorithms are also presented.
3. The JERL outperforms the best NER record on the CoNLL'03 data set, which demonstrates how NER could be improved further by leveraging knowledge base and linking techniques.

The remainder of this paper is organized as follows: the next section discusses related works on NER, entity linking, and joint optimization; section 3 presents our Joint Entity Recognition and Linking model in detail; section 4 describes experiments, results, and analysis; and section 5 concludes.

2 Related Work

The NER problem has been widely addressed by symbolic, statistical, as well as hybrid approaches. It has been encouraged by several editions of evaluation campaigns such as MUC (Chinchor and Marsh, 1998), the CoNLL 2003 NER shared task (Tjong Kim Sang and De Meulder, 2003) and ACE (Doddington et al., 2004). Along with the improvement of Machine Learning techniques, statistical approaches have become a major direction for research on NER, especially after Conditional Random Field is proposed by Lafferty et al. (2001). The well known state-of-art NER systems are Stanford NER (Finkel et al., 2005) and UIUC

NER (Ratinov and Roth, 2009). Liang (2005) compares the performance of the 2nd order linear chain CRF and Semi-CRF (Sarawagi and Cohen, 2004) in his thesis. Lin and Wu (2009) cluster tens of millions of phrases and use the resulting clusters as features in NER reporting the best performance on the CoNLL'03 English NER data set. Recent works on NER have started to focus on multi-lingual named entity recognition or NER on short text, e.g. Twitter.

Entity linking was initiated with Wikipedia-based works on entity disambiguation (Bunescu and Pasca, 2006; Cucerzan, 2007). This task is encouraged by the TAC 2009 KB population task[1] first and receives more and more attention from the research community (Hoffart et al., 2011; Ratinov et al., 2011; Han and Sun, 2011). Linking usually takes mentions detected by NER as its input. Stern et al. (2012) and Wang et al. (2012) present joint NER and linking systems and evaluate their systems on French and Chinese data sets. Sil and Yates (2013) take a re-ranking based approach and achieve the best result on the AIDA data set. In 2014, Microsoft and Google jointly hosted "Entity Recognition and Disambiguation Challenge" which focused on the end to end performance of linking system [2].

Joint optimization models have been studied at great length. E.g. Dynamic CRF (McCallum et al., 2003) has been proposed to conduct Part-of-Speech Tagging and Chunking tasks together. Finkel and Manning (2009) show how to model parsing and named entity recognition together. Yu et al. (2011) work on jointly entity identification and relation extraction from Wikipedia. Sil's (2013) work on jointly NER and linking is described in the introduction section of this paper. It is worth noting that joint optimization does not always work. The CoNLL 2008 shared task (Surdeanu et al., 2008) was intended to encourage jointly optimize parsing and semantic role labeling, but the top performing systems decoupled the two tasks.

3 Joint Entity Recognition and Linking

Named entity recognition is usually formalized as a sequence labeling task, in which each word is classified to not-an-entity or entity labels. Conditional Random Fields (CRFs) is one of the popular models used. Most features used in NER are word-level (e.g. a word sequence appears at position i or whether a word contains exactly four digits). It is hard, if not impossible, to encode entity-level features (such as "entity length" and "correlation to known entities") in traditional CRF. Entity linking is typically formalized as a ranking task. Features used for entity linking are at entity-level inherently (such as entity prior probability; whether there are any related entity names or discriminative keywords occurring in the context).

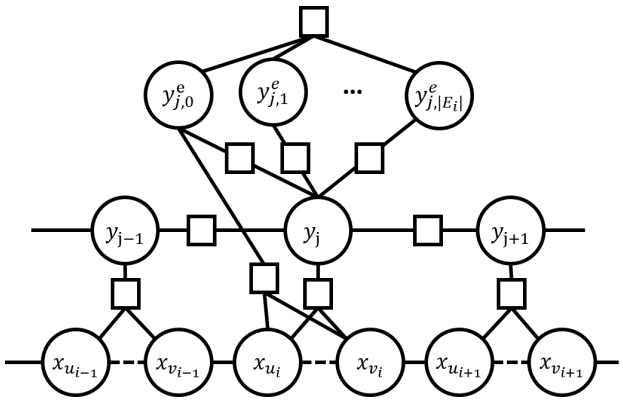


Figure 1: The factor graph of JERL model

The main challenges of joint optimization between NER and linking are: how to combine a sequence labeling model and a ranking model; and how to incorporate word-level and entity-level features. In a linear chain CRF model, each word's label is assumed to depend on the observations and the label of its previous word. Semi-CRF carefully relaxes the Markov assumption between words in CRF, and models the distribution of segmentation boundaries directly. We further extend Semi-CRF to model entity distribution and mutual dependency over segmentations, and name it Joint Entity Recognition and Linking (JERL). The model is described below.

3.1 JERL

Let $\boldsymbol{x} = \{x_i\}$ be a word sequence containing $|\boldsymbol{x}|$ words. Let $\boldsymbol{s} = \{s_j\}$ be a segmentation assignment over $\boldsymbol{x}$, where segment $s_j = (u_j, v_j)$ consist of a start position u_j and an end position v_j. All segments have a positive length and are adjacent to each other, so every (u_j, v_j) always satisfies $1 \leq u_j \leq v_j \leq |\boldsymbol{x}|$ and $u_{j+1} = v_j + 1$. Let $\boldsymbol{y} = \{y_j\}$ be labels in a fixed label alphabet $\mathcal{Y}$ over a segmentation assignment $\boldsymbol{s}$. Here $\mathcal{Y}$ is the set of types NER to predict. $x_{s_j} = (x_{u_j} \dots x_{v_j})$

[1] http://www.nist.gov/tac/2014/KBP/

[2] http://web-ngram.research.microsoft.com/ERD2014/

is the corresponding word sequence to s_j, and $\mathcal{E}_{s_j} = \{e_{j,k}\}$ is a set of entities in the knowledge base (KB), which may be referred by word sequence x_{s_j} in the entity linking task. Each entity $e_{j,k}$ is associated with a label $y^e_{j,k} \in \{0, 1\}$. Label $y^e_{j,k}$ takes 1 iff x_{s_j} referring to entity $e_{j,k}$, and 0 otherwise. If x_{s_j} does not refer to any entity in the KB, $y^e_{j,0}$ takes 1, which is analogous to the NIL[3] identifier in entity linking.

Based on the preliminaries and notations, Figure 1 shows the factor graph (Kschischang et al., 2001) of JERL. There are similar factor nodes for every $(u_j, v_j, y^e_{j,k})$, we only show the first one $(u_j, v_j, y^e_{j,0})$ for clarity.

Given $\boldsymbol{x}$, let $\boldsymbol{a} = (\boldsymbol{s}, \boldsymbol{y}, \boldsymbol{y^e})$ be a joint assignment, and $\boldsymbol{g}(\boldsymbol{x}, j, \boldsymbol{a})$ be local functions for x_{s_j}, namely features, each of which maps an assignment $\boldsymbol{a}$ to a measurement $g^k(x, j, a) \in \Re$. Then $\boldsymbol{G}(\boldsymbol{x}, \boldsymbol{a}) = \sum_{j=1}^{|s|} \boldsymbol{g}(\boldsymbol{x}, j, \boldsymbol{a})$ is the factor graph defining a probability distribution of assignment $\boldsymbol{a}$ conditioned on word sequence $\boldsymbol{x}$.

Then JERL, conditional probability of $\boldsymbol{a}$ over $\boldsymbol{x}$, is defined as:

$$P(\boldsymbol{a}|\boldsymbol{x}, \boldsymbol{w}) = \frac{1}{Z(\boldsymbol{x})} e^{\boldsymbol{w} \cdot \boldsymbol{G}(\boldsymbol{x}, \boldsymbol{a})} \quad (1)$$

where $\boldsymbol{w}$ is the weight vector corresponding to $\boldsymbol{G}$ will be learned later, and $Z(\boldsymbol{x})$ is the normalization factor $Z(\boldsymbol{x}) = \sum_{\boldsymbol{a} \in A} e^{\boldsymbol{w} \cdot \boldsymbol{G}(\boldsymbol{x}, \boldsymbol{a})}$, in which A is the union of all possible assignments over $\boldsymbol{x}$.

JERL is a probabilistic graphical model. More specificly, as shown in Figure 1, there are three groups of local functions and one constrain introduced. Each of them take a different role in JERL, as described below:

Features defined on $\boldsymbol{x}$, s_j, y_j, y_{j-1} are written as $\boldsymbol{g}^{ner}(\boldsymbol{x}, s_j, y_j, y_{j-1})$. These functions model segmentation and entity types' distribution over $\boldsymbol{x}$. Actually, every local features used in NER can be formulated in this way, and thus can be included in JERL. We thus refer to them as "NER features".

Features defined on $\boldsymbol{x}$, s_j, $y^e_{j,k}$ are written as $\boldsymbol{g}^{el}(\boldsymbol{x}, s_j, y^e_{j,k})$ and are called "linking features". These features model joint probabilities of word sequence x_{s_j} and linking decisions $y^k_{j,k} = 1(0 \leq k \leq |\mathcal{E}_{s_j}|)$ given context $\boldsymbol{x}$. JERL incorporates all linking features in this way.

Features defined on y_j, $y^e_{j,k}$ are written as $\boldsymbol{g}^{cr}(y_j, y^e_{j,k})$. These features model "mutual dependency" between NER and linking's outputs. For each entity $e_{j,k}$, there is additional information available in the knowledge base, e.g. categories information, popularity and relationship to other entities. These features encourage predicting coherent outputs for NER and linking.

There is one constrain for each $\boldsymbol{y^e_j}$ that the corresponding x_{s_j} can refer to only one entity $e_{j,k} \in \mathcal{E}_{sj}$ or NIL. This is equivalent to $\sum_{k=0}^{|\mathcal{E}_{s_j}|} y^e_{j,k} = 1$.

Based on the above description, $\boldsymbol{G}(\boldsymbol{x}, \boldsymbol{a})$ in equation 1 is the sum of conjunction $(\boldsymbol{g}^{ner}, \boldsymbol{g}^{el}, \boldsymbol{g}^{cr})$ over $\boldsymbol{s}$, and can be rewritten as,

$$\begin{aligned} \boldsymbol{G}(\boldsymbol{x}, \boldsymbol{a}) = \textstyle\sum_{j=1}^{|\boldsymbol{s}|} (\quad & \boldsymbol{g}^{ner}(\boldsymbol{x}, s_j, y_j, y_{j-1}) \\ & , \textstyle\sum_{k=0}^{|\mathcal{E}_j|} \boldsymbol{g}^{el}(\boldsymbol{x}, s_j, y^e_{j,k}) \\ & , \textstyle\sum_{k=0}^{|\mathcal{E}_j|} \boldsymbol{g}^{cr}(\boldsymbol{x}, y_j, y^e_{j,k}) \ ) \end{aligned}$$

In summary, JERL jointly models the NER and linking, and leverages mutual dependency between them to predict coherent outputs. Previous works (Cucerzan, 2007; Ratinov et al., 2011; Sil and Yates, 2013) on linking argued that entity linking systems often suffer because of errors involved in mention detection phrase, especially false negative errors, and try to mitigate it via overgenerating mention candidates. From the mention generation perspective, JERL actually considers every possible assignment and is able to find the optimal $\boldsymbol{a}$.

3.2 Parameter Estimation

We describe how to conduct parameter estimation for JERL in this section. Given independent and identically distributed (i.i.d.) training data $\boldsymbol{T} = \{(\boldsymbol{x_t}, \boldsymbol{a}_t)\}_{t=1}^N$, the goal of parameter estimation is to find optimal $\boldsymbol{w^*}$ to maximize the joint probability of the assignments $\{\boldsymbol{a_t}\}$ over $\{\boldsymbol{x_t}\}$.

$$\boldsymbol{w}^* = argmax_{\boldsymbol{w} \in \Re^{|\boldsymbol{G}|}} \prod_{t=1}^{N} P(\boldsymbol{a_t}|\boldsymbol{x_t}, \boldsymbol{w})$$

We use conditional log likelihood with ℓ_2 norm as the objective function in training,

$$\mathcal{L}(\boldsymbol{T}, \boldsymbol{w}) = \ \textstyle\sum_t logP(\boldsymbol{a_t}|\boldsymbol{x_t}, \boldsymbol{w}) - \frac{1}{2\sigma^2}||\boldsymbol{w}||_2^2$$

The above function is concave, adding regularization to ensure that it has exactly one global optimum. We adopt a limited-memory quasi-Newton method (Liu and Nocedal, 1989) to solve the optimization problem.

[3] In the entity linking task, if a given mention refers to an entity which is not in the knowledge base, linking system should return a special identifier "NIL".

The gradient of $\mathcal{L}(\boldsymbol{T}, \boldsymbol{w})$ is derived as,

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{w}} = \sum_t (\boldsymbol{G}(\boldsymbol{x_t}, \boldsymbol{a_t}) - \sum_{a'} \boldsymbol{G}(\boldsymbol{x_t}, \boldsymbol{a'}) P(\boldsymbol{a'}|\boldsymbol{x_t}, \boldsymbol{w})) - \frac{\boldsymbol{w}}{\sigma^2} \quad (2)$$

As shown in Figure 1, our model's factor graph is a tree, which means the calculation of the gradient is tractable.

Inspired by the forward backward algorithm (Sha and Pereira, 2003) and Semi-CRF (Sarawagi and Cohen, 2004), we leverage dynamic programming techniques to compute the normalization factor $Z_{\boldsymbol{w}}$ and marginal probability $P(\boldsymbol{a'_j}|\boldsymbol{x_t}, \boldsymbol{w})$ when $\boldsymbol{w}$ is given.(Sutton and McCallum, 2006) The parameter estimation algorithm is abstracted in Algorithm 1.

Algorithm 1: JERL parameter estimation

input : training data $\boldsymbol{T} = \{(\boldsymbol{x}_t, \boldsymbol{a}_t)\}_{t=1}^N$
output: the optimal $\boldsymbol{w}$

$\boldsymbol{w} \leftarrow \boldsymbol{0}$;
while *weight* $\boldsymbol{w}$ *is not converged* **do**
  $Z \leftarrow 0$;
  $\boldsymbol{w'} \leftarrow \boldsymbol{0}$;
  for $t \leftarrow 1$ **to** N **do**
    calculate $\boldsymbol{\alpha_t}, \boldsymbol{\beta_t}$ according to eq.3;
    calculate Z_t according to eq.4
    calculate $\boldsymbol{w'_t}$ according to eq.2, 5;
    $Z \leftarrow Z + Z_t$;
    $\boldsymbol{w'} \leftarrow \boldsymbol{w'} + \boldsymbol{w'_t}$;
  end
  update $\boldsymbol{w}$ to maximize log likelihood $\mathcal{L}(\boldsymbol{T}, \boldsymbol{w})$ under $(Z, \boldsymbol{w'})$ via L-BFGS;
end

Let $\alpha_{i,y}$ $(i \in [0, |x|], y \in \mathcal{Y})$ be the sum of potential functions of all possible assignments over $(x_1 \dots x_i)$ whose last segmentation's labels are y. Then $\alpha_{i,y}$ can be calculated recursively from $i = 0$ to $i = |\boldsymbol{x}|$ as below.

We first define base cases as $\alpha_{0,y} = 1|_{\{y \in \mathcal{Y}\}}$. When $i \in (0, |x|]$:

$$\alpha_{i,y} = \sum_{d=1}^{L} \sum_{y' \in \mathcal{Y}} \alpha_{i-d,y'} \psi^{ner}_{i-d+1,i,y,y'} \left(\sum_{\boldsymbol{y_j^e} \in Y_j^{e*}} \psi^{el.cr}_{i-d+1,i,y,\boldsymbol{y_j^e}} \right) \quad (3)$$

where L is the max segmentation length in Semi-CRF, and Y_j^{e*} is all valid assignments for $\boldsymbol{y_j^e}$ which satisfies $\sum_{k=0}^{|\mathcal{E}_{s_j}|} y^e_{j,k} = 1$. The $\psi^{ner}_{u_j,v_j,y_j,y_{j-1}}$ and $\psi^{el.cr}_{u_j,v_j,y_j,\boldsymbol{y_j^e}}$ are precomputed ahead as below,

$$\psi^{ner}_{u_j,v_j,y_j,y_{j-1}} = e^{\boldsymbol{w}^{ner} \cdot \boldsymbol{g}^{ner}(\boldsymbol{x}, s_j, y_j, y_{j-1})}$$

$$\psi^{el.cr}_{u_j,v_j,y_j,\boldsymbol{y_j^e}} = \prod_{k=0}^{|\mathcal{E}_j|} e^{\boldsymbol{w}^{el} \boldsymbol{g}^{el}(\boldsymbol{x}, s_j, y^e_{j,k}) + \boldsymbol{w}^{cr} \boldsymbol{g}^{cr}(y_j, y^e_{j,k})}$$

where $\boldsymbol{w}^{ner}$, $\boldsymbol{w}^{el}$ and $\boldsymbol{w}^{cr}$ are weights for $\boldsymbol{g}^{ner}$, $\boldsymbol{g}^{el}$ and $\boldsymbol{g}^{cr}$ in $\boldsymbol{w}$ accordingly.

The value of $Z_{\boldsymbol{w}}$ can then be written as

$$Z_{\boldsymbol{w}}(\boldsymbol{x}) = \sum_y \alpha_{|\boldsymbol{x}|,y} \quad (4)$$

Define $\beta_{i,y}$ $(i \in [0, |x|], y \in \mathcal{Y})$ as the sum of potential functions of all possible assignments over $(x_{i+1} \dots x_{|\boldsymbol{x}|})$ whose first segmentation's labels are y. $\beta_{i,y}$ is calculated in a similar way, except they are calculated from $i = |x|$ to left $i = 0$.

Once we get $\{\alpha_{i,j}\}$ and $\{\beta_{i,j}\}$, the marginal probability of arbitrary assignment $a_j = (s_j, y_j, \boldsymbol{y_j^e})$, where $s_j = (u_j, v_j)$, can be calculated as below:

$$P(s_j, y_j|\boldsymbol{x}, \boldsymbol{w}) = \frac{(\sum_{y' \in \mathcal{Y}} \alpha_{u_j - 1, y'} \psi^{ner}_{u_j,v_j,y_j,y'}) \beta_{v_j,y_j}}{Z_{\boldsymbol{w}}(\boldsymbol{x})}$$

and

$$P(a_j|\boldsymbol{x_t}, \boldsymbol{w}) = P(s_j, y_j|\boldsymbol{x}, \boldsymbol{w}) \frac{\psi^{el.cr}_{u_j,v_j,y_j,\boldsymbol{y_j^e}}}{\sum_{\boldsymbol{y_{j'}^e} \in Y_j^{e*}} \psi^{el.cr}_{u_j,v_j,y_j,\boldsymbol{y_{j'}^e}}} \quad (5)$$

3.3 Inference

Given a new word sequence $\boldsymbol{x}$ and model weights $\boldsymbol{w}$ trained on a training set, the goal of inference is to find the best assignment, $\boldsymbol{a^*} = argmax_{\boldsymbol{a}} P(\boldsymbol{a}|\boldsymbol{x}, \boldsymbol{w})$ for $\boldsymbol{x}$. We extend the Viterbi algorithm to exactly infer the best assignment. The inference algorithm is shown in Algorithm 2.

Let $\phi(u_j, v_j, y_j, y_{j-1})$ be the product of potentials depending on (s_j, y_j, y_{j-1}) as,

$$\phi(u_j, v_j, y_j, y_{j-1}) = \psi^{ner}_{u_j,v_j,y_j,y_{j-1}} \left(\sum_{\boldsymbol{y_j^e} \in Y_j^{e*}} \psi^{el.cr}_{u_j,v_j,y_j,\boldsymbol{y_j^e}} \right) \quad (6)$$

Algorithm 2: JERL inference

input : one word sequence $\boldsymbol{x}$ and weights $\boldsymbol{w}$
output: the best assignment $\boldsymbol{a}$ over $\boldsymbol{x}$

// shrink JERL graph to a Semi-CRF graph;
for $u \leftarrow 1$ **to** $|\boldsymbol{x}|$ **do**
  for $v \leftarrow u+1$ **to** $|\boldsymbol{x}|$ **do**
    for $(y, y') \in \mathcal{Y} \times \mathcal{Y}$ **do**
      calculate $\phi_{u,v,y,y'}$ // see eq.6;
    end
  end
end
// infer the best assignment of $(\boldsymbol{s}^*, \boldsymbol{y}^*)$;
for $i \leftarrow 1$ **to** $|\boldsymbol{x}|$ **do**
  for $y \in \mathcal{Y}$ **do**
    calculate $V_{i,y}$ // see eq.7;
  end
end
$(\boldsymbol{s}^*, \boldsymbol{y}^*) \leftarrow argmax(V_{i,y})$;
// infer the best assignment of $\{\boldsymbol{y}_{\boldsymbol{j}}^{\boldsymbol{e}}\}$;
for $j \leftarrow 1$ **to** $|\boldsymbol{s}^*|$ **do**
  $\boldsymbol{y}_{\boldsymbol{j}}^{\boldsymbol{e}} \leftarrow argmax(P(|\boldsymbol{x}, \boldsymbol{w}, \boldsymbol{s}_{\boldsymbol{j}}^*, \boldsymbol{y}_{\boldsymbol{j}}^*))$
end
$\boldsymbol{a}^* \leftarrow (\boldsymbol{s}^*, \boldsymbol{y}^*, \boldsymbol{y}^{\boldsymbol{e}*})$;

and let $V(i, y)$ denotes the largest value of $(\boldsymbol{w} \cdot \boldsymbol{G}(\boldsymbol{x}, \boldsymbol{a'}))$ where $\boldsymbol{a'}$ could be any possible partial assignment starting from x_1 to x_i. The best $(\boldsymbol{s}^*, \boldsymbol{y}^*)$ are derived during the following recursive calculation,

$$V_{i,y} = \begin{cases} max_{y' \in \mathcal{Y}, d \in [1,L]} \\ (V_{i-d,y'} + \phi(i-d+1, i, y, y')) & i > 0 \\ 0 & i = 0 \\ -\infty & i < 0 \end{cases} \quad (7)$$

where L is the maximum segmentation length for Semi-CRF.

Once $(\boldsymbol{s}^*, \boldsymbol{y}^*)$ are found, the corresponding $\boldsymbol{y}_{\boldsymbol{j}}^{\boldsymbol{e}*} = argmax_{\{\boldsymbol{y}_{\boldsymbol{l}}^{\boldsymbol{e}} \in Y_j^{e*}\}}(\psi^{el.cr}_{u_j^*, v_j^*, y_j^*, \boldsymbol{y}_{\boldsymbol{l}}^{\boldsymbol{e}}})$ is also the optimal one. Then $\boldsymbol{a}^* = (\boldsymbol{s}^*, \boldsymbol{y}^*, \boldsymbol{y}^{\boldsymbol{e}*})$ is the best assignment for the given $\boldsymbol{x}$ and $\boldsymbol{w}$.

4 Experiments

In our experiments, we first construct two baseline models JERL$_{ner}$ and JERL$_{el}$, which use exact NER and EL feature sets used in JERL. Then evaluate JERL and the two baseline models against several state-of-art NER and linking systems. After that, we evaluate JERL under different feature settings to analysis the contributions of each features set, and show some examples we find. We also compare the training speed under different settings.

CoNLL'03	Training	Dev set	Test
Articles	946	216	231
Sentences	14,987	3,466	3,684
Tokens	203,621	51,362	46,435
Entities	23,499	5,942	5,648
NIL Entities	4,857	1,129	1,133

Table 1: Overview of CoNLL'03/AIDA data set

4.1 Data set

We take the CoNLL'03/AIDA English data set to evaluate the performance of NER and linking systems. CoNLL'03 is extensively used in prior work on NER evaluation (Tjong Kim Sang and De Meulder, 2003). The English data is taken from Reuters news articles published between August 1966 and August 1997. Four types of entities persons (PER), organizations (ORG), locations (LOC), and miscellaneous names (MISC) are annotated. Hoffart et al. (2011) hand-annotated all proper nouns with corresponding entities wiht YAGO2, Freebase and Wikipedia IDs. This data is referenced as AIDA here. To the best of our knowledge, this data set is the biggest data set which has been labeled for both NER and linking tasks. It becomes a really good starting point for our work. Table 1 contains of an overview of the CoNLL'03/AIDA data set.

For entity linking, we take Wikipedia as the referent knowledge base. We use a Wikipedia snapshot dumped in May 2013, which contains around 4.8 million articles. We also align our Wikipedia dump with additional knowledge bases, Freebase and Satori (a Microsoft internal knowledge base), to enrich the information of these entities.

4.2 Evaluation Metrics

We follow the CoNLL'03 metrics to evaluate NER performance by precision, recall, and F_1 scores, and follow Hoffart's (2011) experiment setting to evaluate linking performance by micro precision@1. Since the linking labels of CONLL'03 were annotated in 2011, it is not completely consistent with the Wikipedia dump we used in the case. We only consider mention entity pairs where the ground truth are known, and ignore around 20% of NIL mentions in the ground truth.

Category	Features
NER	Word unigram / bigram
	Lower cased unigram / bigram
	Word shape unigram / bigram
	Stemmed unigram / bigram
	POS unigram / bigram
	Chunk unigram / bigram
	Words in the 4 left/right window
	Character n-grams, $n \leq 4$
	Brown clusters
	WordNet clusters
	Dictionaries
Linking	Alternative names
	Entity priors
	Entity name priors
	Entity priors over names
	Context scores
	Geo distance
	Related entities
Mutual	Type-category correlation

Table 2: JERL feature list

4.3 JERL Implementation

Table 2 shows features used in our models. JERL uses all features in the three categories, while JERL$_{ner}$ and JERL$_{el}$ use only one corresponding category. All three models are trained on the train and development set, and evaluated on the test set of CoNLL'03/AIDA.

4.3.1 NER

Features in the NER category are relevant to NER. We considered the most commonly used features in literatures (Finkel et al., 2005; Liang, 2005; Ratinov and Roth, 2009). We collect several known name lists, like popular English first/last names for people, organization lists and so on from Wikipedia and Freebase. UIUC NER's lists are also included. In addition, we extract entity name lists from the knowledge base we used for entity linking, and construct 655 more lists. Although those lists are noisy, we find that statistically they do improve the performance of our NER baseline by a significant amount.

4.3.2 Linking

Features in linking category are relevant to entity linking. An entity can be referred by its canonical name, nick names, alias, and first/last names. Those names are defined as alternative names for this entity. We collect all alternative names for all known entities and build a name to entity index. This index is used to select entity candidates for any word sequence, also known as surface form. Following previous work by Han and Sun (2011), we calculate entity priors and entity name priors from Wikipedia. Context scores are calculated based on discriminative keywords. Geo distance and related entities capture the relatedness among entities in the given context.

4.3.3 Mutual

Features in this category capture the mutual dependency between NER and linking's outputs. For each entity in a knowledge base, there is category information available. We aggregate around 1000 distinct categories from multiple sources. One entity can have multiple categories. For example, London is connected to 29 categories. We use all combinations between NER types and categories as features in JERL, and let the model learn the correlation of each combination. This encourages coherent NER and EL decisions, which is one of the key contributions of our work.

4.3.4 Non-local features

Features capturing long distance dependency between hidden labels are classified as non-local features. Those features are very helpful in improving NER system performance but are costly. Since this is not the focus of this paper, we take a simple approach to incorporate non-local features. We cache history results of previous sentences in a 1000 words window, and adopt several heuristic rules for personal names. This approach contributes 0.2 points to the final NER F_1 score. Non-local features are also considered in linking (Ratinov et al., 2011; Han et al., 2011). We try several features, which has been proved to be helpful in TAC data set. However, the gain on CoNLL'03/AIDA data set is not obvious, we do not optimize linking globally.

Lastly, based on preliminary studies and experiments, we set the maximum segmentation length to 6 and max candidate count per segmentation to 5 for efficient training and inference.

4.4 State-of-Art systems

We take three state-of-art NER systems: NereL (Sil and Yates, 2013), UIUC NER (Ratinov and Roth, 2009) and Stanford NER (Finkel et al., 2005). NereL firstly over generates mentions and decomposes them to sets of connected compo-

Dataset	System	Prec.	Recall	F_1
CoNLL'03	Stanford	**95.1**	78.3	85.9
	UIUC	91.2	90.5	90.8
	NereL	86.8	89.5	88.2
	$JERL_{ner}$	90.0	89.9	89.9
	$JERL$	91.5	**91.4**	**91.2**

Table 3: NER evaluation results

Dataset	System	Precision@1
CoNLL'03	Kul09	76.74
	Hof11	81.91
	Nerel	84.22
	JERL_{el}	81.49
	JERL	**84.58**

Table 4: Linking evaluation results

#	Feature set description	NER F_1
0	JERL_{ner} (baseline)	89.9
1	+ candidate	88.7
2	+ candidate + linking	89.9
3	+ candidate + mutual	90.6
4	+ candidate + mutual + linking	91.2

Table 5: JERL features analysis

nents, then trains a maximum-entropy model to re-rank different assignments. UIUC NER uses a regularized averaged perceptron model and external gazetteers to achieve strong performance. In Addition, NereL also uses UIUC NER to generate mentions. Stanford NER uses Conditional Random Fields and Gibbs sampling to incorporate non-local features into its model.

For entity linking systems, NereL, Kul09 (Kulkarni et al., 2009) and Hof11 (Hoffart et al., 2011) are compared with our models. NereL achieves the best precision@1. Kul09 formulates the local compatibility and global coherence in entity linking, and optimizes the overall entity assignment for all entities in a document via a local hill-climbing approach. Hof11 unifies the prior probability of an entity being mentioned, the similarity between context and entity, and the coherence between entity candidates among all mentions in a dense graph.

4.5 Results

Table 3 shows the performance of different NER systems on the CoNLL'03 testb data set. We refer the numbers of state-of-art systems reported by Sil and Yates (2013). Stanford NER achieves the best precision, but its recall is low. UIUC reports the (almost) best recorded F_1. JERL_{ner} considers features only in the NER category, which could be treated as a pure NER system implemented in Semi-CRF. Actually CRF-based implementation with a similar feature set has comparable performance. Our baseline JERL_{ner} is strong enough. We argue that that it is mainly because of the additional dictionaries derived from the knowledge base. JERL further pushes the F_1 to 91.2, which outperforms UIUC by 0.4 points in F_1 score. To the best of our knowledge, it is the best F_1 on CoNLL'03 since 2009. The reason our model can outperform state-of-art systems is that, it has more knowledge about entities via incorporate entity linking techniques. If an entity can be linked to a well known entity via entity linking in high confidence, its mention boundary and entity type are confirmed implicitly.

Table 4 shows the performance of different entity linking systems on the AIDA test set. Kul09 and Hof11 use only the correct mentions detected by the Stanford NER as input, and thus their recall is bound by the recall of NER. NereL uses its overgeneration techniques to generate mention candidates, and outperforms Hoff11 in both precision and recall. Our baseline model JERL_{el} is also evaluated on Stanford NER generated mentions, which has comparable performance with Kul09 and Hof11. JERL achieves precision@1 84.58 which is better than NereL.

We run 15 trials for both NER and linking's experiments and report the average numbers above. The standard deviations are 0.11% and 0.08% for NER and linking separately, which pass the standard t-test with confidence level 5%, demonstrating the significance of our results.

In order to investigate how different features contribute to the overall gain. We compare JERL_{ner} with four different feature sets. Table 5 summaries the results. In the trial "+candidate", JERL expands every possible segmentation with corresponding entity list and builds its factor graph without any linking and mutual features. This version's F_1 drops to 88.7 which indicates the created structure is quite noisy. In the "+candidate +linking" trial, only linking features are enabled and the F_1 is comparable to the baseline. On the other side, in the "+candidate +mutual" trial when mutual features are enabled the F_1 increases to 90.6. If we combine both linking and mutual features,

Category	PER	LOC	ORG	Other
people.person	3.65	0.817	1.260	-1.782
location.city	-0.187	0.712	0.491	-0.188
sports.team	-0.180	2.382	3.595	-2.019

Table 6: Learned mutual dependency

Setting		NER	Linking	Training
MSL	MRC	F_1	prec@1	time (min)
4	5	87.9	76.74	195
5	5	90.8	84.01	234
6	1	90.8	80.13	37
6	3	91.0	83.21	109
6	5	91.2	84.58	280

Table 7: Training time under different settings

JERL achieves the reported performance. The result indicates that mutual features are the determining factor to the performance gain.

Table 6 shows weights of learned mutual dependency of three categories "people.person", "location.city", and "sports.team". The bigger a weight is, the more consistent this combination would be. From the weights, we find several interesting things. If an entity belongs to any of the three categories, it is less likely to be predicted as non-an-entity by NER. If an entity belongs to the category of "people.person", it more likely to be predicted as PER. When an entity belongs to the category "location.city" or "sports.team", NER may predict it as ORG or LOC. This is because in the CoNLL'03/AIDA data set, there are many sports teams mentioned by their city/country names. JERL successfully models such unexpected mutual dependency.

Table 7 compares the performance and training time under different settings of max segmentation length (MSL) and max referent count (MRC). We use machines with Intel Xeon E5620 @ 2.4GHz CPU (8 cores / 16 logical processors) and 48GB memory. We run every setting 10 times and report the averages. As MSL and MRC increasing, the performance is slightly better, but the training time increased a lot. MSL has linear impact on training time, while MRC affects training time more.

5 Conclusion and Future Work

In this paper, we address the problem of joint optimization of named entity recognition and linking. We propose a novel model, JERL, to jointly train and infer for NER and linking tasks. To the best of our knowledge, this is the first model which trains two tasks at the same time. The joint model is able to leverage mutual dependency of the two tasks, and predict coherent outputs. JERL outperforms the state-of-art systems on both NER and linking tasks on the CoNLL'03/AIDA data set.

For future works, we would like to study how to leverage existing partial labeled data, either for NER or for linking only, in joint optimization, and incorporate more NLP tasks together for multi-tasks joint optimization.

Acknowledgments

This work was performed when the first author was working at Microsoft Research. The first author is sponsored by Microsoft Bing Core Relevance team. Thanks Shuming Shi, Bin Gao, and Yohn Cao for their helpful guidance and valuable discussions. Additionally, we would like to thank the three anonymous reviewers for their insightful suggestions and detailed comments.

References

Razvan C Bunescu and Marius Pasca. 2006. Using encyclopedic knowledge for named entity disambiguation. In *EACL*, volume 6, pages 9–16.

Nancy Chinchor and Elaine Marsh. 1998. Muc-7 information extraction task definition. In *Proceeding of the seventh message understanding conference (MUC-7), Appendices*, pages 359–367.

Silviu Cucerzan. 2007. Large-scale named entity disambiguation based on wikipedia data. In *EMNLP-CoNLL*, volume 7, pages 708–716.

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

Jenny Rose Finkel and Christopher D Manning. 2009. Joint parsing and named entity recognition. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 326–334. Association for Computational Linguistics.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 363–370. Association for Computational Linguistics.

Xianpei Han and Le Sun. 2011. A generative entity-mention model for linking entities with knowledge base. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 945–954. Association for Computational Linguistics.

Xianpei Han, Le Sun, and Jun Zhao. 2011. Collective entity linking in web text: a graph-based method. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 765–774. ACM.

Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. 2011. Robust disambiguation of named entities in text. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 782–792. Association for Computational Linguistics.

Frank R Kschischang, Brendan J Frey, and H-A Loeliger. 2001. Factor graphs and the sum-product algorithm. *Information Theory, IEEE Transactions on*, 47(2):498–519.

Sayali Kulkarni, Amit Singh, Ganesh Ramakrishnan, and Soumen Chakrabarti. 2009. Collective annotation of wikipedia entities in web text. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 457–466. ACM.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data.

Percy Liang. 2005. *Semi-supervised learning for natural language*. Ph.D. thesis, Massachusetts Institute of Technology.

Dekang Lin and Xiaoyun Wu. 2009. Phrase clustering for discriminative learning. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 1030–1038. Association for Computational Linguistics.

Dong C Liu and Jorge Nocedal. 1989. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1-3):503–528.

Andrew McCallum, Khashayar Rohanimanesh, and Charles Sutton. 2003. Dynamic conditional random fields for jointly labeling multiple sequences. In *nips workshop on syntax, semantics and statistics*.

Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*, pages 147–155. Association for Computational Linguistics.

Lev Ratinov, Dan Roth, Doug Downey, and Mike Anderson. 2011. Local and global algorithms for disambiguation to wikipedia. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 1375–1384. Association for Computational Linguistics.

Sunita Sarawagi and William W Cohen. 2004. Semi-markov conditional random fields for information extraction. In *Advances in Neural Information Processing Systems*, pages 1185–1192.

Fei Sha and Fernando Pereira. 2003. Shallow parsing with conditional random fields. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 134–141. Association for Computational Linguistics.

Avirup Sil and Alexander Yates. 2013. Re-ranking for joint named-entity recognition and linking. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*, pages 2369–2374. ACM.

Rosa Stern, Benoît Sagot, and Frédéric Béchet. 2012. A joint named entity recognition and entity linking system. In *Proceedings of the Workshop on Innovative Hybrid Approaches to the Processing of Textual Data*, pages 52–60. Association for Computational Linguistics.

Mihai Surdeanu, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. The conll-2008 shared task on joint parsing of syntactic and semantic dependencies. In *Proceedings of the Twelfth Conference on Computational Natural Language Learning*, pages 159–177. Association for Computational Linguistics.

Charles Sutton and Andrew McCallum. 2006. An introduction to conditional random fields for relational learning. *Introduction to statistical relational learning*, pages 93–128.

Erik F Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 142–147. Association for Computational Linguistics.

Longyue Wang, Shuo Li, Derek F Wong, and Lidia S Chao. 2012. A joint chinese named entity recognition and disambiguation system. *CLP 2012*, page 146.

Xiaofeng Yu, Irwin King, and Michael R Lyu. 2011. Towards a top-down and bottom-up bidirectional approach to joint information extraction. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 847–856. ACM.

How Much Information Does a Human Translator Add to the Original?

Barret Zoph, Marjan Ghazvininejad and Kevin Knight
Information Sciences Institute
Department of Computer Science
University of Southern California
{zoph,ghazvini,knight}@isi.edu

Abstract

We ask how much information a human translator adds to an original text, and we provide a bound. We address this question in the context of bilingual text compression: given a source text, how many bits of additional information are required to specify the target text produced by a human translator? We develop new compression algorithms and establish a benchmark task.

1 Introduction

Text compression exploits redundancy in human language to store documents compactly, and transmit them quickly. It is natural to think about compressing bilingual texts, which have even more redundancy:

> "From an information theoretic point of view, accurately translated copies of the original text would be expected to contain almost no extra information if the original text is available, so in principle it should be possible to store and transmit these texts with very little extra cost." (Nevill and Bell, 1992)

Of course, if we look at actual translation data (Figure 1), we see that there is quite a bit of unpredictability. But the intuition is sound. If there were a million equally-likely translations of a short sentence, it would only take us $\log_2(1\text{m}) = 20$ bits to specify which one.

By finding and exploiting patterns in bilingual data, we want to provide an upper bound for this question: *How much information does a human translator add to the original?* We do this in the context of building a practical compressor for bilingual text.

上个星期的战斗至少夺取12个人的生命。
At least 12 people were killed in the battle last week.
Last week's fight took at least 12 lives.
The fighting last week killed at least 12.
The battle of last week killed at least 12 persons.
At least 12 people lost their lives in last week's fighting.
At least 12 persons died in the fighting last week.
At least 12 died in the battle last week.
At least 12 people were killed in the fighting last week.
During last week's fighting, at least 12 people died.
Last week at least twelve people died in the fighting.
Last week's fighting took the lives of twelve people.

Figure 1: Eleven human translations of the same source sentence (LDC2002T01).

We adopt the same scheme used in monolingual text compression benchmark evaluations, such as the Hutter Prize (Hutter, 2006), a competition to compress a 100m-word extract of English Wikipedia. A valid entry is an executable, or self-extracting archive, that prints out Wikipedia, byte-for-byte. Decompression code, dictionaries, and/or other resources must be embedded in the executable—we cannot assume that the recipient of the compressed file has access to those resources. This view of compression goes by the name of algorithmic information theory (or Kolmogorov complexity).

Any executable is permitted. For example, if our job were to compress the first million digits of π, then we might submit a very short piece of code that prints those digits. The brevity of that compression would demonstrate our understanding of the sequence. Of course, in our application, we will find it useful to develop generic algorithms that can compress any text.

Our approach will be as follows. Given a bilingual text (`file1` and `file2`), we develop this compression interface:

```
% compress file1 > file1.exe
% bicompress file2 file1 > file2.exe
```

The second command compresses `file2` while looking at `file1`. We take the size of `file1.exe`

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 889–898,
Lisbon, Portugal, 17-21 September 2015.

	Spanish	English
Uncompressed size	324.9 Mb	294.5 Mb
Word tokens	57,068,133	54,364,566
Vocabulary size	195,314	140,340
Distinct word cooc	93,184,127	
Segment pairs	1,965,734	
Ave. segment length (word tokens)	29.0	27.7
Longest segment (word tokens)	809	880

Figure 2: Large Europarl Spanish/English corpus.

	Spanish	English
Uncompressed size	32.3 Mb	29.3 Mb
Word tokens	5,682,667	5,426,131
Vocabulary size	73,726	45,423
Distinct word cooc	21,231,874	
Segment pairs	196,573	
Ave. segment length (word tokens)	28.9	27.6
Longest segment (word tokens)	733	682

Figure 3: Small Europarl Spanish/English corpus.

as the amount of information in the original text. We bound how much information the translator adds to the original by:

$|$`file2.exe`$|$ / $|$`file1.exe`$|$

We can say that bilingual compression is more effective that monolingual compression if:

$|$`file2.exe`$| < |$`file3.exe`$|$, where

```
% compress file2 > file3.exe
```

Our decompression interface is:

```
% file1.exe > file1
% file2.exe file1 > file2
```

The second command decompresses `file2` while looking at (uncompressed) `file1`.

The contributions of this paper are:

1. We provide a new quantitative bound for how much information a translator adds to an original text.
2. We present practical software to compress bilingual text with compression rates that exceed the previous state-of-the-art.
3. We set up a public benchmark bilingual text compression challenge to stimulate new researchers to find and exploit patterns in bilingual text. Ultimately, we want to feed those ideas into practical machine translation systems.

2 Data

We propose the widely accessible Spanish/English Europarl corpus v7 (Koehn, 2005) as a benchmark for bilingual text compression (Figure 2). Portions of this large corpus have been used in previous compression work (Sánchez-Martínez et al., 2012). The Spanish side is in UTF-8. For English, we have removed accent marks and further eliminated all but the 95 printable ASCII characters (Brown et al., 1992), plus newline.

Our task is to compress the data "as is": untokenized, but already segment aligned. We also include a tokenized version with 334 manually word-aligned segment pairs (Lambert et al., 2005) distributed throughout the corpus.

For rapid development and testing, we have arranged a smaller corpus that is 10% the size of the full corpus (Figure 3).

3 Monolingual compression

Compression captures patterns in data. Language modeling also captures patterns, but at first blush, these two areas seem distinct. In compression, we seek a small executable that prints out a text, while in language modeling, we seek an executable that assigns low perplexity to held-out test data.[1] Actually, the two areas have much more in common, as a review of compression algorithms reveals.

Huffman coding. A well-known compression technique is to create a binary Huffman tree whose leaves are characters in the text,[2] and whose edges are labeled 0 or 1 (Huffman and others, 1952). The tree is arranged so that frequent characters have short binary codes (edge sequences). It is very important that the Huffman tree for a particular text be included at the beginning of the compressed file, so that decompression knows how to process the compressed bit string.

Adaptive Huffman. Actually, we can avoid shipping the Huffman tree inside the compressed file, by building the tree adaptively, as the compressor processes the input text. If we start with a uniform distribution, the first few characters may not compress very well, but soon we will converge onto a good tree and good compression. It is very

[1] File size has advantages, as perplexity computations are often buggy, and they usually gloss over how probability is apportioned to out-of-vocabulary items.

[2] Or other symbols, such as words, bytes, or unicode sequences.

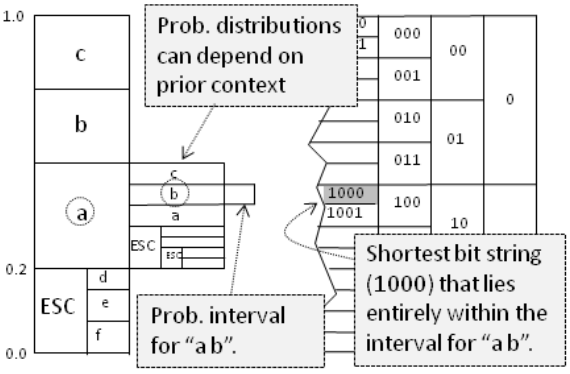


Figure 4: Arithmetic coding.

important that the decompressor exactly recapitulate the same sequence of Huffman trees that the compressor made. It can do this by counting characters as it outputs them, just as the compressor counted characters as it consumed them.

Adaptive compression can also nicely accommodate shifting topics in text, if we give higher counts to recent events. By its single-pass nature, it is also good for streaming data.

Arithmetic coding. Huffman coding exploits a predictive unigram distribution over the next character. If we use more context, we can make sharper distributions. An n-gram table is one way to map contexts onto predictions.

How do we convert good predictions into good compression? The solution is called *arithmetic coding* (Rissanen and Langdon Jr., 1981; Witten et al., 1987). Figure 4 sketches the technique. We produce context-dependent probability intervals, and each time we observe a character, we move to its interval. Our working interval becomes smaller and smaller, but the better our predictions, the wider it stays. A document's compression is the shortest bit string that fits inside the final interval. In practice, we do the bit-coding as we navigate probability intervals.

Arithmetic coding separates modeling and compression, making our job similar to language modeling, where we use try to use context to predict the next symbol.

3.1 PPM

PPM is the most well-known adaptive, predictive compression technique (Cleary and Witten, 1984). PPM updates character n-gram tables (usually n=1..5) as it compresses. In a given context, an n-gram table may predict only a subset of characters, so PPM reserves some probability mass for an escape (ESC), after which it executes a hard backoff to the (n-1)-gram table. In PPMA, P(ESC) is 1/(1+D), where D is the number of times the context has been seen. PPMB uses q/D, where q is the number of distinct character types seen in the context. PPMC uses q/(q+D), aka Witten-Bell. PPMD uses q/2D.

PPM* uses the shortest previously-seen deterministic context, which may be quite long. If there is no deterministic context, PPM* goes to the longest matching context and starts PPMD. Instead of the longest context, PPMZ rates all contexts between lengths 0 and 12 according to each context's most probable character. PPMZ also implements an adaptive P(ESC) that combines context length, number of previous ESC in the context, etc.

We use our own C++ implementation of PPMC for monolingual compression experiments in this paper. When we pass over a set of characters in favor of ESC, we remove those characters from the hard backoff.

3.2 PAQ

PAQ (Mahoney, 2005) is a family of state-of-the-art compression algorithms and a perennial Hutter Prize winner. PAQ combines hundreds of models with a logistic unit when making a prediction. This is most efficient when predictions are at the bit-level instead of the character-level. The unit's model weights are adaptively updated by:

$w_i \leftarrow w_i + \eta x_i(correct - \mathrm{P}(1))$, where
$x_i = ln(\mathrm{P}_i(1)/(1 - \mathrm{P}_i(1))$
η = fixed learning rate
$\mathrm{P}_i(1)$ = ith model's prediction

PAQ models include a character n-gram model that adapts to recent text, a unigram word model (where word is defined as a subsequence of characters with ASCII > 32), a bigram model, and a skip-bigram model.

4 Bilingual Compression: Prior Work

Nevill and Bell (1992) introduce the concept but actually carry out experiments on paraphrase corpora, such as different English versions of the Bible.

Conley and Klein (2008) and Conley and Klein (2013) compress a target text that has been word-aligned to a source text, to which they add a lemmatizer and bilingual glossary. They obtain a 1%-6% improvement over monolingual compression,

without counting the cost of auxiliary files needed for decompression.

Martínez-Prieto et al. (2009), Adiego et al. (2009), Adiego et al. (2010) rewrite bilingual text by first interleaving source words with their translations, then compressing this sequence of biwords. Sánchez-Martínez et al. (2012) improve the interleaving scheme and include offsets to enable decompression to reconstruct the original word order. They also compare several character-based and word-based compression schemes for biword sequences. On Spanish-English Europarl data, they reach an 18.7% compression rate on word-interleaved text, compared to 20.1% for concatenated texts, a 7.2% improvement.

Al-Onaizan et al. (1999) study the perplexity of learned translation models, i.e., the probability assigned to the target corpus given the source corpus. They observed iterative training to improve training-set perplexity (as guaranteed) but degrade test-set perplexity. They hypothesized that an increasingly tight, unsmoothed translation dictionary might exclude word translations needed to explain test-set data. Subsequently, research moved to extrinsic evaluation of translation models, in the context of end-to-end machine translation.

Foster et al. (2002) and others have used prediction to propose auto-completions to speed up human translation. As we have seen, prediction and compression are highly related.

5 Predictive Bilingual Compression

Our algorithm compresses target-language `file2` while looking at source-language `file1`:

```
% bicompress file2 file1 > file2.exe
```

To make use of arithmetic coding, we consider the task of predicting the next target character, given the source sentence and target string so far:[3]

$$\mathrm{P}(e_j | f_1 \ldots f_l, e_1 \ldots e_{j-1})$$

If we are able to accurately predict what a human translator will type next, then we should be able to build a good machine translator. Here is an example of the task:

Spanish:	Pido que hagamos un minuto de silencio.
English so far:	I should like to ob_

[3] We predict e from f in this paper, reversed from Brown et al. (1993), who predict f from e.

	Absolute offsets	Relative offsets
Uncompressed	105.3 Mb	88.9 Mb
Huffman coding	36.6 Mb	24.4 Mb
PPMC	12.4 Mb	13.2 Mb

Figure 5: Compressing a file of (unidirectional) automatic Viterbi word alignments computed from our large Spanish/English corpus (sentences less than 50 words).

5.1 Word alignment

Let us first work at the word level instead of the character level. If we are predicting the jth English word, and we know that it translates f_i ("aligns to f_i"), and if f_i has only a handful of translations, then we may be able to specify e_j with just a few bits. We may therefore suppose that a set of Viterbi word alignments may be useful for compression (Conley and Klein, 2008; Sánchez-Martínez et al., 2012).

We consider unidirectional alignments that link each target position j to a single source position i (including the null word at $i = 0$). Such alignments can be computed automatically using EM (Brown et al., 1993), and stored in one of two formats:

Absolute: 1 2 5 5 7 0 3 6 ...
Relative: +1 +1 +3 0 +2 null -4 +3 ...

In order to interpret the bits produced by the compressor, our decompressor *must also have access to the same Viterbi alignments*. Therefore, we must include those alignments at the beginning of the compressed file. So let's compress them too.

How compressible are alignment sequences? Figure 5 gives results for Viterbi alignments derived from our large parallel Spanish/English corpus. First, some interesting facts:

- Huffman works better on relative offsets, because the common "+1" gets a short bit code.
- PPMC's use of context makes it impressively insensitive to alignment format.
- PPMC beats Huffman on relative offsets. This would not happen if relative offset integers were independent of one another, as assumed by (Brown et al., 1993) and (Vogel et al., 1996). Bigram statistics bear this out:
  P(+1 | -2) = 0.20 P(+1 | +1) = 0.59
  P(+1 | -1) = 0.20 P(+1 | +2) = 0.49
  P(+1 | 0) = 0.52

So this small compression experiment already

suggests that translation aligners might want to model more context than just P(offset).

However, the main point of Figure 5 is that the compressed alignment file requires 12.4 Mb! This is too large for us to prepend to our compressed file, for the sake of enabling decompression.

5.2 Translation dictionary

Another approach is to forget Viterbi alignments and instead exploit a probabilistic translation dictionary table $t(e|f)$. To predict the next target word e_j, we admit the possibility that e_j might be translating *any* of the source tokens. IBM Model 2 (Brown et al., 1993) tells us how to do this:

Given $f_1 \ldots f_l$:

1. Choose English length m $\quad \epsilon(m|l)$
2. For $j = 1..m$, choose alignment a_j $\quad a(a_j|j,l)$
3. For $j = 1..m$, choose translation e_j $\quad t(e_j|f_{a_j})$

which, via the "IBM trick" implies:

$\mathrm{P}(e_1 \ldots e_m|f_1 \ldots f_l) =$
$\epsilon(m|l) \prod_{j=1}^{m} \sum_{i=0}^{l} a(i|j,l)t(e_j|f_i)$

In compression, we must predict English words incrementally, before seeing the whole string. Furthermore, we must predict $\mathrm{P}(STOP)$ to end the English sentence. We can adapt IBM Model 2 to make incremental predictions:

$\mathrm{P}(STOP|f_1 \ldots f_l, e_1 \ldots e_{j-1}) \sim$
$\mathrm{P}(STOP|j,l) =$
$\epsilon(j-1|l) / \sum_{k=j-1}^{max} \epsilon(k|l)$

$\mathrm{P}(e_j|f_1 \ldots f_l, e_1 \ldots e_{j-1}) \sim$
$\mathrm{P}(e_j|f_1 \ldots f_l) =$
$[1 - \mathrm{P}(STOP|j,l)] \sum_{i=0}^{l} a(i|j,l)t(e_j|f_i)$

We can train t, a, and ϵ on our bilingual text using EM (Brown et al., 1993). However, the t-table is still too large to prepend to the compressed English file.

5.3 Adaptive translation modeling

Instead, inspired by PPM, we build up translation tables in RAM, during a single pass of our compressor. Our decompressor then rebuilds these same tables, in the same way, in order to interpret the compressed bit string.

Neal and Hinton (1998) describe *online EM*, which updates probability tables after each training example. Liang and Klein (2009) and Levenberg et al. (2010) apply online EM to a number of language tasks, including word alignment. Here we concentrate on the single-pass case.

We initialize a uniform translation model, use it to collect fractional counts from the first segment pair, normalize those counts to probabilities, use those new probabilities to collect fractional counts from the second segment pair, and so on. Because we pass through the data only once, we hope to converge quickly to high-quality tables for compressing the bulk of the text.

Unlike in batch EM, we need not keep separate count and probability tables. We only need count tables, including summary counts for normalization groups, so memory savings are significant. Whenever we need a probability, we compute it on the fly. To avoid zeroes being immediately locked in, we invoke add-λ smoothing every time we compute a probability from counts:[4]

$t(e|f) = \frac{count(e,f)+\lambda_t}{count(f)+\lambda_t|V_E|}$

$a(i|j,l) = \frac{count(i,j,l)+\lambda_a}{count(j,l)+\lambda_a(l+1)}$

where $|V_E|$ is the size of the English vocabulary. We determine $|V_E|$ via a quick initial pass through the data, then include it at the top of our compressed file.

In batch EM, we usually run IBM Model 1 for a few iterations before Model 2, gripped by an atavistic fear that the a probabilities will enforce rigid alignments before word co-occurrences have a chance to settle in. It turns out this fear is justified in online EM! Because the a table initially learns to align most words to null, we smooth it more heavily ($\lambda_a = 10^2$, $\lambda_t = 10^{-4}$).

We also implement a single-pass HMM alignment model (Vogel et al., 1996). In the IBM models, we can either collect fractional counts after we have compressed a whole sentence, or we can do it word-by-word. In the HMM model, alignment choices are no longer independent of one another:

Given $f_1 \ldots f_l$:

1. Choose English length m w/prob $\epsilon(m|l)$
2. For $j = 1..m$:
   2a. set a_j to null w/prob p_1, *or*
   2b. choose non-null a_j w/prob $(1-p_1)o(a_j - a_k)$
3. For $j = 1..m$, choose translation e_j w/prob $t(e_j|f_{a_j})$

In the expression $o(a_j - a_k)$, k is the maximum English index ($k < j$) such that $a_k \neq 0$. The relative offset o-table learns to encourage adjacent English words to align to adjacent Spanish words.

Batch HMM performs poorly under uniform initialization, with two causes of failure. First, EM training sets $o(0)$ too high, leading to absolute alignments like "1 2 2 2 5 5 5 5 ...". We avoid

[4] In their online EM Model 1 aligner, Liang and Klein (p.c.) skirt the smoothing issue by running an epoch of batch EM to initialize a full set of probabilities before starting.

		Against silver standard (Batch EM)			Against gold standard (Human)		
		IBM1	IBM2	HMM	IBM1	IBM2	HMM
Batch EM		100.0	100.0	100.0	55.4	66.3	70.2
Online EM	Full data	82.2	81.7	89.5	54.9	63.4	70.0
	First 50%	81.4	79.5	88.9	54.9	62.3	70.6
	Last 50%	83.0	83.9	90.0	54.9	64.5	69.4
	Reordered	83.7	83.3	88.1	56.8	65.4	69.5

Figure 6: Word alignment f-scores. Batch EM for IBM 1 is run for 5 iterations; Batch IBM2 adds 5 further iterations of IBM2; Batch HMM adds a further 5 iterations of HMM. Online EM is single-pass. Against the silver standard, alignments are unidirectional; against gold, they are bidirectional and symmetrized with grow-dial-final (Koehn et al., 2003). First and last 50% report on different portions of the corpus. Reordered is on segment pairs ordered short to long. All runs exclude segment pairs with segments longer than 50 words.

this with a standard schedule of 5 IBM1 iterations, 5 IBM2 iterations, then 5 HMM iterations. However, HMM still learns a very high value for p_1, aligning most tokens to null, so we fix $p_1 = 0.1$ for the duration of training.

Single-pass, online HMM suffers the same two problems, both solved when we smooth differentially ($\lambda_o = 10^2, \lambda_t = 10^{-4}$) and fix $p_1 = 0.1$.

Two quick asides before we examine the effectiveness of our online methods:

- Translation researchers often drop long segment pairs that slow down HMM model processing. In compression, we cannot drop any of the text. Therefore, if the source segment contains more than 50 words, we use only monolingual PPMC to compress the target. This affects 26.5% of our word tokens.
- We might assist an online aligner by permuting our n segment pairs to place shorter, less ambiguous ones at the top. However, we would have to communicate the permutation to the decompressor, at a prohibitive cost of $\log_2(n!)/(8 \cdot 10^6)$ = 4.8 Mb.

We next look at alignment accuracy (f-score) on our large Spanish/English corpus (Figure 6). We evaluate against both a silver standard (Batch EM Viterbi alignments[5]) and a gold standard of 334 human-aligned segment pairs distributed throughout the corpus. We see that online methods generate competitive translation dictionaries. Because single-pass alignment is significantly faster than traditional multi-pass, we also investigate its impact on an overall Moses pipeline for phrase-based machine translation (Koehn et al., 2007). Figure 7 shows that we can achieve competitive translation accuracy using fast, single-pass alignment, speeding up the system development cycle. For this use case, we can get an additional +0.3 alignment f-score (just as fast) if we print Viterbi alignments in a second pass instead of during training.

[5]We confirm that our Batch HMM implementation gives f-scores (f=70.2, p=80.4, r=62.3) similar to GIZA++ (f=71.2, p=85.5, r=61.0), and its differently parameterized HMM.

	Alignment speed	Test Set Bleu	
		Europarl	News
Batch HMM	1230.78 min	30.2	26.2
Online HMM	711.87 min	30.0	25.3

Figure 7: Fast, single-pass HMM alignment yields competitive Spanish-English Moses phrase-based translation accuracy, as measured by Bleu (Papineni et al., 2002). In-domain (Europarl) and out-of-domain (SMAT-07 News Commentary) tune/test sets each consist of approximately 1000 sentences, all longer than 50 words to avoid overlap with training.

5.4 Word tokenization

We now want our continuously-improving translation model (TM) to predict target text, and to combine its predictions with PPM's. For that to happen, our TM will need to predict the exact text, including spurious double-spaces, how parentheses combine with quotation marks, and so on.

We devise a tokenization scheme that records spacing information in the word tokens, which allows us to recover the original text uniquely. First, we identify word tokens as subsequences of [a-zA-Z]*, [0-9]*, and [other]*, appending to each token the number of spaces following it (e.g., "...@2"). Next, we remove all "@1", which leaves unique

recoverability intact. Finally, we move any suffix on an alpha-numeric word i to become a prefix on a non-alpha-numeric word $i + 1$. This reduces the vocabulary size for TM learning. An example:

```
"String-theory?" he  asked.
  <=>
S@0 "@0 String@0 -@0 theory@0 ?@0 "@1
he@2 asked@0 .@0
  <=>
S@0 "@0 String@0 -@0 theory@0 ?@0 "
he@2 asked@0 .@0
  <=>
S@0 "@0 String @0-@0 theory @0?@0 "
he@2 asked @0.@0
```

5.5 Predicting target words

Under this tokenization scheme, we now ask our TM to give us a probability distribution over possible next words. The TM knows the entire source word sequence $f_1 ... f_l$ and the target words $e_1 ... e_{j-1}$ seen so far. As candidates, we consider target words that can be produced, via the current t-table, from any (non-NULL) source words with probability greater than 10^{-4}.

For HMM, we compute a prediction lattice that gives a distribution over possible *source alignment positions* for the current word we are predicting. Intuitively, the prediction lattice tells us "where we currently are" in translating the source string, and it prefers translations of source words in that vicinity. We efficiently reuse the lattice as we make predictions for each subsequent target word.

To make the TM's prediction more accurate, we weight its prediction for each word with a smoothed, adapted English bigram word language model (LM). This discourages the TM from trying to predict the first character of a word by simply using the most frequent source words. We found that exponentiating the LM's score by 0.2 before weighting keeps it from overpowering the HMM predictions.

5.6 Predicting target characters

To convert word predictions into character predictions, we combine scores for words that share the next character. For example, if the TM predicts "monkey 0.4, car 0.3, cat 0.2, dog 0.1", then we have "P(c) 0.5, P(m) 0.4, P(d) 0.1". Additionally, we restrict ourselves to words prefixed by the portion of e_j already observed. The TM predicts the space character when a predicted word fully matches the observed prefix.

We also adjust PPM to produce a full distribution over the 96 possible next characters. PPM normally computes a distribution over only characters previously seen in the current context (plus ESC). We now back off to the lowest context for every prediction.

We interpolate PPM and TM probabilities:

$$\begin{aligned}&\mathrm{P}(e_k|f_1 \ldots f_l, e_1 \ldots e_{k-1}) = \\ &\quad \mu\, \mathrm{P}_{PPM}(e_k|e_1 \ldots e_{k-1}) + \\ &\quad (1-\mu)\, \mathrm{P}_{TM}(e_k|f_1 \ldots f_l, e_1 \ldots e_{k-1})\end{aligned}$$

We adjust μ dynamically based on the relative confidence of the models:

$$\mu = \frac{max(PPM)^{2.5}}{max(PPM)^{2.5} + max(HMM)^{2.5}}$$

Here, $max(model)$ refers to the highest probability assigned to any character in the current context by the model. This yields better compression rates than simply setting μ to a constant. When the TM is unable to extend a word, we set $\mu = 1$.

	File size	Compression rate	bpb
Uncompressed	324.9 Mb	100.0%	8.00
Huffman coding	172.8 Mb	53.2%	4.26
PPMC	51.4 Mb	15.8%	1.26

Figure 8: Compression of the Spanish side of the bilingual corpus. bpb = bits per byte.

	File size	Compression rate	bpb
Uncompressed	294.5 Mb	100.0%	8.00
Huffman coding	160.7 Mb	54.6%	4.37
PPMC	48.5 Mb	16.5%	1.32
Bilingual (this paper)	35.0 Mb	11.9 %	0.95
Shannon monolingual			1.61
Shannon bilingual			0.51

Figure 9: **Main results.** Compression of the English side of the bilingual corpus. Bilingual compression improves results. For Shannon game studies, bpb are estimated as cross-entropies of n-gram models fitted to human guess sequences.

6 Results

Figure 8 shows that monolingual PPM compresses the Spanish side of our corpus to 15.8% of the original. Figure 9 **(Main results)** shows results for the English side of the corpus. Monolingual PPM compresses to 16.5%, while our HMM-based bilingual compression compresses to 11.9%.[6]

We can say that a human translation is characterized by an additional 0.95 bits per byte on top of the original, rather than the 1.32 bits per byte we

[6]For this result, we divide the English corpus into two pieces and compress them in parallel, and we further increase the sentence length threshold from 50 to 60, incurring a speed penalty. Our fictional Weissman score is 0.676.

	File size	Compression rate	bpb
Uncompressed	619.4 Mb	100.0%	8.00
Huffman coding	336.8 Mb	54.4%	4.35
PPMC	101.6 Mb	16.4%	1.31
Bilingual (this paper)	86.4 Mb	13.9%	1.12
(Sánchez-Martínez et al., 2012) PPMDI	different corpus	20.1%	1.61
(Sánchez-Martínez et al., 2012) bilingual	different corpus	18.7%	1.50

Figure 10: Compression of Spanish plus English. All methods are run on a single file of Spanish concatenated with English, except for "Bilingual (this paper)," which records the sum of (1) Spanish compression and (2) English-given-Spanish compression. Comparative numbers copied from Sánchez-Martínez et al (2012) are for a different subset of Europarl data.

would need if the English were independent text. Assuming our Spanish compression is good, we can also say that the human translator produces at most 68.1% (35.0/51.4) of the information that the original Spanish author produced. Intuitively, we feel this bound is high and should be reduced with better translation modeling.

Figure 9 also reports our Shannon game experiments in which bilingual humans guessed subsequent characters of the English text. As suggested by Shannon, we upper-bound bpb as the cross-entropy of a unigram model over a human *guess sequence* (e.g., 1 1 2 5 17 1 1 ...), which records how many guesses it took to identify each subsequent English character, given context. For a 502-character English sequence, a team of four bilinguals working together gave us an upper-bound bpb of 0.51. This team had access to the original Spanish, plus a Google translation. Monolinguals guessing on the same data (minus the Spanish and Google translation) yielded an upper-bound bpb of 1.61. These human-level models indicate that human translators are actually only adding $\sim 32\%$ more information on top of the original, and that our current translation models are only capturing some fraction of this redundancy.[7]

Figure 10 shows compression of the entire bilingual corpus, allowing us to compare with the previous state-of-the-art (Sánchez-Martínez et al., 2012), which compresses a single, word-interleaved bilingual corpus. It shows how PPMC does on a concatenated Spanish/English file.

Uncompressed English (294.5 Mb) is 90.6% the size of uncompressed Spanish (324.9 Mb). Huffman narrows this gap to 93.0%, and PPM narrows it further to 94.4%, consistent with Behr et al. (2003) and Liberman (2008). Spanish redundancies like adjective-noun agreement and balanced question marks ("¿ ... ?") may remain unexploited.

7 Conclusion

We have created a bilingual text compression challenge web site.[8] This web site contains standard bilingual data, specifies what a valid compression is, and maintains benchmark results.

There are many future directions to pursue. First, we would like to develop and exploit better predictive translation modeling. We have so far adapted machine translation technology circa only 1996. For example, the HMM alignment model cannot "cross off" a source word and stop trying to translate it. Also possible are phrase-based translation, neural nets, or as-yet-unanticipated pattern-finding algorithms. We only require an executable that prints the bilingual text.

Our current method requires segment-aligned input. To work with real-life bilingual corpora, the compressor should take care of segment alignment, in a way that allows decompression back to the original text. Similarly, we are currently restricted to texts written in the Latin alphabet, per our definition of "word."

More broadly, we would also like to import more compression ideas into NLP. Compression has so far appeared sporadically in NLP tasks like native language ID (Bobicev, 2013), text input methods (Powers and Huang, 2004), word segmentation (Teahan et al., 2000; Sornil and Chaiwanarom, 2004; Hutchens and Alder, 1998), alignment (Liu et al., 2014), and text categorization (Caruana & Lang, unpub. 1995).

Translation researchers may also view bilingual compression as an alternate, reference-free evaluation metric for translation models. We anticipate that future ideas from bilingual compression can be brought back into translation. Like Brown et al. (1992), with their *gauntlet thrown down* and *fury of competitive energy*, we hope that cross-fertilizing compression and translation will bring fresh ideas to both areas.

[7] Machine models can also generate guess sequences, and we see that entropy of a 30m-character PPMC guess sequence (1.43) upper-bounds actual PPMC bpb (1.28).

[8] www.isi.edu/natural-language/compression

Acknowledgements

This work was supported by a USC Provost Fellowship and ARO grant W911NF-10-1-0533.

References

Joaquín Adiego, Nieves R. Brisaboa, Miguel A. Martínez-Prieto, and Felipe Sánchez-Martínez. 2009. A two-level structure for compressing aligned bitexts. In *String Processing and Information Retrieval*. Springer-Verlag.

Joaquín Adiego, Miguel A. Martínez-Prieto, Javier E. Hoyos-Torío, and Felipe Sánchez-Martínez. 2010. Modelling parallel texts for boosting compression. In *Proc. Data Compression Conference (DCC)*.

Yaser Al-Onaizan, Jan Curin, Michael Jahr, Kevin Knight, John Lafferty, Dan Melamed, Franz-Josef Och, David Purdy, Noah A. Smith, and David Yarowsky. 1999. Statistical machine translation. Technical Report http://bit.ly/1u9jJsx, Johns Hopkins University.

F. Behr, Victoria Fossum, Michael Mitzenmacher, and David Xiao. 2003. Estimating and comparing entropies across written natural languages using PPM compression. In *Proc. Data Compression Conference (DCC)*.

Victoria Bobicev. 2013. Native language identification with PPM. In *Proc. NAACL*.

Peter F. Brown, Vincent J. Della Pietra, Robert L. Mercer, Stephen A Della Pietra, and Jennifer C. Lai. 1992. An estimate of an upper bound for the entropy of English. *Computational Linguistics*, 18(1):31–40.

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

John G. Cleary and Ian Witten. 1984. Data compression using adaptive coding and partial string matching. *IEEE Transactions on Communications*, 32(4):396–402.

Ehud S. Conley and Shmuel T. Klein. 2008. Using alignment for multilingual text compression. *International Journal of Foundations of Computer Science*, 19(01):89–101.

Ehud S. Conley and Shmuel T. Klein. 2013. Improved alignment-based algorithm for multilingual text compression. *Mathematics in Computer Science*, 7(2):137–153.

George Foster, Philippe Langlais, and Guy Lapalme. 2002. User-friendly text prediction for translators. In *Proc. EMNLP*.

David A. Huffman et al. 1952. A method for the construction of minimum redundancy codes. *Proc. IRE*, 40(9):1098–1101.

Jason L. Hutchens and Michael D Alder. 1998. Finding structure via compression. In *Proc. Joint Conferences on New Methods in Language Processing and Computational Natural Language Learning*.

Marcus Hutter. 2006. 50,000 Euro prize for compressing human knowledge. `http://prize.hutter1.net`. Accessed: 2015-02-04.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proc. NAACL*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proc. ACL Poster and Demo Session*.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *Proc. MT Summit X*.

Patrik Lambert, Adrià De Gispert, Rafael Banchs, and José B Mariño. 2005. Guidelines for word alignment evaluation and manual alignment. *Language Resources and Evaluation*, 39.

Abby Levenberg, Chris Callison-Burch, and Miles Osborne. 2010. Stream-based translation models for statistical machine translation. In *Proc. NAACL*, pages 394–402. Association for Computational Linguistics.

Percy Liang and Dan Klein. 2009. Online EM for unsupervised models. In *Proc. HLT-NAACL*.

Mark Liberman. 2008. Is English more efficient than Chinese after all? `http://languagelog.ldc.upenn.edu/nll/?p=93`. Accessed: 2015-02-04.

Wei Liu, Zhipeng Chang, and William J. Teahan. 2014. Experiments with a PPM compression-based method for English-Chinese bilingual sentence alignment. In *Statistical Language and Speech Processing*, pages 70–81. Springer.

M. V. Mahoney. 2005. Adaptive weighting of context models for lossless data compression. Technical Report CS-2005-16, Florida Institute of Technology.

M. A. Martínez-Prieto, J. Adiego, F. Sánchez-Martínez, P. de la Fuente, and R. C. Carrasco. 2009. On the use of word alignments to enhance bitext compression. In *Proc. Data Compression Conference (DCC)*.

Radford M. Neal and Geoffrey E Hinton. 1998. A view of the EM algorithm that justifies incremental, sparse, and other variants. In *Learning in Graphical Models*, pages 355–368. Springer.

Craig Nevill and Timothy Bell. 1992. Compression of parallel texts. *Information Processing & Management*, 28(6):781–793.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proc. ACL*.

David Martin Powers and Jin Hu Huang. 2004. Adaptive compression-based approach for chinese pinyin input. In *Proc. Third SIGHAN Workshop on Chinese Language Learning*.

Jorma Rissanen and Glen G Langdon Jr. 1981. Universal modeling and coding. *Information Theory, IEEE Transactions on*, 27(1):12–23.

Felipe Sánchez-Martínez, Rafael C. Carrasco, Miguel A. Martínez-Prieto, and Joaquin Adiego. 2012. Generalized biwords for bitext compression and translation spotting. *Journal of Artificial Intelligence Research*, 43:389–418.

Ohm Sornil and Paweena Chaiwanarom. 2004. Combining prediction by partial matching and logistic regression for thai word segmentation. In *Proc. COLING*, page 1208. Association for Computational Linguistics.

William John Teahan, Yingying Wen, Rodger McNab, and Ian H. Witten. 2000. A compression-based algorithm for Chinese word segmentation. *Computational Linguistics*, 26(3):375–393.

Stephan Vogel, Hermann Ney, and Christoph Tillmann. 1996. HMM-based word alignment in statistical translation. In *Proc. COLING*.

Ian H. Witten, Radford M. Neal, and John G. Cleary. 1987. Arithmetic coding for data compression. *Communications of the ACM*, 30(6):520–540.

Hierarchical Recurrent Neural Network for Document Modeling

Rui Lin[†]*, **Shujie Liu**[‡], **Muyun Yang**[†], **Mu Li**[‡], **Ming Zhou**[‡], **Sheng Li**[†]
[†]Harbin Institute of Technology
{linrui,ymy}@mtlab.hit.edu.cn lisheng@hit.edu.cn
[‡]Microsoft Research
{shujliu,muli,mingzhou}@microsoft.com

Abstract

This paper proposes a novel hierarchical recurrent neural network language model (HRNNLM) for document modeling. After establishing a RNN to capture the coherence between sentences in a document, HRNNLM integrates it as the sentence history information into the word level RNN to predict the word sequence with cross-sentence contextual information. A two-step training approach is designed, in which sentence-level and word-level language models are approximated for the convergence in a pipeline style. Examined by the standard sentence reordering scenario, HRNNLM is proved for its better accuracy in modeling the sentence coherence. And at the word level, experimental results also indicate a significant lower model perplexity, followed by a practical better translation result when applied to a Chinese-English document translation reranking task.

1 Introduction

Deep Neural Network (DNN), a neural network with multiple layers, has been proven powerful in many different domains, such as visual recognition (Kavukcuoglu et al., 2010) and speech recognition (Dahl et al., 2012), ever since Hinton et al. (2006) formulated an efficient training method for it.

In addition to the applications mentioned above, many neural network based methods have also been applied to natural language processing (NLP) tasks with great success. For example, Collobert et al. (2011) propose a generalized DNN framework for a variety of fundamental NLP tasks, including part-of-speech tagging (postag), chunking, named entity recognition (NER), and semantic role labeling.

DNN is successfully introduced to do word-level language modeling, aka., to predict the next word given the history words. Bengio et al. (2003) propose a feedforward neural network to train a word-level language model with a limited n-gram history. To leverage as much history as possible, Mikolov et al. (2010) apply recurrent neural network to word-level language modeling. The model absorbs one word each time, keeps the information in a history vector, and predicts the next word with all the word history in the vector.

Word-level language model can only learn the relationship between words in one sentence. For sentences in one document which talks about one or several specific topics, the words in the next sentence are chosen partially in accordance with the previous sentences. To model this kind of coherence of sentences, Le and Mikolov (2014) extend word embedding learning network (Mikolov et al., 2013) to learn the paragraph embedding as a fixed-length vector representation for paragraph or sentence. Li and Hovy (2014) propose a neural network coherence model which employs distributed sentence representation and then predict the probability of whether a sequence of sentences is coherent or not.

In contrast to the methods mentioned above which learn the word relationship in or between the sentences separately, we propose a hierarchical recurrent neural network language model (HRNNLM) to capture the word sequence across the sentence boundaries at the document level. HRNNLM is essentially a combination of a word-level language model and a sentence-level language model, both of which are recurrent neural networks. The word-level recurrent neural network follows (Mikolov et al., 2010). The sentence-level language model is another recurrent neural network that takes sentence represen-

*Contribution during internship at Microsoft Research.

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 899–907,
Lisbon, Portugal, 17-21 September 2015.

tation as input, and predicts the words in the next sentence. Similar to (Mikolov et al., 2010), the hidden layer in the sentence-level recurrent neural network contains the sentence history information. The hidden layer containing the history information of previous sentences is then linked as an input to the word-level recurrent neural network to predict the next word together with the word-level history vector. This allows the language model to predict the next word probability distribution beyond the words in the current sentence.

We propose a two-step training approach to optimize the parameters of HRNNLM. In the first step, we train the sentence-level language models independently . And then, we connect the hidden layer of the sentence-level language model to the input of word-level RNNLM and train the two models jointly until converged. At sentence level, we evaluate our model with a sentence ordering task and the result shows our method can outperform a maximum entropy based and another state-of-the-art solution. At word level, we compare our method with the conventional recurrent neural network based language model, finding the perplexity is reduced significantly. We also apply our method to rank machine translation output and conduct experiments on a Chinese-English document translation task, yielding a better translation results compared with a state-of-the-art baseline system.

The rest of this paper is organized as follows: Section 2 introduces work related to applying neural network to document modeling and SMT. Section 3 introduces the general framework for document modeling. Our sentence-level language model and its training is described in Section 4, and the overall HRNNLM and its training is presented in Section 5. Section 6 presents our experiments and their results. Finally, we conclude in Section 7.

2 Related work

In this section, we introduce previous efforts on applying neural network to model words coherence across sentence boundaries as well as works on improving machine translation performance at discourse level.

Mikolov and Zweig (2012) propose a RNN-LDA model to implement a context dependent language model. They augment the contextual information into the conventional RNNLM via a real-valued input vector, which is the probability distribution computed by LDA topics for using a block of preceding text. They train a Latent Dirichlet Allocation (LDA) model using documents consisting of about 10 sentences long text from Penn Treebank (PTB) training data. Their approach outperforms RNNLM in perplexity on PTB data with a limited context history over topics instead of complete information of preceding sentences.

Le and Mikolov (2014) extend the Continuous Bag-of-Words Model (CBOW) and Continuous Skip-gram Model (Skip-gram) (Mikolov et al., 2013) by introducing a paragraph vector. In their method, the paragraph vector is learnt in a similar way of word vector model, and there will be $N \times P$ parameters, if there are N paragraphs and each paragraph is mapped to P dimensions. Different from them, the sentence vectors of our model are learnt with nearly unlimited sentence history based on a RNN framework, in which, bag of words in the sentence are used as input. The sentence vector is no longer related with the sentence id, but only based on the words in the sentence. And our sentence vector also integrates nearly all the history information of previous sentences, while their model cannot.

Li and Hovy (2014) implement a neural network model to predict discourse coherence quality in essays. In their work, recurrent (Sutskever et al., 2011) and recursive (Socher et al., 2013) neural networks are both examined to learn distributed sentence representation given pre-trained word embedding. The distributed sentence representation is assigned to capture both syntactic and semantic information. With a slide window of the distributed sentence representation, a neural network classifier is trained to evaluate the coherence of the text. Successful as it is in scoring the coherence for a given sequence of sentences, this method is attempted to discriminate the different word order within a sentence.

An attempt of introducing RNN into convolutional neural network (CNN) is investigated by (Xu and Sarikaya, 2014) for spoken language understanding (SLU). To alleviate more contextual information, they apply a CNN with Jordan-type (Jordan, 1997) recurrent connections. The recurrent connections send the distribution of the last softmax layer's output to the current input layer as additional features. Aimed to improve SLU domain classification, their model is essentially a kind of document representation with certain text

information, neglecting the coherence information between sentences.

Following the thread modeling the word sequence relationship within and across sentences, we propose a hierarchical recurrent neural network language model consist of a sentence-level language model and a word-level language model. This overall network is trained to capture the coherence between sentences and predict words sequence with preceding sentence contexts.

For statistical machine translation (SMT) in which we checked out model as a scenario, DNN has also been revealed for certain good results in several components. Yang et al. (2013) adapt and extend the CD-DNN-HMM (Dahl et al., 2012) model to the HMM-based word alignment model. In their method, they use bilingual word embedding to capture the lexical translation information and modeling the context with surrounding words. Liu et al. (2014) propose a recursive recurrent neural network (R^2NN) for end-to-end decoding to help improve translation quality. And Cho et al. (2014) propose a RNN Encoder-Decoder which is a joint recurrent neural network model at the sentence level as conventional SMT decoder does. However, at the discourse level, there is little report on applying DNN to boost the translation result of a document.

3 Document Language Modeling

Statistical language model assigns a probability to a natural language sequence. Conventional language models only focus on the word sequence within a sentence. For sentences in one document talking about one or several specific topics, the adjacent sentences should be in a coherent order. Therefore, the words in the next sentence are also dependent on the preceding sentences. To model the coherence of sentences in the document D, which contains N sentences S_1, S_2, S_3, ..., S_N, we need to maximize the objective as follow:

$$\begin{aligned} p(D) =& p(S_1, S_2, ..., S_N) \\ =& p(S_1) \cdot p(S_2|S_1) \cdot p(S_3|S_1, S_2) \\ & ...p(S_N|S_1, S_2, ..., S_{N-1}) \end{aligned} \quad (1)$$

For the sentence S_k containing words w_1, w_2, w_3, ..., w_T, $p(S_k|S_1, S_2, ..., S_{k-1})$ is defined as:

$$\begin{aligned} & p(S_k|S_1, S_2, ..., S_{k-1}) \\ & = p(w_1, w_2, ..., w_T|S_1, ..., S_{k-1}) \\ & = p(w_1|S_1, ..., S_{k-1}) \cdot p(w_2|w_1, S_1, ..., S_{k-1}) \\ & ...p(w_T|w_1, w_2, ..., w_{T-1}, S_1, ..., S_{k-1}) \end{aligned} \quad (2)$$

As a special case of approximation to this, classical n-gram language model keep only several words as history, discarding any information across the sentence boundaries. Recurrent neural network language model (Mikolov et al., 2010) uses a hidden layer which employs a real-valued vector recurrently as network's input to keep as many history as possible. This makes RNNLM be able to extend for capturing history beyond a sentence.

To prevent the potential exponential decay of the history, the history length in RNN can not be too long. Here we approximate the history information of previous sentences, $p(S_k|S_1, S_2, ..., S_{k-1})$, by the following:

$$\begin{aligned} & p(S_k|S_1, S_2, ..., S_{k-1}) = \\ & p(BoW_{S_k}|BoW_{S_1}, ..., BoW_{S_{k-1}}) \cdot p(S_k|BoW_{S_k}) \end{aligned} \quad (3)$$

where BoW_{S_k} denotes the bag of words for the sentence S_k. The document is thus generated in two steps.

- Given the previous sentences BoW_{S_1} ,..., $BoW_{S_{k-1}}$ (treating them as bag of words here), first generate the words which will show in the next sentence without considering their order with $p(BoW_{S_k}|BoW_{S_1}, ..., BoW_{S_{k-1}})$
- Generate the words one by one with $p(S_k|BoW_{S_k})$.

The first phase actually completes sentence-level language modeling, and the second addresses the word-level language modeling. Because recurrent neural network has a natural advantage in processing sequential data, we investigate how to model the whole process under a unified framework of recurrent neural network.

4 Sentence-level Language Model

In this section, we describe how to leverage recurrent neural network for sentence-level language modeling. Mikolov et al. (2010) demonstrate a recurrent neural network language model (RNNLM)

for word ordering. It overcomes the limitations of classical language model in capturing only a fixed-length history, yielding a significant performance improvements in terms of perplexity reduction and speech recognition accuracy. Here we adept this framework for a RNN based sentence-level language modeling, i.e. RNNSLM.

4.1 Model

A conventional language model reads a word each time, keeps several words as history and then predict the probability distribution of the next word. Similar to this, our sentence-level language model reads a sentence which is a bag of words representation. And then it stores the sentence history which captures coherence of sentences in a real-valued history vector. With the history vector, our model can predict which words are most likely to appear in the next sentence. All these will be modeled by a recurrent neural network.

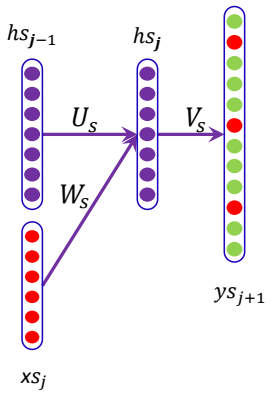


Figure 1: Recurrent Neural Network for Sentence-level Language Modeling

As shown in Figure 1, similar to the conventional recurrent neural network, for the sentence j, our network has two input layers xs_j and hs_{j-1}. xs_j is the current sentence representation, and hs_{j-1} is the history information vector before sentence j. The model has a hidden layer hs_j, which will combine the history information of hs_{j-1} and the current sentence input xs_j, and an output layer ys_{j+1}, which generates the probabilities of the words in the sentence $j+1$. The layers are computed as follows:

$$hs_j = f(U_s \cdot hs_{j-1} + W_s \cdot xs_j) \quad (4)$$

$$ys_{j+1} = g(V_s \cdot hs_j) \quad (5)$$

where W_s, U_s and V_s denote the weight matrix. $f(z)$ is a $HTanh$ function:

$$f(z_j) = \begin{cases} -1 & z_j < -1 \\ z & -1 < z_j < 1 \\ 1 & z_j > 1 \end{cases} \quad (6)$$

and $g(z)$ is a softmax function:

$$g(z_j) = \frac{e^{z_j}}{\sum_k e^{z_k}} \quad (7)$$

The output layer ys_{j+1} is a $1 \times V$ vector that represents probability distribution of words in the next sentence given the current sentence xs_j and previous history hs_{j-1}, where V denotes vocabulary size.

To emphasize coherence between the adjacent sentences, we further add some bigram-like bag of words feature to the output layer. As mentioned in (Mikolov, 2012), this is kind of maximum entropy feature which can be derived by a two-layer neural network. Some experiments show that perplexity significantly decreases after adding these features. Following (Mikolov, 2012), where, the maximum entropy bigram features are added to our RNNSLM by a direct connection between the feature input array and output layer ys_{j+1}. Following (Mahoney, 2000), we map bigram maximum entropy features to a fixed-length array to reduce the memory complexity of direct connections with feature hashing. Then the output layer can be computed as follow:

$$ys_{j+1}(t) = g(V_s(t) \cdot hs_j + \sum_{w \in xs_j} D_{hash(w,t)}) \quad (8)$$

where (t) denotes the t-th row of a vector or a matrix. D denotes that the hash array contains feature weights and $hash(w_i, w_j)$ denotes the hash function for mapping bigram features to a fixed-length array. For a output ys_{j+1}, multiple connections may be activated according to the words in sentence xs_j.

4.2 Training

The training objective of our RNNSLM is to find the best parameters for predicting the words of next sentence. Formally, given the next sentence S_k containing words w_1, w_2, w_3, ..., w_T. The training objective according to (Mikolov et al.,

2013) can be denoted by:

$$log(p(BoW_{S_k}|BoW_{S_1}, ..., BoW_{S_{k-1}})) = \frac{1}{T}\sum_{t=1}^{T} logp(w_t|BoW_{S_1}, ..., BoW_{S_{k-1}}) \quad (9)$$

For weight matrix W_s, U_s, V_s and hash feature weight D, the parameter are trained similar to the conventional recurrent neural network. The learning rate α is set to 0.1 at the start of the training as suggested in (Mikolov et al., 2010). After each epoch, it can be determined by the training loss of network. If the loss decreases significantly, training continues with the same learning rate. Otherwise, if the loss increases, the training will be executed with a new learning rate $\alpha/2$. The training process will be terminated after about 30 epochs.

4.3 Initialization

All elements in weight matrix W_s and U_s are initialized by randomly sampling from a uniform distribution $[-\frac{1}{K_1}, \frac{1}{K_1}]$, where K_1 is the size of the input layer. Elements in weight matrix V_s are initialized by randomly sampling from a uniform distribution $[-\frac{1}{K_2}, \frac{1}{K_2}]$, where K_2 denotes the size of the hidden layer. The hash feature weight array D is initialized as 0.

For the initialization of hs_0, it can be set to a vector of the same values, which is 0.1.

5 Hierarchical Recurrent Neural Network

In the previous section, we propose a RNNSLM which models the coherence between sentences but ignores the word sequence within a sentence. Ideally, a perfect document model should not only capture the information between sentences but also the information with sentence. So we propose a hierarchical recurrent neural network language model (HRNNLM) to fulfill this issue.

5.1 Model

A hierarchical recurrent neural network consists of two independent recurrent neural network. For a conventional word-level language model, it predict the next word only using the word history within the sentence. To capture the longer history, we integrate the sentence history into the word-level language model from sentence-level language model, which forms a hierarchical recurrent neural network.

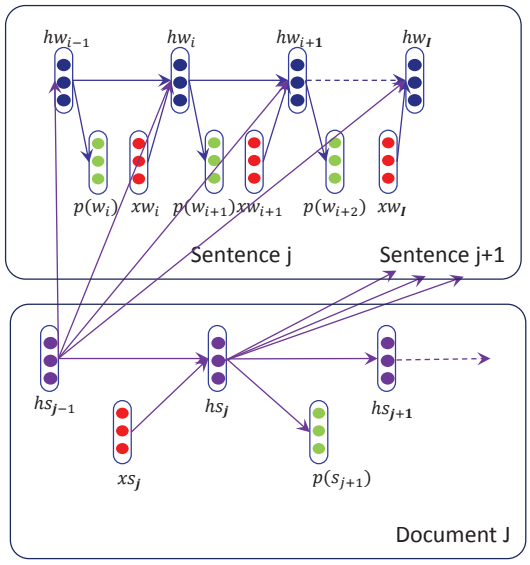


Figure 2: Hierarchical recurrent neural network

As illustrated Figure 2, the upper part is the unfolded illustration of conventional recurrent neural network based language model. It takes one word w_i each time with the previous history information hw_{i-1} together and predicts the probability of the next word $p(w_{i+1})$ with the information kept in the history vector hw_i. The lower part is our RNNSLM, which takes the bag of words representation of a sentence xs_j each time with the history information of previous sentences hs_{j-1} together and predicts the bag of words in the next sentence $p(s_{j+1})$ with the information kept in hs_j.

We integrate these two recurrent neural networks together by adding connections between the sentence-level history vector hs_{j-1} and word level history vector hw_i. So while predicting the next word w_{i+1} of the current sentence, our model will consider the current word w_i, history of previous sentences hs_{j-1} and history of previous words hw_{i-1}. The new word-level history vector hw_i is computed as:

$$hw_i = f(U_w \cdot hw_{i-1} + W_w \cdot xw_i + U_{sw} \cdot hs_{j-1}) \quad (10)$$

where $f(z)$ is a $HTanh$ function. For HRNNLM, we also add a bigram hash feature, similar as we do for RNNSLM.

5.2 Training

The HRNNLM can be trained from scratch following Mikolov et al. (2010) with a dual objective. But this is not without problem. Beginning

of training phase, the sentence history is unstable since the parameters of sentence-level language model are kept updating. Consequently, the training of HRNNLM will be also unstable and hard to converge with unstable sentence history.

In this paper, we approximate the whole training of HRNNLM by a two-step training method. We first train a RNNSLM until it converges. Then we connect the hidden layer of RNNSLM to the hidden layer of RNNWLM. To increase the training speed, all the parameters of RNNSLM are fixed while training HRNNLM. We only update the random initialized parameters in HRNNLM, though ideally the gradient of the sentence history vector could change and the RNNSLM could be updated again. The learning rate α is set to 0.1 and the updating of learning rate is the same as suggested in Section 4.2. All the parameters can be initialize as suggested in Section 4.3.

6 Experiments

We evaluate the sentence-level performance of HRNNLM by the common coherence evaluation of sentence ordering task, its word-level performance by perplexity measure. We also apply our HRNNLM to SMT reranking task in an open Chinese-English translation dataset. The translation performance index is the IBM version of BLEU-4 (Papineni et al., 2002).

6.1 Sentence Ordering

We follow (Barzilay and Lapata, 2008) to evaluate our sentence-level language model via a sentence ordering task with test set 2010 (tst2010), 2011 (tst2011) and 2012 (tst2012) from IWSLT 2014, totaling 37 English documents. 20 random permutations of sentences for each document are generated. Each permutation and its original document are combined as an article pair. Our goal is to find the original one among all the article pairs.

The training data for sentence-level language model is the 1,414 English documents from the parallel corpus also provided by the IWSLT 2014 spoken language translation task. 90% of the documents are for training and the rest are reserved for validation. The size of the hidden layer is set to 30 and hash array size is 10^7.

We define the log probability of a given document as its coherence score. The document with the higher score is regarded as the original document.

We provide two baselines for sentence ordering. One is the state-of-the-art recursive neural network based method proposed by (Li and Hovy, 2014). We implement their model trained and tested with our data. The other is a maximum entropy classifier trained with bag of words features of adjacent sentences which can generate a coherent probability of adjacent sentences. The document with the higher sum of log probability for each adjacency sentences is regarded as the original document. Table 1 shows the accuracy of our system and baseline.

Setting	Accuracy
Recursive	91.39%
ME system	91.89%
Our system	95.68%

Table 1: Accuracy of the sentence ordering task for each system

From Table 1 we can see that the maximum entropy model and the recursive neural network model has almost the same performance. Compared with the baseline systems, the proposed HRNNLM achieves significant improvement with nearly 4.3% improvement in term of accuracy. The experimental result shows that the HRNNLM can model document coherence and capture cross-sentence information.

6.2 Word-level Model Perplexity

We compare the word level performance of HRNNLM with the most popular RNNLM in terms of model perplexity. For a fair comparison, we follow (Mikolov et al., 2010) and train the model also on 90% of the 1.414 English documents form IWSLT 2014, totaling about 3M words. Then we train our model with the same hidden layer size and hash array size as the baseline system. The perplexity of these two models is evaluated on held-out documents, about 370K words. The results are shown in Table 2.

Setting	Perplexity
RNNLM-30	183
HRNNLM-30	174

Table 2: Perplexity of the different language model

According to Table 2, it is reasonable to claim that, by integrating history information of previous

sentences, the model perplexity decreased significantly. Empirically, this confirms the hypothesis that the words selection for the next sentence is dependent on its preceding sentences in the same document.

6.3 Spoken Language Translation

The conventional SMT systems translate sentences independently, without considering the coherence of the sentences in the same document. In order to learn translation coherence between sentences, we apply the HRNNLM to machine translation reranking task.

6.3.1 Data Setting and Baselines

The data comes from the IWSLT 2014 spoken language translation task. The training data consists of 1,414 documents on TED talks, and contains 179k sentence pairs, about 3M Chinese words, and 3.3M English words. The language model for SMT is a 4-gram language model trained with the English documents in the training data. The development set is specified by IWSLT as dev2010, and the test set contains 37 documents from tst2010, tst2011 and tst2012.

The IWSLT 2014 baseline system is built upon the open-source machine translation toolkit Moses at the default configuration, proposed by (Cettolo et al., 2012). We also train a decoder, which is an in-house Bracketing Transduction Grammar (BTG) (Wu, 1997) in a CKY-style decoder with a lexical reordering model trained with maximum entropy (Xiong et al., 2006). The decoder uses commonly used features, such as translation probabilities, lexical weights, a language model, word penalty, and distortion probabilities.

6.3.2 Rerank System

Our reranking system is a linear model with several features, including the SMT system final scores, sentence-level language model scores, and HRNNLM scores. It should be noted all these features are actually employed by the SMT model except for the HRNNLM score. Since Minimum Error Rate Training (MERT) (Och, 2003) is the most general method adopted in SMT systems for tuning, the feature weights are fixed by MERT.

For our reranking system, to score the translation of one sentence we need the translation results of all the previous sentences in the document. Our SMT decoder generates 10-best results of all the sentences of the documents and the reranking system select the best translation result for the first sentence at first. With the translation of first sentence, we score all the translation candidates of the second sentence and select the best one as the result. Following this procedure, we can get the translation results for all the sentences in the document.

6.3.3 Results

The HRNNLM focus on exploiting longer context, esp. cross-sentence word dependencies. Therefor the translation data for IWSLT 2014 is organized as documents instead of sentences for our rerank system. We hope HRNNLM will enable a context-sensitive reranking process, capturing the syntactic and logic relationships between the sentences in the same document.

Setting	tst2010	tst2011	tst2012
IWSLT	11.12	13.34	-
Baseline	12.40	15.09	13.52
SMT + Rerank	12.55	15.23	13.70

Table 3: BLEU scores of SMT systems. The IWSLT is a public baseline which issued by the organizer of IWSLT 2014, as described in (Cettolo et al., 2012).

The translation performance comparison is shown in Table 3. From Table 3, we can find that the rerank system improves SMT performance consistently. For a single sentence without the context information, there are several appropriate translations and it is hard to tell which one is better. When considering the context of a document (previous sentences for our model), some translation candidates may not be coherent with the others which should not be selected. Our model can generate the most coherent translation results by considering previous sentence history.

For example, we have the following two Chinese sentence in one document together with their correct translation:

我 拍摄 过 的 冰山, 有些 冰 是 非常 年轻 - - 几千 年 年龄
Some of the ice in the icebergs that I photograph is very young - - a couple thousand years old.
有些 冰 超过 十万 年
And some of the ice is over 100,000 years old.

Chinese word " 有些" means "some" in English. But when it is used in parallelism sentences, it means "some of" instead of "some". The traditional SMT system translates the italics part without considering the context. The translation result for this kind of system is:

Some ice more than 100,000 years.

For our system, the HRNNLM can take previous sentence as context and learn the parallelism between the two sentences. It can select the best translation "some of" for 有些, and the output of our system is:

Some of the ice more than 100,000 years.

We also calculate the BLEU increase ratio of our system on document level. The ratio is defined as $\frac{1}{N}\#(BleuD_{rerank} > BleuD_{baseline})$, where N denotes the number of documents, and $\#(BleuD_{rerank} > BleuD_{baseline})$ denotes the number of documents for which document level BLEU score of reranking system is higher than the baselines. The results are shown in Table 4.

tst2010	tst2011	tst2012
72.73%	71.43%	75%

Table 4: Experimental results to test BLEU increase ratio after reranking

From Table 4, we can find that, for all the three test data sets, our reranking system can achieve better performance for more than 70% documents.

7 Conclusion and Future Work

In this paper, we propose a hierarchical recurrent neural network language model for document modeling. We first built a RNNSLM to capture the information between sentences. Then we integrate the hidden layer of RNNSLM into the input layer of word-level language model to form a hierarchical recurrent neural network. This enables the model be able to capture both in-sentence and cross-sentence information in a unified RNN. Compared with conventional language models, our model can perceive a longer history than other language models and captures the context patterns in the previous sentences. At sentence level, we examine our model with sentence ordering task. At word level, we test the model perplexity. We also conduct a SMT rerank experiment on IWSLT 2014 data set. All these experimental results show that our hierarchical recurrent neural network has a satisfying performance.

In the future, we will explore better sentence representation such as distributed sentence representation as input for our sentence-level language model to better model document coherence. We can even update the gradient from different RNN to get a better performance.

Acknowledgments

We are grateful to the three anonymous reviewers for their helpful comments and suggestions. We also thank Dongdong Zhang, Lei Cui for useful discussions. This paper is supported by the project of National Natural Science Foundation of China (Grant No. 61272384, 61370170 &61402134).

References

Regina Barzilay and Mirella Lapata. 2008. Modeling local coherence: An entity-based approach. *Computational Linguistics*, 34(1):1–34.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *The Journal of Machine Learning Research*, 3:1137–1155.

Mauro Cettolo, Christian Girardi, and Marcello Federico. 2012. Wit^3: Web inventory of transcribed and translated talks. In *Proceedings of the 16th Conference of the European Association for Machine Translation (EAMT)*, pages 261–268, Trento, Italy, May.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. pages 1724–1734, October.

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

George E Dahl, Dong Yu, Li Deng, and Alex Acero. 2012. Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *Audio, Speech, and Language Processing, IEEE Transactions on*, 20(1):30–42.

Geoffrey Hinton, Simon Osindero, and Yee-Whye Teh. 2006. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554.

Michael I Jordan. 1997. Serial order: A parallel distributed processing approach. *Advances in psychology*, 121:471–495.

Koray Kavukcuoglu, Pierre Sermanet, Y-Lan Boureau, Karol Gregor, Michaël Mathieu, and Yann L Cun. 2010. Learning convolutional feature hierarchies for visual recognition. In *Advances in neural information processing systems*, pages 1090–1098.

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

Jiwei Li and Eduard Hovy. 2014. A model of coherence based on distributed sentence representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2039–2048. Association for Computational Linguistics.

Shujie Liu, Nan Yang, Mu Li, and Ming Zhou. 2014. A recursive recurrent neural network for statistical machine translation. In *Proceedings of ACL*, pages 1491–1500.

Matthew V Mahoney. 2000. Fast text compression with neural networks. In *FLAIRS Conference*, pages 230–234.

Tomas Mikolov and Geoffrey Zweig. 2012. Context dependent recurrent neural network language model. In *Spoken Language Technology Workshop, IEEE*, pages 234 – 239.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010*, pages 1045–1048.

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

Tomáš Mikolov. 2012. *Statistical language models based on neural networks*. Ph.D. thesis, Ph. D. thesis, Brno University of Technology.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 160–167. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, volume 1631, page 1642. Citeseer.

Ilya Sutskever, James Martens, and Geoffrey E Hinton. 2011. Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 1017–1024.

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

Deyi Xiong, Qun Liu, and Shouxun Lin. 2006. Maximum entropy based phrase reordering model for statistical machine translation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 521–528. Association for Computational Linguistics.

Puyang Xu and Ruhi Sarikaya. 2014. Contextual domain classification in spoken language understanding systems using recurrent neural network. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 136–140. IEEE.

Nan Yang, Shujie Liu, Mu Li, Ming Zhou, and Nenghai Yu. 2013. Word alignment modeling with context dependent deep neural network. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 166–175, Sofia, Bulgaria, August. Association for Computational Linguistics.

Auto-Sizing Neural Networks: With Applications to n-gram Language Models

Kenton Murray and **David Chiang**
Department of Computer Science and Engineering
University of Notre Dame
{kmurray4,dchiang}@nd.edu

Abstract

Neural networks have been shown to improve performance across a range of natural-language tasks. However, designing and training them can be complicated. Frequently, researchers resort to repeated experimentation to pick optimal settings. In this paper, we address the issue of choosing the correct number of units in hidden layers. We introduce a method for automatically adjusting network size by pruning out hidden units through $\ell_{\infty,1}$ and $\ell_{2,1}$ regularization. We apply this method to language modeling and demonstrate its ability to correctly choose the number of hidden units while maintaining perplexity. We also include these models in a machine translation decoder and show that these smaller neural models maintain the significant improvements of their unpruned versions.

1 Introduction

Neural networks have proven to be highly effective at many tasks in natural language. For example, neural language models and joint language/translation models improve machine translation quality significantly (Vaswani et al., 2013; Devlin et al., 2014). However, neural networks can be complicated to design and train well. Many decisions need to be made, and performance can be highly dependent on making them correctly. Yet the optimal settings are non-obvious and can be laborious to find, often requiring an extensive grid search involving numerous experiments.

In this paper, we focus on the choice of the sizes of hidden layers. We introduce a method for automatically pruning out hidden layer units, by adding a sparsity-inducing regularizer that encourages units to deactivate if not needed, so that they can be removed from the network. Thus, after training with more units than necessary, a network is produced that has hidden layers correctly sized, saving both time and memory when actually putting the network to use.

Using a neural n-gram language model (Bengio et al., 2003), we are able to show that our novel auto-sizing method is able to learn models that are smaller than models trained without the method, while maintaining nearly the same perplexity. The method has only a single hyperparameter to adjust (as opposed to adjusting the sizes of each of the hidden layers), and we find that the same setting works consistently well across different training data sizes, vocabulary sizes, and n-gram sizes. In addition, we show that incorporating these models into a machine translation decoder still results in large BLEU point improvements. The result is that fewer experiments are needed to obtain models that perform well and are correctly sized.

2 Background

Language models are often used in natural language processing tasks involving generation of text. For instance, in machine translation, the language model helps to output fluent translations, and in speech recognition, the language model helps to disambiguate among possible utterances.

Current language models are usually n-gram models, which look at the previous $(n-1)$ words to predict the nth word in a sequence, based on (smoothed) counts of n-grams collected from training data. These models are simple but very effective in improving the performance of natural language systems.

However, n-gram models suffer from some limitations, such as data sparsity and memory usage. As an alternative, researchers have begun exploring the use of neural networks for language modeling. For modeling n-grams, the most common approach is the feedforward network of Bengio et

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 908–916,
Lisbon, Portugal, 17-21 September 2015.

al. (2003), shown in Figure 1.

Each node represents a *unit* or "neuron," which has a real valued *activation*. The units are organized into real-vector valued *layers*. The activations at each layer are computed as follows. (We assume $n = 3$; the generalization is easy.) The two preceding words, w_1, w_2, are mapped into lower-dimensional word embeddings,

$$\mathbf{x}^1 = \mathbf{A}_{:w_1}$$
$$\mathbf{x}^2 = \mathbf{A}_{:w_2}$$

then passed through two hidden layers,

$$\mathbf{y} = f(\mathbf{B}^1\mathbf{x}^1 + \mathbf{B}^2\mathbf{x}^2 + \mathbf{b})$$
$$\mathbf{z} = f(\mathbf{C}\mathbf{y} + \mathbf{c})$$

where f is an elementwise nonlinear *activation* (or *transfer*) function. Commonly used activation functions are the hyperbolic tangent, logistic function, and rectified linear units, to name a few. Finally, the result is mapped via a softmax to an output probability distribution,

$$P(w_n \mid w_1 \cdots w_{n-1}) \propto \exp([\mathbf{D}\mathbf{z} + \mathbf{d}]_{w_n}).$$

The parameters of the model are $\mathbf{A}$, $\mathbf{B}^1$, $\mathbf{B}^2$, $\mathbf{b}$, $\mathbf{C}$, $\mathbf{c}$, $\mathbf{D}$, and $\mathbf{d}$, which are learned by minimizing the negative log-likelihood of the the training data using stochastic gradient descent (also known as *backpropagation*) or variants.

Vaswani et al. (2013) showed that this model, with some improvements, can be used effectively during decoding in machine translation. In this paper, we use and extend their implementation.

3 Methods

Our method is focused on the challenge of choosing the number of units in the hidden layers of a feed-forward neural network. The networks used for different tasks require different numbers of units, and the layers in a single network also require different numbers of units. Choosing too few units can impair the performance of the network, and choosing too many units can lead to overfitting. It can also slow down computations with the network, which can be a major concern for many applications such as integrating neural language models into a machine translation decoder.

Our method starts out with a large number of units in each layer and then jointly trains the network while pruning out individual units when possible. The goal is to end up with a trained network

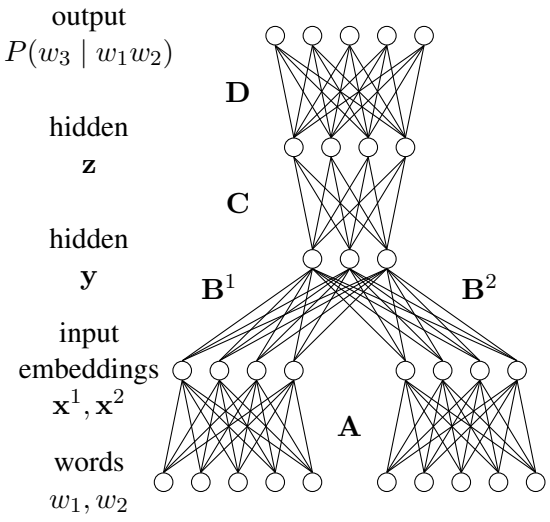


Figure 1: Neural probabilistic language model (Bengio et al., 2003), adapted from Vaswani et al. (2013).

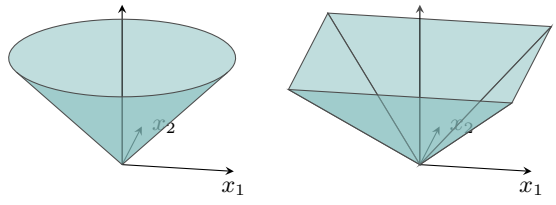
that also has the optimal number of units in each layer.

We do this by adding a regularizer to the objective function. For simplicity, consider a single layer without bias, $\mathbf{y} = f(\mathbf{W}\mathbf{x})$. Let $L(\mathbf{W})$ be the negative log-likelihood of the model. Instead of minimizing $L(\mathbf{W})$ alone, we want to minimize $L(\mathbf{W}) + \lambda R(\mathbf{W})$, where $R(\mathbf{W})$ is a convex regularizer. The ℓ_1 norm, $R(\mathbf{W}) = \|\mathbf{W}\|_1 = \sum_{i,j} |W_{ij}|$, is a common choice for pushing parameters to zero, which can be useful for preventing overfitting and reducing model size. However, we are interested not only in reducing the number of parameters but the number of units. To do this, we need a different regularizer.

We assume activation functions that satisfy $f(0) = 0$, such as the hyperbolic tangent or rectified linear unit ($f(x) = \max\{0, x\}$). Then, if we push the incoming weights of a unit y_i to zero, that is, $W_{ij} = 0$ for all j (as well as the bias, if any: $b_i = 0$), then $y_i = f(0) = 0$ is independent of the previous layers and contributes nothing to subsequent layers. So the unit can be removed without affecting the network at all. Therefore, we need a regularizer that pushes all the incoming connection weights to a unit together towards zero.

Here, we experiment with two, the $\ell_{2,1}$ norm and the $\ell_{\infty,1}$ norm.[1] The $\ell_{2,1}$ norm on a ma-

[1] In the notation $\ell_{p,q}$, the subscript p corresponds to the norm over each group of parameters, and q corresponds to the norm over the group norms. Contrary to more common usage, in this paper, the groups are rows, not columns.



ℓ_2 ℓ_∞

Figure 2: The (unsquared) ℓ_2 norm and ℓ_∞ norm both have sharp tips at the origin that encourage sparsity.

trix $\mathbf{W}$ is

$$R(\mathbf{W}) = \sum_i \|W_{i:}\|_2 = \sum_i \left(\sum_j W_{ij}^2 \right)^{\frac{1}{2}}. \quad (1)$$

(If there are biases b_i, they should be included as well.) This puts equal pressure on each row, but within each row, the larger values contribute more, and therefore there is more pressure on larger values towards zero. The $\ell_{\infty,1}$ norm is

$$R(\mathbf{W}) = \sum_i \|W_{i:}\|_\infty = \sum_i \max_j |W_{ij}|. \quad (2)$$

Again, this puts equal pressure on each row, but within each row, only the maximum value (or values) matter, and therefore the pressure towards zero is entirely on the maximum value(s).

Figure 2 visualizes the sparsity-inducing behavior of the two regularizers on a single row. Both have a sharp tip at the origin that encourages all the parameters in a row to become exactly zero.

4 Optimization

However, this also means that sparsity-inducing regularizers are not differentiable at zero, making gradient-based optimization methods trickier to apply. The methods we use are discussed in detail elsewhere (Duchi et al., 2008; Duchi and Singer, 2009); in this section, we include a short description of these methods for completeness.

4.1 Proximal gradient method

Most work on learning with regularizers, including this work, can be thought of as instances of the *proximal gradient* method (Parikh and Boyd, 2014). Our objective function can be split into two parts, a convex and differentiable part (L) and a convex but non-differentiable part (λR). In proximal gradient descent, we alternate between improving L alone and λR alone. Let $\mathbf{u}$ be the parameter values from the previous iteration. We compute new parameter values $\mathbf{w}$ using:

$$\mathbf{v} \leftarrow \mathbf{u} - \eta \nabla L(\mathbf{u}) \quad (3)$$

$$\mathbf{w} \leftarrow \arg\max_{\mathbf{w}} \left(\frac{1}{2\eta} \|\mathbf{w} - \mathbf{v}\|^2 + \lambda R(\mathbf{w}) \right) \quad (4)$$

and repeat until convergence. The first update is just a standard gradient descent update on L; the second is known as the *proximal operator* for λR and in many cases has a closed-form solution. In the rest of this section, we provide some justification for this method, and in Sections 4.2 and 4.3 we show how to compute the proximal operator for the ℓ_2 and ℓ_∞ norms.

We can think of the gradient descent update (3) on L as follows. Approximate L around $\mathbf{u}$ by the tangent plane,

$$\bar{L}(\mathbf{v}) = L(\mathbf{u}) + \nabla L(\mathbf{u})(\mathbf{v} - \mathbf{u}) \quad (5)$$

and move $\mathbf{v}$ to minimize $\bar{L}$, but don't move it too far from $\mathbf{u}$; that is, minimize

$$F(\mathbf{v}) = \frac{1}{2\eta} \|\mathbf{v} - \mathbf{u}\|^2 + \bar{L}(\mathbf{v}).$$

Setting partial derivatives to zero, we get

$$\frac{\partial F}{\partial \mathbf{v}} = \frac{1}{\eta}(\mathbf{v} - \mathbf{u}) + \nabla L(\mathbf{u}) = 0$$
$$\mathbf{v} = \mathbf{u} - \eta \nabla L(\mathbf{u}).$$

By a similar strategy, we can derive the second step (4). Again we want to move $\mathbf{w}$ to minimize the objective function, but don't want to move it too far from $\mathbf{u}$; that is, we want to minimize:

$$G(\mathbf{w}) = \frac{1}{2\eta} \|\mathbf{w} - \mathbf{u}\|^2 + \bar{L}(\mathbf{w}) + \lambda R(\mathbf{w}).$$

Note that we have *not* approximated R by a tangent plane. We can simplify this by substituting in (3). The first term becomes

$$\frac{1}{2\eta} \|\mathbf{w} - \mathbf{u}\|^2 = \frac{1}{2\eta} \|\mathbf{w} - \mathbf{v} - \eta \nabla L(\mathbf{u})\|^2$$
$$= \frac{1}{2\eta} \|\mathbf{w} - \mathbf{v}\|^2 - \nabla L(\mathbf{u})(\mathbf{w} - \mathbf{v}) + \frac{\eta}{2} \|\nabla L(\mathbf{u})\|^2$$

and the second term becomes

$$\begin{aligned}\bar{L}(\mathbf{w}) &= L(\mathbf{u}) + \nabla L(\mathbf{u})(\mathbf{w} - \mathbf{u}) \\ &= L(\mathbf{u}) + \nabla L(\mathbf{u})(\mathbf{w} - \mathbf{v} - \eta \nabla L(\mathbf{u})).\end{aligned}$$

The $\nabla L(\mathbf{u})(\mathbf{w} - \mathbf{v})$ terms cancel out, and we can ignore terms not involving $\mathbf{w}$, giving

$$G(\mathbf{w}) = \frac{1}{2\eta}\|\mathbf{w} - \mathbf{v}\|^2 + \lambda R(\mathbf{w}) + \text{const.}$$

which is minimized by the update (4). Thus, we have split the optimization step into two easier steps: first, do the update for L (3), then do the update for λR (4). The latter can often be done exactly (without approximating R by a tangent plane). We show next how to do this for the ℓ_2 and ℓ_∞ norms.

4.2 ℓ_2 and $\ell_{2,1}$ regularization

Since the $\ell_{2,1}$ norm on matrices (1) is separable into the ℓ_2 norm of each row, we can treat each row separately. Thus, for simplicity, assume that we have a single row and want to minimize

$$G(\mathbf{w}) = \frac{1}{2\eta}\|\mathbf{w} - \mathbf{v}\|^2 + \lambda\|\mathbf{w}\| + \text{const.}$$

The minimum is either at $\mathbf{w} = 0$ (the tip of the cone) or where the partial derivatives are zero (Figure 3):

$$\frac{\partial G}{\partial \mathbf{w}} = \frac{1}{\eta}(\mathbf{w} - \mathbf{v}) + \lambda\frac{\mathbf{w}}{\|\mathbf{w}\|} = 0.$$

Clearly, $\mathbf{w}$ and $\mathbf{v}$ must have the same direction and differ only in magnitude, that is, $\mathbf{w} = \alpha\frac{\mathbf{v}}{\|\mathbf{v}\|}$. Substituting this into the above equation, we get the solution

$$\alpha = \|\mathbf{v}\| - \eta\lambda.$$

Therefore the update is

$$\begin{aligned}\mathbf{w} &= \alpha\frac{\mathbf{v}}{\|\mathbf{v}\|} \\ \alpha &= \max(0, \|\mathbf{v}\| - \eta\lambda).\end{aligned}$$

4.3 ℓ_∞ and $\ell_{\infty,1}$ regularization

As above, since the $\ell_{\infty,1}$ norm on matrices (2) is separable into the ℓ_∞ norm of each row, we can treat each row separately; thus, we want to minimize

$$G(\mathbf{w}) = \frac{1}{2\eta}\|\mathbf{w} - \mathbf{v}\|^2 + \lambda\max_j |x_j| + \text{const.}$$

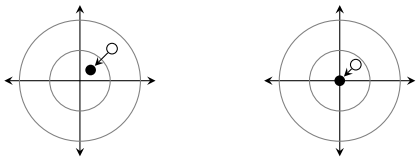


Figure 3: Examples of the two possible cases for the ℓ_2 gradient update. Point $\mathbf{v}$ is drawn with a hollow dot, and point $\mathbf{w}$ is drawn with a solid dot.

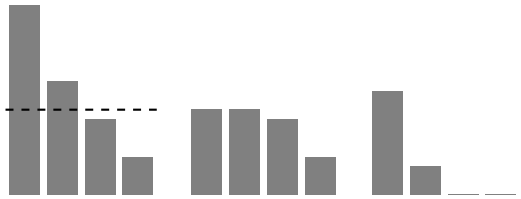


Figure 4: The proximal operator for the ℓ_∞ norm (with strength $\eta\lambda$) decreases the maximal components until the total decrease sums to $\eta\lambda$. Projection onto the ℓ_1-ball (of radius $\eta\lambda$) decreases each component by an equal amount until they sum to $\eta\lambda$.

Intuitively, the solution can be characterized as: Decrease all of the maximal $|x_j|$ until the total decrease reaches $\eta\lambda$ or all the x_j are zero. See Figure 4.

If we pre-sort the $|x_j|$ in nonincreasing order, it's easy to see how to compute this: for $\rho = 1, \ldots, n$, see if there is a value $\xi \leq x_\rho$ such that decreasing all the $x_1, \ldots, x_\rho$ to ξ amounts to a total decrease of $\eta\lambda$. The largest ρ for which this is possible gives the correct solution.

But this situation seems similar to another optimization problem, projection onto the ℓ_1-ball, which Duchi et al. (2008) solve in linear time without pre-sorting. In fact, the two problems can be solved by nearly identical algorithms, because they are convex conjugates of each other (Duchi and Singer, 2009; Bach et al., 2012). Intuitively, the ℓ_1 projection of v is exactly what is cut out by the ℓ_∞ proximal operator, and vice versa (Figure 4).

Duchi et al.'s algorithm modified for the present problem is shown as Algorithm 1. It partitions the x_j about a pivot element (line 6) and tests whether it and the elements to its left can be decreased to a value ξ such that the total decrease is δ (line 8). If so, it recursively searches the right side; if not, the

left side. At the conclusion of the algorithm, ρ is set to the largest value that passes the test (line 13), and finally the new x_j are computed (line 16) – the only difference from Duchi et al.'s algorithm.

This algorithm is asymptotically faster than that of Quattoni et al. (2009). They reformulate $\ell_{\infty,1}$ regularization as a constrained optimization problem (in which the $\ell_{\infty,1}$ norm is bounded by μ) and provide a solution in $\mathcal{O}(n \log n)$ time. The method shown here is simpler and faster because it can work on each row separately.

Algorithm 1 Linear-time algorithm for the proximal operator of the ℓ_∞ norm.

```
1:  procedure UPDATE(w, δ)
2:      lo, hi ← 1, n
3:      s ← 0
4:      while lo ≤ hi do
5:          select md randomly from lo, ..., hi
6:          ρ ← PARTITION(w, lo, md, hi)
7:          ξ ← (1/ρ) (s + Σ_{i=lo}^{ρ} |x_i| − δ)
8:          if ξ ≤ |x_ρ| then
9:              s ← s + Σ_{i=lo}^{ρ} |x_i|
10:             lo ← ρ + 1
11:         else
12:             hi ← ρ − 1
13:     ρ ← hi
14:     ξ ← (1/ρ) (s − δ)
15:     for i ← 1, ..., n do
16:         x_i ← min(max(x_i, −ξ), ξ)
17: procedure PARTITION(w, lo, md, hi)
18:     swap x_lo and x_md
19:     i ← lo + 1
20:     for j ← lo + 1, ..., hi do
21:         if x_j ≥ x_lo then
22:             swap x_i and x_j
23:             i ← i + 1
24:     swap x_lo and x_{i−1}
25:     return i − 1
```

5 Experiments

We evaluate our model using the open-source NPLM toolkit released by Vaswani et al. (2013), extending it to use the additional regularizers as described in this paper.[2] We use a vocabulary size of 100k and word embeddings with 50 dimensions. We use two hidden layers of rectified linear units (Nair and Hinton, 2010).

[2]These extensions have been contributed to the NPLM project.

We train neural language models (LMs) on two natural language corpora, Europarl v7 English and the AFP portion of English Gigaword 5. After tokenization, Europarl has 56M tokens and Gigaword AFP has 870M tokens. For both corpora, we hold out a validation set of 5,000 tokens. We train each model for 10 iterations over the training data.

Our experiments break down into three parts. First, we look at the impact of our pruning method on perplexity of a held-out validation set, across a variety of settings. Second, we take a closer look at how the model evolves through the training process. Finally, we explore the downstream impact of our method on a statistical phrase-based machine translation system.

5.1 Evaluating perplexity and network size

We first look at the impact that the $\ell_{\infty,1}$ regularizer has on the perplexity of our validation set. The main results are shown in Table 1. For $\lambda \leq 0.01$, the regularizer seems to have little impact: no hidden units are pruned, and perplexity is also not affected. For $\lambda = 1$, on the other hand, most hidden units are pruned – apparently too many, since perplexity is worse. But for $\lambda = 0.1$, we see that we are able to prune out many hidden units: up to half of the first layer, with little impact on perplexity. We found this to be consistent across all our experiments, varying n-gram size, initial hidden layer size, and vocabulary size.

Table 2 shows the same information for 5-gram models trained on the larger Gigaword AFP corpus. These numbers look very similar to those on Europarl: again $\lambda = 0.1$ works best, and, counter to expectation, even the final number of units is similar.

Table 3 shows the result of varying the vocabulary size: again $\lambda = 0.1$ works best, and, although it is not shown in the table, we also found that the final number of units did not depend strongly on the vocabulary size.

Table 4 shows results using the $\ell_{2,1}$ norm (Europarl corpus, 5-grams, 100k vocabulary). Since this is a different regularizer, there isn't any reason to expect that λ behaves the same way, and indeed, a smaller value of λ seems to work best.

5.2 A closer look at training

We also studied the evolution of the network over the training process to gain some insights into how the method works. The first question we want to

	2-gram			3-gram			5-gram		
λ	layer 1	layer 2	ppl	layer 1	layer 2	ppl	layer 1	layer 2	ppl
0	1,000	50	103	1,000	50	66	1,000	50	55
0.001	1,000	50	104	1,000	50	66	1,000	50	54
0.01	1,000	50	104	1,000	50	63	1,000	50	55
0.1	499	47	105	652	49	66	784	50	55
1.0	50	24	111	128	32	76	144	29	68

Table 1: Comparison of $\ell_{\infty,1}$ regularization on 2-gram, 3-gram, and 5-gram neural language models. The network initially started with 1,000 units in the first hidden layer and 50 in the second. A regularization strength of $\lambda = 0.1$ consistently is able to prune units while maintaining perplexity, even though the final number of units varies considerably across models. The vocabulary size is 100k.

λ	layer 1	layer 2	perplexity
0	1,000	50	100
0.001	1,000	50	99
0.01	1,000	50	101
0.1	742	50	107
1.0	24	17	173

Table 2: Results from training a 5-gram neural LM on the AFP portion of the Gigaword dataset. As with the smaller Europarl corpus (Table 1), a regularization strength of $\lambda = 0.1$ is able to prune units while maintaining perplexity.

	vocabulary size			
λ	10k	25k	50k	100k
0	47	60	54	55
0.001	47	54	54	54
0.01	47	58	55	55
0.1	48	62	55	55
1.0	61	64	65	68

Table 3: A regularization strength of $\lambda = 0.1$ is best across different vocabulary sizes.

λ	layer 1	layer 2	perplexity
0	1,000	50	100
0.0001	1,000	50	54
0.001	1,000	50	55
0.01	616	50	57
0.1	199	32	65

Table 4: Results using $\ell_{2,1}$ regularization.

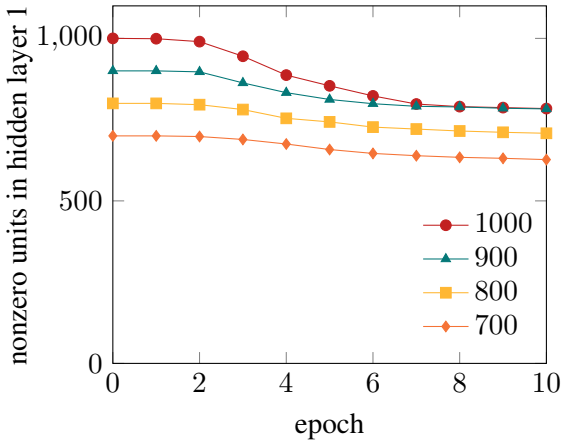


Figure 5: Number of units in first hidden layer over time, with various starting sizes ($\lambda = 0.1$). If we start with too many units, we end up with the same number, although if we start with a smaller number of units, a few are still pruned away.

answer is whether the method is simply removing units, or converging on an optimal number of units. Figure 5 suggests that it is a little of both: if we start with too many units (900 or 1000), the method converges to the same number regardless of how many extra units there were initially. But if we start with a smaller number of units, the method still prunes away about 50 units.

Next, we look at the behavior over time of different regularization strengths λ. We found that not only does $\lambda = 1$ prune out too many units, it does so at the very first iteration (Figure 6, above), perhaps prematurely. By contrast, the $\lambda = 0.1$ run prunes out units gradually. By plotting these curves together with perplexity (Figure 6, below), we can see that the $\lambda = 0.1$ run is fitting the model and pruning it at the same time, which seems preferable to fitting without any pruning ($\lambda =$

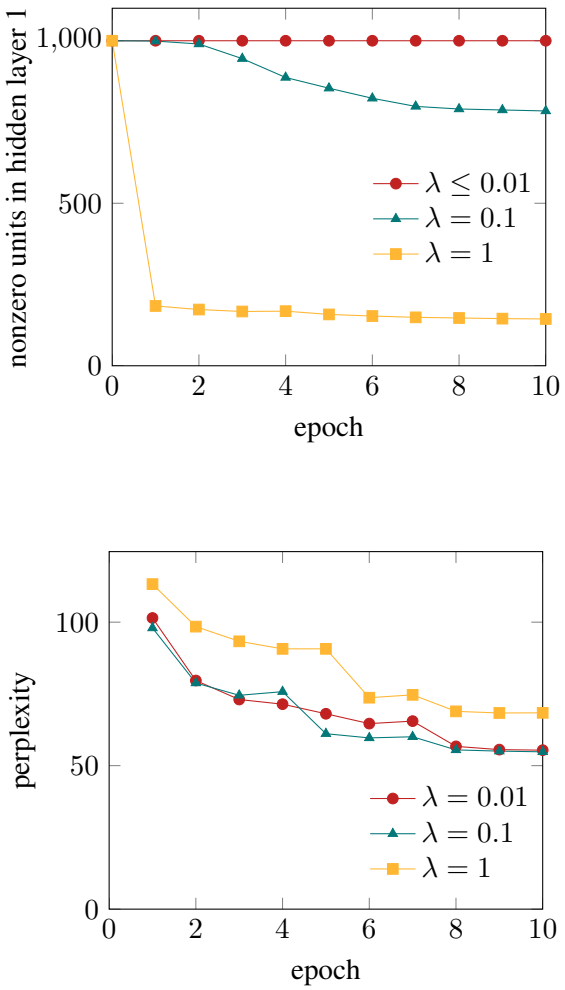


Figure 6: Above: Number of units in first hidden layer over time, for various regularization strengths λ. A regularization strength of ≤ 0.01 does not zero out any rows, while a strength of 1 zeros out rows right away. Below: Perplexity over time. The runs with $\lambda \leq 0.1$ have very similar learning curves, whereas $\lambda = 1$ is worse from the beginning.

	neural LM		
λ	none	Europarl	Gigaword AFP
0 (none)	23.2	24.7 (+1.5)	25.2 (+2.0)
0.1		24.6 (+1.4)	24.9 (+1.7)

Table 5: The improvements in translation accuracy due to the neural LM (shown in parentheses) are affected only slightly by $\ell_{\infty,1}$ regularization. For the Europarl LM, there is no statistically significant difference, and for the Gigaword AFP LM, a statistically significant but small decrease of -0.3.

0.01) or pruning first and then fitting ($\lambda = 1$).

We can also visualize the weight matrix itself over time (Figure 7), for $\lambda = 0.1$. It is striking that although this setting fits the model and prunes it at the same time, as argued above, by the first iteration it already seems to have decided roughly how many units it will eventually prune.

5.3 Evaluating on machine translation

We also looked at the impact of our method on statistical machine translation systems. We used the Moses toolkit (Koehn et al., 2007) to build a phrase based machine translation system with a traditional 5-gram LM trained on the target side of our bitext. We augmented this system with neural LMs trained on the Europarl data and the Gigaword AFP data. Based on the results from the perplexity experiments, we looked at models both built with a $\lambda = 0.1$ regularizer, and without regularization ($\lambda = 0$).

We built our system using the newscommentary dataset v8. We tuned our model using newstest13 and evaluated using newstest14. After standard cleaning and tokenization, there were 155k parallel sentences in the newscommentary dataset, and 3,000 sentences each for the tuning and test sets.

Table 5 shows that the addition of a neural LM helps substantially over the baseline, with improvements of up to 2 BLEU. Using the Europarl model, the BLEU scores obtained without and with regularization were not significantly different ($p \geq 0.05$), consistent with the negligible perplexity difference between these models. On the Gigaword AFP model, regularization did decrease the BLEU score by 0.3, consistent with the small perplexity increase of the regularized model. The decrease is statistically significant, but small compared with the overall benefit of adding a neural LM.

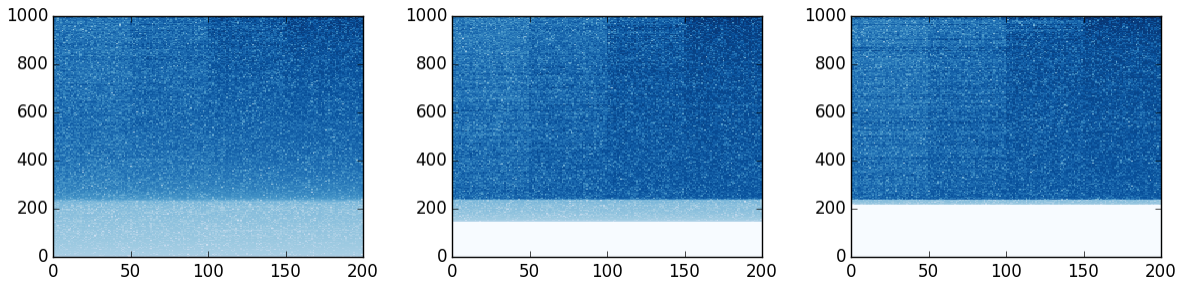


Figure 7: Evolution of the first hidden layer weight matrix after 1, 5, and 10 iterations (with rows sorted by ℓ_∞ norm). A nonlinear color scale is used to show small values more clearly. The four vertical blocks correspond to the four context words. The light bar at the bottom is the rows that are close to zero, and the white bar is the rows that are exactly zero.

6 Related Work

Researchers have been exploring the use of neural networks for language modeling for a long time. Schmidhuber and Heil (1996) proposed a character n-gram model using neural networks which they used for text compression. Xu and Rudnicky (2000) proposed a word-based probability model using a softmax output layer trained using cross-entropy, but only for bigrams. Bengio et al. (2003) defined a probabilistic word n-gram model and demonstrated improvements over conventional smoothed language models. Mnih and Teh (2012) sped up training of log-bilinear language models through the use of noise-contrastive estimation (NCE). Vaswani et al. (2013) also used NCE to train the architecture of Bengio et al. (2003), and were able to integrate a large-vocabulary language model directly into a machine translation decoder. Baltescu et al. (2014) describe a similar model, with extensions like a hierarchical softmax (based on Brown clustering) and direct n-gram features.

Beyond feed-forward neural network language models, researchers have explored using more complicated neural network architectures. RNNLM is an open-source implementation of a language model using recurrent neural networks (RNN) where connections between units can form directed cycles (Mikolov et al., 2011). Sundermeyer et al. (2015) use the long-short term memory (LSTM) neural architecture to show a perplexity improvement over the RNNLM toolkit. In future work, we plan on exploring how our method could improve these more complicated neural models as well.

Automatically limiting the size of neural networks is an old idea. The "Optimal Brain Damage" (OBD) technique (LeCun et al., 1989) computes a *saliency* based on the second derivative of the objective function with respect to each parameter. The parameters are then sorted by saliency, and the lowest-saliency parameters are pruned. The pruning process is separate from the training process, whereas regularization performs training and pruning simultaneously. Regularization in neural networks is also an old idea; for example, Nowland and Hinton (1992) mention both ℓ_2^2 and ℓ_0 regularization. Our method develops on this idea by using a mixed norm to prune units, rather than parameters.

Srivastava et al. introduce a method called *dropout* in which units are directly deactivated at random during training (Srivastava et al., 2014), which induces sparsity in the hidden unit activations. However, at the end of training, all units are reactivated, as the goal of dropout is to reduce overfitting, not to reduce network size. Thus, dropout and our method seem to be complementary.

7 Conclusion

We have presented a method for auto-sizing a neural network during training by removing units using a $\ell_{\infty,1}$ regularizer. This regularizer drives a unit's input weights as a group down to zero, allowing the unit to be pruned. We can thus prune units out of our network during training with minimal impact to held-out perplexity or downstream performance of a machine translation system.

Our results showed empirically that the choice

of a regularization coefficient of 0.1 was robust to initial configuration parameters of initial network size, vocabulary size, n-gram order, and training corpus. Furthermore, imposing a single regularizer on the objective function can tune all of the hidden layers of a network with one setting. This reduces the need to conduct expensive, multi-dimensional grid searches in order to determine optimal sizes.

We have demonstrated the power and efficacy of this method on a feed-forward neural network for language modeling though experiments on perplexity and machine translation. However, this method is general enough that it should be applicable to other domains, both inside natural language processing and outside. As neural models become more pervasive in natural language processing, the ability to auto-size networks for fast experimentation and quick exploration will become increasingly important.

Acknowledgments

We would like to thank Tomer Levinboim, Antonios Anastasopoulos, and Ashish Vaswani for their helpful discussions, as well as the reviewers for their assistance and feedback.

References

Francis Bach, Rodolphe Jenatton, Julien Mairal, and Guillaume Obozinski. 2012. Optimization with sparsity-inducing penalties. *Foundations and Trends in Machine Learning*, 4(1):1–106.

Paul Baltescu, Phil Blunsom, and Hieu Hoang. 2014. OxLM: A neural language modelling framework for machine translation. *Prague Bulletin of Mathematical Linguistics*, 102(1):81–92.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *J. Machine Learning Research*, 3:1137–1155.

Jacob Devlin, Rabih Zbib, Zhongqiang Huang, Thomas Lamar, Richard Schwartz, and John Makhoul. 2014. Fast and robust neural network joint models for statistical machine translation. In *Proc. ACL*, pages 1370–1380.

John Duchi and Yoram Singer. 2009. Efficient online and batch learning using forward backward splitting. *J. Machine Learning Research*, 10:2899–2934.

John Duchi, Shai Shalev-Shwartz, Yoram Singer, and Tushar Chandra. 2008. Efficient projections onto the ℓ_1-ball for learning in high dimensions. In *Proc. ICML*, pages 272–279.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proc. ACL, Interactive Poster and Demonstration Sessions*, pages 177–180.

Yann LeCun, John S. Denker, Sara A. Solla, Richard E. Howard, and Lawrence D. Jackel. 1989. Optimal brain damage. In *Proc. NIPS*, volume 2, pages 598–605.

Tomas Mikolov, Stefan Kombrink, Anoop Deoras, Lukar Burget, and Jan Cernocky. 2011. RNNLM - recurrent neural network language modeling toolkit. In *Proc. ASRU*, pages 196–201.

Andriy Mnih and Yee Whye Teh. 2012. A fast and simple algorithm for training neural probabilistic language models. In *Proc. ICML*, pages 1751–1758.

Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve Restricted Boltzmann Machines. In *Proc. ICML*, pages 807–814.

Steven J. Nowland and Geoffrey E. Hinton. 1992. Simplifying neural networks by soft weight-sharing. *Neural Computation*, 4:473–493.

Neal Parikh and Stephen Boyd. 2014. Proximal algorithms. *Foundations and Trends in Optimization*, 1(3):127–239.

Ariadna Quattoni, Xavier Carreras, Michael Collins, and Trevor Darrell. 2009. An efficient projection for $l_{1,\infty}$ regularization. In *Proc. ICML*, pages 857–864.

Jurgen Schmidhuber and Stefan Heil. 1996. Sequential neural text compression. *IEEE Transactions on Neural Networks*, 7:142–146.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *J. Machine Learning Research*, 15(1):1929–1958.

Martin Sundermeyer, Hermann Ney, and Ralf Schlüter. 2015. From feedforward to recurrent LSTM neural networks for language modeling. *Trans. Audio, Speech, and Language*, 23(3):517–529.

Ashish Vaswani, Yinggong Zhao, Victoria Fossum, and David Chiang. 2013. Decoding with large-scale neural language models improves translation. In *Proc. EMNLP*, pages 1387–1392.

Wei Xu and Alexander I. Rudnicky. 2000. Can artificial neural networks learn language models? In *Proc. International Conference on Statistical Language Processing*, pages M1–13.

Dual Decomposition Inference for Graphical Models over Strings*

Nanyun Peng and **Ryan Cotterell** and **Jason Eisner**
Department of Computer Science, Johns Hopkins University
{npeng1,ryan.cotterell,eisner}@jhu.edu

Abstract

We investigate dual decomposition for joint MAP inference of many strings. Given an arbitrary graphical model, we decompose it into small acyclic sub-models, whose MAP configurations can be found by finite-state composition and dynamic programming. We force the solutions of these subproblems to agree on overlapping variables, by tuning Lagrange multipliers for an adaptively expanding set of variable-length n-gram count features.

This is the first inference method for arbitrary graphical models over strings that does not require approximations such as random sampling, message simplification, or a bound on string length. *Provided that* the inference method terminates, it gives a certificate of global optimality (though MAP inference in our setting is undecidable in general). On our global phonological inference problems, it always terminates, and achieves more accurate results than max-product and sum-product loopy belief propagation.

1 Introduction

Graphical models allow expert modeling of complex relations and interactions between random variables. Since a graphical model with given parameters defines a probability distribution, it can be used to reconstruct values for unobserved variables. The **marginal inference** problem is to compute the posterior marginal distributions of these variables. The **MAP inference** (or **MPE**) problem is to compute the single highest-probability joint assignment to all the unobserved variables.

Inference in general graphical models is NP-hard even when the variables' values are finite discrete values such as categories, tags or domains. In this paper, we address the more challenging setting where the variables in the graphical models range over strings. Thus, the domain of the variables is an *infinite* space of discrete structures.

In NLP, such graphical models can deal with large, incompletely observed lexicons. They could be used to model diverse relationships among strings that represent spellings or pronunciations; morphemes, words, phrases (such as named entities and URLs), or utterances; standard or variant forms; clean or noisy forms; contemporary or historical forms; underlying or surface forms; source or target language forms. Such relationships arise in domains such as morphology, phonology, historical linguistics, translation between related languages, and social media text analysis.

In this paper, we assume a *given* graphical model, whose factors evaluate the relationships among observed and unobserved strings.[1] We present a dual decomposition algorithm for MAP inference, which returns a certifiably optimal solution when it converges. We demonstrate our method on a graphical model for phonology proposed by Cotterell et al. (2015). We show that the method generally converges and that it achieves better results than alternatives.

The rest of the paper is arranged as follows: We will review graphical models over strings in section 2, and briefly introduce our sample problem in section 3. Section 4 develops dual decomposition inference for graphical models over strings. Then our experimental setup and results are presented in sections 5 and 6, with some discussion.

2 Graphical Models Over Strings

2.1 Factor Graphs and MAP Inference

To perform inference on a graphical model (directed or undirected), one first converts the model to a factor graph representation (Kschischang et al., 2001). A **factor graph** is a finite bipartite

*This material is based upon work supported by the National Science Foundation under Grant No. 1423276.

[1] In some task settings, it is also necessary to discover the model topology along with the model parameters. In this paper we do not treat that structure learning problem. However, both structure learning and parameter learning need to call inference—such as the method presented here—in order to evaluate proposed topologies or improve their parameters.

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 917–927,
Lisbon, Portugal, 17-21 September 2015.

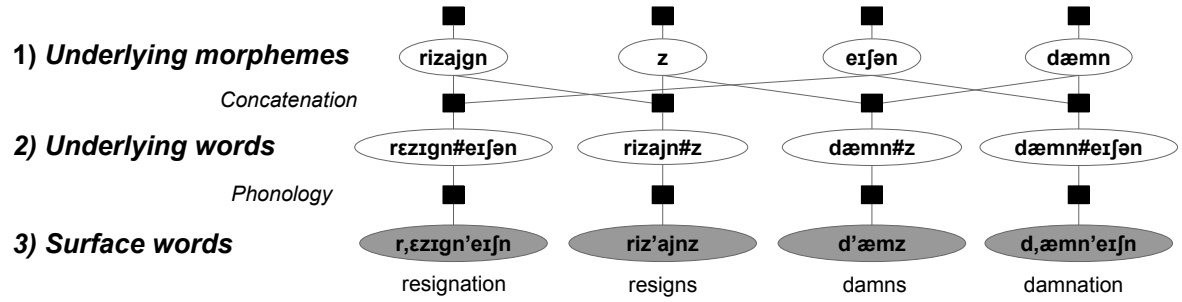


Figure 1: A fragment of the factor graph for the directed graphical model of Cotterell et al. (2015), displaying a possible assignment to the variables (ellipses). The model explains each observed surface word as the result of applying phonology to a concatenation of underlying morphemes. Shaded variables show the observed surface forms for four words: *resignation*, *resigns*, *damns*, and *damnation*. The underlying pronunciations of these words are assumed to be more similar than their surface pronunciations, because the words are known to share latent morphemes. The factor graph encodes what is shared. Each observed word at layer 3 has a latent underlying form at layer 2, which is a deterministic concatenation of latent morphemes at layer 1. The binary factors between layers 2 and 3 score each (underlying,surface) pair for its phonological plausibility. The unary factors at layer 1 score each morpheme for its lexical plausibility. See Cotterell et al. (2015) for discussion of alternatives.

graph over a set $\mathcal{X} = \{X_1, X_2, \ldots\}$ of variables and a set $\mathcal{F}$ of factors. An **assignment** to the variables is a vector of values $\mathbf{x} = (x_1, x_2, \ldots)$. Each factor $F \in \mathcal{F}$ is a real-valued function of $\mathbf{x}$, but it depends on a given x_i *only if* F is connected to X_i in the graph. Thus, a degree d-factor scores some length-d subtuple of $\mathbf{x}$. The score of the whole joint assignment simply sums over all factors:

$$\text{score}(\mathbf{x}) \stackrel{\text{def}}{=} \sum_{F \in \mathcal{F}} F(\mathbf{x}). \quad (1)$$

We seek the $\mathbf{x}$ of maximum score that is consistent with our partial observation of $\mathbf{x}$. This is a generic constraint satisfaction problem with soft constraints. While our algorithm does not depend on a probabilistic interpretation of the factor graph,[2] it can be regarded as peforming maximum *a posteriori* (MAP) inference of the unobserved variables, under the probability distribution $p(\mathbf{x}) \stackrel{\text{def}}{=} (1/Z) \exp \text{score}(\mathbf{x})$.

2.2 The String Case

Graphical models over strings have enjoyed some attention in the NLP community. Tree-shaped graphical models naturally model the evolutionary tree of word forms (Bouchard-Côté et al., 2007; Bouchard-Côté et al., 2008; Hall and Klein, 2010; Hall and Klein, 2011). Cyclic graphical models have been used to model morphological paradigms (Dreyer and Eisner, 2009; Dreyer and Eisner, 2011) and to reconstruct phonological underlying forms of words (Cotterell et al., 2015).

The variables in such a model are strings of unbounded length: each variable X_i is permitted to range over Σ^* where Σ is some fixed, finite alphabet. As in previous work, we assume that a degree-d factor is a d-way rational relation, i.e., a function of d strings that can be computed by a d-tape **weighted finite-state machine (WFSM)** (Mohri et al., 2002; Kempe et al., 2004). Such a machine is called an **acceptor (WFSA)** if $d = 1$ or a **transducer (WFST)** if $d = 2$.[3]

Past work has shown how to approximately *sample* from the distribution over $\mathbf{x}$ defined by such a model (Bouchard-Côté et al., 2007), or approximately compute the distribution's *marginals* using variants of **sum-product belief propagation** (BP) (Dreyer and Eisner, 2009) and expectation propagation (EP) (Cotterell and Eisner, 2015).

2.3 Finite-State Belief Propagation

BP iteratively updates **messages** between factors and variables. Each message is a vector whose elements score the possible values of a variable.

Murphy et al. (1999) discusses BP on cyclic ("loopy") graphs. For pedagogical reasons, suppose momentarily that all factors have degree ≤ 2 (this loses no power). Then BP manipulates only vectors and matrices—whose *dimensionality* depends on the number of possible values of the vari-

[2] E.g., it could be used for exactly computing the separation oracle when training a structural SVM (Tsochantaridis et al., 2005; Finley and Joachims, 2007). Another use is minimum Bayes risk decoding—computing the joint assignment having minimum expected loss—if the loss function does not decompose over the variables, but a factor graph can be constructed that evaluates the expected loss of any assignment.

[3] Finite-state software libraries often support only these cases. Accordingly, Cotterell and Eisner (2015, Appendix B.10) explain how to eliminate factors of degree $d > 2$.

ables. In the string case, they have *infinitely* many rows and columns, indexed by possible strings.

Dreyer and Eisner (2009) represented these infinite vectors and matrices by WFSAs and WFSTs, respectively. They observed that the simple linear-algebra operations used by BP can be implemented by finite-state constructions. The pointwise product of two vectors is the intersection of their WFSAs; the marginalization of a matrix is the projection of its WFST; a vector-matrix product is computed by composing the WFSA with the WFST and then projecting onto the output tape. For degree > 2, BP's rank-d tensors become d-tape WFSMs, and these constructions generalize.

Unfortunately, except in small acyclic models, the BP messages—which are WFSAs—usually become impractically large. Each intersection or composition involves a cross-product construction. For example, when finding the marginal distribution at a degree-d variable, intersecting d WFSA messages having m states each may yield a WFSA with up to m^d states. (Our models in section 6 include variables with d up to 156.) Combining *many* cross products, as BP iteratively passes messages along a path in the factor graph, leads to blowup that is exponential in the length of the path—which in turn is unbounded if the graph has cycles (Dreyer and Eisner, 2009), as ours do.

The usual solution is to prune or otherwise approximate the messages at each step. In particular, Cotterell and Eisner (2015) gave a principled way to approximate the messages using variable-length n-gram models, using an adaptive variant of Expectation Propagation (Minka, 2001).

2.4 Dual Decomposition Inference

In section 4, we will present a dual decomposition (DD) method that decomposes the original complex problem into many small subproblems that are free of cycles and high degree nodes. BP can solve each subproblem without approximation.[4]

The subproblems "communicate" through Lagrange multipliers that guide them towards agreement on a single global solution. This information is encoded in WFSAs that score possible values of a string variable. DD incrementally adjusts the WFSAs so as to encourage values that agree with the variable's average value across subproblems.

Unlike BP messages, the WFSAs in our DD method will be restricted to be variable-length n-gram models, similar to Cotterell and Eisner (2015). They may still grow over time; but DD often halts while the WFSAs are still small. It halts when its strings agree exactly, rather than when it has converged up to a numerical tolerance, like BP.

2.5 Switching Between Semirings

Our factors may be *nondeterministic* WFSMs. So when $F \in \mathcal{F}$ scores a given d-tuple of string values, it may accept that d-tuple along multiple different WFSM paths with different scores, corresponding to different alignments of the strings.

For purposes of MAP inference, we define F to return the *maximum* of these path scores. That is, we take the WFSMs to be defined with weights in the $(\max, +)$ semiring (Mohri et al., 2002). Equivalently, we are seeking the "best global solution" in the sense of choosing not only the strings x_i but also the alignments of the d-tuples.[5]

To do so, we must solve each DD subproblem in the same sense. We use **max-product BP**. This still applies the Dreyer-Eisner method of section 2.3. Since these WFSMs are defined in the $(\max, +)$ semiring, the method's finite-state operations will combine weights using $\max$ and $+$.

MAP inference in our setting is in general computationally undecidable.[6] However, if DD converges (as in our experiments), then *its solution is guaranteed to be the true MAP assignment.*

In section 6, we will compare DD with (loopy) max-product BP and (loopy) sum-product BP. These respectively approximate MAP inference and marginal inference over the entire factor graph. Marginal inference computes marginal string probabilities that sum (rather than maximize) over the choices of other strings *and the choices of paths*. Thus, for sum-product BP, we re-interpret the factor WFSMs as defined over the $(\text{logadd}, +)$ semiring. This means that the exponentiated score assigned by a WFSM is the sum of the exponentiated scores of the accepting paths.

[4] Such small BP problems commonly arise in NLP. In particular, using finite-state methods to decode a composition of several finite-state noisy channels (Pereira and Riley, 1997; Knight and Graehl, 1998) can be regarded as BP on a graphical model over strings that has a linear-chain topology.

[5] This problem is more specifically called **MPE inference**.

[6] The trouble is that we cannot bound the length of the latent strings. If we could, then we could encode them using a finite set of boolean variables, and solve as an ILP problem. But that would allow us to determine whether there exists a MAP assignment with score ≥ 0. That is impossible in general, because it would solve Post's Correspondence Problem as a simple special case (see Dreyer and Eisner (2009)).

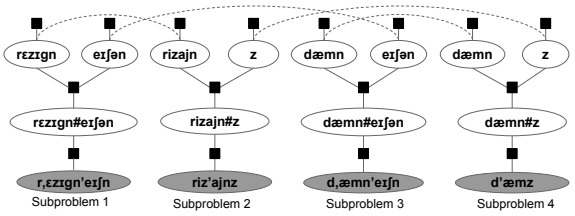


Figure 2: To apply dual decomposition, we choose to decompose 1 into one subproblem per surface word. Dashed lines connect two or more variables from different subproblems that correspond to the same variable in the original graph. The method of Lagrange multipliers is used to force these variables to have identical values. An additional unary factor attached to each subproblem variable (not shown) is used to incorporate its Lagrangian term.

3 A Sample Task: Generative Phonology

Before giving the formal details of our DD method, we give a motivating example: a recently proposed graphical model for morphophonology. Cotterell et al. (2015) defined a Bayesian network to describe the generative process of phonological words. Our Figure 1 shows a conversion of their model to a factor graph and explains what the variables and factors mean.

Inference on this graph performs *unsupervised discovery of latent strings*. Given observed surface representations of words (SRs), inference aims to recover the underlying representations (URs) of the words and their shared constituent morphemes. The latter can then be used to predict held-out SRs.

Notice that the 8 edges in the first layer of Figure 1 form a cycle; such cycles make BP inexact. Moreover, the figure shows only a schematic fragment of the graphical model. In the actual experiments, the graphical models have up to 829 variables, and the variables representing morpheme URs are connected to up to 156 factors (because many words share the same affix).

To handle the above challenges without approximation, we want to decompose the original problem into subproblems where each subproblem can be solved efficiently. In particular, we want the subproblems to be free of cycles and high-degree nodes. In our phonology example, each observed word along with its correspondent latent URs forms an ideal subproblem. This decomposition is shown in Figure 2.

While the subproblems can be solved efficiently in isolation, they may share variables, as shown by the dashed lines in Figure 2. DD repeatedly modifies and re-solves the subproblems until they agree on their shared variables.

4 Dual Decomposition

Dual decomposition is a general technique for solving constrained optimization problems. It has been widely used for MAP inference in graphical models (Komodakis et al., 2007; Komodakis and Paragios, 2009; Koo et al., 2010; Martins et al., 2011; Sontag et al., 2011; Rush and Collins, 2014). However, previous work has focused on variables X_i whose values are in $\mathbb{R}$ or a small finite set; we will consider the infinite set Σ^*.

4.1 Review of Dual Decomposition

To apply dual decomposition, we must partition the original problem into a union of K subproblems, each of which can be solved exactly and efficiently (and in parallel). For example, our experiments partition Figure 1 as shown in Figure 2.

Specifically, we partition the factors into K sets $\mathcal{F}^1, \ldots, \mathcal{F}^K$. Each factor $F \in \mathcal{F}$ appears in exactly one of these sets. This lets us rewrite the score (1) as $\sum_k \sum_{F \in \mathcal{F}^k} F(\mathbf{x})$. Instead of simply seeking its maximizer $\mathbf{x}$, we equivalently seek

$$\operatorname*{argmax}_{\mathbf{x}^1,\ldots,\mathbf{x}^K} \sum_{k=1}^{K} \Big(\sum_{F \in \mathcal{F}^k} F(\mathbf{x}^k) \Big) \text{ s.t. } \mathbf{x}^1 = \cdots = \mathbf{x}^K \tag{2}$$

If we dropped the equality constraint, (2) could be solved by *separately* maximizing $\sum_{F \in \mathcal{F}^k} F(\mathbf{x}^k)$ for each k. This "subproblem" is itself a MAP problem which considers only the factors $\mathcal{F}^k$ and the variables $\mathcal{X}^k$ adjacent to them in the original factor graph. The subproblem objective does not depend on the other variables.

We now attempt to enforce the equality constraint indirectly, by adding Lagrange multipliers that encourage agreement among the subproblems. Assume for the moment that the variables in the factor graph are real-valued (each x_i^k is in $\mathbb{R}$). Then consider the **Lagrangian relaxation** of (2),

$$\max_{\mathbf{x}^1,\ldots,\mathbf{x}^K} \sum_{k=1}^{K} \Big(\sum_{F \in \mathcal{F}^k} F(\mathbf{x}^k) + \sum_i \lambda_i^k \cdot x_i^k \Big) \tag{3}$$

This can still be solved by separate maximizations. For *any* choices of $\lambda_i^k \in \mathbb{R}$ having $(\forall i) \sum_k \lambda_i^k = 0$, it upper-bounds the objective of (2). Why? The solution to (2) achieves the same value in (3), yet (3) may do even better by considering solutions that do not satisfy the constraint. Our goal is to find λ_i^k values that tighten this upper bound as much as possible. *If* we can find λ_i^k values so that

the optimum of (3) satisfies the equality constraint, then we have a tight bound and a solution to (2).

To improve the method, recall that subproblem k considers only variables $\mathcal{X}^k$. It is indifferent to the value of X_i if $X_i \notin \mathcal{X}^k$, so we just leave x_i^k undefined in the subproblem's solution. We treat that as automatically satisfying the equality constraint; thus we do not need any Lagrange multiplier λ_i^k to force equality. Our final solution $\mathbf{x}$ ignores undefined values, and sets x_i to the value agreed on by the subproblems that *did* consider X_i.[7]

4.2 Substring Count Features

But what do we do if the variables are strings? The Lagrangian term $\lambda_i^k \cdot x_i^k$ in (3) is now ill-typed. We replace it with $\boldsymbol{\lambda}_i^k \cdot \boldsymbol{\gamma}(x_i^k)$, where $\boldsymbol{\gamma}(\cdot)$ extracts a real-valued **feature vector** from a string, and $\boldsymbol{\lambda}_i^k$ is a vector of Lagrange multipliers.

This corresponds to changing the constraint in (2). Instead of requiring $x_i^1 = \cdots = x_i^K$ for each i, we are now requiring $\boldsymbol{\gamma}(x_i^1) = \cdots = \boldsymbol{\gamma}(x_i^K)$, i.e., these strings must agree *in their features*.

We want each possible string to have a unique feature vector, so that matching features forces the actual strings to match. We follow Paul and Eisner (2012) and use a **substring count feature** for each $w \in \Sigma^*$. In other words, $\boldsymbol{\gamma}(x)$ is an infinitely long vector, which maps each w to the number of times that w appears in x as a substring.[8]

Computing $\boldsymbol{\lambda}_i^k \cdot \boldsymbol{\gamma}(x_i^k)$ in (3) remains possible because in practice, $\boldsymbol{\lambda}_i^k$ will have only finitely many nonzeros. This is so because our feature vector $\boldsymbol{\gamma}(x)$ has only finitely many nonzeros for any string x, and the subgradient algorithm in section 4.3 below always updates $\boldsymbol{\lambda}_i^k$ by adding multiples of such $\boldsymbol{\gamma}(x)$ vectors.

We will use a further trick below to prevent rapid growth of this finite set of nonzeros. Each variable X_i maintains an **active set** of features, $\mathcal{W}_i$. Only these features may have nonzero Lagrange multipliers. While the active set can grow over time, it will be finite at any given step.

Given the Lagrange multipliers, subproblem k of (3) is simply MAP inference on the factor graph consisting of the variables $\mathcal{X}^k$ and factors $\mathcal{F}^k$ *as well as* an extra unary factor G_i^k at each $X_i \in \mathcal{X}^k$:

$$G_i^k(\mathbf{x}^k) \stackrel{\text{def}}{=} \boldsymbol{\lambda}_i^k \cdot \boldsymbol{\gamma}(x_i^k) \tag{4}$$

These unary factors penalize strings according to the Lagrange multipliers. They can be encoded as WFSAs (Allauzen et al., 2003; Cotterell and Eisner, 2015, Appendices B.1–B.5), allowing us to solve the subproblem by max-product BP as usual. The topology of the WFSA for G_i^k depends only on $\mathcal{W}_i$, while its weights come from $\boldsymbol{\lambda}_i^k$.

4.3 Projected Subgradient Method

We aim to adjust the collection $\boldsymbol{\lambda}$ of Lagrange multipliers to *minimize* the upper bound (3). Following Komodakis et al. (2007), we solve this convex **dual problem** using a projected subgradient method. We initialize $\boldsymbol{\lambda} = \mathbf{0}$ and compute (3) by solving the K subproblems. Then we take a step to adjust $\boldsymbol{\lambda}$, and repeat in hopes of eventually satisfying the equality condition.

The projected subgradient step is

$$\boldsymbol{\lambda}_i^k := \boldsymbol{\lambda}_i^k + \eta \cdot \Big(\mu_i - \boldsymbol{\gamma}(x_i^k)\Big) \tag{5}$$

where $\eta > 0$ is the current step size, and μ_i is the mean of $\boldsymbol{\gamma}(x_i^{k'})$ over all subproblems k' that consider X_i. This update modifies (3) to encourage solutions x^k such that $\gamma(x_i^k)$ comes closer to μ_i.

For each i, we update all $\boldsymbol{\lambda}_i^k$ at once to preserve the property that $(\forall i) \sum_k \lambda_i^k = 0$. However, we are only allowed to update components of the $\boldsymbol{\lambda}_i^k$ that correspond to features in the active set $\mathcal{W}_i$. To ensure that we continue to make progress even after we agree on these features, we first expand $\mathcal{W}_i$ by adding the *minimal* strings (if any) on which the x_i^k do not yet all agree. For example, we will add the `abc` feature only when the x_i^k already agree on their counts of its substrings `ab` and `bc`.[9]

Algorithm 1 summarizes the whole method. Table 1 illustrates how one active set $\mathcal{W}_i$ (section 4.3) evolves, in our experiments, as it tries to enforce agreement on a particular string x_i.

4.4 Past Work: Implicit Intersection

Our DD algorithm is an extension of one that Paul and Eisner (2012) developed for the simpler **implicit intersection** problem. Given many WFSAs $F_1, \ldots, F_K$, they were able to find the string x with maximum total score $\sum_{k=1}^K F_k(x)$. (They applied this to solve instances of the NP-hard **Steiner**

[7] Without this optimization, the Lagrangian term $\lambda_i^k \cdot x_i^k$ would have driven x_i^k to match that value anyway.

[8] More precisely, the number of times that w appears in BOS x EOS, where BOS, EOS are distinguished boundary symbols. We allow w to start with BOS and/or end with EOS, which yields prefix and suffix indicator features.

[9] In principle, we should check that they also (still) agree on `a`, `b`, and `c`, but we skip this check. Our active set heuristic is almost identical to that of Paul and Eisner (2012).

Algorithm 1 DD for graphical models over strings

initialize the active set $\mathcal{W}_i$ for each variable $X_i \in \mathcal{X}$
initialize $\boldsymbol{\lambda}_i^k = \mathbf{0}$ for each X_i and each subproblem k
for t = 1 to T **do** ▷ max number of iterations
  for k = 1 to K **do** ▷ solve all primal subproblems
    if any of the $\boldsymbol{\lambda}_i^k$ have changed **then**
      run max-product BP on the acyclic graph defined by variables $\mathcal{X}^k$ and factors $\mathcal{F}^k$ and G_i^k
      extract MAP strings: $\forall i$ with $X_i \in \mathcal{X}^k$, x_i^k is the label of the max-scoring accepting path in the WFSA that represents the belief at X_i
  for each $X_i \in \mathcal{X}$ **do** ▷ improve dual bound
    if the defined strings x_i^k are not all equal **then**
      Expand active feature set $\mathcal{W}_i$ ▷ section 4.3
      Update each $\boldsymbol{\lambda}_i^k$ ▷ equation (5)
      Update each G_i^k from $\Theta_i, \boldsymbol{\lambda}_i^k$ ▷ see (4)
  if none of the X_i required updates **then**
    return any defined x_i^k (all are equal) for each i
return $\{x_i^1, \ldots, x_i^K\}$ for each i ▷ failed to converge

string problem, i.e., finding the string x of minimum total edit distance to a collection of $K \approx 100$ given strings.) The naive solution to this problem would be to find the highest-weighted path in the intersection $F_1 \cap \cdots \cap F_K$. Unfortunately, the intersection of WFSAs takes the Cartesian product of their state sets. Thus materializing this intersection would have taken time exponential in K.

To put this another way, inference is NP-hard even on a "trivial" factor graph: a *single* variable X_1 attached to K factors. Recall from section 2.3 that BP would solve this via the expensive intersection above. Paul and Eisner (2012) instead applied DD with one subproblem per factor. We generalize their method to handle arbitrary factor graphs, with multiple latent variables and cycles.

4.5 Block Coordinate Update

We also explored a possible speedup for our algorithm. We used a **block coordinate update** variant of the algorithm when performing inference on the phonology problem and observed an empirical speedup. Block coordinate updates are widely used in Lagrangian relaxation and have also been explored specifically for dual decomposition.

In general, block algorithms minimize the objective by holding some variables fixed while updating others. Sontag et al. (2011) proposed a sophisticated block method called MPLP that considers all values of variable X_i instead of the ones obtained from the best assignments for the subproblems. However, it is not clear how to apply their technique to string-valued variables. Instead, the algorithm we propose here is much simpler—it divides the *primal* variables into groups and updates each group's associated *dual* variables in turn, using a single subgradient step (5). Note that this way of partitioning the dual variables has the nice property that we can still use the projected subgradient update we gave in (5) and preserve the property that $(\forall i) \sum_k \lambda_i^k = 0$.

In the graphical model for generative phonology, there are two types of underlying morphemes in the first layer: word stems and word affixes. Our block coordinate update algorithm thus alternates between subgradient updates to the dual variables for the stems and the dual variables for the affixes. Note that when performing block coordinate update on the dual variables, the primal variables are *not* held constant, but rather are chosen by optimizing the corresponding subproblem.

Iter#	x_i^1	x_i^2	x_i^3	x_i^4	$\Delta\mathcal{W}_i$
1	ϵ	ϵ	ϵ	ϵ	∅
3	g	g	g	g	∅
4	gris	griz	griz	griz	{s, z, is, iz, s$ z$ }
5	gris	grizo	griz	griz	{o, zo, o$ }
14	griz	grizo	griz	griz	∅
17	griz	griz	griz	griz	∅
18	griz	griz	grize	griz	{ e, ze, e$ }
19	gris	griz	griz	griz	∅
31	griz	griz	griz	griz	∅

Table 1: One variable's active set as DD runs. This variable is the unobserved stem morpheme shared by the Catalan words gris, grizos, grize, grizes. The second column shows the current set of solutions from the 4 subproblems having copies of this variable. The third column shows the new substrings that are then added to the active set, to try to enforce agreement via their Lagrange multipliers. The table does not show iterations in which these columns have not changed. However, those iterations still update the Lagrange multipliers to more strongly encourage agreement (if needed). Although agreement is achieved at iterations 1, 3, and 17, it is then disrupted—the subproblems' solutions change because of Lagrange-multiplier pressures on their *other* variables (suffixes that do *not* agree yet). At iteration 31, the variable returns to agreement on griz, and never changes again.

5 Experimental Setup

5.1 Datasets

We compare DD to belief propagation, using the graphical model for generative phonology discussed in section 3. Inference in this model aims to reconstruct underlying morphemes. Since our focus is inference, we will evaluate these reconstructions directly (whereas Cotterell et al. (2015) evaluated their ability to predict novel surface forms using the reconstructions).

Our factor graphs have a similar topology to the pedagogical fragment shown in Figure 1. How-

ever, they are actually derived from datasets constructed by Cotterell et al. (2015), which are available with full descriptions at `http://hubal.cs.jhu.edu/tacl2015/`. Briefly:

EXERCISE Small datasets of Catalan, English, Maori, and Tangale, drawn from phonology textbooks. Each dataset contains 55 to 106 surface words, formed from a collection of 16 to 55 morphemes.

CELEX Larger datasets of German, English, and Dutch, drawn from the CELEX database (Baayen et al., 1995). Each dataset contains 1000 surface words, formed from 341 to 381 underlying morphemes.

5.2 Evaluation Scheme

We compared three types of inference:

DD Use DD to perform *exact MAP inference*.

SP Perform *approximate marginal inference* by sum-product loopy BP with pruning (Cotterell et al., 2015).

MP Perform *approximate MAP inference* by max-product loopy BP with pruning. DD and SP improve this baseline in different ways.

DD predicts a string value for each variable. For SP and MP, we deem the prediction at a variable to be the string that is scored most highly by the belief at that variable.

We report the fraction of predicted morpheme URs that exactly match the gold-standard URs proposed by a human (Cotterell et al., 2015). We also compare these predicted URs to one another, to see how well the methods agree.

5.3 Parameterization

The model of Cotterell et al. (2015) has two factor types whose parameters must be chosen.[10] The first is a unary factor M_ϕ. Each underlying-morpheme variable (layer 1 of Figure 1) is connected to a copy of M_ϕ, which gives the prior distribution over its values. The second is a binary factor S_θ. For each surface word (layer 3), a copy of S_θ gives its conditional distribution given the corresponding underlying word (layer 2). M_ϕ and S_θ respectively model the *lexicon* and the *phonology* of the specific language; both are encoded as WFSMs.

M_ϕ is a 0-gram generative model: at each step it emits a character chosen uniformly from the alphabet Σ with probability ϕ, or halts with probability $1-\phi$. It favors shorter strings in general, but ϕ determines how weak this preference is.

S_θ is a sequential edit model that produces a word's SR by stochastically copying, inserting, substituting, and deleting the phonemes of its UR. We explore two ways of parameterizing it.

Model 1 is a simple model in which θ is a scalar, specifying the probability of copying the next character of the underlying word as it is transduced to the surface word. The remaining probability mass $1-\theta$ is apportioned equally among insertion, substitution and deletion operations.[11] This models phonology as "noisy concatenation"—the minimum necessary to account for the fact that surface words cannot quite be obtained as simple concatenations of their shared underlying morphemes.

Model 2 is a replication of the much more complicated parametric model of Cotterell et al. (2015), which can handle linguistic phonology. Here the factor S_θ is a *contextual* edit FST (Cotterell et al., 2014). The probabilities of competing edits in a given context are determined by a log-linear model with weight vector θ and features that are meant to pick up on phonological phenomena.

5.4 Training

When evaluating an inference method from section 5.2, we use the same inference method both for prediction and within training.

We train Model 1 by grid search. Specifically, we choose $\phi \in [0.65, 1)$ and $\theta \in [0.25, 1)$ such that the predicted forms maximize the joint score (1) (always using the $(\max, +)$ semiring).

For Model 2, we compared two methods for training the ϕ and θ parameters (θ is a vector):

Model 2S Supervised training, which *observes* the "true" (hand-constructed) values of the URs. This idealized setting uses the best possible parameters (trained on the test data).

Model 2E Expectation maximization (EM), whose E step *imputes* the unobserved URs.

EM's E step calls for *exact marginal inference*, which is intractable for our model. So we substitute the same inference method that we are test-

[10] The model also has a three-way factor, connecting layers 1 and 2 of Figure 1. This represents deterministic concatenation (appropriate for these languages) and has no parameters.

[11] That is, probability mass of $(1-\theta)/3$ is divided equally among the $|\Sigma|$ possible insertions; another $(1-\theta)/3$ is divided equally among the $|\Sigma|-1$ possible substitutions; and the final $(1-\theta)/3$ is allocated to deletion.

ing. This gives us three approximations to EM, based on DD, SP and MP. Note that DD specifically gives the Viterbi approximation to EM—which sometimes gets better results than true EM (Spitkovsky et al., 2010). For MP (but not SP), we extract only the 1-best predictions for the E step, since we study MP as an approximation to DD.

As initialization, our first E step uses the trained version of Model 1 for the same inference method.

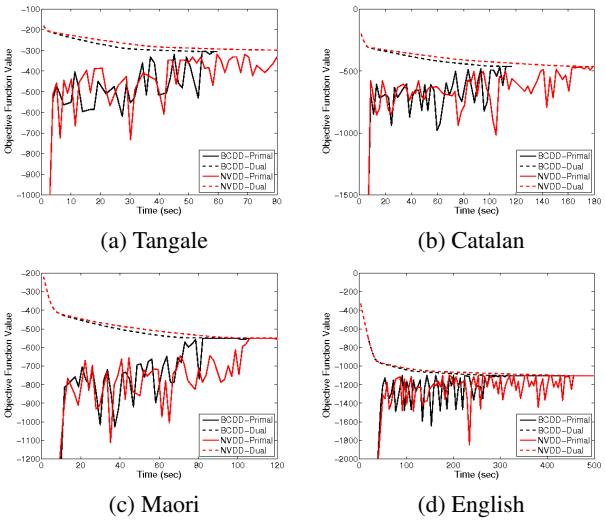
5.5 Inference Details

We run SP and MP for 20 iterations (usually the predictions converge within 10 iterations). We run DD to convergence (usually < 600 iterations). DD iterations are much faster since each variable considers d strings, not d *distributions* over strings. Hence DD does not intersect distributions, and many parts of the graph settle down early because discrete values can converge in finite time.[12]

We follow Paul and Eisner (2012, section 5.1) fairly closely. In particular: Our stepsize in (5) is $\eta = \alpha/(t + 500)$, where t is the iteration number; $\alpha = 1$ for Model 2S and $\alpha = 10$ otherwise. We proactively include all 1-gram and 2-gram substring features in the active sets $\mathcal{W}_i$ at initialization, rather than adding them only as needed. At iterations 200, 400, and 600, we proactively add all 3-, 4-, and 5-gram features (respectively) on which the counts still disagree; this accelerates convergence on the few variables that have not already converged. We handle negative-weight cycles as Paul and Eisner do. If we had ever failed to converge within 2000 iterations, we would have used their heuristic to extract a prediction anyway.

Model 1 suffers from a symmetry-breaking problem. Many edits have identical probability, and when we run inference, many assignments will tie for highest scoring configuration. This can prevent DD from converging and makes performance hard to measure. To break these ties, we add "jitter" separately to each copy of M_ϕ in Figure 1. Specifically, if F_i is the unary factor attached to X_i, we expand our 0-gram model $F_i(x) = \log((p/|\Sigma|)^{|x|} \cdot (1-p))$ to become $F_i(x) = \log(\prod_{c\in\Sigma} p_{c,i}^{|x|_c} \cdot (1-p))$, where $|x|_c$ denotes the count of character c in string x, and $p_{c,i} \propto (p/|\Sigma|) \cdot \exp \varepsilon_{c,i}$ where $\varepsilon_{c,i} \sim N(0, 0.01)$ and we preserve $\sum_{c\in\Sigma} p_{c,i} = p$.

[12] A variable need not update $\boldsymbol{\lambda}$ if its strings agree; a subproblem is not re-solved if none of its variables updated $\boldsymbol{\lambda}$.



(a) Tangale (b) Catalan

(c) Maori (d) English

Figure 3: The primal-dual curve of NVDD v.s. BCDD on 4 EXERCISE languages. BCDD always converges faster.

6 Experimental Results

6.1 Convergence and Speed of DD

As linguists know, reconstructing an underlying stem or suffix can be difficult. We may face insufficient evidence or linguistic irregularity—or regularity that goes unrecognized because the phonological model is impoverished (Model 1) or poorly trained (early EM iterations on Model 2). DD may then require extensive negotiation to resolve disagreements among subproblems. Furthermore, DD must renegotiate as conditions change elsewhere in the factor graph (Table 1).

DD converged in all of our experiments. Note that DD (section 4.3) has converged when all the equality constraints in (2) are satisfied. In this case, we have found the true MAP configuration.

In section 4.5, we discussed a block coordinate update variation (BCDD) of our DD algorithm. Figure 3 shows the convergence behavior of BCDD against the naive projected subgradient algorithm (NVDD) on the four EXERCISE languages under Model 1. The dual objective (3) always upper-bounds the primal score (i.e., the score (1) of an assignment derived heuristically from the current subproblem solutions). The dual decreases as the algorithm progresses. When the two objectives meet, we have found an optimal solution to the primal problem. We can see in Figure 3 that our DD algorithm converges quickly on the four EXERCISE languages and BCDD converges consistently faster than NVDD. We use BCDD in the remaining experiments.

When DD runs fast, it is competitive with the

	DD	SP	MP	Gold
DD		92.74%	90.55%	**96.92%**
SP			95.22%	94.63%
MP				90.63%

(a) The 4 EXERCISE languages under Model 1

	DD	SP	MP	Gold
DD		88.05%	85.19%	**89.66%**
SP			92.64%	85.71%
MP				83.46%

(b) The 3 CELEX languages under Model 1

	DD	SP	MP	Gold
DD		96.53%	100%	**98.67%**
SP			96.53%	96.05%
MP				**98.67%**

(c) The 3 CELEX languages under Model 2S (EXERCISE dataset gives 100% everywhere)

	DD	SP	MP	Gold
DD		92.43%	89.39%	**98.18%**
SP			96.73%	95.42%
MP				90.74%

(d) The 4 EXERCISE languages under Model 2E

Table 2: Pairwise agreement (on morpheme URs) of DD, SP, MP and the gold standard, for each group of inference problems. Boldface is highest accuracy (agreement with gold).

other methods. It is typically faster on the EXERCISE data, and a few times slower on the CELEX data. But we stop the other methods after 20 iterations, whereas *DD runs until it gets an exact answer*. We find that this runtime is unpredictable and sometimes quite long. In the grid search for training Model 1, we observed that changes in the parameters (ϕ, θ) could cause the runtime of DD inference to vary by 2 orders of magnitude. Similarly, on the CELEX data, the runtime on Model 1 (over 10 different $N = 600$ subsets of English) varied from about 1 hour to nearly 2 days.[13]

6.2 Comparison of Inference

For each language, we constructed several different unsupervised prediction problems. In each problem, we observe some size-N subset of the words in our dataset, and we attempt to predict the URs of the morphemes in those words. For each CELEX language, we took $N = 600$, and used three of the size-N training sets from (Cotterell et al., 2015). For each EXERCISE language, we took N to be one less than the dataset size, and used all $N + 1$ subsets of size N, again similar to (Cotterell et al., 2015). We report the unweighted macro-average of all these accuracy numbers.

[13]Note that our implementation is not optimized; e.g., it uses Python (not Cython).

We compare DD, SP, and MP inference on each language under different settings. Table 2 shows aggregate results, as an unweighted average over multiple languages and training sets. We present various additional results at `http://cs.jhu.edu/~npeng/emnlp2015/`, including a per-language breakdown of the results, runtime numbers, and significance tests.

The results for Model 1 are shown in Tables 2a and 2b. As we can see, in both datasets, dual decomposition performed the best at recovering the URs, while MP performed the worst. Both DD and MP are doing MAP inference, so the differences reflect the search error in MP. Interestingly, DD agrees more with SP than with MP, even though SP uses marginal inference.

Although the aggregate results on the EXERCISE dataset show a large improvement of DD over both of the BP algorithms, the gain all comes from the English language. SP actually does better than DD on Catalan and Maori, and MP also gets better results than DD on Maori, tying with SP.

For Model 2S, all inference methods achieved 100% accuracy on the EXERCISE dataset, so we do not show a table. The results on the CELEX dataset are shown in Table 2c. Here both DD and MP performed equally well, and outperformed BP—a result like (Spitkovsky et al., 2010). This trend is consistent over all three languages: DD and MP always achieve similar results and both outperform SP. Of course, one advantage of DD in the setting is that it actually finds the true MAP prediction of the model; the errors are known to be due to the model, not the search procedure.

For Model 2E, we show results on the EXERCISE dataset in Table 2d. Here the results resemble the pattern of Model 1.

7 Conclusion and Future Work

We presented a general dual decomposition algorithm for MAP inference on graphical models over strings, and applied it to an unsupervised learning task in phonology. The experiments show that our DD algorithm converges and gets better results than both max-product and sum-product BP.

Techniques should be explored to speed up the DD method. Adapting the MPLP algorithm (Sontag et al., 2011) to the string-valued case would be a nontrivial extension. We could also explore other serial update schemes, which generally speed up message-passing algorithms over parallel update.

References

Cyril Allauzen, Mehryar Mohri, and Brian Roark. 2003. Generalized algorithms for constructing statistical language models. In *Proceedings of ACL*, pages 40–47.

R. Harald Baayen, Richard Piepenbrock, and Leon Gulikers. 1995. The CELEX lexical database on CD-ROM.

Alexandre Bouchard-Côté, Percy Liang, Thomas L Griffiths, and Dan Klein. 2007. A probabilistic approach to diachronic phonology. In *Proceedings of EMNLP-CoNLL*, pages 887–896.

Alexandre Bouchard-Côté, Percy Liang, Thomas Griffiths, and Dan Klein. 2008. A probabilistic approach to language change. In *Proceedings of NIPS*.

Ryan Cotterell and Jason Eisner. 2015. Penalized expectation propagation for graphical models over strings. In *Proceedings of NAACL-HLT*, pages 932–942, Denver, June. Supplementary material (11 pages) also available.

Ryan Cotterell, Nanyun Peng, and Jason Eisner. 2014. Stochastic contextual edit distance and probabilistic FSTs. In *Proceedings of ACL*, Baltimore, June. 6 pages.

Ryan Cotterell, Nanyun Peng, and Jason Eisner. 2015. Modeling word forms using latent underlying morphs and phonology. *Transactions of the Association for Computational Linguistics*, 3:433–447.

Markus Dreyer and Jason Eisner. 2009. Graphical models over multiple strings. In *Proceedings of EMNLP*, pages 101–110, Singapore, August.

Markus Dreyer and Jason Eisner. 2011. Discovering morphological paradigms from plain text using a Dirichlet process mixture model. In *Proceedings of EMNLP*, pages 616–627, Edinburgh, July.

Thomas Finley and Thorsten Joachims. 2007. Parameter learning for loopy markov random fields with structural support vector machines. In *ICML Workshop on Constrained Optimization and Structured Output Spaces*.

David Hall and Dan Klein. 2010. Finding cognate groups using phylogenies. In *Proceedings of ACL*.

David Hall and Dan Klein. 2011. Large-scale cognate recovery. In *Proceedings of EMNLP*.

André Kempe, Jean-Marc Champarnaud, and Jason Eisner. 2004. A note on join and auto-intersection of n-ary rational relations. In Loek Cleophas and Bruce Watson, editors, *Proceedings of the Eindhoven FASTAR Days (Computer Science Technical Report 04-40)*, pages 64–78. Department of Mathematics and Computer Science, Technische Universiteit Eindhoven, Netherlands, December.

Kevin Knight and Jonathan Graehl. 1998. Machine transliteration. *Computational Linguistics*, 24(4).

Nikos Komodakis and Nikos Paragios. 2009. Beyond pairwise energies: Efficient optimization for higher-order MRFs. In *Proceedings of CVPR*, pages 2985–2992. IEEE.

Nikos Komodakis, Nikos Paragios, and Georgios Tziritas. 2007. MRF optimization via dual decomposition: Message-passing revisited. In *Proceedings of ICCV*, pages 1–8. IEEE.

Terry Koo, Alexander M. Rush, Michael Collins, Tommi Jaakkola, and David Sontag. 2010. Dual decomposition for parsing with non-projective head automata. In *Proceedings of EMNLP*, pages 1288–1298.

F. R. Kschischang, B. J. Frey, and H. A. Loeliger. 2001. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47(2):498–519, February.

André Martins, Mário Figueiredo, Pedro Aguiar, Eric P. Xing, and Noah A. Smith. 2011. An augmented lagrangian approach to constrained map inference. In *Proceedings of ICML*, pages 169–176.

Thomas P. Minka. 2001. Expectation propagation for approximate Bayesian inference. In *Proceedings of UAI*, pages 362–369.

Mehryar Mohri, Fernando Pereira, and Michael Riley. 2002. Weighted finite-state transducers in speech recognition. *Computer Speech & Language*, 16(1):69–88.

Kevin P. Murphy, Yair Weiss, and Michael I. Jordan. 1999. Loopy belief propagation for approximate inference: An empirical study. In *Proceedings of UAI*, pages 467–475.

Michael J. Paul and Jason Eisner. 2012. Implicitly intersecting weighted automata using dual decomposition. In *Proceedings of NAACL*, pages 232–242.

Fernando C. N. Pereira and Michael Riley. 1997. Speech recognition by composition of weighted finite automata. In Emmanuel Roche and Yves Schabes, editors, *Finite-State Language Processing*. MIT Press, Cambridge, MA.

Alexander M. Rush and Michael Collins. 2014. A tutorial on dual decomposition and Lagrangian relaxation for inference in natural language processing. Technical report available from `arXiv.org` as `arXiv:1405.5208`.

David Sontag, Amir Globerson, and Tommi Jaakkola. 2011. Introduction to dual decomposition for inference. *Optimization for Machine Learning*, 1:219–254.

Valentin I. Spitkovsky, Hiyan Alshawi, Daniel Jurafsky, and Christopher D. Manning. 2010. Viterbi training improves unsupervised dependency parsing. In *Proceedings of CoNLL*, page 917, Uppsala, Sweden, July.

I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. 2005. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6:1453–1484, September.

Discourse parsing for multi-party chat dialogues

Stergos Afantenos Eric Kow Nicholas Asher Jérémy Perret
IRIT, Université Toulouse & CNRS, Univ. Paul Sabatier,
118 Route de Narbonne, 31062 Toulouse
{firstname.lastname@irit.fr}

Abstract

In this paper we present the first ever, to the best of our knowledge, discourse parser for multi-party chat dialogues. Discourse in multi-party dialogues dramatically differs from monologues since threaded conversations are commonplace rendering prediction of the discourse structure compelling. Moreover, the fact that our data come from chats renders the use of syntactic and lexical information useless since people take great liberties in expressing themselves lexically and syntactically. We use the dependency parsing paradigm as has been done in the past (Muller et al., 2012; Li et al., 2014). We learn local probability distributions and then use MST for decoding. We achieve 0.680 F_1 on unlabelled structures and 0.516 F_1 on fully labeled structures which is better than many state of the art systems for monologues, despite the inherent difficulties that multi-party chat dialogues have.

1 Introduction

Discourse parsing is a difficult, multifaceted problem involving the understanding and modeling of various semantic and pragmatic phenomena as well as understanding the structural properties that a discourse graph can have. Unsurprisingly, most extant theories and computational approaches postulate an extremely simplified version of discourse structure. One of the most widely cited theories, Rhetorical Structure Theory (RST) (Mann and Thompson, 1987; Mann and Thompson, 1988; Taboada and Mann, 2006), requires that only *adjacent* discourse units be connected together with a discourse relation. Another widely cited approach, the Penn Discourse Treebank (PDTB) (Prasad et al., 2008), focuses on decisions about the discourse connectives that label the attachment of potentially arbitrary text spans but does not make any claims as to what the overall discourse structure of the resulting annotation looks like. Further, all computational work on the PDTB takes the attachments as given in discourse parsing tasks. In both cases, the attachment problem, finding which discourse units are attached to which, is vastly simplified, though this has enabled researchers to explore various approaches for discourse parsing (Marcu, 2000; Sagae, 2009; Hernault et al., 2010; Joty et al., 2012).

Our paper's main contribution is to provide a discourse parsing model for multi-party chat dialogue (*i.e.* typed online dialogue), trained on a large corpus we have developed annotated with full discourse structures. We study attachment problem in detail for this genre, without using the simplifying hypotheses mentioned above that we know to be inadequate. In the following section, we describe the *Settlers of Catan* game and our corpus in more detail and discuss some problematic structures for discourse parsing from our corpus. We motivate our choice of a particular discourse theory, the Segmented Discourse Representation Theory (SDRT), as the underlying theoretical model for our annotations. In section 4 we present our parsing approach, which consists of building a local probability distribution model which serves as input to a series of decoder mechanisms. We present and discuss the results we obtain in section 5, while related work and conclusions are presented in sections 6 and 7 respectively.

2 What's so special about multi party dialogue?

Multi-party dialogue or multi-party chat involves multiple interlocutors that may address one or more interlocutors during their turn. For example,

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 928–937,
Lisbon, Portugal, 17-21 September 2015.

a person might pose a question that concerns all the participants; and once everybody has replied, that same person might reply to all of them with a single comment (e.g. thanking them) or with a single acknowledgment. Figure 1 provides an example from our corpus. In turn 234, gotwood4sheep asks a question and makes an underspecified offer to all the players. He then gets back negative responses to his question from inca, CheshireCatGrin and dmm; and then he broadcasts in 239 an acknowledgment of all the negative responses. That is, we have 235, 236 & 238 all attached to 234 as answers to the question in 234; and we have 239 that is attached to 235, 236 & 238 as an acknowledgment of the contents of those turns. A graphical representation is shown on the right of the same picture.

The presence of such structures makes a powerful case that the general framework guiding the annotation of multi-party dialogues should take *non-tree-like* graphs as the basic form of discourse structures. This will require then rethinking the task of discourse parsing when attempting to learn such structures. In particular, the following questions present themselves: 1) how many non-tree-like structures are there? 2) what are the constraints on discourse graphs, if they are not trees? 3) how far can traditional tree-based decoding mechanisms get us in dealing with such data?

Another complicated phenomenon in multi-party chat dialogues is the presence of crossing dependencies. Many theories of discourse structure like RST, given that they allow attachment only of adjacent spans will perforce not allow structures with crossing dependencies. Also theories that postulate a simple *right frontier constraint*, according to which only elements on the right frontier of a discourse structure (whether graph or tree) will in general not generate structures with crossing dependencies. However, crossing dependencies are commonplace in multi-party chat. Several subgroups of interlocutors can and do momentarily form and carry on a discussion amongst themselves, forming thus multiple concurrent discussion threads. Since, though, what is being written is publicly available to all involved parties, it can be the case that participants of one thread might reply or comment to something said to another thread. Figure 2 contains an example from our corpus.

There are at least three threads in this excerpt, and we have given them different fonts to aid the reader. The intuitive attachments in this excerpt involve the following crossing dependencies: (165, 168), (167,170), (176, 178), (177, 179), (175, 181), (177,182), and (180,183). We note also the lack of standard discourse markers such as those found in the PDTB or RST manuals, "personalized" orthography, the lack of elaborate syntactic structure and the frequent presence of sentence fragments, all of which means we cannot rely on sentential syntax to aid with discourse parsing (syntax is very useful in monologue discourse parsing, as witnessed by the dramatically higher scores for intra-sentential discourse parsing (Joty et al., 2015)). Multi-party dialogue presents a discourse parsing problem free of syntactic crutches.

The phenomena we have just described are just some of the complications that appear in the discourse representation of multi-party dialogues, unfortunately rendering discourse theories based on attaching only adjacent units unsuitable for the representation of multi-party dialogues. In order to be able to capture the discourse phenomena present in our chat corpus, we have decided to use the Segmented Discourse Representation Theory (SDRT) (Asher and Lascarides, 2003). This theory not only allows long distance attachments, which (Ginzburg, 2012) finds attested in multilogue, but also has semantics capable of dealing with fragments or non sentential utterances (Schlangen, 2003), which are frequent in our corpus. Also, it can model non-tree like structures, like that shown in Figure 1, which account for at least 9% of the links in our corpus. Such structures make theories that model discourse structures with rooted trees, like *Rhetorical Structure Theory* (RST) (Mann and Thompson, 1987) or simple dialogue models where attachments are always made to *Last*—cf. (Schegloff, 2007; Poesio and Traum, 1997)—unsuitable.

A final feature of discourse annotations that multi-party dialogue and monologue share is the presence of complex discourse units or CDUs. CDUs are in fact subgraphs of the discourse graph that have a rhetorical function or bear some discourse relation to another constituent. Examples are easy to come by. Consider the following example:

(1) gw: Do you have a sheep?
th: I do, if you give me an ore

234	18:55:02:745	gotwood4sheep	anyone got wheat for a sheep?
235	18:55:10:047	inca	sorry, not me
236	18:55:18:787	CheshireCatGrin	nope. you seem to have lots of sheep!
237	18:55:23:428	gotwood4sheep	yup baaa
238	18:55:32:308	dmm	i think i'd rather hang on to my wheat i'm afraid
239	18:55:47:845	gotwood4sheep	kk I'll take my chances then...

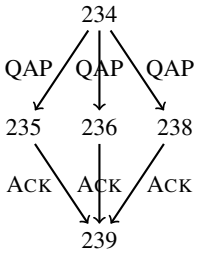


Figure 1: Dialogue excerpt showing the need for general graphs instead of trees.

165	lj	**anyone want sheep for clay?**
166	gw	**got none, sorry :(**
167	gw	so how do people know about the league?
168	wm	**no**
170	lj	i did the trials
174	tk	i know about it from my gf
175	gw	[yeah me too,]$_a$
		[*are you an Informatics student then, lj?*]$_b$
176	tk	did not do the trials
177	wm	**has anyone got wood for me?**
178	gw	[I did them]$_a$ [because a friend did]$_b$
179	gw	**lol william, you cad**
180	gw	**afraid not :(**
181	lj	*no, I'm about to start math*
182	tk	**sry no**
183	gw	**my single wood is precious**
184	wm	**what's a cad?**

Figure 2: Example of interleaved threads

th: or a wood.

Clearly, th's two turns combine to form a CDU that is then related by a conditional discourse relation to *I do*. That is *you give me an ore* and *or a wood* form together the antecedent to the conditional that he expresses. In order to reflect this semantic dependency, SDRT creates collections of Elementary Discourse Units (EDUs) forming a coherent discourse unit (called Complex Discourse Unit, CDU) and link it to any other discourse unit. The end result of this process is the creation of a hypergraph or, equivalently, a graph with two types of edges. Thus, our general conception of a discourse structure for a discourse $D = \{e_1, \ldots, e_n\}$, where e_i are the EDUs of D, is a tuple (V, E_1, E_2, ℓ), where V is a set of nodes or discourse units including $\{e_1, \ldots, e_n\}$, $E_1 \subseteq V \times V$ a set of edges representing discourse relations and $E_2 \subseteq V \times V$ a set of edges that represents parthood in the sense that if $(x, y) \in E_2$, then x is a discourse unit that is an element of the CDU y. ℓ: $E_1 \rightarrow$ *Relations* is a function that assigns each arc a discourse relation type. Our corpus contains many instances of CDUs, some of which are quite large, encompassing an entire question answering session like that seen in Figure 1.

3 The STAC corpus

The corpus that we use was collected from an online version of the game *The Settlers of Catan*. *Settlers* is a multi-party, win-lose game in which players use resources such as wood and sheep to build roads and settlements. In the standard online version, players interact solely through the game interface, making trades and building roads, etc., without saying a word. In our online version, players were asked to discuss and negotiate their trades via a chat interface before finalizing them non-linguistically via the game interface. As a result, players frequently chatted not only to negotiate trades, but to discuss numerous topics, some unrelated to the task at hand.

The *Settlers* corpus is ideal for studying multilogue. First, while the chats maintain the advantage of written text (no need for transcription), they approximate spoken communication. We have to deal with many sentence fragments, non-standard orthography and sometimes lack of syntax. Second, they manifest phenomena particular to multilogue, such as multiple conversation threads and non-tree-like structures.

The corpus consists of 59 games out of which 36 games have so far been annotated for discourse structure in the style of SDRT. Each game consists of several dialogues representing a single turn of the game. Each dialogue is treated as a separate document. About 10% of our corpus was held out for evaluation purposes while the rest was kept for training. Detailed statistics on the number of dialogues, EDUs and relations contained in each subcorpus can be found in table 1.

The dialogues in our corpus have an average size of 10 EDUs with 8 speaker turns, though the longest has 156 EDUs and 119 turns. The vast majority of our discourse connections thus lie between turns. All dialogues also have a dialogue act

	Total	Training	Testing
Dialogues	1091	968	123
EDUs	10677	9545	1132
Relations	11348	10158	1190

Table 1: Dataset overview

style annotation in which each EDU is assigned a particular type (it can be an offer or counter-offer, an acceptance or refusal, or other) (Sidner, 1994a; Sidner, 1994b). The dialogue act annotations have been used to train an automatic classifier for EDUs (Cadilhac et al., 2013). This large annotation effort was carried out by 4 annotators who had no special knowledge of linguistics, but who received training over 22 negotiation dialogues with 560 turns. Because annotating full discourse structures is a very complex task (using an exact match criterion of success, the inter annotator agreement score was a Kappa of 0.72 attachment on structures, 0.58 on labelling (Afantenos et al., 2012a)), experts made several passes over the annotations from the naive annotators, improving the data and debugging it.

The discourse graphs in our development corpus exhibited several interesting properties. First of all they are DAGs with a unique root, one unit that has no incoming edges. Secondly, the graphs are weakly connected in almost all cases: i.e., every discourse unit in it is connected to some other discourse unit. Thirdly, our graphs are *reactive* in the sense that speakers' contributions are reactions and attach anaphorically to prior contributions of other speakers. This means that edges between the contributions of different speakers are always oriented in one direction. This is a general feature of dialogue, as we explain in the next section.

4 Model

4.1 Dependency Structures

For a given discourse graph for SDRT of the form (V, E_1, E_2, ℓ), we have as yet no general and reliable method to calculate edges in E_2; and no such method has been presented in the literature. In order to perform constrained decoding over local probability distributions, we have opted for a strategy first presented in Muller et al. (2012) for SDRT. The strategy involves transforming hypergraphs into dependency graphs. We transform our full graphs (V, E_1, E_2, ℓ) into dependency structures (V', E_1, ℓ), with $V' \subset V$ the set of EDUs in V by replacing any attachment to a CDU with an attachment to the CDU's head—the textually first EDU within the CDU which has no incoming links. Our transformation in effect sets E_2 in our general definition of a graph to $\emptyset$. In the case that we have a discourse relation between two EDUs, this relation is kept intact since it already represents a dependency arc. In case a discourse relation has one or two CDUs as arguments, the CDUs need to be replaced with their *recursive head*. In order to calculate the recursive head we identify all the DUs with no incoming links; if they are CDUs we recursively apply the algorithm until we get an EDU. If there is more than one EDU with no incoming links we pick the leftmost, i.e. the one firstly introduced in the text.

Hirao et al. (2013) and Li et al. (2014) later followed a similar strategy for the creation of dependency structures for RST. Every single nucleus-satellite relation was transformed into a dependency relation with the governor being the EDU representing the nucleus and the dependent being the satellite. For relations between non-EDU higher spans, the recursive head was used. It is unclear how Li et al. (2014) deal with binary multinucleus relations like CONTRAST for example; it is not clear how to calculate the recursive head of the span.[1] In such cases an arbitrary decision—like always taking as the nucleus the leftmost or the rightmost span—has to be taken. In the SDRT annotations, however, every edge in the graph is already directed and so such arbitrary decisions can be avoided.

Ideally, what one then wants is to learn a function

$$h : \mathcal{X}_{E^n} \mapsto \mathcal{Y}_{\mathcal{G}}$$

where $\mathcal{X}_{E^n}$ is the domain of instances representing a collection of EDUs for each dialogue and $\mathcal{Y}_{\mathcal{G}}$ is the set of all possible SDRT graphs. However, given the complexity of this task and the fact that it would require an amount of training data that we currently lack in the community, we aim at the more modest goal of learning a function

$$h : \mathcal{X}_{E^2} \mapsto \mathcal{Y}_R$$

where the domain of instances $\mathcal{X}_{E^2}$ represents features for a pair of EDUs and $\mathcal{Y}_R$ represents the

[1] Although Li et al. (2014) do explain how to treat n-ary multinuclear relations, following others (Hernault et al., 2010, for example).

set of SDRT relations. The upshot of this is that we are building a local sort of model that learns relations between individual EDUs with a certain probability but that it does not learn a local or even global structure.

One of the drawbacks of this approach, however, is that it does not guarantee an object that is well formed. Learning a probability distribution over EDUs and then choosing the most probable relation or attachment for each pair of EDUs potentially leads to structures that contain cycles. To avoid this, we can't blindly choose the most probable relation or attachment decision for each pair of EDUs. Instead, we should use this probability distribution as an input to a decoding mechanism.

4.2 Local probability distributions

We used a regularized maximum entropy (shortened as MaxEnt) model (Berger et al., 1996). In MaxEnt, we estimate the parameters of an exponential model of the following form:

$$P(r|p) = \frac{1}{Z(c)} \exp\left(\sum_{i=1}^{m} w_i f_i(p, r)\right)$$

where p represents the current pair of EDUs and r the learnt label (i.e. the type of relation, or a binary attachment value between the two EDUs). Each pair of EDUs p is encoded as a vector of m indicator features f_i (see table 2 for more details). There is one weight/parameter w_i for each feature f_i that predicts its classification behavior. Finally, $Z(c)$ is a normalization factor over the different class labels, which guarantees that the model outputs probabilities. In MaxEnt, the values for the different parameters $\hat{w}$ are obtained by maximizing the log-likelihood of the training data T with respect to the model (Berger et al., 1996):

$$\hat{w} = \operatorname*{argmax}_{w} \sum_{i}^{T} \log P(r^{(i)}|p^{(i)})$$

4.3 The turn constraint

Given our observations about the structure of dialogues in our corpus, we hypothesize that a dialogue is fundamentally sequential: first one person talks and then others react to them or ignore them, but the discourse links that do occur between speaker turns are *reactive*. In other words, a turn can't be anaphorically and rhetorically dependent on a turn that comes after it. Thus, the nature of dialogue imposes an essential and important constraint on the attachment process that is not present for monologue or single-authored text, where an EDU may be dependent upon any EDU, later in the ordering or not: in dialogue there are no "backwards" rhetorical links such that an EDU in turn n by speaker a is rhetorically and anaphorically dependent upon an EDU in turn $n + m$ of speaker b with $a \neq b$. We call this the *Turn Constraint*. Within a turn, however, just as in monologue (as is evident from a study of most styles of discourse annotations of text), backwards links are allowed.

Given this observation, we decided to split our local model into two different ones. The first one concerns the learning of a model for intra-turn utterances,[2] while the second models inter-turn utterances. The intra-turn model considers as input during learning all pairs of EDUs (i, j) with $i \neq j$. The inter-turn model on the other hand does not contain any backward links during learning. In other words it takes as input all pairs of EDUs (i, j) with $i < j$. We apply the turn constraint not only during learning of the local models, but also during decoding. This practice is also followed—at the sentence level—for monologues (Wellner and Pustejovsky, 2007; Joty et al., 2012; Joty et al., 2013), though our turn constraint, we believe, is firmly supported not only by our data but also by a good theoretical model of dialogue.

4.4 Decoders

The local probability distributions obtained are used as decoder inputs. We have experimented with several decoders. As a baseline measure we have included what we call a LOCAL decoder which creates a simple classifier out of the raw local probability distribution. In the case of MaxEnt, for example, this decoder selects

$$\hat{r} = \operatorname*{argmax}_{r} \left(\frac{1}{Z(c)} \exp\left(\sum_{i=1}^{m} w_i f_i(p, r)\right)\right)$$

with r representing a relation type or a binary attachment value. We also used the baseline LAST, where each EDU is attached to the immediately preceding EDU in the linear, textual order.

[2] EDUs are considered as belonging to the same turn if they are by the same speaker without any interjection from an other speaker. In other words any consecutive EDU by the same speaker is considered as belonging to the same turn.

Maximum Spanning Trees To answer our questions, "how many non-tree-like structures are there?" and "how far can tree decoding algorithms get us in multi-party dialogue?", our first decoder is the classic Maximum Spanning Trees (MST) algorithm— used by McDonald et al. (2005) for syntactic dependency parsing as well as by Muller et al. (2012) and Li et al. (2014) for discourse parsing—tweaking it in order to produce structures that are closer to the ones specific to multi-party dialogue. We are looking for:

$$T^* = \underset{T \text{ a spanning tree of } G}{\operatorname{argmax}} \sum_{e \in E(T)} w(e)$$

$$w(e) = \log\left(\frac{p(e)}{1 - p(e)}\right)$$

G being the complete graph of possible edges returned by the classifiers ; $E(D)$ representing the edges of D. The weight function w computes the log-odds of the probability returned by the model.

We used Chu-Liu-Edmonds version of the MST algorithm (Chu and Liu, 1965; Edmonds, 1967), which requires a specific node to be the root, i.e. a node without any incoming edges, of the initial complete graph. For each dialogue, we made an artificial node as the root with special dummy features. At the end of the procedure, this node points the real root of the discourse graph.

Combining intra- and inter-turn models with the turn constraint As described above, we learn a separate local model for intra- and inter-turn EDUs. We also use the turn constraint during decoding. For the intra-turn decoding, we have experimented with various options. One concerns the creation of a classifier out of the local probability distribution. Another intra-turn decoder is Last, which always takes the last EDU for attachment. Finally we also used MST.

We used the exact same decoding approaches for inter-turn decoding. With the structure for the inter-turn EDUs produced separately, we replace those structures with their heads. The detection of a structure's head, be it intra- or inter-turn, uses the same trick as McDonald et al. (2005) did for syntactic parsing: inserting a dummy node as a fake head which contains only outgoing links enabling us essentially to learn the real head of our structures. Our best overall model used Last to link EDUs inside the turn together with MST and the turn constraint for predicting the global structure.

Category	Description
Positional	Speaker initiated the dialogue
-	First utterance of the speaker in the dialogue
-	Position in dialogue
-	*Distance between EDUs*
-	*EDUs have the same speaker*
Lexical	Ends with exclamation mark
-	Ends with interrogation mark
-	Contains possessive pronouns
-	Contains modal modifiers
-	Contains words in lexicons
-	Contains question words
-	Contains a player's name
-	Contains emoticons
-	First and last words
Parsing	Subject lemmas given by syntactic dependency parsing
-	Dialogue act according to predicting model

Table 2: Feature set description. Pair features are italicized.

5 Experiments and Results

To train our local models, we extracted features for every pair of EDUs in a given dialogue. Our features concern the pair of EDUs as well as features related to each EDU specifically. The feature set, detailed in Table 2, can be summarized as follows:

- Positional features: (related to) the non-linguistic context of the pair;
- Lexical features: single words[3] and punctuation present in the EDUs;
- Parsing features: dependency[4] and dialogue act[5] tagging.

Table 3 shows our results on our unseen test corpus, which contains a randomly selected 10% of dialogues in our corpus. The best configuration was selected after performing ten-fold cross validation on the training corpus. The reported results implement the turn-constraint during training

[3]We use a number of lexicons (opinion markers, quantifiers, etc.), each corresponding to a feature

[4]Provided by the Stanford CoreNLP pipeline (Manning et al., 2014).

[5]The prediction model of Cadilhac et al. (2013) generates EDU tags such as *Offer*, *Refusal*, etc.

for the local models. In other words, training instances for the local models include only forward links.

We used two baselines. The first one, Last, simply attaches every EDU to its previous one. This is a very strong baseline in discourse parsing (Muller et al., 2012, for example). The second baseline is essentially the local classifier without any further decoding; in other words, we simply select the class with the highest probability both for attachment and labeling. Attaching to last gives us an F-score of 0.584 for attachment and 0.391 when we add the relations as well. Using only classification from the local probability distribution without decoding gives 0.541 for attachment and 0.446 for attachments and relations.

The best results for the global parsing problem exploited the turn constraint both during learning the local model and during decoding. Within a turn, our discourse structures are simple and largely linear; the best intra-turn results came from using Last. Most of our interlocutors did not create elaborate discourse structures with long-distance attachments within the same turn. The inter-turn level was a different story, as the figures show. For inter-turn and the global problem, MST using the heads of the intra-turn substructures computed with Last, produced the best results. The F1 score for unlabeled structures is at 0.671 while for labelled structures we have 0.516. To enable a comparison with RST style parsing where exact arguments for discourse relations are not computed, the undirected attachment F1 score = 0.68 for the global parsing problem.

Despite the inherent difficulty of discourse parsing on multi-party chat dialogues (simultaneous, multiple discussion threads, lack of syntax) our results are close to or better than the current state of the art for discourse parsing on monologue. There are two approaches currently that use dependency parsing strategies for discourse, thoroughly described in the next section. Li et al. (2014) report an accuracy of 0.7506 for unlabelled structures and 0.4309 for the full labelled structures. Muller et al. (2012) report 0.662 for unlabelled structure and 0.361 for labelled structures. We outperform both systems for full labelled structures, and despite our non-tree-like structures beat or are close to these on unlabelled attachments. Though comparisons across different corpora are difficult, the numbers suggest that our results are more than competitive. Our results also suggest that one can get quite far with tree-based decoding algorithms, though we know that in principle MST cannot do better than 91% even with a perfect local model (a model in which an arc is giving probability 1 just in case it occurs in the gold standard annotation).

6 Related Work

To date, discourse parsing has almost exclusively been applied to monologue. Multi-party chat dialogues have never been considered before. Baldridge and Lascarides (2005) predicted tree discourse structures for 2 party "directed" dialogues from the Verbmobil corpus by training a PCFG that exploited the structure of the underlying task. Elsner and Charniak (2010), Elsner and Charniak (2011) are presenting a combination of local coherence models initially provided for monologues showing that those models can satisfactorily model local coherence in chat dialogues. Nonetheless they do not present a full-fledged discourse parsing model. Our data required a more open domain approach and a more sophisticated approach to structure.

Our use of dependency parsing for learning discourse structure has a few antecedents in the literature on monologue. One of the first papers to introduce this technique is Muller et al. (2012), working with a small French language corpus, ANNODIS (Afantenos et al., 2012a). They use a similar approach to us, including the classic version of the MST decoder. They also used an A^* search as another decoding mechanism but it gave the same results as MST. As we have said, our results better theirs both on attachment and full labeled structures. In the context of RST, Hirao et al. (2013) and Li et al. (2014) transform RST trees into dependency structures; we have discussed their in section 4.1. Li et al. (2014) use both the Eisner algorithm (Eisner, 1996) as well as the MST algorithm from McDonald et al. (2005). As we mentioned, our labelled scores are higher than theirs, though we are cautious of making comparisons across such different corpora.

Most work on discourse parsing focuses on the task of discourse relation labeling between pairs of discourse units—e.g., Marcu and Echihabi (2002) Sporleder and Lascarides (2005) and Lin et al. (2009). This corresponds to our local model. As we have shown in this paper, this setting makes an unwarranted assumption, as it assumes inde-

Method	Undirected Attachment			Directed Attachment			Full Labelled Structure		
	prec	rec	F1	prec	rec	F1	prec	rec	F1
LAST	0.602	0.566	0.584	0.602	0.566	0.584	0.403	0.379	0.391
LOCAL	0.698	0.488	0.574	0.623	0.478	0.541	0.513	0.394	0.446
INTRA-TURN	0.837	0.955	0.892	0.808	0.922	0.861	0.489	0.558	0.521
INTER-TURN	0.617	0.516	0.562	0.616	0.514	0.561	0.492	0.411	0.448
GLOBAL	0.697	0.663	0.680	0.688	0.655	0.671	0.529	0.503	0.516

Table 3: Evaluation results.

pendence of local attachment decisions. There is also work on discourse structure within a single sentence; e.g., Soricut and Marcu (2003) makes use of dynamic programming along with a standard bottom-up chart parsing, while Sagae (2009) uses shift-reduce algorithm for intra-sentential discourse analysis. Such approaches do not apply to our data, as most of the structure in our dialogues lies beyond the sentence level.

As for document-level discourse parsers, Subba and Di Eugenio (2009) use a transition-based approach, following the paradigm of Sagae (2009). duVerle and Prendinger (2009) and Hernault et al. (2010) both rely on locally greedy methods. Like us, they treat attachment prediction and relation label prediction as independent problems. Feng and Hirst (2012) extend this approach by additional feature engineering but is restricted to sentence-level parsing. Finally, Joty et al. (2012) present a sentence-level discourse parser that uses Conditional Random Fields to capture label interdependencies and chart parsing for decoding. Joty et al. (2013) and Joty et al. (2015) extend this approach on the level of documents and have the best results non-dependency based discourse parsing, with an F1 of 0.689 on unlabelled structures and 0.5587 on labelled structures. Our scores are very close to Joty et al.'s, however, and achieved with much simpler methods than theirs.

7 Conclusions

As far as we know, this is the first paper to deal with discourse parsing in multi-party chat dialogues. We believe that such data will be useful for other discourse parsing tasks like analyzing fora with multi-threads. We have used the STAC corpus (Afantenos et al., 2012b) for our data. To simplify the parsing task, we transformed our SDRT structures into dependency ones. We used two different local probability distribution models as input to several decoding mechanisms, including one based on the Maximum Spanning Tree algorithm, and an enhanced version of it in order to produce structures closer to the ones we observe. We obtain the best results using the enhanced version of the MST algorithm. In future work, we plan to investigate ILP constraints in greater depth to develop a plausible alternative to MST on DAGs.

References

Stergos Afantenos, Nicholas Asher, Farah Benamara, Myriam Bras, Cecile Fabre, Mai Ho-Dac, Anne Le Draoulec, Philippe Muller, Marie-Paul Pery-Woodley, Laurent Prevot, Josette Rebeyrolles, Ludovic Tanguy, Marianne Vergez-Couret, and Laure Vieu. 2012a. An empirical resource for discovering cognitive principles of discourse organisation: the ANNODIS corpus. In Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Mehmet Uğur Doğan, Bente Maegaard, Joseph Mariani, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey, may. European Language Resources Association (ELRA).

Stergos Afantenos, Nicholas Asher, Farah Benamara, Anaïs Cadilhac, Cédric Degremont, Pascal Denis, Markus Guhe, Simon Keizer, Alex Lascarides, Oliver Lemon, Philippe Muller, Soumya Paul, Verena Rieser, and Laure Vieu. 2012b. Developing a corpus of strategic conversation in the settlers of catan. In Noriko Tomuro and Jose Zagal, editors, *Workshop on Games and NLP (GAMNLP-12)*, Kanazawa, Japan.

N. Asher and A. Lascarides. 2003. *Logics of Conversation*. Cambridge University Press.

Jason Baldridge and Alex Lascarides. 2005. Probabilistic head-driven parsing for discourse structure. In *Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL)*.

A. Berger, S. Della Pietra, and V. Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71.

Anais Cadilhac, Nicholas Asher, Farah Benamara, and Alex Lascarides. 2013. Grounding strategic conversation: Using negotiation dialogues to predict

trades in a win-lose game. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 357–368, Seattle, Washington, USA, October. Association for Computational Linguistics.

Y. J. Chu and T. H. Liu. 1965. On the shortest arborescence of a directed graph. *Science Sinica*, 14:1396–1400.

David duVerle and Helmut Prendinger. 2009. A novel discourse parser based on support vector machine classification. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 665–673, Suntec, Singapore, August. Association for Computational Linguistics.

Jack Edmonds. 1967. Optimum branchings. *Journal of Research of the National Bureau of Standards*, 71B(233–240).

Jason M. Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING-96)*, volume 1, pages 340–345, Copenhagen, Denmark.

Micha Elsner and Eugene Charniak. 2010. Disentangling chat. *Computational Linguistics*, 36(3):389–409.

Micha Elsner and Eugene Charniak. 2011. Disentangling chat with local coherence models. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1179–1189, Portland, Oregon, USA, June. Association for Computational Linguistics.

Vanessa Wei Feng and Graeme Hirst. 2012. Text-level discourse parsing with rich linguistic features. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 60–68, Jeju Island, Korea, July. Association for Computational Linguistics.

Jonathan Ginzburg. 2012. *The interactive stance*. Oxford University Press.

Hugo Hernault, Helmut Prendinger, David A. duVerle, and Mitsuru Ishizuka. 2010. HILDA: A Discourse Parser Using Support Vector Machine Classification. *Dialogue and Discourse*, 1(3):1–33.

Tsutomu Hirao, Yasuhisa Yoshida, Masaaki Nishino, Norihito Yasuda, and Masaaki Nagata. 2013. Single-document summarization as a tree knapsack problem. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1515–1520, Seattle, Washington, USA, October. Association for Computational Linguistics.

Shafiq Joty, Giuseppe Carenini, and Raymond Ng. 2012. A Novel Discriminative Framework for Sentence-Level Discourse Analysis. In *EMNLP-CoNLL*.

Shafiq Joty, Giuseppe Carenini, Raymond Ng, and Yashar Mehdad. 2013. Combining intra- and multi-sentential rhetorical parsing for document-level discourse analysis. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 486–496, Sofia, Bulgaria, August. Association for Computational Linguistics.

Shafiq Joty, Giuseppe Carenini, and Raymond Ng. 2015. Codra: A novel discriminative framework for rhetorical analysis. *Computational Linguistics*.

Sujian Li, Liang Wang, Ziqiang Cao, and Wenjie Li. 2014. Text-level discourse dependency parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 25–35, Baltimore, Maryland, June. Association for Computational Linguistics.

Ziheng Lin, Min-Yen Kan, and Hwee Tou Ng. 2009. Recognizing implicit discourse relations in the Penn Discourse Treebank. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 343–351, Singapore, August. Association for Computational Linguistics.

William C. Mann and Sandra A. Thompson. 1987. Rhetorical Structure Theory: A Framework for the Analysis of Texts. Technical Report ISI/RS-87-185, Information Sciences Institute, Marina del Rey, California.

William C. Mann and Sandra A. Thompson. 1988. Rhetorical Structure Theory: Towards a Functional Theory of Text Organization. *Text*, 8(3):243–281.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60.

Daniel Marcu and Abdessamad Echihabi. 2002. An unsupervised approach to recognizing discourse relations. In *Proceedings of ACL*, pages 368–375.

Daniel Marcu. 2000. *The Theory and Practice of Discourse Parsing and Summarization*. The MIT Press.

Ryan T. McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajic. 2005. Non-projective dependency parsing using spanning tree algorithms. In *HLT/EMNLP*.

Philippe Muller, Stergos Afantenos, Pascal Denis, and Nicholas Asher. 2012. Constrained decoding for text-level discourse parsing. In *Proceedings of COLING 2012*, pages 1883–1900, Mumbai, India,

December. The COLING 2012 Organizing Committee.

Massimo Poesio and David R Traum. 1997. Conversational actions and discourse situations. *Computational intelligence*, 13(3):309–347.

Rashmi Prasad, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Livio Robaldo, Aravind Joshi, and Bonnie L. Webber. 2008. The Penn Discourse TreeBank 2.0. In *Proceedings of LREC 2008*.

Kenji Sagae. 2009. Analysis of discourse structure with syntactic dependencies and data-driven shift-reduce parsing. In *Proceedings of the 11th International Conference on Parsing Technologies*, IWPT '09, pages 81–84, Stroudsburg, PA, USA. Association for Computational Linguistics.

Emanuel A Schegloff. 2007. *Sequence organization in interaction: Volume 1: A primer in conversation analysis*, volume 1. Cambridge University Press.

David Schlangen. 2003. *A coherence-based approach to the interpretation of non-sentential utterances in dialogue*. Ph.D. thesis, University of Edinburgh. College of Science and Engineering. School of Informatics.

C. Sidner. 1994a. An artificial discourse language for collaborative negotiation. In *AAAI*, volume 1, pages 814–819. MIT Press, Cambridge.

C. Sidner. 1994b. Negotiation in collaborative activity: a discourse analysis. *Knowledge-Based Systems*, 7(4):265 – 267.

R. Soricut and D. Marcu. 2003. Sentence level discourse parsing using syntactic and lexical information. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 149–156. Association for Computational Linguistics.

Caroline Sporleder and Alex Lascarides. 2005. Exploiting linguistic cues to classify rhetorical relations. In *Proceedings of Recent Advances in Natural Langauge Processing (RANLP)*, Bulgaria.

Rajen Subba and Barbara Di Eugenio. 2009. An effective discourse parser that uses rich linguistic information. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 566–574, Boulder, Colorado, June. Association for Computational Linguistics.

Maite Taboada and William C. Mann. 2006. Rhetorical Structure Theory: Looking Back and Moving Ahead. *Discourse Studies*, 8(3):423–459, June.

Ben Wellner and James Pustejovsky. 2007. Automatically Identifying the Arguments of Discourse Connectives. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 92–101, Prague, Czech Republic, June. Association for Computational Linguistics.

Joint prediction in MST-style discourse parsing for argumentation mining

Andreas Peldszus
Applied Computational Linguistics
UFS Cognitive Science
University of Potsdam
peldszus@uni-potsdam.de

Manfred Stede
Applied Computational Linguistics
UFS Cognitive Science
University of Potsdam
stede@uni-potsdam.de

Abstract

We introduce a new approach to argumentation mining that we applied to a parallel German/English corpus of short texts annotated with argumentation structure. We focus on structure prediction, which we break into a number of subtasks: relation identification, central claim identification, role classification, and function classification. Our new model jointly predicts different aspects of the structure by combining the different subtask predictions in the edge weights of an evidence graph; we then apply a standard MST decoding algorithm. This model not only outperforms two reasonable baselines and two data-driven models of global argument structure for the difficult subtask of relation identification, but also improves the results for central claim identification and function classification and it compares favorably to a complex mstparser pipeline.

1 Introduction

Argumentation mining is a task that has drawn increased interest in the last years. In its full-fledged version, it seeks to automatically recognize the structure of argumentation in a text by identifying and connecting the central claim of the text, supporting premises, possible objections, and counter-objections to these objections.[1]

A variety of applications can profit from access to the argumentative structure of text, including the retrieval of relevant court decisions from legal databases (Palau and Moens, 2011), automatic document summarization systems (Teufel and Moens, 2002), the analysis of scientific papers in biomedical text mining (Teufel, 2010; Liakata et al., 2012), or essay scoring. Importantly, argument analysis can also be an extension of opinion mining applications.

To make argumentation structures available for these applications, their robust automatic recognition is required, a task that is very challenging: argumentative strategies and styles vary across text genres and languages; classifying arguments might require domain knowledge; furthermore, argumentation can often rely on implicitly conveyed messages.

The full-fledged task can be decomposed into several subtasks:

- Segmentation: splitting the text into elementary discourse units (EDUs as used in general kinds of discourse parsing, typically sentences or clauses)
- Identification of argumentative discourse units (ADUs): discarding argumentatively irrelevant EDUs, joining adjacent EDUs to form larger ADUs
- ADU type classification: determining the type of argumentative unit; different schemes have been proposed, involving stance, evidence types, rhetorical status, argumentative function
- Relation identification: building a connected tree- or graph-structure to represent argumentative relations between the ADUs
- Relation type classification: determining the type of argumentative relation (e.g. supporting versus attacking relations or more fine-grained types)

In this paper, we address the last three subtasks: Given a text segmented into relevant ADUs, identify the argumentation structure. We will work with a bilingual corpus of short texts that have been generated in a text production experiment.

[1] A comprehensive overview of the research field is given in (Peldszus and Stede, 2013).

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 938–948,
Lisbon, Portugal, 17-21 September 2015.

The next section describes related work. In section 3, we present the dataset used in our experiments. Section 4 gives a more detailed description of the task. The baselines and the models are presented in section 5. We then report the result of our experiments in section 6 and close with some concluding remarks.

2 Related Work

In our discussion of related work, we focus on the three subtasks addressed in this paper:

ADU type classification: One typical classification task concerns the properties of a segment in the argumentation structure: Burstein and Marcu (2003) trained classifiers for identifying thesis and conclusion statements in student essays, using additional automatic discourse parse features and cue words, resulting in an average F-score of 53% for thesis and 80% for conclusion segments. For legal texts, Palau and Moens (2011) demonstrated in their influential work how to classify the segment of a text into premises and conclusions, obtaining an F-score of 68% and 74% for the two classes. More recently, Stab and Gurevych (2014) classified segments in student essays into the classes major claim (of the text), claim (of the paragraph), premise and irrelevant. The macro average F-score for all classes is 73%, the F-score for the claim of the paragraph 54% and for the major claim 63%.

Besides structural segment-wise classification tasks, there is also work on more semantic tasks: The rhetorical status of a segment is classified in the argumentative zoning approaches (Teufel and Moens, 2002; Teufel and Kan, 2011; Liakata et al., 2012), where certain coarse-grained patterns of argumentation in scholarly papers can be captured. Park and Cardie (2014) focus on supporting segments and classify which type of evidence is presented in it. Finally, stance classification (Hasan and Ng, 2013) might be of interest to identify possible objections, although it is typically applied on full comments and not on single segments.

Relation identification: Much less prior work can be found for the process of building argumentation structures. Palau and Moens (2011) used a hand-written context-free grammar to predict argumentation trees on legal documents, achieving an accuracy of 60%. Only recently, data-driven approaches have been applied. Lawrence et al. (2014) construct tree structures on philosophical texts using unsupervised methods based on topical distance between the segments. The relations in the tree are neither labeled not directed. Unfortunately, the method was evaluated on only a few annotated items, which is why we cannot comment on the results. Finally, Stab and Gurevych (2014) present a supervised data-driven approach for relation identification. They predict attachment for support-graphs spanning over paragraphs of English essays and obtain a macro F1 score of 72%, and an F1 score of 52% for positive attachment. No decoding is used to optimize global predictions per text.

Relation type classification: The only study on explicitly classifying argumentative relations we are aware of is (Feng and Hirst, 2011). They classify pairs of premise and conclusion from newswire text into a set of five frequently used argumentation schemes in the sense of Walton et al. (2008). In one-against-others classification, the system yields best average accuracies of over 90% for two schemes, while for the other three schemes the results are between 63% and 70%.

To the best of our knowledge, no data-driven model of argumentation structure has been proposed yet that would optimize argumentation structure globally for the complete input text, as it is done in other discourse parsing tasks, e.g. in (Muller et al., 2012).

3 Dataset

Texts: We use the *arg-microtext* corpus (Peldszus and Stede, to appear), a freely available[2] parallel corpus of 112 short texts with 576 ADUs. The texts are authentic discussions of controversial issues. They were originally written in German and have been professionally translated to English, preserving the segmentation and if possible the usage of discourse markers. The texts have been collected in a controlled text generation experiment, with the result that all of them fulfill the following criteria: (i) The length of each text is about 5 ADUs (henceforth: segments). (ii) One segment explicitly states the central claim. (iii) Each segment is argumentatively relevant. (iv) At least one objection to the central claim is considered.

Scheme: The argumentation structure of every text has been annotated according to a scheme (Peldszus and Stede, 2013) based on Freeman's theory of argumentation structures (Free-

[2]https://github.com/peldszus/arg-microtexts

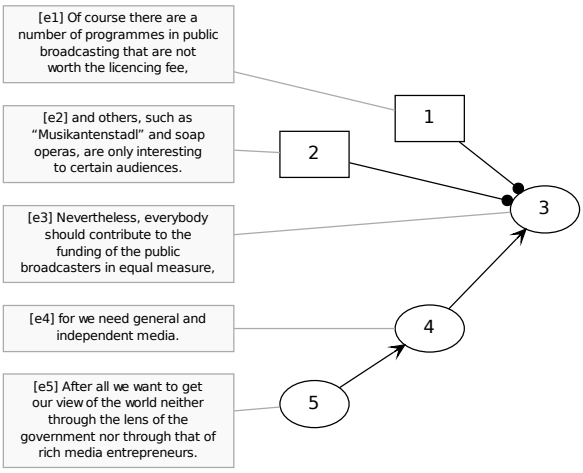


Figure 1: An example text and its reduced argumentation structure: Text segments, proponent (round) and opponent (box) nodes, supporting (arrow-head) and attacking (circle-head) relations.

man, 1991; Freeman, 2011), that has been proven to yield reliable structures in annotation experiments (Peldszus, 2014). The argumentation structure of a text is defined as a graph with the text segments as nodes. Each node is associated with one argumentative *role*: the proponent who presents and defends the central claim, or the opponent who critically questions the proponent's claims. Edges between the nodes represent argumentative relations, and each edge is of one specific argumentative *function*: support or attack. The scheme allows to discriminate between "rebutting" attacks, targeting another node and thereby challenging its acceptability, and "undercutting" attacks, targeting an edge and thereby challenging the acceptability of the inference from the source to the target node. It can also represent linked support, where multiple premises jointly support a claim.

Transformation: The annotated graph structures can be quite complex, especially when they involve undercutting relations and linked support. For the purpose of this study, we thus reduce the graphs to a simpler tree-like representation. All relations pointing to edges are rewritten to point to the source node of the original target edge, which enables the use of standard graph algorithms (like MST). Also, this is a loss-less mapping, given that every segment has only one outgoing arc (as generally done in argumentation models). Furthermore, the set of relation types is reduced to the simple binary distinction between support and attack. We think this is a reasonable simplification that facilitates comparisons with slightly different approaches/datasets (we are not aware of any dataset that makes use of the full granularity proposed in our scheme).

An example text from the corpus in its reduced form is shown in Figure 1. Text boxes are EDUs, each of which constitutes also an ADU. Proponent ADUs are round nodes, opponent ADUs are box nodes. Supporting relations have a normal arrowhead, while attacking relations have a circle arrowhead.

All statistics on the annotated argumentation structures apply equally for the German and the English version of the parallel corpus.

4 Task

Identifying the structure of argumentation according to our scheme involves choosing one segment as the central claim of the text, deciding how the other segments are related to the central claim and to each other, identifying the argumentative role of each segment, and finally the argumentative function of each relation.

Our prior experiments on automating the recognition of argumentation structure approached the problem as a segment-wise classification task (Peldszus, 2014). Formulating the task this way was successful for the recognition of argumentative role and function of a segment. For the automation of the structure building however, the segment-wise classification of attachment with only a small context window around the target segment proved to be a very hard task. This is due to the long-distance dependencies frequently found in argumentation graphs. For example, 46% of the relations marked in the corpus used for this study involve non-adjacent segments. For longer texts this number might increase further: Stab and Gurevych (2014) report a rate of 63% of non-adjacent relations in their corpus.

In this study we therefore frame the task of attachment classification as a binary decision, where the classifier, when given a pair of a source and a target segment, chooses whether or not to establish a relation from the source to the target. Since these relations can hold not only between adjacent but between arbitrary segments of the text, all possible combinations of segments are required to be tested. Consequently, the class distribution is very skewed.

- **attachment (at):** Is there an argumentative

connection between the source and the target segment? In the corpus, a relation has been annotated for 464 segment pairs, no relation has been annotated for the combinatorially remaining 2000 pairs of segments.

In this paper we first address only the task of attachment classification, and then the prediction of the full graph, involving all other levels:

- **central claim (cc):** Is the current segment the central claim of the text? In our data 112 of the 576 segments are central claims.
- **role (ro):** Does the current segment present a claim of the proponent or the opponent? In our data 451 of the 576 segments are proponent segments and 125 are opponent segments.
- **function (fu):** Has the current segment a supporting or an attacking function? In our data, 290 segments are supports, 174 are attacks and 112 are the central claim and thus have no own function.

5 Models

We compare two heuristic baseline models and different data-driven models that we developed, each of them trained and evaluated separately on both language versions of the corpus. All models are evaluated on the basis of 10 iterations of 5x3-fold nested cross validation (CV). The outer 5-fold CV is for evaluation only, i.e. to ensure that the model is trained only on training data and tested only on test data. If a model requires hyperparameters to be tuned or multiple passes, then this is achieved via one (or multiple) inner 3-fold CV over the training data only. The folding is stratified, randomly distributing the texts of the corpus while aiming to reproduce the overall label distribution in both training and test set.

5.1 Baseline: attach to first

In the English-speaking school of essay writing and debating, there is the tendency to state the central claim of a text or a paragraph in the very first sentence, followed by supporting arguments. It is therefore a reasonable baseline to assume that all segments attach to the first segment. In our corpus, the first segment is the central claim in 50 of the 112 texts (44.6%).

This baseline (**BL-first**) will not be able to capture serial argumentation, where one more general argument is supported or attacked by a more specific one. However, it will cover convergent argumentation, where separate arguments are put forward in favor of the central claim (given that it is expressed in the first segment). It will always produce flat trees. In our corpus, 176 of the 464 relations (37.9%) attach to the first segment.

5.2 Baseline: attach to preceding

A typically very strong baseline in discourse parsing is attaching to the immediately preceding segment (Muller et al., 2012). Possibly, this holds more for corpora with relations often or always being adjacent, as in rhetorical structure trees. Since argumentation structures often exhibit non-adjacent relations (see above), this heuristic might be easier to beat in our scenario.

This baseline (**BL-preced.**) will always produce chain trees and thus cover serial argumentation, but not convergent argumentation. In our corpus, 210 of all 464 relations (45.3%) attach to the preceding segment.

5.3 Learned attachment without decoding

We train a linear log-loss model (**simple**) using stochastic gradient descent (SGD) learning, with elastic net regularization, the learning rate set to optimal decrease and class weight adjusted according to class distribution (Pedregosa et al., 2011). The following hyper parameters are tuned in the inner CV: the regularization parameter alpha, the elastic net mixing parameter and the number of iterations. We optimize macro averaged F1-score.

For each text segment, we extract binary features for lemma, pos-tags, lemma- and pos-tag-based dependency-parse triples and the main verb morphology (Bohnet, 2010), and discourse connectives (Stede, 2002), furthermore simple statistics like relative segment position, segment length and punctuation count. For each pair of text segments, we extract relative distance between the segments and their linear order (is the source before or after the target). The feature vector for each pair then contains both the pair features and the segment features for source and target segment and their adjacent segments.[3]

[3] We experimented with several features, some of which were dismissed from the final evaluation runs due to lacking impact: sentiment values and the presence of negation for

5.4 Learned attachment with MST decoding

The simple model just described might be able to learn which segment pairs actually attach, i.e., correspond to some argumentative relation in the corpus. However it is not guaranteed to yield predictions that can be combined to a tree structure again. A more appropriate model would enforce global constraints on its predictions. In the **simple+MST** model, this is achieved by a *minimum spanning tree* (MST) decoding, which has first been applied for syntactic dependency parsing (McDonald et al., 2005a; McDonald et al., 2005b) and later for discourse parsing (Baldridge et al., 2007; Muller et al., 2012). First, we build a fully-connected directed graph, with one node for each text segment. The weight of each edge is the attachment probability predicted by the learned classifier for the corresponding pair of source and target segment. We then apply the Chu-Liu-Edmonds algorithm (Chu and Liu, 1965; Edmonds, 1967) to determine the minimum spanning tree, i.e., the subgraph connecting all nodes with minimal total edge cost (in our case highest total edge probability). This resulting tree then represents the best global attachment structure for a text given the predicted probabilities.

5.5 Joint prediction with MST decoding

All models presented in the previous subsections have in common that they do not rely on other features of the argumentation graph. However, it is fair to assume that knowledge about the argumentative role and function of a segment or its likeliness to be the central claim might improve the attachment classification. Consequently, our next model considers not only the predicted probability of attachment for a segment pair, but also the predicted probabilities of argumentative role, function and of being the central claim for each segment. The predictions of all levels are combined in one *evidence graph*.

Additional segment-wise base classifiers: We train base classifiers for the role, function and central claim level using the same learning regime as described in Section 5.3. Contrary to the attachment classification, the items are not segment pairs but single segments. We thus extract all segment-based features as described above for the target segment and its adjacent segments.

segments, and distance measures between pairs of segments in terms of word-overlap, td-idf and LDA distributions.

Combining segment and segment-pair predictions: Our goal in this model is to combine the predicted probabilities of all levels in one edge score, so that the MST decoding can be applied as before. Figure 2 depicts the situation before and after the combination, first with separate prediction for segments and segment pairs and then with the combined edge scores.

The evidence graph is constructed as follows: First, we build a fully connected multigraph over all segments with as many edges per segment-pair as there are edge types. In our scenario there are two edge types, supporting and attacking edges. Then we translate the segment-wise predictions into level-specific edge scores.

The edge-score for the central claim level $\overline{\text{cc}}_{i,j}$ is equal to the probability of the edge's source not being the central claim, which is capturing the intuition that central claims are unlikely to have outgoing edges:

$$\overline{\text{cc}}_{i,j} = \quad p(\text{cc}_i = \text{no}) \tag{1}$$

The edge-score for the argumentative function level $\overline{\text{fu}}_{i,j}$ is equal to the probability of the source being the corresponding segment for the edge type:

$$\overline{\text{fu}}_{i,j} = \begin{cases} p(\text{fu}_i = \text{sup}) & \text{for sup. edges} \\ p(\text{fu}_i = \text{att}) & \text{for att. edges} \end{cases} \tag{2}$$

The edge-score for the argumentative role level $\overline{\text{ro}}_{i,j}$ is also determined by the edge type. Attacking edges involve a role switch (proponent or opponent would not attack their own claims), while supporting edges preserve the role (proponent or opponent will only support their own claims):

$$\overline{\text{ro}}_{i,j} = \begin{cases} p(\text{ro}_i = \text{pro}) \times p(\text{ro}_j = \text{pro}) + \\ p(\text{ro}_i = \text{opp}) \times p(\text{ro}_j = \text{opp}) & \text{for sup. edges} \\ \\ p(\text{ro}_i = \text{pro}) \times p(\text{ro}_j = \text{opp}) + \\ p(\text{ro}_i = \text{opp}) \times p(\text{ro}_j = \text{pro}) & \text{for att. edges} \end{cases} \tag{3}$$

Finally, of course the edge-score for the attachment level $\overline{\text{at}}_{i,j}$ is equal to the probability of attachment between the segment pair:

$$\overline{\text{at}}_{i,j} = \quad p(\text{at}_{i,j} = \text{yes}) \tag{4}$$

The combined score of an edge $w_{i,j}$ is then defined as the weighted sum of the level-specific edge score:

$$w_{i,j} = \frac{\phi_1 \overline{\text{ro}}_{i,j} + \phi_2 \overline{\text{fu}}_{i,j} + \phi_3 \overline{\text{cc}}_{i,j} + \phi_4 \overline{\text{at}}_{i,j}}{\sum \phi_n} \tag{5}$$

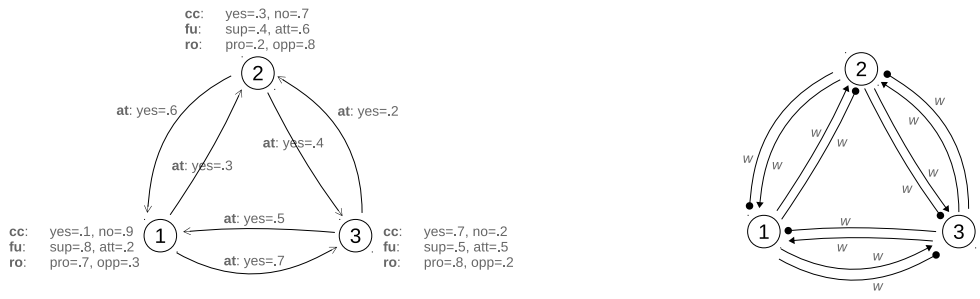


Figure 2: An example evidence graph before (left) and after (right) the predicted probabilities of the different levels have been combined in a single edge score.

In our implementation, the combined evidence graphs can be constructed without a weighting, and then be instantiated with a specific weighting to yield the combined edge scores $w_{i,j}$.

Procedure: As before, we first tune the hyperparameters in the inner CV, train the model on the whole training data and predict probabilities on all items of the test set. Also, we predict all items in the training data "as unseen" in a second inner CV using the best hyperparameters. This procedure is executed for every level. Using the predictions of all four levels, we then build the evidence graphs for training and test set.

Finding the right weighting: We evaluate two versions of the evidence graph model. The first version (**EG equal**) gives equal weight to each level-specific edge score. The second version (**EG best**) optimizes the weighting of the base classifiers with a simple evolutionary search on all evidence graphs of the training set, i.e. it searches for a weighting that maximizes the average level evaluation score of the decoded argumentation structures in the training set. Finally, all evidence graphs of the test set are instantiated with the selected weighting (the equal one or the optimized one) and evaluated.

5.6 Comparison: MST parser

Finally, we compare our models to the well-known mstparser[4], which was also used in the discourse parsing experiments of Baldridge et al. (2007). The mstparser applies 1-best MIRA structured learning, a learning regime that we expect to be superior over the simple training in the previous models. In all experiments in this paper, we use 10 iterations for training, the non-projective 1-best MST decoding, and no second order features. The base mstparser model (**MP**) evaluated here uses the same features as above, as well as its own features extracted from the dependency structure. Second, we evaluate a pre-classification scenario (**MP+p**), where the predictions of the base classifiers trained in the above models for central claim, role and function are added as additional features. We expect this to improve the central claim identification as well as the edge labeling.

For the full task involving all levels, we combine the mstparser with an external edge labeler, as the internal edge labeler is reported to be weak. In this setting (**MP+r**), we replace the edge labels predicted by the mstparser with the predictions of the base classifier for argumentative function. Furthermore, the combination of pre-classification, mstparser and external relation labeler (**MP+p+r**) is evaluated. Finally, we evaluate a scenario (**MPϵ+p+r**) where the mstparser has access only to its own features and to those of the pre-classification, but not to the features described in Section 5.3, and the external relation labeller is used. In this scenario, the mstparser exclusively serves as a meta-model on the base classifier's predictions.

6 Results

All results are reported as average and standard deviation over the 50 folds resulting from 10 iterations of (the outer) 5-fold cross validation. We use the following metrics: macro averaged F1, F1 for positive attachment, and Cohen's Kappa κ. For significance testing, we apply the Wilcoxon signed-rank test on the macro averaged F1 scores and assume a significance level of α=0.01.

6.1 Attachment task

Table 1 shows the results in the attachment task. The rule-based baseline scores are equal for both

[4] http://sourceforge.net/projects/mstparser/

	BL-first	BL-preced.	simple	simple+MST	EG equal	EG best	MP	MP+p
F1 macro	.618±.041	.662±.025	.679±.025	.688±.032	.712±.026	.710±.028	.724±.030	**.728**±.033
attach F1	.380±.067	.452±.039	.504±.038	.494±.053	.533±.042	.530±.044	.553±.048	**.559**±.053
κ	.236±.081	.325±.050	.365±.048	.377±.064	.424±.052	.421±.055	.449±.060	**.456**±.066
trees	100%	100%	15.4%	100%	100%	100%	100%	100%
	BL-first	BL-preced.	simple	simple+MST	EG equal	EG best	MP	MP+p
F1 macro	.618±.041	.662±.025	.663±.030	.674±.036	.692±.034	.693±.031	.707±.035	**.720**±.034
attach F1	.380±.067	.452±.039	.478±.049	.470±.058	.501±.056	.502±.052	.524±.056	**.546**±.056
κ	.236±.081	.325±.050	.333±.059	.347±.071	.384±.068	.386±.063	.414±.070	**.440**±.069
trees	100%	100%	11.6%	100%	100%	100%	100%	100%

Table 1: Results for the attachment task: for German (above) and English (below), best values highlighted.

	German		English	
total graphs	1120	100.0%	1120	100.0%
rooted	1091	97.4%	1088	97.1%
cycle free	1059	94.6%	995	88.8%
full span	908	81.1%	864	77.1%
out degree	298	26.6%	283	25.3%
trees	173	15.4%	120	10.7%

Table 2: Number and percentage of valid trees for the "simple" attachment model

languages, since they rely only on the annotated structure of the parallel corpus. Here, attach-to-first is the lower bound, attach-to-preceding is a more competitive baseline, as we had hypothesized in section 5.2.

The learned classifier (simple) beats both baselines in both languages, although the improvement is much smaller for English than for German. In general, the classifier lacks precision compared to recall: It predicts too many edges. As a result, the graph constructed from the predicted edges for one text very often does not form a tree. In Table 2, we give a summary of how often tree constraints are fulfilled, showing that without decoding, valid trees can only be predicted for 15.4% of the texts in German and for 10.7% of the texts in English. The most frequently violated constraint is "out degree", stating that every node in the graph should have at most one outgoing edge. Note that all other models, the baselines as well as the MST decoding models, are guaranteed to predict tree structures.

The simple+MST model yields slightly lower F1-scores for positive attachment than without decoding, trading off a loss of 10 points in recall of the over-optimistic base classifier against a gain of 5 in precision. However, the output graphs are constrained to be trees now, which is rewarded by a slight increase in the summarizing metrics macro F1 and κ.

The evidence graph models (EG equal & EG best) clearly outperform the simple and simple+MST model, indicating that the attachment classification can benefit from jointly predicting the four different levels. Note, that the EG model with equal weighting scores slightly better than the one with optimized weighting for German but not for English. However, this difference is not significant ($p>0.5$) for both languages, which indicates that the search for an optimal weighting is not necessary for the attachment task.

The overall best result is achieved by the mstparser model. We attribute this to the superior structured learning regime. The improvement of MP over EP equal and best is significant in both languages ($p<0.008$). Using pre-classification further improves the results, although difference is neither significant for German ($p=0.4$) nor for English ($p=0.016$).

6.2 Full task

Until now, we only focused on the attachment task. In this subsection we will present results on the impact of joint prediction for all levels.

The results in Table 3 show significant improvements of the EG models over the base-classifiers on the central claim, the function and the attachment levels ($p<0.0001$). This demonstrates the positive impact of jointly predicting all levels. The EG models achieve the best scores in central claim identification and function classification, and the second best result in role identification. The differences between EG equal and EG best are not significant on any level, which again indicates that

		simple	EG equal	EG best	MP	MP+p	MP+r	MP+p+r	MPϵ+p+r
cc	maF1	.849±.035	.879±.042	**.890**±.037	.825±.055	.855±.055	.825±.055	.855±.055	.854±.053
	κ	.698±.071	.759±.085	**.780**±.073	.650±.111	.710±.110	.650±.111	.710±.110	.707±.105
ro	maF1	**.755**±.049	.737±.052	.734±.046	.464±.042	.477±.047	.656±.054	.669±.062	.664±.053
	κ	**.511**±.097	.477±.103	.472±.092	.014±.049	.022±.063	.315±.106	.340±.122	.330±.105
fu	maF1	.703±.046	.735±.045	**.736**±.043	.499±.054	.527±.047	.698±.054	.723±.052	.723±.050
	κ	.528±.068	.573±.066	**.570**±.063	.293±.056	.326±.056	.522±.076	.557±.075	.560±.073
at	maF1	.679±.025	.712±.026	.710±.028	.724±.030	**.728**±.033	.724±.030	**.728**±.033	.724±.029
	κ	.365±.048	.424±.052	.421±.055	.449±.060	**.456**±.066	.449±.060	**.456**±.066	.448±.059
		simple	EG equal	EG best	MP	MP+p	MP+r	MP+p+r	MPϵ+p+r
cc	maF1	.817±.045	.860±.051	**.869**±.053	.780±.063	.831±.059	.780±.063	.831±.059	.823±.063
	κ	.634±.090	.720±.103	**.737**±.107	.559±.126	.661±.118	.559±.126	.661±.118	.647±.122
ro	maF1	**.750**±.045	.721±.051	.720±.047	.482±.053	.475±.047	.620±.064	.638±.057	.641±.062
	κ	**.502**±.090	.445±.098	.442±.092	.024±.068	.015±.060	.243±.126	.280±.114	.285±.122
fu	maF1	.671±.049	.707±.048	**.710**±.050	.489±.062	.514±.059	.642±.057	.681±.057	.677±.059
	κ	.475±.074	.529±.070	**.530**±.072	.254±.058	.296±.063	.440±.081	.491±.083	.486±.083
at	maF1	.663±.030	.692±.034	.693±.031	.707±.035	**.720**±.034	.707±.035	**.720**±.034	.713±.033
	κ	.333±.095	.384±.068	.386±.063	.414±.070	**.440**±.069	.414±.070	**.440**±.069	.427±.066

Table 3: Results for the full task: for German (above) and English (below), best values highlighted.

we can dispense with the extra step of optimizing the weighting and use the simple equal weighting. These result are consistent across both languages.

The pure labeled mstparser model (MP) performs worse than the base classifiers on all levels except for the attachment task. Adding pre-classification yields significant improvements on all levels but role identification. Using the external relation labeler drastically improves function classification and indirectly also role identification. The combined model (MP+p+r) yields best results for all mstparser models, but is still significantly outperformed by EG equal in all tasks except attachment classification. There, the mstparser models achieve best results, the improvement of MP+p+r over EG equal is significant for English ($p<0.0001$) and for German ($p=0.001$). Interestingly, the meta-model (MPϵ+p+r) which has access to its own features and to those of the pre-classification, but not to the features described in Section 5.3, performs nearly as good as or equal to the combined model (MP+p+r).

The only level not benefiting from any MST model in comparison with the base classifier is the role classification: In the final MST, the role of each segment is only implicitly represented, and can be determined by following the series of the role-switches of each argumentative function from the segment to the root. The loss of accuracy for predicting the argumentative role is much smaller for German than for English, probably due to the better attachment classification in the first place.

Finally, note that the EG best model gives the highest total score when summed over all levels, followed by EG equal and then MP+p+r.

Projecting further improvements: We have shown that joint prediction of all levels in the evidence graph models helps to improve the classification on single levels. To measure exactly how much a level contributes to the predictions of other levels, we simulate better base classifiers and study their impact. To achieve this, we artificially improved the classification of one target level by overwriting a percentage of its predictions with ground truth. The overwritten predictions where drawn randomly, corresponding to the label distribution of the target level. E.g. for a 20% improvement on the argumentative function level, the predictions of 20% of the true "attack"-items were set to attack and the predictions of 20% of the true "support"-items were set to support, irrespective of whether the classifier already chose the correct label.

The results of the simulations are presented in Figure 3 for English only, due to space constraints. The results for German exhibit the same trends. The figure plots the κ-score on the y-axis against the percentage of improvement on the x-axis. Artificially improved levels are drawn as a dashed line. As the first plot shows, function classifica-

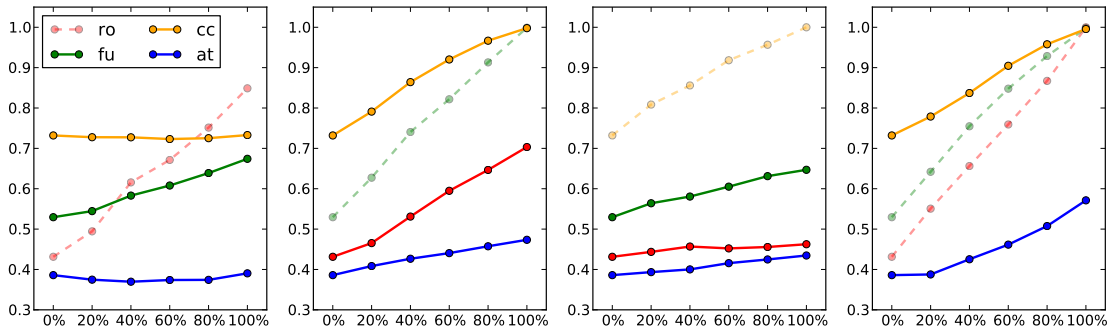


Figure 3: Simulations of the effect of better base classifiers in the EG equal model for English: dashed levels artificially improved, x = number of predictions overwritten with ground truth; y = average κ score in 10 iterations of 5fold CV.

tion is greatly improved by a better role classification (due to the logical connection between them), whereas the other levels are unaffected. In contrast, all levels would benefit from a better function classification, most importantly even the attachment classification. Potential improvements in the central claim identification mostly affect function classification (as these classification tasks partly overlap: central claims will not be assigned a function they cannot have). Finally, a combined improvement on the logically coupled task of role and function identification, would even more help the attachment classification. It might thus be useful to work on a better joint role and function classifier in near future.

Evidence combination: As pointed out by one reviewer, combining the evidence in an edge score as a weighted sum, see (5), instead of a product of probabilities might be inadequate and could result in a model that optimizes the highest scored but not the most probable structure. We compared the EG equal against an EG model with a product of probability. The model scores are nearly identical and do not show a significant difference.

7 Summary and Outlook

We introduced a new approach to argumentation mining that we applied to a parallel German/English corpus of 112 short texts. For the purposes of automatic mining, the original more fine-grained annotation in the corpus was reduced to a slightly simplified scheme consisting of support and attack relations among argumentative discourse units. We did not address the segmentation step here but focused on structure prediction, which we broke into a number of subtasks. Our new *evidence graph* model jointly predicts different aspects of the structure by combining the different subtask predictions in the edge weights of an evidence graph; we then apply a standard MST decoding algorithm. This model not only outperforms two reasonable baselines and two simple models for the difficult subtask of attachment/relation identification, but also improves the results for central claim identification and relation classification, and it compares favorably to a 3-pass mstparser pipeline.

To the best of our knowledge, this is the first data-driven model of argumentation structure that optimizes argumentation structure globally for the complete sequence of input segments. Furthermore, it is the first model jointly tackling segment type classification, relation identification and relation type classification.

Although a direct comparison with results from related work on other corpora is not possible, we can draw indirect comparisons. The first learned model without decoding (simple) is similar to the one presented by Stab and Gurevych (2014). Since it is outperformed by our joint MST decoding model on our data, we assume similar gains could be accomplished on their student essay dataset.

Our next step is to apply the method to other corpora and to more complex text, where the identification of non-participating segments (which are irrelevant for the argumentation) needs to be accounted for. Furthermore, we plan to investigate structured models that not only jointly predict but jointly learn the different aspects of the argumentation graph.

Acknowledgments

We are grateful to the anonymous reviewers for their thoughtful comments. We want to thank Željko Agić, Stergos Afantenos and Christoph Teichmann for fruitful discussions and Wladimir Sidorenko for feedback on an earlier version of the paper. The first author was supported by a grant from Cusanuswerk.

References

Jason Baldridge, Nicholas Asher, and Julie Hunter. 2007. Annotation for and robust parsing of discourse structure on unrestricted texts. *Zeitschrift für Sprachwissenschaft*, 26:213–239.

Bernd Bohnet. 2010. Very high accuracy and fast dependency parsing is not a contradiction. In *Proceedings of the 23rd International Conference on Computational Linguistics*, COLING '10, pages 89–97, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jill Burstein and Daniel Marcu. 2003. A machine learning approach for identification thesis and conclusion statements in student essays. *Computers and the Humanities*, 37(4):455–467.

Y. J. Chu and T. H. Liu. 1965. On the shortest arborescence of a directed graph. *Science Sinica*, 14:1396–1400.

Jack Edmonds. 1967. Optimum Branchings. *Journal of Research of the National Bureau of Standards*, 71B:233–240.

Vanessa Wei Feng and Graeme Hirst. 2011. Classifying arguments by scheme. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 987–996, Stroudsburg, PA, USA. Association for Computational Linguistics.

James B. Freeman. 1991. *Dialectics and the Macrostructure of Argument*. Foris, Berlin.

James B. Freeman. 2011. *Argument Structure: Representation and Theory*. Argumentation Library (18). Springer.

Saidul Kazi Hasan and Vincent Ng. 2013. Stance classification of ideological debates: Data, models, features, and constraints. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 1348–1356. Asian Federation of Natural Language Processing.

John Lawrence, Chris Reed, Colin Allen, Simon McAlister, and Andrew Ravenscroft. 2014. Mining arguments from 19th century philosophical texts using topic based modelling. In *Proceedings of the First Workshop on Argumentation Mining*, pages 79–87, Baltimore, Maryland, June. Association for Computational Linguistics.

Maria Liakata, Shyamasree Saha, Simon Dobnik, Colin R. Batchelor, and Dietrich Rebholz-Schuhmann. 2012. Automatic recognition of conceptualization zones in scientific articles and two life science applications. *Bioinformatics*, 28(7):991–1000.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005a. Online large-margin training of dependency parsers. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 91–98, Ann Arbor, Michigan, June. Association for Computational Linguistics.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajic. 2005b. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 523–530, Vancouver, British Columbia, Canada, October. Association for Computational Linguistics.

Philippe Muller, Stergos Afantenos, Pascal Denis, and Nicholas Asher. 2012. Constrained decoding for text-level discourse parsing. In *Proceedings of COLING 2012*, pages 1883–1900, Mumbai, India, December. The COLING 2012 Organizing Committee.

Raquel Mochales Palau and Marie-Francine Moens. 2011. Argumentation mining. *Artificial Intelligence and Law*, 19(1):1–22.

Joonsuk Park and Claire Cardie. 2014. Identifying appropriate support for propositions in online user comments. In *Proceedings of the First Workshop on Argumentation Mining*, pages 29–38, Baltimore, Maryland, June. Association for Computational Linguistics.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Andreas Peldszus and Manfred Stede. 2013. From argument diagrams to automatic argument mining: A survey. *International Journal of Cognitive Informatics and Natural Intelligence (IJCINI)*, 7(1):1–31.

Andreas Peldszus and Manfred Stede. to appear. An annotated corpus of argumentative microtexts. In *Proceedings of the First European Conference on Argumentation: Argumentation and Reasoned Action*, Lisbon, Portugal, June 2015.

Andreas Peldszus. 2014. Towards segment-based recognition of argumentation structure in short texts. In *Proceedings of the First Workshop on Argumentation Mining*, Baltimore, U.S., June. Association for Computational Linguistics.

Christian Stab and Iryna Gurevych. 2014. Identifying argumentative discourse structures in persuasive essays. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 46–56, Doha, Qatar, October. Association for Computational Linguistics.

Manfred Stede. 2002. DiMLex: A Lexical Approach to Discourse Markers. In Vittorio Di Tomaso Alessandro Lenci, editor, *Exploring the Lexicon - Theory and Computation*. Edizioni dell'Orso, Alessandria, Italy.

Simone Teufel and Min-Yen Kan. 2011. Robust argumentative zoning for sensemaking in scholarly documents. In Raffaella Bernadi, Sally Chambers, Björn Gottfried, Frédérique Segond, and Ilya Zaihrayeu, editors, *Advanced Language Technologies for Digital Libraries*, volume 6699 of *Lecture Notes in Computer Science*, pages 154–170. Springer Berlin Heidelberg.

Simone Teufel and Marc Moens. 2002. Summarizing scientific articles: Experiments with relevance and rhetorical status. *Comput. Linguist.*, 28(4):409–445, December.

Simone Teufel. 2010. *The Structure of Scientific Articles: Applications to Citation Indexing and Summarization*. CSLI Studies in Computational Linguistics. CSLI Publications.

Douglas Walton, Chris Reed, and Fabrizio Macagno. 2008. *Argumentation Schemes*. Cambridge University Press.

Feature-Rich Two-Stage Logistic Regression for Monolingual Alignment

Md Arafat Sultan[†], **Steven Bethard**[‡] **and Tamara Sumner**[†]
[†]Institute of Cognitive Science and Department of Computer Science
University of Colorado Boulder
[‡]Department of Computer and Information Sciences
University of Alabama at Birmingham
arafat.sultan@colorado.edu, bethard@cis.uab.edu, sumner@colorado.edu

Abstract

Monolingual alignment is the task of pairing semantically similar units from two pieces of text. We report a top-performing supervised aligner that operates on short text snippets. We employ a large feature set to (1) encode similarities among semantic units (words and named entities) in context, and (2) address cooperation and competition for alignment among units in the same snippet. These features are deployed in a two-stage logistic regression framework for alignment. On two benchmark data sets, our aligner achieves F_1 scores of 92.1% and 88.5%, with statistically significant error reductions of 4.8% and 7.3% over the previous best aligner. It produces top results in extrinsic evaluation as well.

1 Introduction

Computer applications frequently require semantic comparison between short snippets of natural language text. Such comparisons are key to paraphrase detection (Das and Smith, 2009; Madnani et al., 2012), textual similarity identification (Agirre et al., 2015; Sultan et al., 2015) and recognition of textual entailment (Dagan and Glickman, 2004; Padó et al., 2015). And they underpin applications such as short answer grading (Mohler et al., 2011), question answering (Hixon et al., 2015), machine translation evaluation (Padó et al., 2009), and machine reading (de Marneffe et al., 2007).

A central problem underlying all text comparison tasks is that of *alignment*: pairing related semantic units (i.e. words and phrases) across the two snippets (MacCartney et al., 2008; Thadani and McKeown, 2011; Thadani et al., 2012; Yao et al., 2013a; Yao et al., 2013b; Sultan et al., 2014a). Studies have shown that such tasks can benefit from an explicit alignment component (Hickl and Bensley, 2007; Sultan et al., 2014b; Sultan et al., 2015). However, alignment is still an open research problem. We present a supervised monolingual aligner that produces top results in several intrinsic and extrinsic evaluation experiments. We pinpoint a set of key challenges for alignment and design a model with components targeted at each.

Lexical and phrasal alignments can both be represented as pairs of words – in the form of many-to-many mappings among the two phrases' component words in the latter case. Thus without loss of generality, we formulate alignment as a binary classification task where given all word pairs across two sentences, the goal is to assign each a class label in {*aligned*, *not aligned*}. However, this is not a straightforward classification scenario where each word pair can be treated independently – words in the same snippet can play both mutually cooperating and competing roles in complex ways. For example, semantically similar words in a snippet can be in competition for alignment with a word in the other snippet, whereas words that constitute a phrase can provide supporting evidence for one another (e.g. in named entity alignments such as `Watson` ↔ `John Hamish Watson`). To handle such interdependencies, we employ a two-stage logistic regression model – stage 1 computes an alignment probability for each word pair based solely on its own feature values, and stage 2 assigns the eventual alignment labels to all pairs following a comparative assessment of stage 1 probabilities of cooperating and competing pairs.

On two alignment data sets reported in (Brockett, 2007) and (Thadani et al., 2012), our aligner demonstrates respective F_1 scores of 92.1% and 88.5%, with statistically significant error reductions of 4.8% and 7.3% over the previous best aligner (Sultan et al., 2014a). We also present extrinsic evaluation of the aligner within two text comparison tasks, namely sentence similarity identification and paraphrase detection, where it demonstrates state-of-the-art results.

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 949–959,
Lisbon, Portugal, 17-21 September 2015.

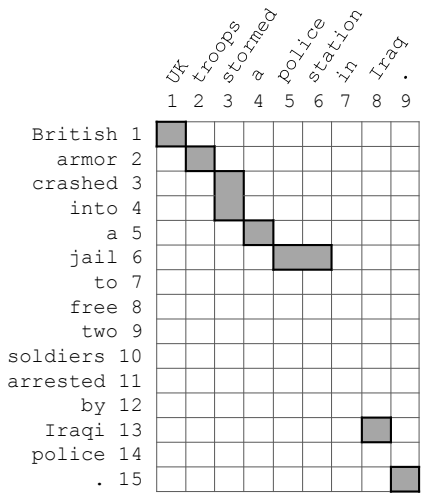


Figure 1: A human-aligned sentence pair from the MSR alignment corpus. The shaded cells depict the alignment, which can also be represented as the set of word index pairs $\{(1,1),(2,2),(3,3),(4,3),(5,4),\ldots,(15,9)\}$.

2 Alignment: Key Pieces of the Puzzle

We illustrate with examples key pieces of the alignment puzzle and discuss techniques used by existing aligners to solve them. We use the term 'unit' to refer to both words and phrases in a snippet.

Figure 1 shows a shortened version of sentence pair 712 in the MSR alignment corpus *dev* set (Brockett, 2007) with related units aligned by human annotators. Evident from these alignments is the fact that aligned units are typically semantically similar or related. Existing aligners utilize a variety of resources and techniques for computing similarity between units: WordNet (MacCartney et al., 2008; Thadani and McKeown, 2011), PPDB (Yao et al., 2013b; Sultan et al., 2014a), distributional similarity measures (MacCartney et al., 2008; Yao et al., 2013b) and string similarity measures (MacCartney et al., 2008; Yao et al., 2013a). Recent work on neural word embeddings (Mikolov et al., 2013; Baroni et al., 2014) have advanced the state of distributional similarity, but remain largely unexplored in the context of alignment.

Lexical or phrasal similarity does not entail alignment, however. Consider function words: the alignment $(5,4)$ in Figure 1 exists not just because both units are the word `a`, but also because they modify semantically equivalent units: `jail` and `police station`. The influence of context on content word alignment becomes salient particularly in the presence of competing words. In Figure 1, `(soldiers(10), troops(2))` are not aligned despite the two words' semantic equivalence *in isolation*, due to the presence of a competing pair, `(armor(2), troops(2))`, which is a better fit *in context*.

The above examples reveal a second aligner requirement: the ability to incorporate context into similarity calculations. Existing supervised aligners use various contextual features within a learning algorithm for this purpose. Such features include both shallow surface measures (e.g., the relative positions of the tokens being aligned in the respective sentences, similarities in the immediate left or right words) (MacCartney et al., 2008; Thadani and McKeown, 2011; Yao et al., 2013a) and syntactic measures like typed dependencies (Thadani and McKeown, 2011; Thadani et al., 2012). Sultan et al. (2014a) design an unsupervised model that more directly encodes context, via surface and dependency-based neighbors which allow contextual similarity to be represented as a weighted sum of lexical similarity. But their model lacks a key structural advantage of supervised models: to be able to use an arbitrarily large feature set to robustly encode lexical and/or contextual similarity.

The third and final key component of an aligner is a mechanism to combine lexical/phrasal and contextual similarities to produce alignments. This task is non-trivial due to the presence of cooperating and competing units. We first discuss competing units: semantically similar units in one snippet, each of which is a potential candidate for alignment with one or more units in the other snippet. At least three different possible scenarios of varying difficulty exist concerning such units:

- Scenario 1: No competing units. In Figure 1, the aligned pair `(British(1), UK(1))` represents this scenario.
- Scenario 2: Many-to-one competition: when multiple units in one snippet are similar to a single unit in the other snippet. In Figure 1, pairs `(armors(2), troops(2))` and `(soldiers(10), troops(2))` are in such competition.
- Scenario 3: Many-to-many competition: when similar units in one snippet have multiple potential alignments in the other snippet.

Groups of mutually cooperating units can also exist where one unit provides supporting evidence

for the alignment of other units in the group. Examples (besides named entities) include individual words in one snippet that are grouped together in the other snippet (e.g., `state of the art` ↔ `state-of-the-art` or `headquarters in Paris` ↔ `Paris-based`).

We briefly discuss the working principles of existing aligners to show how they respsond to these challenges. MacCartney et al. (2008), Thadani and McKeown (2011) and Thadani et al. (2012) frame alignment as a set of phrase edit (insertion, deletion and substitution) operations that transform one snippet into the other. Each edit operation is scored as a weighted sum of feature values (including lexical and contextual similarity features), and an optimal set of edits is computed. Yao et al. (2013a; 2013b) take a sequence labeling approach: input snippets are considered sequences of units and for each unit in one snippet, units in the other snippet are considered potential labels. A first order conditional random field is used for prediction. Sultan et al. (2014a) treat alignment as a bipartite matching problem and use a greedy algorithm to perform one-to-one word alignment. A weighted sum of two words' lexical and contextual similarities serves as the pair's edge weight.

Noticeable in the designs of the supervised aligners is a lack of attention to the scenarios competing units can pose – alignment of a unit depends only on its own feature values. While the unsupervised aligner by Sultan et al. (2014a) employs techniques to deal with such scenarios, it allows only one-to-one alignment, which fundamentally limits the set of reachable alignments.

3 Approach

We primarily focus on word alignment, which Yao et al. (2013b) report to cover more than 95% of all alignments in multiple human-annotated corpora. Named entities are the only phrasal units we consider for alignment; in a later section we discuss how our techniques can be extended to general phrasal alignment.

Figure 2 shows our two-stage logistic regression model. We address the first two challenges, namely identifying lexical and contextual similarities, in stage 1 of the model. Given input text snippets $\boldsymbol{T}^{(1)} = (T_1^{(1)}, ..., T_n^{(1)})$ and $\boldsymbol{T}^{(2)} = (T_1^{(2)}, ..., T_m^{(2)})$ where $T_k^{(t)}$ is the k-th word of snippet $\boldsymbol{T}^{(t)}$, the goal of this stage is to assign each word pair of the form $(T_i^{(1)}, T_j^{(2)})$ an alignment

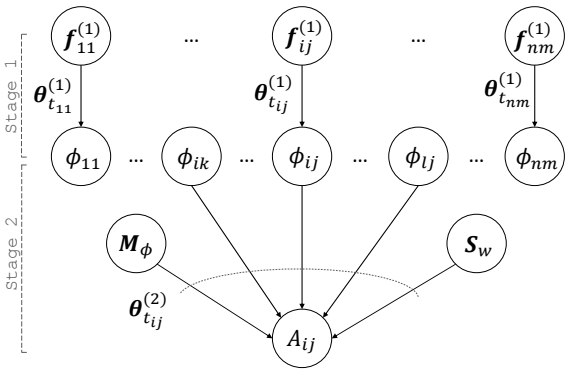


Figure 2: Two-stage logistic regression for alignment. Stage 1 computes an alignment probability ϕ_{ij} for each word pair based on local features $\boldsymbol{f}_{ij}^{(1)}$ and learned weights $\boldsymbol{\theta}_{t_{ij}}^{(1)}$ (see Section 4.1). Stage 2 assigns each pair a label $A_{ij} \in \{$*aligned*, *not aligned*$\}$ based on its own ϕ, the ϕ of its cooperating and competing pairs, a max-weighted bipartite matching $\boldsymbol{M}_\phi$ with all ϕ values as edge weights, the semantic similarities $\boldsymbol{S}_w$ of the pair's words and words in all cooperating pairs, and learned weights $\boldsymbol{\theta}_{t_{ij}}^{(2)}$ for these global features.

probability ϕ_{ij}, based on the pair's lexical and contextual similarity features. We discuss our stage 1 features in Section 4.1.

We categorize each word along two different dimensions: (1) whether or not it is part of a named entity, and (2) which of the following groups it belongs to: content words, function words, and punctuation marks. This distinction is important because, (1) certain features apply only to certain types of words (e.g., acronymy applies only to named entities; punctuation marks do not participate in dependency relationships), and (2) certain features can be more important for certain types of words (e.g., the role of a function word depends heavily on its surrounding words and therefore contextual features can be more important for function words). Combined, the two above dimensions form a domain of six possible values which can be represented as the Cartesian product $\{$*non-named entity*, *named entity*$\} \times \{$*content word*, *function word*, *punctuation mark*$\}$. Each member of this set is a *word type* in our model; for instance, *named entity function word* is a word type.

This notion of types is then extended to word pairs in $\boldsymbol{T}^{(1)} \times \boldsymbol{T}^{(2)}$: the type of pair $(T_i^{(1)}, T_j^{(2)})$ is the union of the types of $T_i^{(1)}$ and $T_j^{(2)}$. Given

the pair's stage 1 feature vector $\boldsymbol{f}_{ij}^{(1)}$ and the stage 1 weight vector $\boldsymbol{\theta}_{t_{ij}}^{(1)}$ for its type t_{ij}, we compute its stage 1 alignment probability ϕ_{ij} as:

$$\phi_{ij} = \frac{1}{1 + e^{-\boldsymbol{\theta}_{t_{ij}}^{(1)} \cdot \boldsymbol{f}_{ij}^{(1)}}}$$

The weight vector $\boldsymbol{\theta}_t^{(1)}$ for word pair type t is derived by minimizing the L1-regularized loss:

$$J(\boldsymbol{\theta}_t^{(1)}) = -\frac{1}{N_t} \sum_{p=1}^{N_t} \Big[y_t^{(p)} \log(\phi_t^{(p)}) + (1 - y_t^{(p)}) \log(1 - \phi_t^{(p)}) \Big] + \lambda \|\boldsymbol{\theta}_t^{(1)}\|_1$$

where N_t is the number of word pairs of type t over all sentence pairs in the training data, $y_t^{(p)}$ is the gold label for pair p of type t ($1 =$ *aligned*, $0 =$ *not aligned*), and $\phi_t^{(p)}$ is its stage 1 alignment probability.

Stage 2 of the model assigns the final alignment label $A_{ij} \in \{0, 1\}$ to $(T_i^{(1)}, T_j^{(2)})$. Like stage 1, it uses L1-regularized logistic regression to compute an alignment probability for each word pair, but additionally assigns a final $0/1$ label using a 0.5 threshold. Stage 2 factors in the stage 1 probabilities of cooperating and competing pairs as well as a maximum-weighted matching $\boldsymbol{M}_\phi$ between $\boldsymbol{T}^{(1)}$ and $\boldsymbol{T}^{(2)}$, where word pairs in $\boldsymbol{T}^{(1)} \times \boldsymbol{T}^{(2)}$ are weighted by their stage 1 ϕ values. Such global knowledge is useful in addressing cooperation and competition among words. We describe our stage 2 features in Section 4.2.

The two stages are trained separately, each as *n* standard logistic regression models where *n* is the number of word pair types for which at least one instance per class is observed in the training data. The stage 1 models are first trained and used to make predictions for each training sentence pair (for each training pair, all other training pairs are used to train the model). Given all the stage 1 alignment probabilities and the other stage 2 features, the stage 2 models are then trained. At test time, the two sets of trained models (i.e. stage 1 and 2 models) are successively applied to each input sentence pair.

4 Features

As mentioned above, we train a separate model for each individual word pair type. Our feature set is largely the same across word pair types, with some differences. In the two following sections, we discuss these features and indicate the associated word pair types. We assume alignment of the two words $T_i^{(1)} \in \boldsymbol{T}^{(1)}$ and $T_j^{(2)} \in \boldsymbol{T}^{(2)}$.

4.1 Stage 1: Assessing Pairs Individually

4.1.1 Word Similarity Features

Our first feature combines neural word embeddings, used previously for word similarity prediction (Mikolov et al., 2013; Baroni et al., 2014), with a paraphrase database (Ganitkevitch et al., 2013). Our feature is the output of a ridge regression model trained on human annotations of word similarity (Radinsky et al., 2011; Halawi et al., 2012; Bruni et al., 2014) with two features: the cosine similarity between the neural embedding vectors of the two words (using a publicly available set of 400-dimensional word vectors (Baroni et al., 2014)), and the presence/absence of the word pair in the PPDB XXXL database. This regression model produces similarities (sim henceforth) in $[0, 1]$, though we only consider similarities above 0.5 as lower scores are often noisy. To deal with single-letter spelling errors, we consider $T_i^{(1)}$ and $T_j^{(2)}$ to be an exact match if exactly one of the two is correctly spelled and their Levenshtein distance is 1 (words of length ≥ 3 only).

We also use the following semantic and string similarity features: a boolean feature that is 1 iff one of $T_i^{(1)}$ and $T_j^{(2)}$ is hyphenated and the other is identical to a hyphen-delimited part of the first, the same feature for highly similar ($sim \geq 0.9$) words, two features that show what proportion of the characters of one word is covered by the other if the latter is a prefix or a suffix of the former and zero otherwise (length < 3 words are discarded).

For named entities, we (1) consider acronymy as exact match, (2) use membership in two lists of alternative country names and country-nationality pairs (from Wikipedia) as features, and (3) include a feature that encodes whether $T_i^{(1)}$ and $T_j^{(2)}$ belong to the same named entity (determined by one mention containing all words of the other, e.g., `Einstein` and `Albert Einstein`).

4.1.2 Contextual Features

Effective identification of contextual similarity calls for a robust representation of word context in a sentence. Our contextual features are based on two different sentence representations. The word-

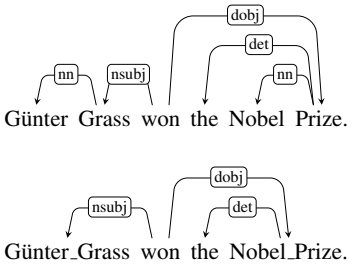


Figure 3: Word and entity-based representations of a sentence. Words in the same named entity are grouped together in the latter representation.

based representation treats each individual word as a semantic unit whereas the entity-based representation (1) groups together words in a multi-word named entity, and (2) treats non-name words as individual entities. Figure 3 shows an example. The two representations are complementary – the entity-based representation can capture equivalences between mentions of different lengths of a named entity, while the word-based representation allows the use of similarity resources for named entity words. Non-name words are treated identically. For simplicity we only discuss our word-based features below, but each feature also has an entity-based variant.

Dependency-based context. These features apply only if neither of $T_i^{(1)}$ and $T_j^{(2)}$ is a punctuation mark. We compute the proportion of identical and highly similar ($sim \geq 0.9$) parents and children of $T_i^{(1)}$ and $T_j^{(2)}$ in the dependency trees of $\boldsymbol{T}^{(1)}$ and $\boldsymbol{T}^{(2)}$ (Stanford collapsed dependencies (de Marneffe et al., 2006)). Equivalent dependency types (Sultan et al., 2014a) are included in the above computation, which encode semantic equivalences between typed dependencies (e.g., *nsubjpass* and *dobj*). We employ separate features for identicality and similarity. Similar features are also computed for a dependency neighborhood of size 2 (parents, grandparents, children and grandchildren), where we consider only content word neighbors.

Dependency neighbors of $T_i^{(1)}$ and $T_j^{(2)}$ that are less similar ($0.9 > sim \geq 0.5$; e.g., (`gas, energy`) or (`award, winner`)) can also contain useful semantic information for an aligner. To accommodate this relatively large range of word similarities, rather than counting such pairs, we find a maximum-weighted bipartite matching of $T_i^{(1)}$ and $T_j^{(2)}$ neighbors in a neighborhood of size 2 using the primal-dual algorithm (content words only), where word similarities across the two neighborhoods serve as edge weights. We use as a feature the sum of similarities between the matched neighbors, normalized by the total number of content words in the two neighborhoods.

Surface-form context. We draw several contextual features from nearby words of $T_i^{(1)}$ and $T_j^{(2)}$ in the surface forms of $\boldsymbol{T}^{(1)}$ and $\boldsymbol{T}^{(2)}$: (1) whether the left and/or the right word/lemma is identical , (2) whether the two are highly similar ($sim \geq 0.9$), (3) the longest common word/lemma sequence containing $T_i^{(1)}$ and $T_j^{(2)}$ such that at least one word in the sequence is a content word, (4) proportion of identical and highly similar ($sim \geq 0.9$) words in a neighborhood of 3 content words to the left and 3 content words to the right; we use two versions of this feature, one compares neighbors only in the same direction (i.e. left with left, right with right) and the other compares neighbors across the two directions, (5) similarly to dependency-based context, similarity in a max-weighted matching of all neighbors with $sim \in [0.5, 0.9)$ in the above $[-3, 3]$ window. For punctuation mark pairs, we use an additional feature indicating whether or not they both mark the end of their respective sentences.

4.2 Stage 2: Cooperation and Competition

We consider two groups of mutually cooperating words in a sentence: (1) words that belong to the same named entity, and (2) words in a sentence that are joined together to form a larger word in the other sentence (e.g., `state-of-the-art`). Speaking in terms of $T_i^{(1)}$, the goal is to be able to use any evidence present for a $(T_k^{(1)}, T_j^{(2)})$ alignment also as evidence for a $(T_i^{(1)}, T_j^{(2)})$ alignment if $T_i^{(1)}$ and $T_k^{(1)}$ both belong to such a group. We call $T_i^{(1)}$ and $T_k^{(1)}$ mutually cooperating words with respect to $T_j^{(2)}$ in such cases. Any word $T_l^{(1)} \in \boldsymbol{T}^{(1)}$ which is not a cooperating word for $T_i^{(1)}$ is a competing word: a word that can potentially make $(T_i^{(1)}, T_j^{(2)})$ a less viable alignment by having a larger stage 1 alignment probability in $(T_l^{(1)}, T_j^{(2)})$. We call a pair $(T_k^{(1)}, T_j^{(2)})$ a cooperating (competing) pair for $(T_i^{(1)}, T_j^{(2)})$ if $T_k^{(1)}$ is a cooperating (competing) word for $T_i^{(1)}$ with respect to $T_j^{(2)}$. With a reversal of word order and appropriate substitution of indexes, the above discussion

equally holds for $T_j^{(2)}$.

Given sets of stage 1 probabilities $\mathbf{\Phi}_{ij}^{cop}$ and $\mathbf{\Phi}_{ij}^{cmp}$ of cooperating and competing pairs for the pair $(T_i^{(1)}, T_j^{(2)})$, we employ three features to deal with scenario 2 of Section 2: (1) $\max(\phi_{ij}, \max(\mathbf{\Phi}_{ij}^{cop}))$: the greater of the pair's own stage 1 alignment probability and the highest among all cooperating pair probabilities, (2) $\max(\mathbf{\Phi}_{ij}^{cmp})$: the highest of all competing pair probabilities, and (3) a binary feature indicating which of the two above is larger.

To address scenario 3, we construct a weighted bipartite graph: nodes represent words in $\boldsymbol{T}^{(1)}$ and $\boldsymbol{T}^{(2)}$ and the weight of each edge represents the stage 1 alignment probability of a word pair in $\boldsymbol{T}^{(1)} \times \boldsymbol{T}^{(2)}$. We find a max-weighted bipartite matching $\boldsymbol{M}_\phi$ of word pairs in this graph. For each word pair, we employ a feature indicating whether or not it is in $\boldsymbol{M}_\phi$. The presence of $(T_i^{(1)}, T_j^{(2)})$ and $(T_k^{(1)}, T_l^{(2)})$ in $\boldsymbol{M}_\phi$, where all four words are similar, is a potential indicator that $(T_i^{(1)}, T_l^{(2)})$ and $(T_k^{(1)}, T_j^{(2)})$ are no longer viable alignments.

Low recall has traditionally been the primary weakness of supervised aligners (as we later show in Table 1). Our observation of the aligner's behavior on the *dev* set of the MSR alignment corpus (Brockett, 2007) suggests that this happens primarily due to highly similar word pairs being left unaligned even in the absence of competing pairs because of relatively low contextual evidence. Consequently, aligner performance suffers in sentences with few common or similar words. To promote high recall, we employ the higher of a word pair's own lexical similarity and the lexical similarity of the cooperating pair with the highest stage 1 probability as a stage 2 feature.

The stage 2 feature set is identical across word pair types, but as in stage 1, we train individual models for different pair types.

5 Experiments

5.1 System Evaluation

We report evaluation on two alignment data sets and extrinsic evaluation on two tasks: sentence similarity identification and paraphrase detection.

5.1.1 Alignment

We adopt the evaluation procedure for aligners reported in prior work (MacCartney et al., 2008; Thadani and McKeown, 2011; Yao et al., 2013a).

	Aligner	P %	R %	F_1 %	E %
MSR	MacCartney et al. (2008)	85.4	85.3	85.3	21.3
	Thadani & McKeown (2011)	89.5	86.2	87.8	33.0
	Yao et al. (2013a)	93.7	84.0	88.6	35.3
	Yao et al. (2013b)	92.1	82.8	86.8	29.1
	Sultan et al. (2014a)	93.7	**89.8**	91.7	43.8
	Our Aligner	**95.4**	89.0	**92.1**	**47.3**
EDB++	Thadani et al. (2012)	76.6	83.8	79.2	12.2
	Yao et al. (2013a)	91.3	82.0	86.4	15.0
	Yao et al. (2013b)	90.4	81.9	85.9	13.7
	Sultan et al. (2014a)	**93.5**	82.5	87.6	**18.3**
	Our Aligner	92.1	**85.2**	**88.5**	**18.3**

Table 1: Performance on two alignment data sets. Improvements in F_1 are statistically significant.

Data. The MSR alignment corpus (Brockett, 2007) contains 800 *dev* and 800 *test* sentence pairs from the PASCAL RTE 2006 challenge. Each pair is aligned by three human annotators; Fleiss Kappa agreement of about 0.73 ("substantial agreement") is reported on both sets. Following prior work, we only consider the *sure* alignments, take the majority opinion on each word pair, and leave out three-way disagreements.

The Edinburgh++ corpus (Thadani et al., 2012) contains 714 training and 306 test sentence pairs. Each test pair is aligned by two annotators and the final gold alignments consist of a random but even selection of the two sets of annotations.

Evaluation metrics. Our primary evaluation metrics are macro-averaged precision (*P*), recall (*R*) and F_1 score. A fourth metric E measures the proportion of sentence pairs for which the system alignments are identical to the gold alignments.

Model setup. For each corpus, we train our model using the dev set and evaluate on the test set. We use the logistic regression implementation of Scikit-learn (Pedregosa et al., 2011) and use leave-one-out cross-validation on the dev pairs to set the regularization parameter C.

Results. Table 1 shows the performance of different aligners on the two test sets. Our aligner demonstrates the best overall performance in terms of both F_1 and E. Wilcoxon signed-rank tests (with Pratt's treatment for zero-difference pairs) show that the improvements in F_1 over the previous best aligner (Sultan et al., 2014a) are statistically significant at $p < 0.01$ for both test sets.

5.1.2 Identification of Sentence Similarity

Given two input sentences, the goal in this task, known also as Semantic Textual Similarity (STS), is to output a real-valued semantic similarity score.

System	Pearson's r	Rank
Han et al. (2013)	**73.7**	1
Yao et al. (2013a)	46.2	66
Sultan et al. (2014a)	67.2	7
Our Aligner	67.8	4

Table 2: STS results. Performances of past systems are reported by Sultan et al. (2014a).

Data. To be able to directly compare with past aligners, we select three data sets (*headlines*: pairs of news headlines; *OnWN*, *FNWN*: gloss pairs) from the 2013 *SEM STS corpus (Agirre et al., 2013), containing 1500 sentence pairs in total. Sultan et al. (2014a) reports the performance of two state-of-the-art aligners on these pairs.

Evaluation metric. At SemEval, STS systems output a similarity score in $[0, 5]$. For each individual test set, the Pearson product-moment correlation coefficient (Pearson's r) is computed between system scores and human annotations. The final evaluation metric is a weighted sum of r's over all test sets, where the weight assigned to a set is proportional to its number of pairs.

Method. Being a logistic regression model, stage 2 of our aligner assigns each word pair an alignment probability. For STS, we compute a length-normalized sum of alignment probabilities of content word pairs across the two sentences. We include all pairs with probability > 0.5; the remaining pairs are included in decreasing order of their probabilities and already included words are ignored. Following (Sultan et al., 2014a), we normalize by dividing with the harmonic mean of the numbers of content words in the two sentences.

Results. Table 2 shows the performance of different aligners on the three STS 2013 test sets. We also show the performance of the contest-winning system (Han et al., 2013). Our STS system demonstrates a weighted correlation of 67.8%, which is better than similar STS systems based on the two previous best aligners. The difference with the next best aligner is statistically significant at $p < 0.05$ (two-sample one-tailed z-test). Overall, our system outperforms 86 of the 89 participating systems.

5.1.3 Paraphrase Detection

Given two input sentences, the goal in this task is to determine if their meanings are the same.

Data. The MSR paraphrase corpus (Dolan et al., 2004) contains 4076 *dev* and 1725 *test* sentence pairs; a paraphrase label (*true*/*false*) for each pair is provided by human annotators.

System	A %	P %	R %	F_1 %
Madnani et al. (2012)	**77.4**	**79.0**	89.9	**84.1**
Yao et al. (2013a)	70.0	72.6	88.1	79.6
Yao et al. (2013b)	68.1	68.6	**95.8**	79.9
Sultan et al. (2014a)	73.4	76.6	86.4	81.2
Our Aligner	73.2	75.3	88.8	81.5

Table 3: Paraphrase results. Performances of past systems are taken from (Sultan et al., 2014a).

Evaluation Metrics. We report performance in terms of: (1) accuracy in classifying the sentences into *true* and *false* classes (A), and (2) true class precision (P), recall (R) and F_1 score.

Method. Following prior aligners (MacCartney et al., 2008; Yao et al., 2013b; Sultan et al., 2014a), we output a *true* decision for a test sentence pair iff the length-normalized alignment score for the pair exceeds a threshold derived from the dev set.

Results. The top row of Table 3 shows the best result by any system on the MSR test set. Among all aligners (all other rows), ours achieves the best F_1 score and the second best accuracy.

We report paraphrase detection results primarily to allow comparison with past aligners. However, this simplistic application to a complex task only gives a ballpark estimate of an aligner's quality.

	Model	P %	R %	F_1 %	E %
MSR	Two-Stage Model	**95.4**	**89.0**	**92.1**	**47.3**
	Stage 1 Only	92.9	85.6	89.1	28.0
EDB++	Two-Stage Model	92.1	**85.2**	**88.5**	**18.3**
	Stage 1 Only	**93.0**	79.0	85.4	13.7

Table 4: Performance with and without stage 2.

5.2 Ablation

We perform ablation tests to find out how important (1) the two-stage framework, and (2) the different features are for our aligner.

5.2.1 Results without Stage 2

Stage 1 of our aligner can operate as an aligner by itself by mapping each alignment probability to a 0/1 alignment decision based on a threshold of 0.5. From a design perspective, this is an aligner that does not address scenarios 2 and 3 of Section 2.

The performance of the aligner with and without stage 2 is shown in Table 4. On each test set, the F_1 and E scores increase with the addition of stage 2. On the MSR test set, performance improves along all dimensions. On the Edinburgh++ test set, the

	MSR			EDB++		
Features	P %	R %	F_1 %	P %	R %	F_1 %
All Features	95.4	**89.0**	**92.1**	92.1	**85.2**	**88.5**
- Lexical	95.1	82.8	88.5	90.9	84.0	87.3
- Resources	**96.0**	87.0	91.3	**92.2**	84.3	88.1
- Contextual	89.0	79.2	83.9	89.9	66.3	76.3
- Dependency	95.3	88.2	91.6	91.9	84.9	88.3
- Surface	94.4	85.6	89.8	90.6	76.9	83.2
- Word-Based	94.6	87.7	91.0	92.0	85.1	88.4
- Entity-Based	95.5	**89.0**	**92.1**	92.1	85.1	**88.5**

Table 5: Results without different stage 1 features.

	MSR			EDB++		
Features	P %	R %	F_1 %	P %	R %	F_1 %
All Features	95.4	89.0	**92.1**	92.1	85.2	**88.5**
- ϕ values	87.3	**90.7**	88.9	86.4	**86.9**	86.7
- Matching	95.3	87.9	91.5	92.3	84.6	88.3
- Word Sim	**95.5**	88.4	91.8	**92.7**	84.7	**88.5**

Table 6: Results without different stage 2 features.

precision drops a little, but this effect is offset by a larger improvement in recall. These results show that stage 2 is central to the aligner's success.

5.2.2 Without Different Stage 1 Features

We exclude different stage 1 features (which fall into one of two groups: lexical and contextual) and examine the resulting model's performance. Table 5 shows the results. The subtraction sign represents the exclusion of the corresponding feature.

Without any lexical feature (i.e., if the model relies only on contextual features), both precision and recall decrease, resulting in a considerable overall performance drop. Exclusion of word similarity resources (i.e. embeddings and PPDB) improves precision, but again harms overall performance.

Without any contextual features, the model suffers badly in both precision and recall. The extreme overall performance degradation indicates that contextual features are more important for the aligner than lexical features. Leaving out surface-form neighbors results in a larger performance drop than when dependency-based neighbors are excluded, pointing to a more robust role of the former group in representing context. Finally, our entity-based representation of context neither helps nor harms system performance, but relying only on entity-based neighbors has detrimental effects. Factoring in semantic similarities of named entities should improve the utility of these features.

5.2.3 Without Different Stage 2 Features

Table 6 shows the aligner's performance after the exclusion of different stage 2 features. Leaving out the stage 1 alignment probabilities harms overall performance the most by causing a large drop in precision. Exclusion of the maximum-weighted bipartite matching feature results in worse recall and overall performance. The lexical similarity feature improves overall results only on the MSR test set but increases recall on both test sets.

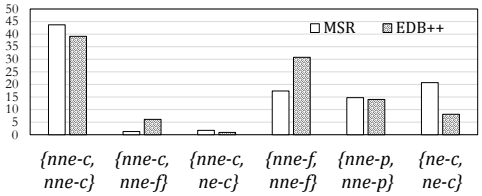


Figure 4: % distribution of aligned word pair types; *nne*: non-named entity, *ne*: named entity, *c*: content word, *f*: function word, *p*: punctuation mark.

	MSR			EDB++		
Pair Type	P %	R %	F_1 %	P %	R %	F_1 %
{*nne-c*, *nne-c*}	95.7	84.3	89.7	92.2	89.2	90.7
{*nne-c*, *nne-f*}	100.0	2.7	5.3	61.4	7.7	13.6
{*nne-c*, *ne-c*}	89.2	66.7	76.3	71.9	43.4	54.1
{*nne-f*, *nne-f*}	90.7	86.0	88.3	93.4	86.5	89.8
{*nne-p*, *nne-p*}	99.4	99.2	99.3	93.0	91.5	92.2
{*ne-c*, *ne-c*}	96.2	97.8	97.0	90.9	94.2	92.6

Table 7: Performance on different word pair types.

5.3 Error Analysis

We examine the aligner's performance on different word pair types. Figure 4 shows the % distribution of word pair types with at least 20 aligned instances in at least one test set. These six types account for more than 99% of all alignments in both test sets.

Table 7 shows the results. We ignore punctuation mark pairs in the following discussion. Performance is worst on the two rarest types: {*nne-c*, *nne-f*} and {*nne-c*, *ne-c*}, due primarily to very low recall. A relatively low availability of positive examples in the training sets (in the hundreds, in contrast to thousands of examples for each of the other three types) is a primary factor affecting classifier performance on these two pair types. The {*nne-c*, *nne-f*} pairs, nonetheless, are intrinsically the most difficult type for a word aligner because they occur frequently as part of phrasal alignments. On {*nne-c*, *ne-c*} pairs, errors also occur due to failure in recognition of certain named entity types (e.g., acronyms and multiword named entities) and the aligner's lack of world knowledge (e.g., in `daughter` ↔ `Chelsea`).

Low recall remains the primary issue, albeit to a much lesser extent, for {*nne-c*, *nne-c*} and {*nne-f*, *nne-f*} pairs. For the former, two major sources of error are: (1) inability to utilize contextual evidence outside the local neighborhood examined by the aligner, and (2) failure to address one-to-many alignments. Low recall for the latter follows naturally, as function word alignment is heavily dependent on related content word alignment. On {*ne-c*, *ne-c*} pairs the aligner performs the best, but still suffers from the two above issues.

6 Related Work

We mentioned major standalone monolingual aligners and briefly discussed their working principles in Section 2. There are, however, at least two additional groups of related work which can inform future research on monolingual alignment. First, alignment is often performed in the context of extrinsic tasks, e.g., textual entailment recognition (Wang and Manning, 2010), question answering (Heilman and Smith, 2010), discourse generation (Roth and Frank, 2012) and redundancy detection (Thadani and McKeown, 2008). Such systems may contain useful design elements yet to be utilized by standalone aligners. Second, a large body of work exists in the bilingual alignment literature (Och and Ney, 2003; Blunsom and Cohn, 2006; Chang et al., 2014), elements of which (such as the machine learning models) can be useful for monolingual aligners (see (Yao et al., 2013a) for an example).

7 Conclusions and Future Work

We present a two-stage classification framework for monolingual alignment that demonstrates top results in intrinsic and extrinsic evaluation experiments. While our work focuses primarily on word alignment, given a mechanism to compute phrasal similarity, the notion of cooperating words can be exploited to extend our model for phrasal alignment. Another important future direction is the construction of a robust representation of context, as our model currently utilizes contextual information only within a local neighborhood of a predefined size and therefore fails to utilize long-distance semantic relationships between words. Incorporating a single background model which is trained on all word pair types might also improve performance, especially on types that are rare in the training data. Finally, studying the explicit requirements of different extrinsic tasks can shed light on the design of a robust aligner.

Acknowledgments

This material is based in part upon work supported by the National Science Foundation under Grant Numbers EHR/0835393 and EHR/0835381.

References

Eneko Agirre, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. 2013. *SEM 2013 Shared Task: Semantic Textual Similarity. In *Proceedings of the Second Joint Conference on Lexical and Computational Semantics*, *SEM '13, pages 32-43, Atlanta, Georgia, USA.

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Iñigo Lopez-Gazpio, Montse Maritxalar, Rada Mihalcea, German Rigau, Larraitz Uria, and Janyce Wiebe. 2015. SemEval-2015 Task 2: Semantic Textual Similarity, English, Spanish and Pilot on Interpretability. In *Proceedings of the 9th International Workshop on Semantic Evaluation*, pages 252-263, Denver, Colorado, USA.

Marco Baroni, Georgiana Dinu, and Germán Kruszewski. 2014. Don't Count, Predict! A Systematic Comparison of Context-Counting vs. Context-Predicting Semantic Vectors. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, ACL '14, pages 238-247, Baltimore, Maryland, USA.

Phil Blunsom and Trevor Cohn. 2006. Discriminative Word Alignment with Conditional Random Fields. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL*, pages 65-72, Sydney, Australia.

Chris Brockett. 2007. Aligning the RTE 2006 Corpus. Technical Report MSR-TR-2007-77, Microsoft Research.

Elia Bruni, Nam Khanh Tran, and Marco Baroni. 2014. Multimodal Distributional Semantics. *Journal of Artificial Intelligence Research*, vol. 49, pages 1-47.

Yin-Wen Chang, Alexander M. Rush, John DeNero, and Michael Collins. 2014. A Constrained Viterbi Relaxation for Bidirectional Word Alignment. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 14811490, Baltimore, Maryland, USA.

Ido Dagan and Oren Glickman. 2004. Probabilistic Textual Entailment: Generic Applied Modeling of Language Variability. In *Proceedings of the PASCAL Workshop on Learning Methods for Text Understanding and Mining*, Grenoble, France.

Dipanjan Das and Noah A. Smith. 2009. Paraphrase Identification as Probabilistic Quasi-Synchronous Recognition. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 468-476, Singapore.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating Typed Dependency Parses from Phrase Structure Parses. In *Proceedings of the International Conference on Language Resources and Evaluation*, pages 449-454, Genoa, Italy.

Marie-Catherine de Marneffe, Trond Grenager, Bill MacCartney, Daniel Cer, Daniel Ramage, Chlo Kiddon, and Christopher D. Manning. 2007. Aligning Semantic Graphs for Textual Inference and Machine Reading. In *Proceedings of the AAAI Spring Symposium*, pages 468-476, Stanford, California, USA.

Bill Dolan, Chris Quirk, and Chris Brockett. 2004. Unsupervised Construction of Large Paraphrase Corpora: Exploiting Massively Parallel News Sources. In *Proceedings of the International Conference on Computational Linguistics*, pages 350-356, Geneva, Switzerland.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: The Paraphrase Database. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics*, pages 758-764, Atlanta, Georgia, USA.

Guy Halawi, Gideon Dror, Evgeniy Gabrilovich, and Yehuda Koren. 2012. Large-Scale Learning of Word Relatedness with Constraints. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1406-1414, Beijing, China.

Lushan Han, Abhay Kashyap, Tim Finin, James Mayfield, and Jonathan Weese. 2013. UMBC EBIQUITY-CORE: Semantic Textual Similarity Systems. In *Proceedings of the Second Joint Conference on Lexical and Computational Semantics*, *SEM '13, pages 44-52, Atlanta, Georgia, USA.

Michael Heilman and Noah A. Smith. 2010. Tree Edit Models for Recognizing Textual Entailments, Paraphrases, and Answers to Questions. In *Proceedings of the 2010 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 10111019, Los Angeles, California, USA.

Andrew Hickl and Jeremy Bensley. 2007. A Discourse Commitment-Based Framework for Recognizing Textual Entailment. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pages 171-176, Prague, Czech Republic.

Ben Hixon, Peter Clark, and Hannaneh Hajishirzi. 2015. Learning Knowledge Graphs for Question Answering through Conversational Dialog. In *Proceedings of the the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Denver, Colorado, USA.

Bill MacCartney, Michel Galley, and Christopher D. Manning. 2008. A Phrase-Based Alignment Model for Natural Language Inference. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 802-811, Honolulu, Hawaii, USA.

Nitin Madnani, Joel Tetreault, and Martin Chodorow. 2012. Re-examining Machine Translation Metrics for Paraphrase Identification. In *Proceedings of 2012 Conference of the North American Chapter of the Association for Computational Linguistics*, pages 182-190, Montreal, Canada.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. In *Proceedings of the International Conference on Learning Representations Workshop*, Scottsdale, Arizona, USA.

Michael Mohler, Razvan Bunescu, and Rada Mihalcea. 2011. Learning to Grade Short Answer Questions Using Semantic Similarity Measures and Dependency Graph Alignments. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, pages 752-762, Portland, Oregon, USA.

Franz Josef Och and Hermann Ney. 2003. A Systematic Comparison of Various Statistical Alignment Models. *Computational Linguistics*, 29(1):1951, MIT Press.

Sebastian Padó, Michel Galley, Dan Jurafsky, and Chris Manning. 2009. Robust Machine Translation Evaluation with Entailment Features. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 297-305, Singapore.

Sebastian Padó, Tae-Gil Noh, Asher Stern, Rui Wang, and Roberto Zanoli. 2015. Design and Realization of a Modular Architecture for Textual Entailment. *Natural Language Engineering*, 21 (2), pages 167-200, Cambridge University Press.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, vol. 12, pages 2825-2830.

Kira Radinsky, Eugene Agichtein, Evgeniy Gabrilovich, and Shaul Markovitch. 2011. A Word at a Time: Computing Word Relatedness using Temporal Semantic Analysis. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 337-346, Hyderabad, India.

Michael Roth and Anette Frank. 2012. Aligning Predicates across Monolingual Comparable Texts using Graph-based Clustering. In *Proceedings of the 20th International Conference on World Wide Web*, pages pages 171182, Jeju Island, Korea.

Md Arafat Sultan, Steven Bethard, and Tamara Sumner. 2014a. Back to Basics for Monolingual Alignment: Exploiting Word Similarity and Contextual Evidence. *Transactions of the Association for Computational Linguistics*, 2 (May), pages 219-230.

Md Arafat Sultan, Steven Bethard, and Tamara Sumner. 2014b. DLS@CU: Sentence Similarity from Word Alignment. In *Proceedings of the 8th International Workshop on Semantic Evaluation*, pages 241-246, Dublin, Ireland.

Md Arafat Sultan, Steven Bethard, and Tamara Sumner. 2015. DLS@CU: Sentence Similarity from Word Alignment and Semantic Vector Composition. In *Proceedings of the 9th International Workshop on Semantic Evaluation*, pages 148-153, Denver, Colorado, USA.

Kapil Thadani and Kathleen McKeown. 2008. A Framework for Identifying Textual Redundancy. In *Proceedings of the 22nd International Conference on Computational Linguistics*, pages 873880, Manchester, UK.

Kapil Thadani and Kathleen McKeown. 2011. Optimal and Syntactically-Informed Decoding for Monolingual Phrase-Based Alignment. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, pages 254-259, Portland, Oregon, USA.

Kapil Thadani, Scott Martin, and Michael White. 2012. A Joint Phrasal and Dependency Model for Paraphrase Alignment. In *Proceedings of COLING 2012*, pages 1229-1238, Mumbai, India.

Mengqiu Wang and Christopher D. Manning. 2010. Probabilistic Tree-Edit Models with Structured Latent Variables for Textual Entailment and Question Answering. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 11641172, Beijing, China.

Xuchen Yao, Benjamin Van Durme, Chris Callison-Burch, and Peter Clark. 2013a. A Lightweight and High Performance Monolingual Word Aligner. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 702-707, Sofia, Bulgaria.

Xuchen Yao, Benjamin Van Durme, Chris Callison-Burch, and Peter Clark. 2013b. Semi-Markov Phrase-based Monolingual Alignment. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 590-600, Seattle, Washington, USA.

Semantic Role Labeling with Neural Network Factors

Nicholas FitzGerald[‡*] **Oscar Täckström**[†] **Kuzman Ganchev**[†] **Dipanjan Das**[†]
[‡]Department of Computer Science and Engineering, University of Washington
[†] Google, New York
nfitz@cs.uw.edu
{oscart,kuzman,dipanjand}@google.com

Abstract

We present a new method for semantic role labeling in which arguments and semantic roles are jointly embedded in a shared vector space for a given predicate. These embeddings belong to a neural network, whose output represents the potential functions of a graphical model designed for the SRL task. We consider both local and structured learning methods and obtain strong results on standard PropBank and FrameNet corpora with a straightforward product-of-experts model. We further show how the model can learn jointly from PropBank and FrameNet annotations to obtain additional improvements on the smaller FrameNet dataset.

1 Introduction

Semantic role labeling (SRL) is the task of identifying the semantic arguments of a predicate and labeling them with their semantic roles. A key challenge in this task is sparsity of labeled data: a given predicate-role instance may only occur a handful of times in the training set. Most existing SRL systems model each semantic role as an atomic unit of meaning, ignoring finer-grained semantic similarity between roles that can be leveraged to share context between similar labels, both within and across annotation conventions.

Low-dimensional embedding representations have been shown to be successful in overcoming sparsity and representing label similarity across a wide range of tasks (Weston et al., 2011; Srikumar and Manning, 2014; Hermann et al., 2014; Lei et al., 2015). In this paper, we present a new model for SRL that embeds candidate arguments and semantic roles (in context of a predicate frame) in a shared vector space. A feed-forward neural network is learned to capture correlations of the respective embedding dimensions to create argument and role representations. The similarity of these two representations, as measured by their dot product, is used to score possible roles for candidate arguments within a graphical model. This graphical model jointly models the assignment of semantic roles to all arguments of a predicate, subject to structural linguistic constraints.

Our model has several advantages. Compared to linear multiclass classifiers used in prior work, vector embeddings of the predictions overcome the assumption of modeling each semantic role as a discrete label, thus capturing fine-grained label similarity. Moreover, since predictions and inputs are embedded in the same vector space, and features extracted from inputs and outputs are decoupled, our approach is amenable to joint learning of multiple annotation conventions, such as PropBank and FrameNet, in a single model. Finally, as with other neural network approaches, our model obviates the need to manually engineer feature conjunctions.

Our underlying inference algorithm for SRL follows Täckström et al. (2015), who presented a dynamic program for structured SRL; it is targeted towards the prediction of full argument spans. Hence, we present empirical results on three span-based SRL datasets: CoNLL 2005 and 2012 data annotated with PropBank conventions, as well as FrameNet 1.5 data. We also evaluate our system on the dependency-based CoNLL 2009 shared task by assuming single word argument spans, that represent semantic dependencies, and limit our experiments to English. On all datasets, our model performs on par with a strong linear model baseline that uses hand-engineered conjunctive features. Due to random parameter initialization and stochasticity in the online learning algorithm used to train our models, we observed considerable variance in performance across datasets. To resolve this variance, we adopt a product-of-experts model that

[*]Work carried out during an internship at Google.

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 960–970,
Lisbon, Portugal, 17-21 September 2015.

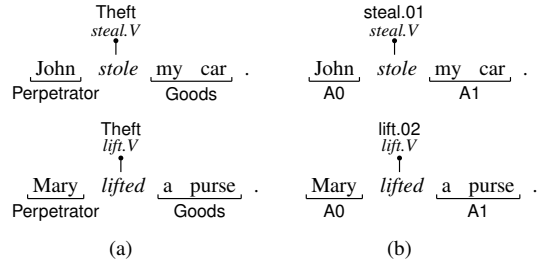


Figure 1: FrameNet (a) and PropBank (b) annotations for two sentences.

combines multiple randomly-initialized instances of our model to achieve state-of-the-art results on the CoNLL 2009 and FrameNet datasets, while coming close to the previous best published results on the other two. Finally, we present even stronger results for FrameNet data (which is scarce) by jointly training the model with PropBank-annotated data.

2 Background

In this section, we briefly describe the SRL task and discuss relevant prior work.

2.1 Semantic Role Labeling

SRL annotations rely on a frame lexicon containing *frames* that could be evoked by one or more *lexical units*. A lexical unit consists of a word lemma conjoined with its coarse-grained part-of-speech tag.[1] Each frame is further associated with a set of possible *core* and *non-core* semantic roles which are used to label its arguments. This description of a frame lexicon covers both PropBank and FrameNet conventions, but there are some differences outlined below. See Figure 1 for example annotations.

PropBank defines frames that are essentially sense distinctions of a given lexical unit. The set of PropBank roles consists of seven generic core roles (labeled A0-A5 and AA) that assume different semantics for different frames, each associating with a subset of the core roles. In addition, there are 21 non-core roles that encapsulate further arguments of a frame, such as temporal (AM-TMP) and locative (AM-LOC) adjuncts. The non-core roles are shared between all frames and assume similar meaning. In contrast, a FrameNet frame often associates with multiple lexical units and the frame lexicon contains several hundred core and non-core roles that are shared across frames. For example, the FrameNet frame Theft could be evoked by the verbs *steal*, *pickpocket*, or *lift*, while PropBank has distinct frames for each of them. The Theft frame also contains the core roles Goods and Perpetrator that additionally belong to the Commercial_transaction and Committing_crime frames respectively.

A typical SRL dataset consists of sentence-level annotations that identify (possibly multiple) target predicates in each sentence, a disambiguated frame for each predicate, and the associated argument spans (or single word argument heads) labeled with their respective semantic roles.

2.2 Related Work

SRL using PropBank conventions (Palmer et al., 2005) has been widely studied. There have been two shared tasks at CoNLL 2004-2005 to identify the phrasal arguments of verbal predicates (Carreras and Màrquez, 2004; Carreras and Màrquez, 2005). The CoNLL 2008-2009 shared tasks introduced a variant where semantic dependencies are annotated rather than phrasal arguments (Surdeanu et al., 2008; Hajič et al., 2009). Similar approaches (Das et al., 2014; Hermann et al., 2014) have been applied to frame-semantic parsing using FrameNet conventions (Baker et al., 1998). We treat PropBank and FrameNet annotations in a common framework, similar to Hermann et al. (2014).

Most prior work on SRL rely on syntactic parses provided as input and use locally estimated classifiers for each span-role pair that are only combined at prediction time.[2] This is done by picking the highest scoring role for each span, subject to a set of structural constraints, such as avoiding overlapping arguments and repeated core roles. Typically, these constraints have been enforced by integer linear programming (ILP), as in Punyakanok et al. (2008). Täckström et al. (2015) interpreted this as a graphical model with local factors for each span-role pair, and global factors that encode the structural constraints. They derived a dynamic program (DP) that enforces most of the constraints proposed by Punyakanok et al. and showed how the DP can be used to take these constraints into account during learning. Here, we use an identical graphical model, but extend the model of Täckström et al. by replacing its linear potential func-

[1] We borrow the term "lexical unit" from the frame semantics literature. The CoNLL 2005 dataset is restricted to verbal lexical units, while the CoNLL 2009 and 2012 datasets contains both verbal and nominal lexical units. FrameNet has lexical units of several coarse syntactic categories.

[2] Some recent work have successfully proposed joint models for syntactic parsing and SRL instead of a pipeline approach (Lewis et al., 2015).

tions with a multi-layer neural network. A similar use of non-linear potential functions in a structured model was proposed by Do and Artières (2010) for speech recognition, and by Durrett and Klein (2015) for syntactic phrase-structure parsing.

Feature-based approaches to SRL employ hand-engineered linguistically-motivated feature templates to represent the semantic structure. Some recent work has focused on low-dimensional representations that reduce the need for intensive feature engineering and lead to better generalization in the face of data sparsity. Lei et al. (2015) employ low-rank tensor factorization to induce a compact representation of the full cross-product of atomic features; akin to this work, they represent semantic roles as real-valued vectors, but use a different scoring formulation for modeling potential arguments. Moreover, they restrict their experiments to CoNLL 2009 semantic dependencies. Roth and Woodsend (2014) improve on the state-of-the-art feature-based system of Björkelund et al. (2010) by adding distributional word representations trained on large unlabeled corpora as features.

Collobert and Weston (2007) use a neural network and do not rely on syntactic parses as input. While they use non-standard evaluation, they report accuracy similar to the ASSERT system (Pradhan et al., 2005), to which we compare in Table 4. Very recently, Zhou and Xu (2015) proposed a deep bidirectional LSTM model for SRL that does not rely on syntax trees as input; their approach achieves the best results on CoNLL 2005 and 2012 corpora to date, but unlike this work, they do not report results on FrameNet and CoNLL 2009 dependencies and do not investigate joint learning approaches involving multiple annotation conventions.

For FrameNet-style SRL, Kshirsagar et al. (2015) recently proposed the use of PropBank-based features, but their system performance falls short of the state of the art. Roth and Lapata (2015) proposed another approach exploring linguistically motivated features tuned towards the FrameNet lexicon, but their performance metrics are significantly worse than our best results.

The inspiration behind our approach stems from recent work on bilinear models (Mnih and Hinton, 2007). There have been several recent studies representing input observations and output labels with distributed representations, for example, in the WSABIE model for image annotation (Weston et al., 2011), in models for embedding labels in structured graphical models (Srikumar and Manning, 2014), and in techniques to learn joint embeddings of predicate words and their semantic frames in a vector space (Hermann et al., 2014).

3 Model

Our model for SRL performs inference separately for each marked predicate in a sentence. It assumes that the predicate has been automatically disambiguated to a semantic frame drawn from a frame lexicon, and the semantic roles of the frame are used for labeling the candidate arguments in the sentence. Formally, we are given a sentence x in which a predicate t, with lexical unit ℓ, has been marked. Assuming that the semantic frame f of the predicate has already been identified (see §4.2 for this step), we seek to predict the set of spans that correspond to its overt semantic arguments and to label each argument with its semantic role. Specifically, we model the problem as that of assigning each span $s \in \mathcal{S}$, from an over-generated set of candidate argument spans $\mathcal{S}$, to a semantic role $r \in \mathcal{R}$. The set of semantic roles $\mathcal{R}$ includes the special null role $\varnothing$, which is used to represent non-overt arguments. Thus, our algorithm performs the SRL task in one step for a single predicate frame. For the span-based SRL task, in a sentence of n words, there could be $O(n^2)$ potential arguments. For statistical and computational reasons we prune the set of spans $\mathcal{S}$ using a set of syntactically-informed heuristics from prior work (see §4.2).

3.1 Graphical Model

We make use of a graphical model that represents global assignment of arguments to their semantic roles, subject to linguistic constraints (Punyakanok et al., 2008; Täckström et al., 2015). Under this graphical model, we assume a parameterized potential function that assigns a real-valued compatibility score $g(s, r; \boldsymbol{\theta})$ to each span-role pair $(s, r) \in \mathcal{S} \times \mathcal{R}$, where $\boldsymbol{\theta}$ denotes the model parameters. Below, we consider two types of potential functions. As a baseline, similar to most prior work, one could use a simple linear function of discrete input features $g_{\text{L}}(s, r; \boldsymbol{\theta}) = \boldsymbol{\theta}^\top \cdot \boldsymbol{\phi}(r, s, x, t, \ell, f)$, where $\boldsymbol{\phi}(\cdot)$ denotes a feature function. In this work, we instead propose a multi-layer feed-forward neural network potential function, specified in §3.2. Given these local factors, we employ the dynamic program of Täckström et al. to enforce global constraints on the inferred output.

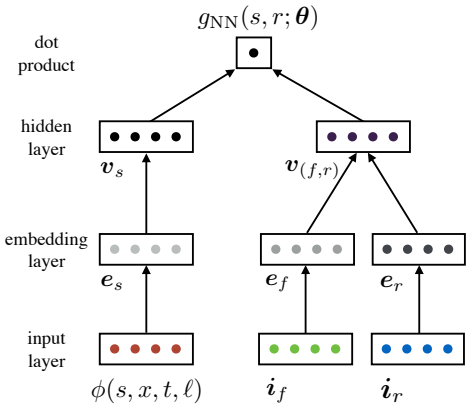


Figure 2: Neural network architecture.

• first word of s	• tag of the first word of s
• last word of s	• tag of the last word of s
• head word of s	• tag of the head word of s
• bag of words in s	• bag of tags in s
• cluster of s's head	• linear *distance* of s from t
• t's children words	• word cluster of s's head
• dependency path between s's head and t	
• subcategorization frame of s	
• *position* of s w.r.t. t (*before, after, overlap or same*)	
• predicate use voice (*active*, *passive*, or *unknown*)	
• whether the subject of t is missing (*missingsubj*)	
• *position* of s w.r.t. t (*before, after, overlap or same*)	
• word, tag, dependency label and cluster of the words immediately to the left and right of s	

Table 1: Span features $\boldsymbol{\phi}(s, x, t, \ell)$ in Figure 2.

Let $\mathcal{R}^{|\mathcal{S}|}$ denote the set of all possible assignments of semantic roles to argument spans (s_i, r_i) for $s_i \in \mathcal{S}$ that satisfy the constraints. Given a potential function $g(s, r) \triangleq g(s, r; \boldsymbol{\theta})$, the probability of a joint assignment $\boldsymbol{r} \in \mathcal{R}^{|\mathcal{S}|}$, subject to the constraints, is given by

$$p(\boldsymbol{r} \mid x, t, \ell, f) = \exp\left(\sum_{s_i \in \mathcal{S}} g(s_i, r_i) - A(\mathcal{S})\right), \quad (1)$$

where the log-partition function $A(\mathcal{S})$ sums over all satisfying joint role assignments:

$$A(\mathcal{S}) = \log \sum_{\boldsymbol{r}' \in \mathcal{R}^{|\mathcal{S}|}} \exp\left(\sum_{s_i \in \mathcal{S}} g(s_i, r_i')\right). \quad (2)$$

3.2 Neural Network Potentials

Our approach replaces the standard linear potential function $g_{\text{L}}(s, r; \boldsymbol{\theta})$ with the real-valued output of a feed forward neural network with non-linear hidden units. The network structure is outlined in Figure 2. The frame f and role r are initially encoded using a one-hot encoding as $\boldsymbol{i}_f$ and $\boldsymbol{i}_r$. In other words, $\boldsymbol{i}_f$ and $\boldsymbol{i}_r$ have all zeros except for one position at f and r respectively. These are passed through fully connected linear layers to give $\boldsymbol{e}_f$ and $\boldsymbol{e}_r$. We call these linear layers the *embedding* layers since $\boldsymbol{i}_f$ selects the embedding of the frame f and $\boldsymbol{i}_r$ for r. Next, $\boldsymbol{e}_f$ and $\boldsymbol{e}_r$ are passed through a fully connected rectified linear layer (Nair and Hinton, 2010), to obtain the final frame-role representation $\boldsymbol{v}_{(f,r)}$. For the candidate span, the process is similar. Atomic features $\boldsymbol{\phi}(s, x, t, \ell)$ for the argument span s are extracted first. (These features are the non-conjoined features used in the linear model of Täckström et al.; see Table 1 for the list). These are next passed through a fully-connected linear embedding layer to get the span embedding $\boldsymbol{e}_s$, which is subsequently passed through a fully connected rectified linear layer to obtain $\boldsymbol{v}_s$, the final span representation. The final output is the dot product of $\boldsymbol{v}_s$ and $\boldsymbol{v}_{(f,r)}$:

$$g_{\text{NN}}(s, r; \boldsymbol{\theta}) = \boldsymbol{v}_s^\top \cdot \boldsymbol{v}_{(f,r)}. \quad (3)$$

The weights of all the layers constitute the parameters $\boldsymbol{\theta}$ of the neural network. We initialize $\boldsymbol{\theta}$ randomly, with the exception of embedding parameters corresponding to words, which are initialized from pre-trained word embeddings (see §4.4 for details). We train the network as described in §3.3.[3]

Note that unlike typical linear models, the atomic span features are not explicitly conjoined with each other, the frame or the role. Instead the hidden layers learn to emulate span feature conjunctions and frame and role feature conjunctions in parallel.[4] Moreover, note that span $\boldsymbol{v}_s$ and frame-role $\boldsymbol{v}_{(f,r)}$ representations are decoupled in this model. This decoupling is important as it allows us to train a single model in a multitask setting. We demonstrate this by successfully combining PropBank and FrameNet training data, as described in §5.

3.3 Parameter Estimation

We consider two methods for parameter estimation.

[3] Various other network structures are worth investigating, such as concatenating the span, frame and role representations and passing them through fully connected layers. This treatment, for example, has been used by Chen and Manning (2014) for syntactic parsing. We leave these explorations to future work.

[4] We found that adding feature conjunctions to the network's input layer did not improve performance in practice.

Local Estimation In local estimation, we treat each span-role assignment pair $(s, r) \in \mathcal{S} \times \mathcal{R}$ as an individual binary decision problem and maximize the corresponding log-likelihood of the training set.[5] Denote by $z_{s,r} \in \{0, 1\}$ the decision variable, such that $z_{s,r} = 1$ iff span s is assigned role r. By passing the potential $g_{\text{NN}}(s, r; \boldsymbol{\theta})$ through the logistic function, we obtain the log-likelihood $l(z_{s,r}; \boldsymbol{\theta}) \triangleq \log p(z_{s,r} \mid x, t, \ell, f)$ of an individual training example. Here,

$$p(z_{s,r} \mid x, t, \ell, f) = \begin{cases} \frac{1}{1+e^{-g_{\text{NN}}(s,r;\boldsymbol{\theta})}} & \text{if } z_{s,r} = 1 \\ \frac{e^{-g_{\text{NN}}(s,r;\boldsymbol{\theta})}}{1+e^{-g_{\text{NN}}(s,r;\boldsymbol{\theta})}} & \text{if } z_{s,r} = 0 \end{cases}$$

Thus, the gold role for a given span according to the training data serves as the positive example, while all the other potential roles serve as negatives. To maximize the log-likelihood, we use Adagrad (Duchi et al., 2011). This requires the gradient of the log-likelihood with respect to the parameters $\boldsymbol{\theta}$, which can be derived using the chain rule.

Structured Estimation In structured estimation, we instead learn a globally normalized probabilistic model that takes the structural constraints into account during training. This method is closely related to the linear approach of Täckström et al. (2015), as well as to the fine-tuning of a neural CRF described by Do and Artières (2010).

We train the model by maximizing the log-likelihood of the training data, again using Adagrad. From Equation (1), we have that the log-likelihood $l(\boldsymbol{r}; \boldsymbol{\theta}) \triangleq \log p(\boldsymbol{r} \mid x, t, \ell, f)$ of a single (structured) training example $(\boldsymbol{r}, \mathcal{S}, x)$ is given by

$$l(\boldsymbol{r}; \boldsymbol{\theta}) = \sum_{s_i \in \mathcal{S}} g(s_i, r_i) - A(\mathcal{S}) \,. \tag{4}$$

By application of the chain rule, the gradient of the log-likelihood factorizes as

$$\frac{\partial l(\boldsymbol{r}; \boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \frac{\partial l(\boldsymbol{r}; \boldsymbol{\theta})}{\partial g_{\text{NN}}} \frac{\partial g_{\text{NN}}}{\partial \boldsymbol{\theta}} \,, \tag{5}$$

where we have used the shorthand g_{NN} for brevity. It is easy to show that the first term $\partial l(\boldsymbol{r}; \boldsymbol{\theta}) / \partial g_{\text{NN}}$ factors into the marginals over edges in the DP lattice, which can be computed with the forward-backward algorithm (recall that the potentials are in simple correspondence with the edge scores in the DP lattice, see Täckström et al. (2015, §4) for details). Again, the chain rule can be used to compute the gradient $\partial g_{\text{NN}} / \partial \boldsymbol{\theta}$ with respect to the parameters of each layer in the network.

[5] An alternate way to locally train the neural network would be to treat the scores as potentials in a multiclass logistic regression model and optimize log-likelihood, as is done with the locally-trained linear model from Täckström et al. (2015), but we did not investigate this alternative in this work.

3.4 Product of Experts

As we will observe in Tables 2 to 5, random initialization of the neural network parameters $\boldsymbol{\theta}$ causes variance in the performance over different runs. We found that using a straightforward product-of-experts (PoE) model (Hinton, 2002) at inference time reduces this variance and results in significantly higher performance. This PoE model is a very simple ensemble, being the factor-wise sum of the potential functions from K independently trained neural networks:

$$g_{\text{PoE}}(s, r; \boldsymbol{\theta}) = \sum_{j=1}^{K} g_{\text{NN}}^{(j)}(s, r, \boldsymbol{\theta}) \,. \tag{6}$$

where $g_{\text{NN}}^{(j)}(s, r, \boldsymbol{\theta})$ is the score from model j.

4 Experimental Setup

In this section we describe the datasets used, the required preprocessing steps, the baselines compared to and the details of our experimental setup.

4.1 Datasets and Significance Testing

We evaluate our approach on four standard datasets. For span-based SRL using PropBank conventions (Palmer et al., 2005), we evaluate on both the CoNLL 2005 shared task dataset (Carreras and Màrquez, 2005), and the larger CoNLL 2012 dataset derived from the OntoNotes 5.0 corpus (Weischedel et al., 2011). We also evaluate our model on the CoNLL 2009 shared task dataset (Hajič et al., 2009), that annotates roles for semantic dependencies, rather than full argument spans. For the CoNLL datasets, we use the standard training, development and test sets. For frame-semantic parsing using FrameNet conventions (Baker et al., 1998), we follow Das et al. (2014) and Hermann et al. (2014) in using the full-text annotations of the FrameNet 1.5 release and follow their data splits.

We use the standard evaluation scripts for each task and use a paired bootstrap test (Efron and Tibshirani, 1994) to assess the statistical significance of the results. For brevity, we only give the p-values for the observed differences between our best and second best models on each of the test sets.

4.2 Preprocessing and Frame Identification

All datasets are preprocessed with a part-of-speech tagger and a syntactic dependency parser, both trained on the CoNLL 2012 training split, after converting the constituency trees to Stanford-style dependencies (De Marneffe and Manning, 2013). The tagger is based on a second-order conditional random field (Lafferty et al., 2001) with standard emission and transition features; for parsing, we use a graph-based parser with structural diversity and cube-pruning (Zhang and McDonald, 2014).

On the WSJ development set (section 22), the labeled attachment score of the parser is 90.9% while the part-of-speech tagger achieves an accuracy of 97.2%. On the CoNLL 2012 development set, the corresponding scores are 90.2% and 97.3%. Both the tagger and the parser, as well as the SRL models use word cluster features (see Table 1). Specifically, we use the clusters with 1000 classes from Turian et al. (2010), which are induced with the Brown algorithm (Brown et al., 1992). To generate the candidate arguments $\mathcal{S}$ (see §3.2) for the CoNLL 2005 and 2012 span-based datasets, we follow Täckström et al. (2015) and adapt the algorithm of Xue and Palmer (2004) to use dependency syntax. For the dependency-based CoNLL 2009 experiments, we modify our approach to assume single length spans and treat every word of the sentence as a candidate argument. For FrameNet, we follow the heuristic of Hermann et al. (2014).

As mentioned in §3, we automatically disambiguate the predicate frames. For FrameNet, we use an embedding-based model described by Hermann et al. (2014). For PropBank, we use a multiclass log-linear model, since Hermann et al. did not observe better results with the embedding model.

To ensure a fair comparison with the closest linear model baseline, we ensured that the preprocessing steps, the argument candidate generation algorithm for the span-based datasets and the frame identification methods are identical to Täckström et al. (2015, §3.2, §6.2-§6.3).

4.3 Baseline Systems

In addition to comparing to Täckström et al. (2015), whose setup is closest to ours, we also compare to prior state-of-the-art systems from the literature.

For CoNLL 2005, we compare to the best non-ensemble and ensemble systems of Surdeanu et al. (2007), Punyakanok et al. (2008) and Toutanova et al. (2008). The ensemble variants of these systems use multiple parses and multiple SRL systems to leverage diversity. In contrast to these ensemble systems, our product-of-experts model uses only a single architecture and one syntactic parse; the constituent models differ only in random initialization. We also compare with the recent deep bidirectional LSTM model of Zhou and Xu (2015).

For CoNLL 2012, we compare to Pradhan et al. (2013), who report results with the (non-ensemble) ASSERT system (Pradhan et al., 2005), and to the model of Zhou and Xu (2015).

For CoNLL 2009, we compare to the top system from the shared task (Zhao et al., 2009), two state-of-the-art systems that employ a reranker (Björkelund et al., 2010; Roth and Woodsend, 2014), and the recent tensor-based model of Lei et al. (2015). We also trained the linear model of Täckström et al. on this dataset (their work omitted this experiment), as a baseline.

Finally, for the FrameNet experiments, we compare to the state-of-the-art system of Hermann et al. (2014), which combines a frame-identification model based on WSABIE (Weston et al., 2011) with a log-linear role labeling model.

4.4 Hyperparameters and Initialization

There are several hyperparameters in our model (§3.2). First, the span embedding dimension of $\boldsymbol{e}_s$ was fixed to 300 to match the dimension of the pre-trained GloVe word embeddings from Pennington et al. (2014) that we use to initialize the embeddings of the word-based features in $\boldsymbol{\phi}(s, x, t, \ell)$. Preliminary experiments showed random initialization of the word-based embeddings to be inferior to pre-trained embeddings. The remaining model parameters were randomly initialized. The frame embedding dimension was chosen from $\{100, 200, 300, 500\}$, while the hidden layer dimension was chosen from $\{300, 500\}$. For PropBank, we fixed the role embedding dimension to 27, which is the number of semantic roles in PropBank datasets (ignoring the AA role, that appears with negligible frequency). For FrameNet, the role embedding dimension was chosen from $\{100, 200, 300, 500\}$. In the Adagrad algorithm, the mini-batch size was fixed to 100 for local estimation (§3.3). For structured estimation (§3.3), a batch size of one was used, since each structured instance contains multiple local factors. The learning rate was chosen from $\{0.1, 0.2, 0.5, 1.0\}$ for local learning and from $\{0.01, 0.02, 0.05, 0.1\}$ for struc-

Method	WSJ Dev				WSJ Test				Brown Test			
	P	R	F1	Comp.	P	R	F1	Comp.	P	R	F1	Comp.
Surdeanu (Single)	–	–	–	–	79.7	74.9	77.2	52.0	–	–	–	–
Surdeanu (Ensemble)	–	–	–	–	87.5	74.7	80.6	51.7	81.8	61.3	70.1	34.3
Toutanova (Single)	–	–	77.9	**57.2**	–	–	79.7	**58.7**	–	–	67.8	39.4
Toutanova (Ensemble)	–	–	78.6	58.7	81.9	78.8	80.3	60.1	–	–	68.8	40.8
Punyakanok (Single)	–	–	–	–	77.1	75.5	76.3	–	–	–	–	–
Punyakanok (Ensemble)	80.1	74.8	77.4	50.7	82.3	76.8	79.4	53.8	73.4	62.9	67.8	32.3
Täckström (Local)	81.3	74.8	77.9	52.4	82.6	76.4	79.3	54.3	74.0	66.8	70.2	38.4
Täckström (Struct.)	81.2	76.2	78.6	54.4	82.3	77.6	79.9	56.0	74.3	68.6	71.3	39.8
Zhou	79.7	**79.4**	**79.6**	–	**82.9**	**82.8**	**82.8**	–	70.7	68.2	69.4	–
This work (Local)	81.4	75.6	78.4	53.7	82.3±0.4	76.8±0.5	79.4±0.1	55.1±0.6	74.1±0.6	68.0±0.7	70.9±0.3	39.1±0.8
This work (Struct.)	80.7	76.1	78.3	54.1	81.8±0.5	77.3±0.3	79.4±0.2	55.6±0.5	73.8±0.7	68.8±0.6	71.2±0.3	40.5±0.8
This work (Local, PoE)	**82.0**	76.6	79.2	55.2	**82.9**	77.8	80.3*	56.7	**75.2**	69.1	72.0	40.8
This work (Struct., PoE)	81.2	76.7	78.9	55.1	82.5	78.2	80.3*	57.3*	74.5	**70.0**	**72.2****	**41.3**

Table 2: PropBank-style SRL results on CoNLL 2005 data. Bold font indicates the best system using a single or no syntactic parse, while the best scores among all systems are underlined. Results from prior work are taken from the respective papers, and '–' indicates performance metrics missing in the original publication. Statistical significance was assessed for F1 and Comp. on the WSJ and Brown test sets with $p < 0.01$ ($*$) and $p < 0.05$ ($**$).

	Excluding predicate senses			Including predicate senses	
	WSJ Dev	WSJ Test	Brown Test	WSJ Test	Brown Test
CoNLL-2009 1st place	–	82.1	69.8	86.2	74.6
Björkelund et al., 2010 + reranking	80.5	82.9	70.9	86.9	75.7
Roth and Woodsend, 2014 + reranking	–	82.1	71.1	86.3	**75.9**
Lei et al. 2015	81.0	82.5	70.8	86.6	75.6
Täckström et al. 2015 (Local)	81.4	83.0	71.2	86.9	74.8
Täckström et al. 2015 (Struct.)	82.4	83.7	72.3	87.3	75.5
This work (Local)	81.2±0.2	82.7±0.3	71.9±0.4	86.7±0.2	75.2±0.3
This work (Struct)	82.3±0.1	83.6±0.1	71.9±0.3	87.3±0.1	75.2±0.2
This work (Local, PoE)	82.4	83.8	**72.8**	87.5	**75.9**
This work (Struct., PoE)	**83.0***	**84.3***	72.4	**87.8***	75.5

Table 3: PropBank-style *semantic dependency* SRL results (labeled F1) on the CoNLL 2009 data set. Bold font indicates the best system. Statistical significance was assessed with $p < 0.01$ ($*$).

tured learning.[6] All hyperparameters were tuned on the respective development sets for each dataset with a straightforward grid-search procedure. In the product-of-experts setup, we train $K = 10$ models, each with a different random seed, and combine them at inference time (see Equation (6)).

5 Empirical Results

Table 2 shows results on the CoNLL 2005 development set and the WSJ and Brown test sets. Our individual neural network models are on par with the best linear single-system baselines that use carefully chosen feature combinations, but has variance across reruns. On the WSJ test set, the product-of-experts model featuring neural networks trained with structured learning achieves higher F_1-score than all non-ensemble baselines, except the LSTM model of Zhou and Xu. It is on par and at times better than ensemble baselines that use diverse syntactic parses. The PoE model outperforms all baselines on the Brown test set, exhibiting its generalization power on out-of-domain text. Overall, using structured learning improves recall at a slight expense of precision when compared to local learning, leading to an increase in the complete argument structure accuracy (*Comp.* in the tables).

Table 3 shows results on the CoNLL 2009 task. Following Lei et al. (2015), we present results using the official evaluation script, along with additional metrics that do not count frame predictions. Note that the linear baseline of Täckström et al.

[6] We observed a strong interaction between learning rate and mini-batch size. Since the number of factors per frame structure is much larger than 100, lower learning rates are better suited for structured estimation.

Method	CoNLL 2012 Development P	R	F1	Comp.
Täckström (Local)	80.6	77.1	78.8	59.0
Täckström (Struct.)	80.5	77.8	79.1	60.1
Zhou	–	–	**81.1**	–
This work (Local)	80.4	77.3	78.8	59.0
This work (Struct)	80.6	77.8	79.2	59.8
This work (Local, PoE)	**81.0**	78.3	79.6	60.6
This work (Struct., PoE)	**81.0**	**78.5**	79.7	**60.9**
Method	CoNLL 2012 Test P	R	F1	Comp.
Pradhan	**81.3**	70.5	75.5	51.7
Pradhan, revised	78.5	76.6	77.5	55.8
Täckström (Local)	80.9	77.7	79.2	60.9
Täckström (Struct.)	80.6	78.2	79.4	61.8
Zhou	–	–	**81.3**	–
This work (Local)	80.6±0.3	77.8±0.2	79.2±0.1	60.8±0.3
This work (Struct.)	80.9±0.2	78.4±0.2	79.6±0.1	61.7±0.2
This work (Local, PoE)	**81.3**	78.8	80.0	62.4
This work (Struct., PoE)	81.2	**79.0**	80.1*	**62.6***

Table 4: PropBank-style SRL results on the CoNLL 2012 development and test sets. Results from prior work are taken from the respective papers, and '–' indicates performance metrics missing in the original publication. Significance was assessed for F1 and Comp. on the test set with $p < 0.01$ ($*$).

Method	FrameNet Development P	R	F1	Comp.
Hermann	78.3	64.5	70.8	–
Täckström (Local)	**80.7**	62.9	70.7	31.2
Täckström (Struct.)	79.6	64.1	71.0	33.3
This work (Local)	78.6	64.6	70.9	32.0
This work (Struct.)	79.6	63.9	70.9	31.8
This work (Local, PoE)	79.0	65.0	71.3	33.1
This work (Struct., PoE)	79.0	64.4	71.0	32.3
This work (Local, PoE, Joint)	79.4	**65.8**	**72.0**	**34.5**
This work (Struct., PoE, Joint)	78.8	65.4	71.5	33.5
Method	FrameNet Test P	R	F1	Comp.
Hermann	74.3	66.0	69.9	–
Täckström (Local)	**76.1**	64.9	70.1	33.0
Täckström (Struct.)	75.4	65.8	70.3	33.8
This work (Local)	73.9±0.6	66.4±0.4	69.9±0.3	33.4±0.6
This work (Struct.)	74.8±0.2	65.5±0.2	69.9±0.1	33.0±0.3
This work (Local, PoE)	74.3	66.9	70.4	33.9
This work (Struct., PoE)	74.6	66.3	70.2	33.3
This work (Local, PoE, Joint)	75.0	**67.3**	**70.9****	**35.4***
This work (Struct., PoE, Joint)	74.2	67.2	70.5	34.2

Table 5: Joint frame and argument identification results for FrameNet. Statistical significance was assessed for F1 and Comp. on the test set with $p < 0.01$ ($*$) and $p < 0.05$ ($**$).

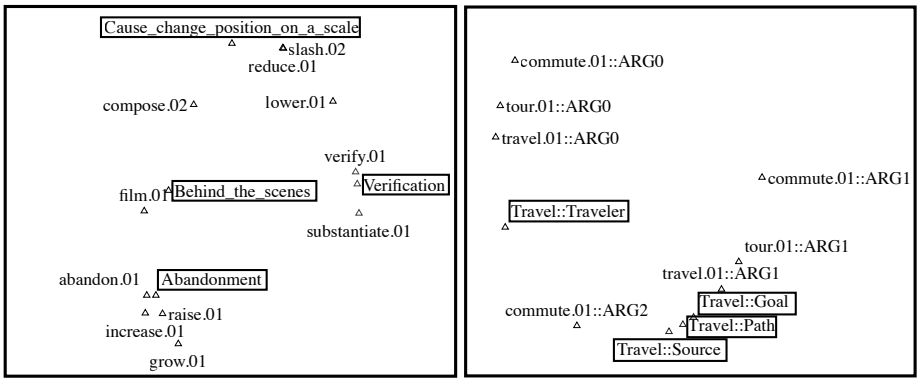
outperforms most prior work, including ones that employs rerankers, except on the Brown test set. Our neural network model performs even better, achieving state-of-the-art results on all metrics.

Table 4 shows the results on the span-based CoNLL 2012 data. The trends observed on the CoNLL 2005 data hold here as well, with structured training yielding an increase in precision at the cost of a small drop in recall. This leads to improvements in both F_1 score and complete structure accuracy. The product-of-experts model trained with structured learning here yields results better than the ASSERT system (Pradhan et al., 2013), but akin to CoNLL 2005, our system falls short in comparison to Zhou and Xu's F_1-score. In contrast to the smaller CoNLL 2005 data, even our single (non-PoE) model outperforms the linear model of Täckström et al. (2015) on the CoNLL 2012 data. We hypothesize that the relative abundance of the latter counteracts the risk for overfitting of the larger number of parameters in our model.

Finally, Table 5 shows the results on FrameNet data, which is very small in size. Here, structured learning does not help and in fact leads to a small drop in performance. Our locally-trained neural network model performs comparably to the linear model of Täckström et al. (2015). However we achieve significant improvements in both F_1-score and full structure accuracy by training our model with a dataset composed of both FrameNet and CoNLL 2005 data.[7] The ability to train in this multitask setting is a unique capability of our approach, and yields state-of-the-art results for FrameNet.

Figure 4 shows the effect of adding increasing amount of CoNLL 2005 data to supplement the FrameNet training corpus in this multitask setting. The Y-axis plots F_1-score on the development data averaged across runs for the local non-PoE model. With increasing amount of PropBank data, performance increases in small steps, and peaks when all the data is added. This shows that with more PropBank data we could further improve performance on the FrameNet task; we leave further exploration of multitask learning of predicate argument structures, including multilingual settings, to future work.

[7] The joint model does not improve results for PropBank. This is likely due to the much larger CoNLL 2005 training set, compared to the FrameNet data (39,832 training sentences in the former as opposed to 3,256 sentences in the latter).



(a) (b)

Figure 3: Two-dimensional t-SNE projections (Van der Maaten and Hinton, 2008) of joint PropBank and FrameNet (boxed) embeddings of frames (a) and frame-role pairs (b).

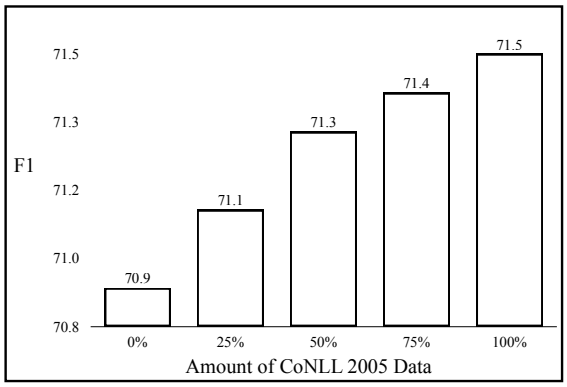


Figure 4: F_1 score on the FrameNet development data averaged over runs versus the percentage of CoNLL 2005 data used to append the FrameNet training corpus. For this plot, we used the locally trained non-PoE model.

5.1 Qualitative Analysis of Embeddings

Figure 3 shows example embeddings from the model trained jointly on FrameNet and PropBank annotations. Figure 3a shows the proximity of the learned embeddings $\boldsymbol{e}_f$ of frames from both FrameNet and PropBank. Figure 3b shows the embeddings for frame-role pairs $\boldsymbol{v}_{(f,r)}$ (the output of the hidden rectified linear layer). Here, we fix the FrameNet frame Travel and the similar PropBank frames commute.01, tour.01 and travel.01 are visualized along with their semantic roles. We observe that the model learns very similar embeddings for the semantically related roles across both datasets. Note that there is a clear separation of the agentive roles from the others for both conventions and how the FrameNet and PropBank counterparts of each type of role are proximate in vector space.

6 Conclusion

We presented a neural network model for semantic role labeling that learns to embed both inputs and outputs in the same vector space. We considered both local and structured training methods for the network parameters from supervised SRL data. Empirically, our approach achieves state-of-the-art results on two standard datasets with a product of experts model, while approaching the performance of a recent deep recurrent neural network model on two other datasets. By training the model jointly on both FrameNet and PropBank data, we achieve the best result to date on the FrameNet test set. Finally, qualitative analysis indicates that the model represents semantically similar annotations with proximate vector-space embeddings.

Acknowledgments

We thank Tom Kwiatkowski, Slav Petrov and Fernando Pereira for comments on early drafts. We are also thankful to Mike Lewis, Mark Yatskar and Luke Zettlemoyer for valuable feedback. Finally, we thank the three anonymous reviewers for suggestions that enriched the final version of the paper.

References

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

Anders Björkelund, Bernd Bohnet, Love Hafdell, and Pierre Nugues. 2010. A high-performance syntactic and semantic dependency parser. In *Proceedings of ICCL: Demonstrations*.

Peter F Brown, Peter V Desouza, Robert L Mercer, Vincent J Della Pietra, and Jenifer C Lai. 1992. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479.

Xavier Carreras and Lluís Màrquez. 2004. Introduction to the CoNLL-2004 shared task: Semantic role labeling. In *Proceedings of CoNLL*.

Xavier Carreras and Lluís Màrquez. 2005. Introduction to the CoNLL-2005 shared task: Semantic role labeling. In *Proceedings of CoNLL*.

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

Ronan Collobert and Jason Weston. 2007. Fast semantic extraction using a novel neural network architecture. In *Proceedings of ACL*.

Dipanjan Das, Desai Chen, Andre F. T. Martins, Nathan Schneider, and Noah A. Smith. 2014. Frame-semantic parsing. *Computational Linguistics*, 40(1):9–56.

Marie-Catherine De Marneffe and Christopher D Manning, 2013. *Stanford typed dependencies manual*.

Trinh-Minh-Tri Do and Thierry Artières. 2010. Neural conditional random fields. In *Proceedings of AISTATS*.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159.

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

Bradley Efron and Robert J Tibshirani. 1994. *An introduction to the bootstrap*. CRC press.

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, et al. 2009. The CoNLL-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of CoNLL*.

Karl Moritz Hermann, Dipanjan Das, Jason Weston, and Kuzman Ganchev. 2014. Semantic frame identification with distributed word representations. In *Proceedings of ACL*.

Geoffrey E. Hinton. 2002. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800.

Meghana Kshirsagar, Sam Thomson, Nathan Schneider, Jaime Carbonell, Noah A. Smith, and Chris Dyer. 2015. Frame-semantic role labeling with heterogeneous annotations. In *Proceedings of ACL*.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML*.

Tao Lei, Yuan Zhang, Lluís Márquez, Alessandro Moschitti, and Regina Barzilay. 2015. High-order low-rank tensors for semantic role labeling. In *Proceedings of NAACL*.

Mike Lewis, Luheng He, and Luke Zettlemoyer. 2015. Joint A* CCG parsing and semantic role labelling. In *Proceedings of EMNLP*.

Andriy Mnih and Geoffrey Hinton. 2007. Three new graphical models for statistical language modelling. In *Proceedings of ICML*.

Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted Boltzmann machines. In *Proceedings of ICML*.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The Proposition Bank: An annotated corpus of semantic roles. *Computational linguistics*, 31(1):71–106.

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

Sameer Pradhan, Kadri Hacioglu, Valerie Krugler, Wayne Ward, James H Martin, and Daniel Jurafsky. 2005. Support vector learning for semantic argument classification. *Machine Learning*, 60(1-3):11–39.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Hwee Tou Ng, Anders Björkelund, Olga Uryupina, Yuchen Zhang, and Zhi Zhong. 2013. Towards robust linguistic analysis using OntoNotes. In *Proceedings of CoNLL*.

Vasin Punyakanok, Dan Roth, and Wen-tau Yih. 2008. The importance of syntactic parsing and inference in semantic role labeling. *Computational Linguistics*, 34(2):257–287.

Michael Roth and Mirella Lapata. 2015. Efficient inference and structured learning for semantic role labeling. *Transactions of the Association for Computational Linguistics*, 3:449–460.

Michael Roth and Kristian Woodsend. 2014. Composition of word representations improves semantic role labelling. In *Proceedings of EMNLP*.

Vivek Srikumar and Christopher D Manning. 2014. Learning distributed representations for structured output prediction. In *Proceedings of NIPS*.

Mihai Surdeanu, Lluís Màrquez, Xavier Carreras, and Pere Comas. 2007. Combination strategies for semantic role labeling. *Journal of Artificial Intelligence Research*, 29:105–151.

Mihai Surdeanu, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. The CoNLL-2008 shared task on joint parsing of syntactic and semantic dependencies. In *Proceedings of CoNLL*.

Oscar Täckström, Kuzman Ganchev, and Dipanjan Das. 2015. Efficient inference and structured learning for semantic role labeling. *Transactions of the Association for Computational Linguistics*, 3:29–41.

Kristina Toutanova, Aria Haghighi, and Christopher D Manning. 2008. A global joint model for semantic role labeling. *Computational Linguistics*, 34(2):161–191.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of ACL*.

Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(85).

Ralph Weischedel, Eduard Hovy, Martha Palmer, Mitch Marcus, Robert Belvin, Sameer Pradhan, Lance Ramshaw, and Nianwen Xue. 2011. OntoNotes: A large training corpus for enhanced processing. In J. Olive, C. Christianson, and J. McCary, editors, *Handbook of Natural Language Processing and Machine Translation*. Springer.

Jason Weston, Samy Bengio, and Nicolas Usunier. 2011. WSABIE: Scaling up to large vocabulary image annotation. In *Proceedings of IJCAI*.

Nianwen Xue and Martha Palmer. 2004. Calibrating features for semantic role labeling. In *Proceedings of EMNLP*.

Hao Zhang and Ryan McDonald. 2014. Enforcing structural diversity in cube-pruned dependency parsing. In *Proceedings of ACL*.

Hai Zhao, Wenliang Chen, Chunyu Kity, and Guodong Zhou. 2009. Multilingual dependency learning: A huge feature engineering method to semantic dependency parsing. In *Proceedings of CoNLL*.

Jie Zhou and Wei Xu. 2015. End-to-end learning of semantic role labeling using recurrent neural networks. In *Proceedings of ACL*.

RELLY: Inferring Hypernym Relationships Between Relational Phrases

Adam Grycner, Gerhard Weikum
Max-Planck Institute for Informatics
Campus E1.4, 66123
Saarbrücken, Germany
{agrycner, weikum}
@mpi-inf.mpg.de

Jay Pujara, James Foulds, Lise Getoor
Computer Science Department
University of California, Santa Cruz
Santa Cruz, CA 95064
{jpujara,jfoulds,getoor}
@soe.ucsc.edu

Abstract

Relational phrases (e.g., "got married to") and their hypernyms (e.g., "is a relative of") are central for many tasks including question answering, open information extraction, paraphrasing, and entailment detection. This has motivated the development of several linguistic resources (e.g. DIRT, PATTY, and WiseNet) which systematically collect and organize relational phrases. These resources have demonstrable practical benefits, but are each limited due to noise, sparsity, or size. We present a new general-purpose method, RELLY, for constructing a large hypernymy graph of relational phrases with high-quality subsumptions using collective probabilistic programming techniques. Our graph induction approach integrates small high-precision knowledge bases together with large automatically curated resources, and reasons collectively to combine these resources into a consistent graph. Using RELLY, we construct a high-coverage, high-precision hypernymy graph consisting of 20K relational phrases and 35K hypernymy links. Our evaluation indicates a hypernymy link precision of 78%, and demonstrates the value of this resource for a document-relevance ranking task.

1 Introduction

One of the many challenges in natural language understanding is interpreting the multi-word phrases that denote relationships between entities. Semantically organizing the complex relationships between diverse phrases is crucial to applications including question answering, open information extraction, paraphrasing, and entailment detection (Yahya et al., 2012; Fader et al., 2011; Madnani et al., 2012; Dagan et al., 2005). For example, a corpus containing the phrase "George Burns was married to Gracie Allen" allows us to answer the query "Who was the spouse of George Burns?" However, "Jay Z is in a relationship with Beyoncé" provides insufficient information to determine whether the couple is married. To capture the knowledge found in text, relational phrases need to be systematically organized with lexical links like synonymy ("married to" and "spouse of") and hypernymy ("in a relationship" generalizing "married to").

Many projects address the challenge of understanding relational phrases, but existing linguistic resources are often limited to synonymy, suffer from low precision, or have low coverage. Systems such as DIRT (Lin and Pantel, 2001), RESOLVER (Yates and Etzioni, 2009), and WiseNet (Moro and Navigli, 2012) have used sophisticated clustering techniques to determine synonymous phrases, but do not provide subsumption information. The PATTY (Nakashole et al., 2012) project goes beyond clustering and introduces a subsumption hierarchy, but suffers from sparsity and contains few hypernymy links. The HARPY (Grycner and Weikum, 2014) project extended PATTY, generating 600K hypernymy links, but with low precision. Berant et al. (2011) introduced entailment graphs that provided a high-quality subsumption hierarchy. This method required partitioning the graph and the largest component consisted of 120 relations. A number of manually-curated relational taxonomies such as WordNet (Fellbaum, 1998), VerbNet (Kipper et al., 2008), and FrameNet (Baker et al., 1998) also offer high-precision hierarchies with limited coverage.

In this paper, we introduce RELLY, a method for producing a hypernymy graph that has both high coverage and precision. We build on previous work, integrating the high-precision knowledge in resources such as YAGO (Suchanek et al.,

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 971–981,
Lisbon, Portugal, 17-21 September 2015.

2007) and WordNet with noisy statistical information from OpenIE projects PATTY and HARPY. RELLY maintains a consistent graph by including collective global constraints such as transitivity, asymmetry, and acyclicity. Scalability is often a concern when employing collective reasoning over large corpora, but our system can produce graphs with over 100K edges on conventional hardware. As a result, we produce a large, complete, and high-precision hypernym graph that includes alignments and type information.

RELLY leverages probabilistic soft logic (PSL) (Bach et al., 2015), a popular probabilistic modeling framework, to collectively infer hypernymy links at scale. PSL uses continuously-valued variables and evidence, allowing easy integration of uncertain statistical information while encoding dependencies between variables using a first-order logic syntax. We define a PSL model with rules that combine statistical features, semantic information, and structural constraints. Statistical features, such as argument overlap and alignments to WordNet verbs senses, allow RELLY to learn from large text collections. Semantic information, such as type information for relation arguments, improves precision of the resulting inferences. Structural constraints, such as transitivity and acyclicity, enforce a complete and consistent set of edges. Using this PSL model, we learn rule weights with a small amount of training data and then perform joint inference over all hypernymy links in the graph.

We highlight three major contributions of our work. First, we introduce RELLY, a scalable method for integrating statistical and semantic signals to produce a hypernymy graph. RELLY is extensible and can easily incorporate additional information sources and features. Second, we generate a complete and precise hypernymy graph over 20K relational phrases and 35K hypernymy links. We have publicly released this hypernymy graph as a resource for the NLP community. Third, we present a thorough empirical evaluation to measure the precision of the hypernymy graph as well as demonstrate its usefulness in a real-world document ranking task. Our results show a high precision (0.78) and superior performance in document ranking compared to state-of-the-art models such as word2vec (Mikolov et al., 2013).

2 Background

Before describing the details of RELLY, we begin with necessary background information on the task of semantically organizing relational phrases, as well as the probabilistic soft logic modeling language which we use to develop our hypernymy graph construction method.

2.1 Relational Phrases

Relational phrases are textual representations of relations which occur between named entities (e.g., "Terry Pratchett") or noun phrases (e.g., "the great writer"). Nakashole et al. (2012) identify relational phrases with the *semantic type signature* of the relation, i.e. the fine-grained lexical types of left- and right-hand side arguments. For example, "Terry Pratchett published his new novel The Colour of Magic" is an instance of the relational phrase "<*person*> published his * ADJ novel <*book*>." In this case, the left-hand argument (the domain of the relation) has the type <*person*> and the right-hand argument (the range of the relation) has the type <*book*>.

Several projects from the Open Information Extraction community have addressed the task of finding synonyms of relational phrases using clustering algorithms. The biggest collection of relational phrases and their synonyms is currently the PATTY project (Nakashole et al., 2012), with around 350,000 semantically typed relational phrases. Prominent alternatives are WiseNet (Moro and Navigli, 2012), which offers 40,000 synsets of relational phrases, PPDB (Ganitkevitch et al., 2013), which contains over 220 million paraphrase pairs, as well as DIRT and VerbOcean (Lin and Pantel, 2001; Chklovski and Pantel, 2004) which inspired the approach and results pursued here.

Relational phrases can be further organized into a hierarchical structure according to their hypernymy (subsumption) relationships. For example, "<*person*> moves to <*country*>" is a hypernym of the relational phrase "<*musician*> emigrates to <*country*>." Of the aforementioned collections, only PATTY attempts to automatically create a subsumption hierarchy for the extracted relational phrases. The authors of the HARPY system argue that the sparseness of PATTY's graph comes from the lack of general phrases in the source corpus. As a solution, they propose using the WordNet verb hierarchy (which contains general

verb senses) to construct a similar hierarchy with PATTY's relational phrases. The graph obtained by HARPY consists of around 600,000 hypernymy links for around 20,000 relational phrases. However, the final graph was not evaluated for precision; rather, the evaluation was instead concentrated on the alignment between verb senses and relations.

In this paper we will make use of several concepts that are closely related to hypernymy, which we define below. Note that although the following definitions concern verbs, we also apply them to relational phrases:

- *hypernym*: the verb Y is a hypernym of the verb X if Y is more general than X. *To perceive* is a hypernym of *to listen* (Bai et al., 2010).
- *troponym*: the verb Y is a troponym of the verb X if doing Y is doing X, in some manner. *To lisp* is a troponym of *to talk* (Bai et al., 2010). Troponym is a verb counterpart for *hyponym*, which applies to nouns. In this work we use these two terms interchangeably.
- *entailment*: the verb Y is entailed by X if, by doing X, you must be doing Y. *To sleep* is entailed by *to snore* (Bai et al., 2010).

2.2 Probabilistic Soft Logic

Our approach is based on probabilistic soft logic (PSL), a popular statistical relational learning system which we briefly describe here. PSL is a templating language for a class of graphical models known as hinge-loss Markov random fields. PSL models are specified using rules in first-order logic syntax, expressing dependencies between interrelated variables. For example, the PSL rule

$$w : \textsc{Hypernym}(P_1, P_2) \wedge \textsc{Hypernym}(P_2, P_3) \Rightarrow \textsc{Hypernym}(P_1, P_3)$$

expresses the transitivity of hypernyms: if phrase P_1 is a hypernym of phrase P_2 and P_2 is a hypernym of P_3, then P_1 is a hypernym of P_3. Rules are weighted (w) to indicate their importance in the model, and weight learning in PSL allows these weights to be learned from training data.

Each rule is ground by substituting the variables in the rule with constants, e.g. "married to" and "relative of" for P_1 and P_2. However, unlike previous approaches such as Markov logic networks, the atoms in each logical rule take values in the [0,1] continuous domain. In addition to providing a natural way of incorporating uncertainty and similarity into models, continuous-valued variables allow the inference objective to be formulated as convex optimization making MAP inference extremely efficient, with empirical performance that scales linearly with the number of ground rules.

3 Hypernymy Graph Construction

In this section we detail RELLY, our system for constructing a hypernymy graph. RELLY incorporates semantic and statistical information from sources such as YAGO, WordNet, PATTY, and HARPY, and uses PSL to combine and reason over these sources. For each source, we introduce a PSL predicate (Table 1). The predicates are divided into three categories: *statistical* (continuous-valued features arising from statistical methods), *semantic* (binary predicates acquired from knowledge bases) and *output* (the target variables). We relate these predicates with a series of rules which combine alignment links, argument similarity, and hierarchical information. The collection of rules defines the PSL model, which we describe in Section 3.1 and Table 2.

In the resulting hypernymy graph, an edge from a relational phrase $R1$ to a relational phrase $R2$ denotes that $R1$ is more specific than $R2$, i.e. $R2$ is a hypernym of $R1$. For example, there is an edge from $R1$ ="<*musician*> emigrates to <*country*>" to $R2$ = "<*person*> moves to <*country*>." In the PSL model the strength of this edge is represented by the confidence score of the predicate $hyponym(R1, R2)$.

3.1 PSL Rules

The PSL rules that define the model are shown in Table 2. Each of the rules is additionally supplied with a weight which describes its importance in the model. The weights are learned from a small hand-crafted hierarchy of relational phrases. The full PSL model combines multiple statistical and semantic signals into the hypernymy graph.

Our model includes rules to encode signals that provide evidence for hypernymy, as well as rules to encode consistency in the graph. One statistical signal for phrase subsumption is *argument overlap*. If the arguments to a relational phrase $R1$ are also found as arguments to another relational phrase $R2$, $R1$ and $R2$ may be synonymous or

Table 1: PSL predicates;
$R1, R2$ are relational phrases, $Vb1, Vb2$ WordNet verb senses and $TL1, TR1, T1, T2$ YAGO types

PSL predicate	Type	Description
$weedsInclusion(R1, R2)$	statistical	degree of inclusion of sets of argument pairs of relations defined as $\frac{\lvert ArgsR1\rvert}{\lvert ArgsR1 \cap ArgsR2\rvert}$ (Weeds and Weir, 2003)
$pattySubsumption(R1, R2)$	statistical	PATTY subsumption (Nakashole et al., 2012)
$harpy(R1, Vb1)$	statistical	alignment links between relational phrases and WordNet verb senses (Grycner and Weikum, 2014)
$wordnetHyponym(Vb1, Vb2)$	semantic	hyponymy link between WordNet verb senses
$lType(R1, TL1)$	semantic	left (domain) type of arguments of a relational phrase
$rType(R1, TR1)$	semantic	right (range) type of arguments of a relational phrase
$yagoHyponym(T1, T2)$	semantic	$T1$ is a subtype of $T2$ in YAGO hierarchy
$candidateHyponym(R1, R2)$	output	relational phrase $R1$ is more specific than $R2$ (without enforcing consistent argument types)
$hyponym(R1, R2)$	output	relational phrase $R1$ is more specific than $R2$

$R2$ may be a hypernym of $R1$. We use two measures of argument overlap, $weedsInclusion$ and $pattySubsumption$, in rules 1 and 2, respectively, to capture the relationship between argument overlap and subsumption. Another signal, used in rule 3, is the *alignment* between relational phrases and WordNet verb senses. If relational phrases $R1$ and $R2$ are aligned to WordNet verb senses $Vb1$ and $Vb2$ which are in a hyponymy relationship, then this is evidence that $R1$ is more specific than $R2$. An example of using HARPY alignment links and WordNet hierarchy is shown in Figure 1.

We encode local consistency requirements using Rules 4–6. Rule 4 (*types compatibility*) is a constraint to restrict hypernymy links to be between relations whose types are compatible, i.e they are identical or the types of the more specific relation are subtypes of the types of the more general relation. Rules 5 and 6 create a *transitive closure* of both WordNet and YAGO hierarchies. As a result of these rules, we can use indirect hyponyms (in rule 3) or indirect subtypes (in rule 4).

Finally, rules 7, 8 and 9 shape the structure of the output graph with collective global constraints. Rule 7 (*asymmetry*) removes bidirectional links, rule 8 (*transitivity*) creates a transitive closure of the graph and rule 9 (*acyclicity*) prevents the creation of small cycles in the graph.

3.2 RELLY Overview

RELLY has four stages: data pre-processing, rule weight learning, inference, and thresholding.

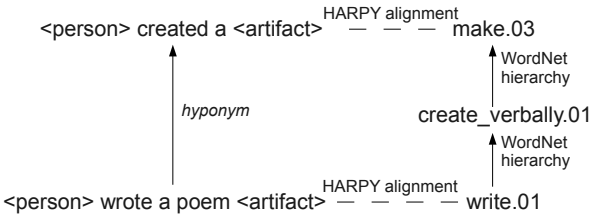


Figure 1: HARPY alignment usage

First, in the data pre-processing stage, we assign confidence scores of 0 or 1 for the binary-valued semantic predicates in the PSL model. For example, the $wordnetHyponym(Vb1, Vb2)$ confidence score is set to 1 if there is a hyponymy link between verb senses $Vb1$ and $Vb2$ and 0 otherwise. In other cases, the confidence is set to a similarity score of a feature which is represented by a predicate. For example, the $weedsInclusion(R1, R2)$ confidence is equal to the Weeds inclusion score between relations $R1$ and $R2$.

In the next stage the weights of the PSL rules described in Table 2 are learned from a small handcrafted graph of relational phrases. The weight learning is performed using an EM algorithm. Later, the most-probable explanation (MPE) state of the output predicates is inferred.

Finally, we export the inferred confidence scores of the predicate $hyponym$ and perform additional cleaning. Whenever two links contradict each other (e.g. we have both $hyponym(R1, R2)$ and $hyponym(R2, R1)$) we remove the link with

Table 2: PSL rules

Id	Feature	PSL rule
1	Weeds inclusion	$weedsInclusion(R1, R2) \Rightarrow candidateHyponym(R1, R2)$
2	Patty subsumption	$pattySubsumption(R1, R2) \Rightarrow candidateHyponym(R1, R2)$
3	Harpy alignment	$wordnetHyponym(Vb1, Vb2) \wedge harpy(R1, Vb1) \wedge harpy(R2, Vb2)$ $\Rightarrow candidateHyponym(R1, R2)$
4	Types compatibility	$candidateHyponym(R1, R2) \wedge lType(R1, TL1) \wedge rType(R1, TR1)$ $\wedge lType(R2, TL2) \wedge rType(R2, TR2) \wedge yagoHyponym(TL1, TL2)$ $\wedge yagoHyponym(TR1, TR2) \Rightarrow hyponym(R1, R2)$
5	WordNet hierarchy	$wordnetHyponym(Vb1, Vb2) \wedge wordnetHyponym(Vb2, Vb3)$ $\Rightarrow wordnetHyponym(Vb1, Vb3)$
6	Yago hierarchy	$yagoHyponym(T1, T2) \wedge yagoHyponym(T2, T3)$ $\Rightarrow yagoHyponym(T1, T3)$
7	Asymmetry	$hyponym(R1, R2) \Rightarrow \neg hyponym(R2, R1)$
8	Transitivity	$hyponym(R1, R2) \wedge hyponym(R2, R3) \Rightarrow hyponym(R1, R3)$
9	Acyclicity	$hyponym(R1, R2) \wedge hyponym(R2, R3) \Rightarrow \neg hyponym(R3, R1)$

the lower confidence score. If both predicates have the same confidence score we exclude them both from the final graph. Additionally, we only consider links with a confidence score above an empirically chosen threshold of 0.2.

4 Evaluation

In our experiments, we use a large corpus of relational phrases to construct a hypernymy graph using RELLY. We evaluate RELLY using both intrinsic and extrinsic evaluation. In the intrinsic evaluation, we asked human annotators to judge the relationship between two relational phrases and compared results from several hypernymy graphs. In the extrinsic evaluation, we used the hypernymy graph for a real-world document ranking task and measured the mean reciprocal rank (MRR) for a number of methods. In both evaluations, the hypernymy graph constructed by RELLY demonstrates significantly better performance than competing algorithms.

4.1 Dataset

We use RELLY to build a hypernymy graph with data from the PATTY and HARPY projects. The input to our system consists of 20,812 relational phrases and the associated argument types extracted from the English-language Wikipedia website using the PATTY system. For simplicity, we only include relational phrases that contain exactly one verb (e.g. "took the throne"), excluding noun phrases (e.g. "member of") and phrases containing multiple verbs (e.g. "hit and run"). The verb "to be" and modal verbs were not considered in the dataset. We also include HARPY alignments to the corresponding verb senses in WordNet for each phrase in the corpus. Additionally, we use a subset of the type-subsumption hierarchy from YAGO consisting of 144 types and 323 subsumption relationships.

During graph inference, RELLY evaluated 7.9M possible hypernymy links using 9.7M ground logical rules and constraints. Ultimately, RELLY produced 35,613 hypernymy links between relational phrases with confidence scores above 0.2. The hypernymy graph consisted of 3,730 roots. Running RELLY on a multi-core 2.27GHz server with 64GB of RAM required approximately 20 hours. For comparison, PATTY produced 8,162 subsumption links out of 350,569 phrases with approximately 2,300 roots.

4.2 Intrinsic Evaluation

In our intrinsic evaluation, we assess the precision of hypernymy links inferred by RELLY and compare with the precision of hypernymy graphs of PATTY and HARPY. In this evaluation, we measure precision for both the most confident hypernymy links in the system (precision@100) and the precision of a random sample of 100 hypernymy links. Each set of hypernymy links were presented to several human annotators for labeling.

To measure precision@100, we choose the top 100 hypernymy links using the confidence scores reported by PSL. We similarly choose the top 100 links from PATTY using the PATTY subsumption

score. Since HARPY does not provide confidence scores, we were unable to compute precision@100 for HARPY.

For each of the three systems, we used the full set of hypernymy links they produce, which consisted of 8K links from PATTY, 600K links from HARPY and 35K links from RELLY. We randomly sampled 100 hypernymy links from each of these systems.

We presented the selected hypernymy links to several human annotators. The labeling task required the annotator to judge the relationship between two relational phrases in a hypernymy link. For each relational phrase, we provided annotators with type information about the phrase arguments (domain and range) and examples of sentences that use the relational phrase. Based on this information, annotators could make one of four judgments: (1) the phrases are unrelated; (2) the phrases are synonymous; (3) the first phrase is more specific than the second phrase; (4) the second phrase is more specific than the first phrase. This evaluation task had good inter-annotator agreement, with a Cohen's Kappa of 0.624. Separately, the precision@100 dataset had Cohen's Kappa of 0.708 and the randomly sampled dataset had Cohen's Kappa of 0.521.

We show the results of the intrinsic evaluation in Table 3 with 0.9-confidence Wilson score interval (Brown et al., 2001). In comparison to HARPY and PATTY, RELLY has higher precision for both precision@100 and random evaluations. Precision in RELLY is comparable to PATTY, but RELLY has more than four times as many hypernym links. HARPY has far more hypernymy links, but with a precision of 0.43, we find that many of these links are incorrect.

Table 4 includes example hypernymy links from RELLY. There are examples where PATTY's subsumption is a dominant signal ("<*person*> publicly accused <*person*>" ⇒ "<*person*> accused <*person*>"). We also observe YAGO type hierarchy influence ("<*athlete*> played for <*team*>" ⇒ "<*person*> played for <*organization*>"), as well as the influence of combined WordNet hierarchy with HARPY alignments ("<*person*> marry daughter <*person*>" ⇒ "<*person*> joins <*person*>"). The advantage of RELLY is that it computes the final graph jointly and incorporates transitivity, asymmetry and acyclicity rules. It leads to less semantic drift in longer hypernymy chains

Table 3: Intrinsic evaluation

	Prec.	**Range**	**Cvg.**
precision@100			
RELLY	0.87	0.81 - 0.92	35K
PATTY	0.83	0.76 - 0.90	8K
random sample			
RELLY	0.78	0.71 - 0.84	35K
PATTY	0.75	0.68 - 0.82	8K
HARPY	0.43	0.35 - 0.52	600K

(e.g. Figure 2) compared with PATTY where "<*organization*> merged <*organization*>" can lead to "<*team*> beat <*team*>".

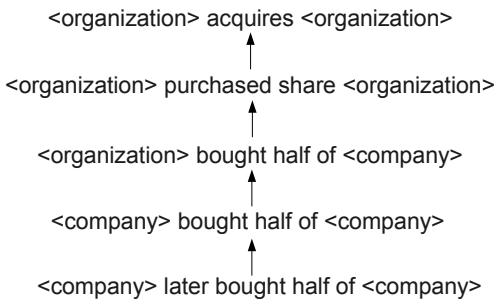


Figure 2: Chain of hypernymy

4.3 Ablation Study

Two advantages of RELLY that we have highlighted are easily incorporating new information sources and collectively enforcing global constraints. To analyze the influence of these system components, we performed an ablation study where we omitted PSL rules corresponding to specific model features. Using this approach, we quantify the importance of these features to RELLY's performance.

First, we demonstrate the value of type information in determining hypernymy. The YAGO type hierarchy allows RELLY to detect hypernymy links between relational phrases where types do not match exactly, but are compatible through type subsumption. When the YAGO type hierarchy rules are omitted from the model, coverage is reduced dramatically; the resulting hypernymy graph contains only 12,000 hypernymy links in contrast to the 35,000 links in the original model. Additionally, removing YAGO type information harms precision, with a precision of 0.75 ± 0.09

Table 4: Example RELLY hypernymy links

Hyponym relational phrase			Hypernym relational phrase		
Domain	Text pattern	Range	Domain	Text pattern	Range
head of state	abdicated in favor of	sovereign	person	resigns as	person
person	publicly accused	person	person	accused	person
person	marry daughter	person	person	joins	person
person	had paid	person	person	interacted with	person
athlete	played for	team	person	played for	organization

Table 5: Results for Entailment graphs induction

	Prec.	Rec.	F1
Berant et al. (2011)	0.422	0.434	0.428
PSL	**0.461**	**0.435**	**0.447**

with 0.9-confidence Wilson score interval for a random sample of 100 examples.

Next, we show how global constraints on the hypernymy graph such as anti-symmetry and acyclicity improve the quality of the hypernymy graph. Since the relational phrases generated by PATTY are clustered to find synonymous relations, these global constraints prevent RELLY from merging clusters. When the anti-symmetry and acyclicity rules were removed from the model, the resulting hypernymy graph included approximately 500 additional hypernymy links, while 10 existing links were removed. We manually evaluated the newly introduced links, and found that the majority of links were false positives.

4.4 Entailment Graph Induction

We compared the performance of PSL against the Integer Linear Programming (ILP) formulation by (Berant et al., 2011). The comparison was performed on the task of creating entailment graphs as described in (Berant et al., 2011). This task is strongly related to finding hypernyms of relational phrases. The experiments were executed on the dataset of 10 manually annotated graphs. In total this dataset contains 3,427 positive and 35,585 negative examples. Our model uses the transitivity rule ($entails(A,B) \wedge entails(B,C) \Rightarrow entails(A,C)$). We also include the local entailment scores ($score(A,B) \Rightarrow entails(A,B)$) which were released by (Berant et al., 2011). Table 5 presents micro-averaged precision, recall and F1 scores for this comparison.

PSL was much faster than the other exact methods used for this problem. To compare efficiency we measured the run-time of our method. Without any graph decomposition it took on average 232 seconds. The experiments were performed on a multi-core 2.67GHz server with 32GB of RAM. The methods reported in (Berant et al., 2012), which did not utilize graph decomposition method, had run-time above 5000 seconds.

4.5 Extrinsic Evaluation

The ultimate goal of producing a high-quality hypernymy graph is to deepen our understanding of natural language and improve performance on the many NLP applications. One such application is document retrieval, where billions of queries are performed each day through search engines. In our extrinsic evaluation, we demonstrate how a hypernymy graph can improve performance on a document ranking and retrieval task.

We consider a task where an input query document is compared to a corpus of documents with the aim of finding the most relevant related documents. To isolate the evaluation to relational phrases, we *anonymize* the documents, by replacing all named entities and noun phrases with placeholders. For example, the sentence "`The villain has already fled to the Republica de Isthmus`" is anonymized to "`⋆ has already fled to ⋆`." Anonymized retrieval has potential applications in security and for sensitive documents.

We collected a dataset consisting of movie plot summaries from two different websites, Wikipedia and the Internet Movie Database (IMDB). We chose plot synopses from 25 James Bond movies and 23 movies based on the Marvel Comics characters. For each plot synopsis, we have two plot descriptions: one from Wikipedia and another from IMDB. Given a query in the form of an anonymized plot description from one website, the task is to rank the anonymized plot descriptions

from the other dataset using relational phrase similarity. For example, given a query plot description of "Iron Man" from Wikipedia, rank plot descriptions from IMDB with the goal of maximizing the ranking of the corresponding "Iron Man" plot summary. We evaluate the quality of these rankings using the mean reciprocal rank (*MRR*) score, $\text{MRR} = \frac{1}{|Q|}\sum_{i=1}^{|Q|}\frac{1}{\text{rank}_i}$. Here, Q is the number of documents in the collection (i.e. 2*48 = 96) and rank_i is the position of the counterpart document in the ranking of document i.

As baseline algorithms, we use a unigram word2vec model and a bigram model. In the unigram word2vec model documents are represented by the average of the 300-dimensional word vectors trained on part of Google News dataset (about 100 billion words) (Mikolov et al., 2013). We could not use the bigram word2vec model because of the frequent occurrence of the placeholder symbol. In the bigram model, documents are represented by vectors in the bag-of-bigrams model with bigram frequency weights. The similarity measure in both cases is the cosine similarity measure.

As the first of our approaches we proposed a solution purely based on relational phrases. In the *relational phrases* model we extract relational phrases from a text and we map them to their synsets from PATTY (clusters of synonyms). A phrase is mapped to a synset if the Jaccard similarity between tokens of extracted relation and tokens of one of the phrases in the synset is above a threshold. Next we represent the document as a vector of the relational phrase synsets weighted by the frequency of the synset in the document (bag-of-relational_phrases). The similarity score between two documents is the cosine similarity between two vectors representing two documents. The ranking is created based on the similarity scores. In the *relational phrases + hypernyms* model we add hypernyms of the extracted relational phrases to the document vector (based on the hypernymy graph). Hypernyms are additionally weighted by the confidence score produced by the algorithm described in the Section 3. In the second approach we combine relational phrases models with the best of the baselines. The similarity score is then equal to $\lambda sim_1 + (1-\lambda) sim_2$. The λ parameter is trained on a different dataset (2*8 plot descriptions of Harry Potter movies). Training was performed by maximization of the MRR score using grid search. We consider the combination of the *bigram* model with *relational phrases*, as well as the combination of the *bigram* model with *relational phrases + hypernyms*.

Table 6: Extrinsic evaluation (Bond & Marvel)

	MRR score
word2vec	0.26
bigram	0.55
relational phrases	0.28
+ hypernyms	0.25
+ bigrams	0.58
+ hypernyms + bigrams	**0.60**

The results of the experiment are presented in Table 6. The best MRR score was obtained by *relational phrases + hypernyms + bigrams* model. The number of samples, 96, was large enough for statistical significance. We performed a paired t-test for MRR between each of these methods. The obtained p-values were below 0.05.

5 Related Work

The biggest sources of hypernyms, subsumptions, and hierarchical structure can be found in existing knowledge bases. Examples of these are Freebase (Bollacker et al., 2008), YAGO, DBPedia (Lehmann et al., 2014), and Google Knowledge Vault (Dong et al., 2014). However, these knowledge bases are mainly concentrated on named entities and noun phrases, and the variety of relations between entities is much smaller. Relations and information about them are underrepresented.

Open Information Extraction systems try to solve this problem by extracting new relations from natural text. These new relations do not necessarily follow the standard schema of knowledge bases. Additionally, these systems often organize the newly extracted relations by clustering or hierarchy construction. A first attempt to extract and cluster similar relations was presented in DIRT. This work was followed by projects such as ReVerb, PATTY, WiseNet, NELL (Carlson et al., 2010), and RESOLVER (Yates and Etzioni, 2009). PATTY and WiseNet also introduced semantic types to their concept of relational phrases. All of these systems rely on the co-occurrence of arguments of clustered relations. A different approach was presented in PPDB, where the authors

cluster phrases based on the similarity of translations to other languages.

Of these systems, only PATTY attempted to create a hierarchy of relations and the result was very sparse. HARPY aimed to overcome this problem by disambiguating and aligning relational phrases with WordNet, and performing a simple reconstruction of the WordNet hierarchy on top of relational phrases from PATTY. A very similar problem was addressed in the entailment graph project (Levy et al., 2014). The authors automatically created graphs of entailments between propositions, using Integer Linear Programing as one of the main components. Propositions can be encoded as triples of form *(subject, relation, object)*. Edges in the entailment graph occur between these triples, whereas edges connect typed relations in PATTY and HARPY. Moreover, the relations in the propositions were mainly limited to single verbs, whereas in our case we also consider longer relational phrases. Relations with semantic types were also used in typed entailment graphs (Berant et al., 2011). However, the type hierarchy was not considered there, which prevented from creating links between two relations with different semantic types. The input dataset was also smaller – the biggest graph consisted of 118 relations.

Although there is a scarcity of automatically created taxonomies of relations, there exist several manually curated taxonomies. Manually crafted verb or relation hierarchies are available in WordNet, VerbNet and FrameNet. WordNet has 13,767 verb synsets, which are organized into a hierarchy with 13,239 hypernymy links.

Automatic construction of taxonomies of named entities or noun phrases has received much more attention than organization of verbs or relations. In (Snow et al., 2006), the WordNet taxonomy was extended by 10,000 novel noun synsets with hypernym-hyponym links. In (Bansal et al., 2014), the authors reconstructed WordNet's noun hypernymy/hyponymy hierarchy from scratch using a probabilistic graphical model formulation. Another method of organizing noun phrases was proposed in (Mehdad et al., 2013), where an entailment graph of noun phrases was constructed.

Building a hypernymy graph for relational phrases is strongly related with the textual entailment task (Dagan et al., 2010). This concept was introduced in the Recognizing Textual Entailment (RTE) shared task (Dagan et al., 2005). Instead of short typed relational phrases, the input data are two texts – the entailing text T and the hypothesis text H. According to (Dagan et al., 2005)'s definition, "T entails H if, typically, a human reading T would infer that H is most probably true."

In RELLY, we use probabilistic soft logic (PSL) as the main ingredient of our approach. PSL was successfully used for numerous other applications including knowledge graph construction (Pujara et al., 2013), trust in social networks (Huang et al., 2012b), ontology alignment (Broecheler and Getoor, 2009), and social group modeling (Huang et al., 2012a).

6 Conclusion

This paper presents RELLY, a scalable method for integrating statistical and semantic signals to produce a hypernymy graph of relational phrases. We used RELLY to create a hypernymy graph that has both high coverage and precision, as shown in our evaluation. RELLY is extensible and can easily incorporate additional information sources and features. The hypernymy graph of relational phrases could potentially be useful for many problems of natural language processing and information retrieval. For example, we applied the hypernymy graph to a document-relevance task, which we used to evaluate RELLY extrinsically. As a future work, RELLY can incorporate more information sources and statistical signals and be expanded to infer multi-verb or noun relational phrases. The RELLY resource is publicly available at `www.mpi-inf.mpg.de/yago-naga/patty/`.

Acknowledgments: This work was partially supported by National Science Foundation (NSF) grant IIS1218488 and by the Intelligence Advanced Research Projects Activity (IARPA) via Department of Interior National Business Center (DoI/NBC) contract number D12PC00337. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of NSF, IARPA, DoI/NBC, or the U.S. Government.

References

Stephen H. Bach, Matthias Broecheler, Ben. Huang, and Lise Getoor. 2015. Hinge-loss Markov random fields and probabilistic soft logic. arXiv:1505.04406 [cs.LG].

Rujiang Bai, Xiaoyue Wang, and Junhua Liao. 2010. Extract semantic information from wordnet to improve text classification performance. In *Advances in Computer Science and Information Technology*, pages 409–420.

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet project. In *Proceedings of Association for Computational Linguistics and Conference on Computational Linguistics (COLING/ACL)*, pages 86–90.

Mohit Bansal, David Burkett, Gerard de Melo, and Dan Klein. 2014. Structured learning for taxonomy induction with belief propagation. In *Proceedings of Association for Computational Linguistics (ACL)*, pages 1041–1051.

Jonathan Berant, Ido Dagan, and Jacob Goldberger. 2011. Global learning of typed entailment rules. In *Proceedings of Association for Computational Linguistics (ACL)*, pages 610–619.

Jonathan Berant, Ido Dagan, Meni Adler, and Jacob Goldberger. 2012. Efficient tree-based approximation for entailment graph learning. In *Proceedings of Association for Computational Linguistics (ACL)*.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of ACM SIGMOD International Conference on Management of Data*, pages 1247–1250.

Matthias Broecheler and Lise Getoor. 2009. Probabilistic similarity logic. In *International Workshop on Statistical Relational Learning*.

Lawrence D. Brown, T. Tony Cai, and Anirban Dasgupta. 2001. Interval estimation for a binomial proportion. *Statistical Science*, 16:101–133.

Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R. Hruschka, and Tom M. Mitchell. 2010. Toward an architecture for never-ending language learning. In *Proceedings of Association for the Advancement of Artificial Intelligence (AAAI)*.

Timothy Chklovski and Patrick Pantel. 2004. VerbOcean: Mining the web for fine-grained semantic verb relations. In *Proceedings of Empirical Methods in Natural Language Processing (EMNLP)*, pages 33–40.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The PASCAL recognising textual entailment challenge. In *In Proceedings of the PASCAL Challenges Workshop on Recognising Textual Entailment*.

Ido Dagan, Bill Dolan, Bernardo Magnini, and Dan Roth. 2010. Recognizing textual entailment: Rational, evaluation and approaches erratum. *Natural Language Engineering*, 16:105–105, 1.

Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. 2014. Knowledge Vault: A web-scale approach to probabilistic knowledge fusion. In *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 601–610.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *Proceedings of Empirical Methods in Natural Language Processing (EMNLP)*, pages 1535–1545.

Christiane Fellbaum, editor. 1998. *WordNet An Electronic Lexical Database*. The MIT Press.

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

Adam Grycner and Gerhard Weikum. 2014. HARPY: Hypernyms and alignment of relational paraphrases. In *Proceedings of Conference on Computational Linguistics (COLING)*, pages 2195–2204.

Bert Huang, Stephen H. Bach, Eric Norris, Jay Pujara, and Lise Getoor. 2012a. Social group modeling with probabilistic soft logic. In *NIPS Workshop on Social Network and Social Media Analysis: Methods, Models, and Applications*.

Bert Huang, Angelika Kimmig, Lise Getoor, and Jennifer Golbeck. 2012b. Probabilistic soft logic for trust analysis in social networks. In *International Workshop on Statistical Relational Artificial Intelligence (StaRAI 2012)*.

Karin Kipper, Anna Korhonen, Neville Ryant, and Martha Palmer. 2008. A large-scale classification of english verbs. *Language Resources and Evaluation*, 42(1):21–40.

Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, and Chris Bizer. 2014. DBpedia - a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web Journal*.

Omer Levy, Ido Dagan, and Jacob Goldberger. 2014. Focused entailment graphs for Open IE propositions. In *Proceedings of Conference on Computational Natural Language Learning (CoNLL)*, pages 87–97.

Dekang Lin and Patrick Pantel. 2001. DIRT @SBT@discovery of inference rules from text. In *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 323–328.

Nitin Madnani, Joel Tetreault, and Martin Chodorow. 2012. Re-examining machine translation metrics for paraphrase identification. In *Proceedings of North American Chapter of the Association for Computational Linguistics Human Language Technologies (NAACL-HLT)*, pages 182–190.

Yashar Mehdad, Giuseppe Carenini, Raymond T. Ng, and Shafiq Joty. 2013. Towards topic labeling with phrase entailment and aggregation. In *Proceedings of North American Chapter of the Association for Computational Linguistics Human Language Technologies (NAACL-HLT)*, pages 179–189.

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

Andrea Moro and Roberto Navigli. 2012. WiseNet: building a wikipedia-based semantic network with ontologized relations. In *Proceedings of ACM International Conference on Information and Knowledge Management (CIKM)*, pages 1672–1676.

Ndapandula Nakashole, Gerhard Weikum, and Fabian Suchanek. 2012. PATTY: A taxonomy of relational patterns with semantic types. In *Proceedings of Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning EMNLP-CoNLL*, pages 1135–1145.

Jay Pujara, Hui Miao, Lise Getoor, and William Cohen. 2013. Knowledge graph identification. In *International Semantic Web Conference (ISWC)*.

Rion Snow, Daniel Jurafsky, and Y. Andrew Ng. 2006. Semantic taxonomy induction from heterogenous evidence. In *Proceedings of Association for Computational Linguistics and International Conference on Computational Linguistics (COLING/ACL)*, pages 801–808.

Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *Proceedings of International Conference on World Wide Web (WWW)*, pages 697–706.

Julie Weeds and David Weir. 2003. A general framework for distributional similarity. In *Proceedings of Empirical Methods in Natural Language Processing (EMNLP)*, pages 81–88.

Mohamed Yahya, Klaus Berberich, Shady Elbassuoni, Maya Ramanath, Volker Tresp, and Gerhard Weikum. 2012. Natural language questions for the web of data. In *Proceedings of Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning EMNLP-CoNLL*, pages 379–390.

Alexander Yates and Oren Etzioni. 2009. Unsupervised methods for determining object and relation synonyms on the web. *Journal of Artificial Intelligence Research*, 34(1):255–296.

Mise en Place: Unsupervised Interpretation of Instructional Recipes

Chloé Kiddon[†], **Ganesa Thandavam Ponnuraj**[‡], **Luke Zettlemoyer**[†], **and Yejin Choi**[†]

[†] Computer Science & Engineering, University of Washington, Seattle, WA
{chloe, lsz, yejin}@cs.washington.edu

[‡] Department of Computer Science, Stony Brook University, Stony Brook, NY
gthandavam@gmail.com

Abstract

We present an unsupervised hard EM approach to automatically mapping instructional recipes to action graphs, which define what actions should be performed on which objects and in what order. Recovering such structures can be challenging, due to unique properties of procedural language where, for example, verbal arguments are commonly elided when they can be inferred from context and disambiguation often requires world knowledge. Our probabilistic model incorporates aspects of procedural semantics and world knowledge, such as likely locations and selectional preferences for different actions. Experiments with cooking recipes demonstrate the ability to recover high quality action graphs, outperforming a strong sequential baseline by 8 points in F1, while also discovering general-purpose knowledge about cooking.

1 Introduction

Instructional language describes how to achieve a wide variety of goals, from traveling successfully to a desired location to cooking a particular dish for dinner. Despite the fact that such language is important to our everyday lives, there has been relatively little effort to design algorithms that can automatically convert it into an actionable form. Existing methods typically assume labeled training data (Lau et al., 2009; Maeta et al., 2015) or access to a physical simulator that can be used to test understanding of the instructions (Branavan et al., 2009; Chen and Mooney, 2011; Bollini et al., 2013). In this paper, we present the first approach for unsupervised learning to interpret instructional recipes using text alone, with application to cooking recipes.

Given a recipe, our task is to segment it into text spans that describe individual actions and construct an *action graph* whose nodes represent actions and edges represent the flow of arguments across actions, for example as seen in Fig. 1. This task poses unique challenges for semantic analysis. First, null arguments and ellipses are extremely common (Zwicky, 1988). For example, sentences such as "Bake for 50 minutes" do not explicitly mention what to bake or where. Second, we must reason about how properties of the physical objects are changed by the described actions, for example to correctly resolve what the phrase "the wet mixture" refers to in a baking recipe. Although linguistic context is important to resolving both of these challenges, more crucial is common sense knowledge about how the world works, including what types of things are typically baked or what ingredients could be referred to as "wet."[1]

These challenges seemingly present a chicken and egg problem — if we had a high quality semantic analyzer for instructions we could learn common sense knowledge simply by reading large bodies of text. However, correctly understanding instructions requires reasoning with exactly this desired knowledge. We show that this conflict can be resolved with an unsupervised learning approach, where we design models to learn various aspects of procedural knowledge and then fit them to unannotated instructional text. Cooking recipes are an ideal domain to study these two challenges simultaneously, as vast amounts of recipes are available online today, with significant redundancy in their coverage that can help bootstrap the overall learning process. For example, there are over 400 variations on "macaroni and cheese" recipes on allrecipes.com, from "chipotle

[1]The goal of representing common sense world knowledge about actions and objects also drives theories of frame semantics (Fillmore, 1982) and script knowledge (Schank and Abelson, 1977). However, our focus is on inducing this style of knowledge automatically from procedural texts.

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 982–992,
Lisbon, Portugal, 17-21 September 2015.

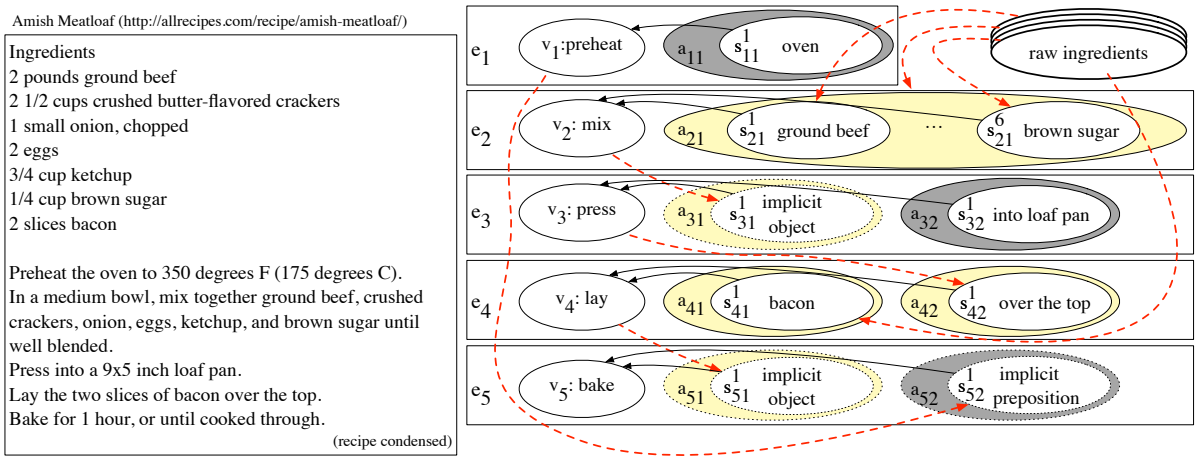


Figure 1: An input recipe (left) and a partial corresponding output action graph (right). Each rectangle (e_i) represents an action. The leftmost oval (v_i) in each action is the action's verb and the following ovals (a_{ij}) represents the verb's arguments. The yellow ovals represent foods; the grey ovals represent locations. Argument ovals with dotted boundaries are implicit, i.e., not present in text. The inner white ovals (s_{ij}^k) are string spans. The red dashed lines represent connections to string spans from their originating verb or raw ingredient. The string spans also connect to their associated verb in the action diagram to model the flow of ingredients. For example, there is a directed path from each raw ingredient to the implicit object of bake, representing that the object being baked is composed of all of the raw ingredients.

macaroni and cheese," to "cheesy salsa mac."

We present two models that are learned with hard EM algorithms: (1) a segmentation model to extract the actions from the recipe text, and (2) a graph model that defines a distribution over the connections between the extracted actions. The common sense knowledge is encoded in the second model which can, for example, prefer graphs that model implicit verb arguments when they better match the learned selectional preferences. The final action graph is constructed with a local search algorithm, that allows for global reasoning about ingredients as they flow through the recipe.

Experiments demonstrate the ability to recover high quality action graphs, gaining up to 8 points in F1 over a strong baseline where the ingredients flow sequentially through the verbs. The learned models are also highly interpretable, specifying for example that "dough" likely contains "flour" and that "add" generally requires two food arguments, even if only one is mentioned in the sentence.

2 Task Definition

Procedural text such as a recipe defines a set of actions, i.e. *predicates*, applied to a set of objects, i.e. *arguments*. A unique challenge in procedural text understanding is to recover how different arguments flow through a chain of actions; the results of intermediate actions (e.g., "Boil the pasta until al dente.") provide the inputs for future actions (e.g., "Drain."). We represent these correspondences with an *action graph*. In this section, we first describe our structured representation of recipe text, then we define how components of the recipe connect. Finally, we will show how given a recipe and a set of connections we can construct an action graph that models the flow of ingredients through the recipe. Fig. 1 provides a detailed running example for the section.

2.1 Recipe R

A recipe R is a piece of text that describes a list of instructions and a (possibly-empty) set of raw ingredients that are required to perform the instructions. The first step is to segment the text into a list of verb-argument tuples, called *actions*, $E_R = \{e_1 = (v_1, \mathbf{a}_1), \ldots, e_n = (v_n, \mathbf{a}_n)\}$. Sec. 6 will describe an unsupervised approach for learning to segment recipes. Each action e_i pairs a verb v_i with a list of arguments $\mathbf{a}_i$, where a_{ij} is the j^{th} argument of verb v_i. In Fig. 1, each row contains an action with a verb in the white oval and its arguments in the yellow and gray ovals.

Each argument is a tuple $a_{ij} = (t_{ij}^{syn}, t_{ij}^{sem}, S_{ij})$ with a syntactic type $t^{syn}(a) \in T^{syn} = \{DOBJ, PP\}$, a semantic type $t^{sem}(a) \in$

$T^{sem} = \{food, location, other\}$, and a list of text string spans $S_{ij} = \{s_{ij}^1, \ldots, s_{ij}^{|S_{ij}|}\}$, where s_{ij}^k is the k^{th} span in the j^{th} argument of verb v_i. In Fig. 1, the spans of each argument are represented by the white ovals inside of the argument ovals. For example, a_{21} contains a span for each raw ingredient being mixed in the second action (e.g., s_{21}^1 ="ground beef," s_{21}^6 ="brown sugar"). The syntactic type determines whether the argument is the direct object or a prepositional phrase argument of the verb in the recipe text. All other syntactic constructs are ignored and left for future work. The semantic types include food, location, and other. In Fig. 1, yellow ovals represent foods and gray ovals represent locations. Arguments of other semantic types are marked as $other$ (e.g., "Mash **using a fork**").

We also augment the set of arguments for each verb to include *implicit* arguments with empty string spans. This allows making connections to arguments that the author does not mention explicitly (e.g., the elided direct object of "bake" in e_5). Every verb is assigned one implicit PP argument, and, if a verb has no argument with syntactic type $DOBJ$, an implicit direct object. These arguments have indeterminate semantic types, which are to be determined based on how they connect to other actions. For example, in Fig. 1, when the implicit object of "bake" is connected to the output of the "lay" action, it is inferred to be of type $food$ since that is what is created by the "lay" action. However, when the implicit PP argument of "bake" is connected to the output of the "preheat" action, it is inferred to be a location since "preheat" does not generate a food.

2.2 Connections C

Given a segmented recipe, we can build graph connections. A *connection* identifies the origin of a given string span as either the output of a previous action or as a new ingredient or entity being introduced into the recipe. A connection is a six-tuple $(o, i, j, k, t^{syn}, t^{sem})$ indicating that there is a connection from the output of v_o to the argument span s_{ij}^k with syntactic type $t^{syn} \in T^{syn}$ and semantic type $t^{sem} \in T^{sem}$. We call o the *origin index* and i the *destination index*. For example, in Fig. 1, the connection from the output of the "press" verb (e_3) to "over the top" (s_{42}^1) would be $(3, 4, 2, 1, PP, food)$. If a span introduces raw ingredient or new location into the recipe, then $o = 0$; in Fig. 1, this occurs for each of the spans that represent raw ingredients as well as "oven" and "into loaf pan."

Given a recipe R, a set of connections C is valid for R if there is a one-to-one correspondence between spans in R and connections in C, and if the origin indexes of connections in C are 0 or valid verb indexes in R, $\forall(o, i, j, k, t^{syn}, t^{sem}) \in C,\ o \in \{\mathbb{Z} \mid 0 \leq o \leq |E_R|\}$.

2.3 Action graph G

A recipe R and a set of connections C define an *action graph*, which is a directed graph $G = (V, E)$. Each raw ingredient, verb, and argument span is represented by a vertex in V. Each argument span vertex is connected to its associated verb vertex, and each connection $c = (o, i, j, k, t^{syn}, t^{sem})$ adds a corresponding edge to E. Edges from connections with semantic type $food$ propagate ingredients through the recipe; edges from connections with semantic type $location$ propagate a location. Fig. 1 shows an action graph. By following the edges, we can tell that the implicit food entity that is being baked in the final action has been formed from the set of ingredients in the mixing action and the bacon from e_4 and that the baking action occurs inside the oven preheated in e_1.

3 Probabilistic connection model

Our goal is, given a segmented recipe R, to determine the most likely set of connections, and thus the most likely action graph. We model (1) a prior probability over C, $P(C)$ (Sec. 3.1), and (2) the probability of seeing a segmented recipe R given a set of connections C, $P(R|C)$ (Sec. 3.2). The most likely set of connections will maximize the joint probability: $P(R|C)P(C)$. A summary of this model is presented in Fig. 2, and the details are described in the this section.

3.1 Connections prior model

The probability of a set of connections C depends on features of the incoming set of connections for each action. Let a *destination subset* $\mathbf{d}_i \subseteq C$ be the subset of C that contains all connections that have i as the destination index. In Fig. 1, $\mathbf{d}_3$ contains the connection from v_2 to the implicit object as well as a connection to "into loaf pan" with an origin index of 0. Using the chain rule, the probability of C is equal to the product of the probability

- **Input:** A set of connections C and a recipe R segmented (**Sec. 6**) into its actions $\{e_1 = (v_1, \mathbf{a}_1), \ldots, e_n = (v_n, \mathbf{a}_n)\}$
- The joint probability of C and R is $P(C, R) = P(C)P(R|C)$, each defined below:

1. **Connections Prior (Sec. 3.1)**: $P(C) = \prod_i P(\mathbf{d}_i|\mathbf{d}_1, \ldots, \mathbf{d}_{i-1})$
   Define $\mathbf{d}_i$ as the list of connections with destination index i. Let $c_p = (o, i, j, k, t^{syn}, t^{sem}) \in \mathbf{d}_i$. Then,
   - $P(\mathbf{d}_i|\mathbf{d}_1, \ldots, \mathbf{d}_{i-1}) = P(vs(\mathbf{d_i})) \prod_{c_p \in \mathbf{d_i}} P(\mathbb{1}(o \rightarrow s^k_{ij})|vs(\mathbf{d_i}), \mathbf{d}_1, \ldots, \mathbf{d}_{i-1}, c_1, \ldots, c_{p-1})$
     (a) $P(vs(\mathbf{d_i}))$: multinomial verb signature model (**Sec. 3.1.1**)
     (b) $P(\mathbb{1}(o \rightarrow s^k_{ij})|vs(\mathbf{d_i}), \mathbf{d}_1, \ldots, \mathbf{d}_{i-1}, c_1, \ldots, c_{p-1})$: multinomial connection origin model, conditioned on the verb signature of $\mathbf{d}_i$ and all previous connections (**Sec. 3.1.2**)
2. **Recipe Model (Sec. 3.2)**: $P(R|C) = \prod_i P(e_i|C, e_1, \ldots, e_{i-1})$
   For brevity, define $\mathbf{h}_i = (e_1, \ldots, e_{i-1})$.
   - $P(e_i|C, \mathbf{h}_i) = P(v_i|C, \mathbf{h}_i) \prod_j P(a_{ij}|C, \mathbf{h}_i)$ (**Sec. 3.2**)
     Define argument a_{ij} by its types and spans, $a_{ij} = (t^{syn}_{ij}, t^{sem}_{ij}, S_{ij})$.
     (a) $P(v_i|C, \mathbf{h}_i) = P(v_i|g_i)$: multinomial verb distribution conditioned on verb signature (**Sec. 3.2**)
     (b) $P(a_{ij}|C, \mathbf{h}_i) = P(t^{syn}_{ij}, t^{sem}_{ij}|C, \mathbf{h}_i) \prod_{s^k_{ij} \in S_{ij}} P(s^k_{ij}|t^{syn}_{ij}, t^{sem}_{ij}, C, \mathbf{h}_i)$
        i. $P(t^{syn}_{ij}, t^{sem}_{ij}|C, \mathbf{h}_i)$: deterministic argument types model given connections (**Sec. 3.2.1**)
        ii. $P(s^k_{ij}|t^{syn}_{ij}, t^{sem}_{ij}, C, \mathbf{h}_i)$: string span model computed by case (**Sec. 3.2.2**):
           A. $t^{sem}_{ij} = food$ and $origin(s^k_{ij}) \neq 0$: IBM Model 1 generating composites (**Part-composite model**)
           B. $t^{sem}_{ij} = food$ and $origin(s^k_{ij}) = 0$: naïve Bayes model generating raw food references (**Raw food model**)
           C. $t^{sem}_{ij} = location$: model for generating location referring expressions (**Location model**)

Figure 2: Summary of the joint probabilistic model $P(C, R)$ over connection set C and recipe R.

of each of the destination subsets:

$$P(C) = \prod_i P(\mathbf{d}_i|\mathbf{d}_1, \ldots, \mathbf{d}_{i-1}).$$

The probability of each destination subset decomposes into two distributions, a verb signature model and a connection origin model:

$$P(\mathbf{d}_i|\mathbf{d}_1, \ldots, \mathbf{d}_{i-1}) = P(vs(\mathbf{d_i})) \times \prod_{c_p \in \mathbf{d_i}} P(\mathbb{1}(o \rightarrow s^k_{ij})|vs(\mathbf{d_i}), \mathbf{d}_1^{i-1}, c_1^{p-1}).$$

We define each of these distributions below.

3.1.1 Verb signature model

A destination subset $\mathbf{d}_i$ deterministically defines a *verb signature* g_i for verb v_i based on the syntactic and semantic types of the connections in $\mathbf{d}_i$ as well as whether or not each connection has a non-zero origin index. If the origin index is 0 for all connections in $\mathbf{d}_i$, we call v_i a *leaf*. (In Fig, 1, v_1 (preheat) and v_2 (mix) are leafs.) The formal definition of a verb signature is as follows:

Definition 1 The *verb signature* g_i for a verb v_i given a destination set $\mathbf{d}_i$ consists of two parts:

1. **type:** $\{t^{syn} \mid \exists (o, i, j, k, t^{syn}, food) \in \mathbf{d}_i\}$
2. **leaf:** true iff $(o, i, j, k, t^{syn}, t^{sem}) \in \mathbf{d}_i \Rightarrow o = 0$

For example, in Fig. 1, the signature for the "mix" action is $g_2 = (\{DOBJ\}, true)$ and the signature for the "lay" action is $g_4 = (\{DOBJ, PP\}, false)$. Given that there are two syntactic types (i.e., $DOBJ$ and PP) and each verb signature can either be labeled as a leaf or not, there are eight possible verb signatures.

We define a deterministic function that returns the verb signature of a destination subset: $vs(\mathbf{d}_i) = g_i$. $P(vs(\mathbf{d}_i))$ is a multinomial distribution over the possible verb signatures.

3.1.2 Connection origin model

We define $\mathbb{1}(o \rightarrow s^k_{ij})$ as an indicator function that is 1 if there is a connection from the action with index o to the span s^k_{ij}. The probability that a string span has a particular origin depends on (1) the verb signature of the span's corresponding verb, and (2) the previous connections. If, for example, g_i has **leaf**$= true$, then the origin of s^k_{ij} must be 0. If an origin has been used in a previous connection, it is much less likely to be used again.[2]

We assume that a destination subset is a list of connections: if $c_p \in \mathbf{d}_i$, we define c_1^{p-1} as the connections that are prior to c_p in the list. Similarly, $\mathbf{d}_1^{i-1}$ is the set of destination sets $(\mathbf{d}_1, \ldots, \mathbf{d}_{i-1})$. The connection origin model is a multinomial distribution that defines the probability of an origin for a span conditioned on the verb signature and all previous connections:

$$P(\mathbb{1}(o \rightarrow s^k_{ij})|vs(\mathbf{d_i}), \mathbf{d}_1^{i-1}, c_1^{p-1}),$$

[2] A counterexample in the cooking domain is separating egg yolks from egg whites to be used in separate components, only to be incorporated again in a later action.

where $c_p = (o, i, j, k, t^{syn}, t^{sem})$.

3.2 Recipe model

Given a set of connections C for a recipe R, we can determine how the actions of the recipe interact and we can calculate the probability of generating a set of recipe actions $E_R = \{e_1 = (v_1, \mathbf{a}_1), \ldots, e_n = (v_n, \mathbf{a}_n)\}$. Intuitively, R is more likely given C if the destinations of the connections are good text representations of the origins. For example, a string span "oven" is much more likely to refer to the output of the action "Preheat the oven" than "Mix flour and sugar."

We define the history $\mathbf{h}_i$ of an action to be the set of all previous actions: $\mathbf{h}_i = (e_1, \ldots, e_{i-1})$. The probability of a recipe R given a set of connections C can be factored by the chain rule:

$$P(R|C) = \prod_i P(e_i|C, \mathbf{h}_i).$$

Given C and a history $\mathbf{h}_i$, we assume the verb and arguments of an action are independent:

$$P(e_i|C, \mathbf{h}_i) = P(v_i|C, \mathbf{h}_i) \prod_j P(a_{ij}|C, \mathbf{h}_i).$$

Since the set of connections deterministically defines a verb signature g_i for a verb v_i, we can simplify $P(v_i|C, \mathbf{h}_i)$ to the multinomial distribution $P(v_i|g_i)$. For example, if g_i defines the verb to have an ingredient direct object, then the probability of "preheat" given that signature will be lower than the probability of "mix."

The probability of an argument $a_{ij} = (t^{syn}_{ij}, t^{sem}_{ij}, S_{ij})$ given the connections and history decomposes as follows:

$$\begin{aligned} P(a_{ij}|C, \mathbf{h}_i) &= P(t^{syn}_{ij}, t^{sem}_{ij}|C, \mathbf{h}_i) \\ &\times P(S_{ij}|t^{syn}_{ij}, t^{sem}_{ij}, C, \mathbf{h}_i). \end{aligned}$$

3.2.1 Argument types model

The first distribution, $P(t^{syn}_{ij}, t^{sem}_{ij}|C, \mathbf{h}_i)$, ensures that the syntactic and semantic types of the argument match the syntactic and semantic type of the incoming connections to spans of that argument. The probability is 1 if all the types match, 0 otherwise. For example, in Fig. 1, this distribution would prevent a connection from the "preheat" action to the food argument a_{42}, i.e., "over the top," since the semantic types would not match.

3.2.2 String span models

The second distribution, $P(S_{ij}|t^{syn}_{ij}, t^{sem}_{ij}, C, \mathbf{h}_i)$, models how likely it is to generate a particular string span given the types of its encompassing argument, the connections, and history. We assume the probability of each span is independent:

$$P(S_{ij}|t^{syn}_{ij}, t^{sem}_{ij}, C, \mathbf{h}_i) = \prod_{s^k_{ij} \in S_{ij}} P(s^k_{ij}|t^{syn}_{ij}, t^{sem}_{ij}, C, \mathbf{h}_i).$$

We break this distribution into three cases. To help describe the separate cases we define the function $origin(s, C)$ to determine the origin index of the connection in C to the span s. That is, $origin(s^k_{ij}, C) = o \Leftrightarrow \exists (o, i, j, k, t^{syn}, t^{sem}) \in C$.

Part-composite model When the encompassing argument is a food and the origin is a previous verb (i.e., $P(s^k_{ij}|t^{syn}_{ij}, t^{sem}_{ij} = food, origin(s^k_{ij}) \neq 0, C, \mathbf{h}_i)$), then the probability of the span depends on the ingredients that the span represents given the connections in C. For example, "dressing" is more likely given ingredients "oil" and "vinegar" than given "chicken" and "noodles". We use IBM Model 1 (Brown et al., 1993) to model the probability of a composite destination phrase given a set of origin food tokens. Let $food(s^k_{ij}, C)$ be the set of spans in food arguments such that there is a directed path from those arguments to s^k_{ij}. IBM Model 1 defines the probability of a span given the propagated food spans, $P(s^k_{ij}|food(s^k_{ij}, C))$.[3]

Raw food model When the encompassing argument is a food but the origin index is 0 (i.e., $P(s^k_{ij}|t^{syn}_{ij}, t^{sem}_{ij} = food, origin(s^k_{ij}) = 0, C, \mathbf{h}_i)$), then there is no flow of ingredients into the span. A span that represents a newly introduced raw ingredient (e.g., "bacon" in e_4 of Fig. 1) should have a high probability. However, spans that denote the output of actions (e.g, 'batter," "banana mixture") should have low probability. We use a naïve Bayes model over the tokens in the span $P(s|\text{is raw}) = \prod_\ell P(w_\ell|\text{is raw})$ where w_ℓ is the ℓ^{th} token in s (e.g., "mixture" would have a very low probability but "flour" would be likely).

Location model When the encompassing argument is a location (i.e., $t^{sem}_{ij} = location$),

[3] IBM Model 1 cannot handle implicit arguments. In this case, we model the probability of having an implicit food argument given the length of the connection (i.e., implicit food arguments nearly deterministically connect to the previous action). The probability of non-empty string spans is scaled accordingly to ensure a valid probability distribution.

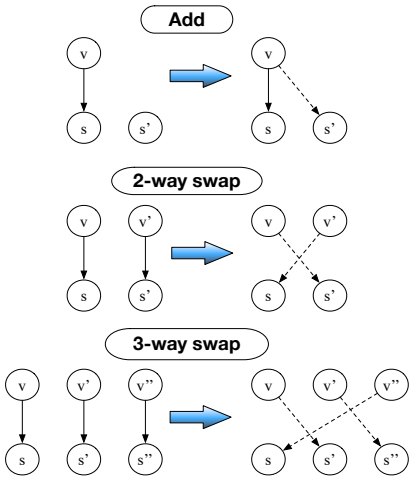


Figure 3: The three types of search operators. For swaps, one of the origins can be 0.

$P(S_{ij}|t_{ij}^{syn}, t_{ij}^{sem}, C, \mathbf{h})$ models the appropriateness of the origin action's location for the destination. If the string span is not implicit, the model deterministically relies on string match between the span and the location argument of the verb at the origin index. For example, the probability of "the preheated oven" conditioned on an origin with location "oven" is 1, but 0 for an origin with location "bowl." If the span s_{ij}^k is empty, we use a multinomial model $P(loc(origin(s_{ij}^k, C))|v_i)$ that determines how likely it is that an action v_i occurs in the location of the origin verb. For example, baking generally happens in an oven and grilling on a grill, but not vice versa. For example, in Fig. 1, the probability of the location span of "bake" is determined by P("oven" | "bake").

4 Local Search

Connections among actions and arguments identify which ingredients are being used by which action. For example, in Fig. 1, we know that we are baking something that contains all the ingredients introduced in e_2 and e_4 because there is a path of connections from the introduction of the raw ingredients to the implicit object of "bake". We cannot make decisions about the origins of arguments independently; the likelihood of each edge depends on the other edges. Identifying the most likely set of connections is, therefore, intractable.

We adopt a local search approach to infer the best set of connections.[4] We initialize the set of connections using a sequential algorithm that connects the output of each event to an argument of the following event, which is a strong baseline as shown in Sec. 8. We score potential local search operators that can be applied to the current set of connections C and make a greedy selection that improves $P(C, R)$ the most until no search operator can improve the probability. We constrain the search so all verbs have a direct object (i.e., implicit direct objects connect to a previous action).

We employ three types of search operators (see Fig. 3 for details). **OP_ADD** changes the origin index of a connection in C from 0 to the index of an event. **OP_2SWAP** swaps the origin indexes of two connections. This works even if one of the origin indexes is 0. **OP_3SWAP** rotates the origin indexes of three connections. This works even if one of the origin indexes is 0. For efficiency reasons, we only allow 3-way swaps with destination indexes within 4 events of each other.

Algorithm 1 Pseudocode for learning $P(C, R)$

Input: Initialized $P(C, R)$, recipe dataset $\mathcal{R}$
Repeat until convergence:
  E-step: Update $C \longleftarrow \arg\max_C P(C, R)$ for each $R \in \mathcal{R}$ using local search (Sec. 4)
  M-step: Update parameters of $P(C, R)$ using action graphs generated in E-step
Return $P(C, R)$

5 Learning

We use hard EM to learn the probabilistic models. Pseudocode is given in Alg. 1. At each iteration, we use our local search algorithm and the current probabilistic models to annotate each recipe in the data set with its most likely set of connections C (Sec. 4). Then, we re-estimate the parameters of the probabilistic models using the recipe-connections pairs as training data. A small (33 recipes) development set was used to determine when to stop the iterations. Experimental details and model initialization are described in Sec. 7.

6 Segmentation

Our inference and learning algorithms assume as input a recipe segmented into a set of events $E_R = \{(v_1, \mathbf{a}_1), \ldots, (v_n, \mathbf{a}_n)\}$. We designed a segmentation system that could be trained on our unannotated data set of mostly imperative sentences.

[4] Similar local search methods have been shown to work well for other NLP tasks, including recent work on dependency parsing (Zhang et al., 2014).

Our system achieves an F1 score of 95.6% on the task of identifying the correct verbs in the test set.[5]

Segmentation model We define a generative model for recipes as:

$$P(R) = P(n) \prod_{i}^{n} P(v_i) P(m \mid v_i) \prod_{j=1}^{m} P(a_{ij}).$$

We first select a number of verbs n in the recipe from a geometric distribution. Given the number of verbs, we select a set of verbs $\mathbf{V} = \{v_1, \ldots, v_n\}$ using a multinomial distribution. For each verb v_i, we select a number of arguments m from a separate multinomial distribution that has the probability of 0, 1, 2, or 3+ arguments given the verb, $P(m \mid v_i)$. For each argument, we generate a string using a bigram model, $P(a_{ij}) = \prod_{\ell} P(w_\ell | w_{\ell-1})$, where w_ℓ is the ℓ^{th} word of a_{ij}.

Inference Given tokenized sentence $T = (w_1, \ldots, w_k)$, we enumerate all possible segmentations and choose the one with the highest probability. To keep this efficient, we use a closed set of possible verbs and assume a closed set of words (e.g., prepositions, adverbs) can only follow the start token in the argument bigram model. Thus, annotating the verbs in a sentence determines a unique set of argument strings. Despite scoring the segmentations for all possible sets of verbs, we found the process to be very efficient in practice.

Learning For unsupervised learning, we again employ a hard EM approach. We initialize our models, segment all of the training data, re-estimate the parameters, and iterate these steps until performance on a development set converges.

We estimate the initial verb multinomial model using counts from the first word of each sentence in the dataset, which are normally verbs in imperative sentences, and filter out any words that have no verb synsets in WordNet (Miller, 1995). All other models are initialized to be uniform.

7 Experimental Setup

Data Set We collected 2456 recipes (with over 23,000 sentences) from allrecipes.com by searching for 20 dish names (e.g., including "banana muffins", and "deviled eggs"). We randomly sampled, removed, and hand labeled 33 recipes for a development set and 100 recipes for test. All models were trained on the unannotated recipes; the dev set was used to determine the stopping point for training. Each recipe in the test set has 13 actions on average.

Recipe pre-processing To pre-process each recipe, we first use the segmentation system described in Sec. 6. Then, we use a string classification model to determine the semantic type (e.g., $food$, $location$, or $other$) of an argument based on its spans. We identify spans as raw ingredients based on string match heuristics (e.g., in Fig. 1, the span "crushed crackers" represents the ingredients "crushed butter-flavored crackers"). We stem all words and ignore function words.

Sequential Baseline Because most connections are sequential – i.e., argument spans are most often connected to the output of the previous verb – sequential connections make a strong baseline; we connect the output of each predicate to the first available argument span of the following predicate. If no argument exists, an implicit argument is created. We run this baseline with and without first identifying raw ingredients in the recipe; if raw ingredient spans are identified, the baseline will not connect the previous event to those spans. Performance suffers significantly if the raw ingredients are not identified beforehand.

Evaluation metrics We report F-measure by comparing the predicted connections from actions to spans (i.e., where the origin index > 0) against gold standard annotations. We don't evaluate connections to raw ingredients as we create those connections during pre-processing (see Sec. 7).

Model initialization The verb signature model (Sec. 3.2) is initialized by first identifying $food$ arguments using string overlap with the ingredient list. All other arguments' types are considered unknown, and partial counts were awarded to all verb signatures consistent with the partial information. The first verb in each recipe was assumed to be the only leaf. The string classification model for the pre-processing step was initialized by using the initialized verb signature model to identify the types of $DOBJ$ arguments. The string classification model was estimated using the argument tokens given the types. We initialized the part-composite model (Sec. 3.2.2) so that exact string matches between ingredients and spans are given

[5] Early efforts using a state-of-the-art parser could only achieve an F1 score of 73.6% for identifying verbs, likely due to a lack of imperative sentences in the training data. This result motivated us to develop our segmentation system.

Algorithm	Prec	Rec	F1
Automatic segmentations			
Sequential baseline	55.7	52.7	54.1
Sequential baseline w/ ingredients	60.4	57.2	58.8
Our model before EM	65.8	62.7	64.2
Our model after EM	**68.7**	**65.0**	**66.8**
Oracle segmentations			
Sequential baseline	67.8	65.2	66.5
Sequential baseline w/ ingredients	73.5	70.7	72.0
Our model before EM	77.1	74.8	75.9
Our model after EM	**81.6**	**78.5**	**80.0**

Table 1: Performance of our algorithm against the sequential baselines.

Verb	Top location tokens	
bake	oven - 55.4%	min - 0.7%
mix	bowl - 32.6%	hand - 0.9%
press	pan - 24.7%	dish - 6.5%
stir	bowl - 5.5%	skillet - 2.0%
fry	heat - 11.9%	skillet - 10.2%
cool	rack - 10.5%	pan - 3.8%
boil	water - 15.5%	saucepan - 5.2%

Table 2: The top scoring location token for example verbs. The percentage is the percent of times the verb has that as a visible location token.

high probabilities and those without are given low probabilities. Given the initialized string classification model, the raw food model (Sec. 3.2.2) is initialized counting whether or not tokens in food arguments occur in the ingredient list. The probability of an implicit location (Sec. 3.2.2) is initialized to a hand-tuned value using the dev set.

8 Results

Quantitative Results We trained our model for four iterations of hard EM until performance converged on the development set. Table 1 presents our results on the test set. We compare our model to the sequential baselines using both the output of our segmentation system and oracle segmentations. We perform significantly better than the sequential baselines, with an increase in F1 of 8 points over the more competitive baseline using our segmentation system and an increase of 8 points using the oracle segmentations.

Qualitative Results We find that the learned models demonstrate interpretable cooking knowledge. Table 3 shows the top composite tokens for different ingredients as learned by the part-composite model (Sec. 3.2.2). The composite tokens show parts of the ingredient (e.g., after "eggs" can be split into "whites" or "yolks") or composites that are likely to contain an ingredient (e.g., "flour" is generally found in "batter" and "dough"). Unsurprisingly, the word "mixture" is one of the top words to describe a combination of ingredients, regardless of the ingredient. The model also learns modifiers that describe key properties of ingredients (e.g., flour is "dry" but bananas are "wet") which is important when evaluating connections for sentences such as "Fold the wet mixture into the dry ingredients."

Verb	Top verb signature	(%)
add	$\{DOBJ, PP\}$	58%
	$\{DOBJ\}$	27%
combine	$\{DOBJ\}$:leaf	68%
	$\{DOBJ\}$	17%
bake	$\{DOBJ\}$	95%
grease	{}:leaf	75%
pour	$\{DOBJ, PP\}$	68%
	$\{DOBJ\}$	27%
reduce	$\{PP\}$	90%
	$\{DOBJ\}$	8%

Table 4: The top verb signatures for example verbs. The syntactic types identify which arguments of the verb are foods and "leaf" means no arguments of the verb connect to previous actions.

Table 2 shows the location preferences of verbs learned by the location model (Sec. 3.2.2). Some verbs show strong preferences on locations (e.g., "bake" occurs in an oven, "mix" in a bowl). The top location for a "boil" action is in "water," but in other recipes "water" is an ingredient.

Finally, Table 4 shows learned verb signatures. For example, "add" tends to be a non-leaf action, and can take one or two food arguments (e.g., one food argument: "Heat the pan. Add onions." vs. two food arguments: "Add the wet mixture to the dry mixture.") We learn that the most likely verb signature for "add" has two food arguments; since over 74% of the occurrences of "add" in the dataset only have one visible argument, the segmentation alone is not enough to determine the signature.

Errors Finally, we performed an error analysis on the development set. 24% of the errors were due to missing or incorrect actions caused by segmentation errors. Among the actions that were segmented correctly, 82% of the outgoing connections were sequential. Of those, our system missed 17.6% of the sequential connections and 18.3% of the non-sequential connections.

Ingredient	Top composite tokens
eggs	egg, yolk, mixture, noodles, whites, cook, top, potato, cold, fill
beef	beef, mixture, grease, meat, excess, cook, top, loaf, sauce, ground
flour	flour, mixture, dough, batter, top, crust, ingredients, sauce, dry, pie
noodles	noodles, cook, mixture, egg, sauce, top, meat, drain, pasta, layer
chicken	chicken, mixture, salad, cook, dressing, pasta, soup, breast, vegetables, noodles
pumpkin	pumpkin, mixture, pie, filling, temperature, seeds, mash, oven, crust, dough
bananas	banana, mixture, batter, muffin, bread, egg, wet, cup, ingredients, slice

Table 3: Examples of ingredients with their top inferred composite words.

9 Related work

Our work relates to a substantial body of research that transforms natural language instructions into actionable plans (Artzi and Zettlemoyer, 2013, Chen and Mooney, 2011, Branavan et al., 2011, Branavan et al., 2009, McMahon et al., 2006). Most of these approaches do interactive learning in virtual environments or simulations, while we learn from the redundancy seen in the text of different instances of similar recipes.

There is also significant related work on supervised learning for instructions. A recent series of studies have explored parsing of cooking recipes (Mori et al., 2012; Mori et al., 2014; Maeta et al., 2015). However, they assume annotated data, study Japanese recipes, and make edge connections independently without taking into account the flow of ingredients. Tasse and Smith (2008) develops annotation for English recipes, but do not mark connections from implicit roles, and only studied segmentation models. Lau et al. (2009) develop models to interpret how-to instructions, but also assume supervision, and do not make connections between different actions.

Data-driven extraction of cooking knowledge has been explored in the context of building a cooking ontology (Gaillard et al., 2012; Nanba et al., 2014). In contrast, our work induces probabilistic cooking knowledge as part of unsupervised learning process for understanding recipes. Cooking knowledge is also closely related to script knowledge, but most prior work focus on newswire and children's books rather than procedural language (Fujiki et al., 2003; Chambers and Jurafsky, 2009; Pichotta and Mooney, 2014; Balasubramanian et al., 2013) or rely on crowdsourced descriptions to learn procedural knowledge (Regneri et al., 2010; Regneri et al., 2011; Frermann et al., 2014). There is work on related, but distinct, tasks that use recipes, including identifying actionable refinements from online recipe reviews (Druck and Pang, 2012) and extracting structured information from ingredient lists (Greene, 2015)

Cooking recipes have also been studied in the context of grounded language learning, e.g., to build robots that can cook (e.g., Bollini et al., 2013, Beetz et al., 2011), or to align cooking videos to natural language descriptions of actions (Regneri et al., 2013) or recipe texts (Malmaud et al., 2014; Malmaud et al., 2015). Our work complements these efforts by recovering fine-grained procedural semantics from text alone.

Finally, detection and resolution of implicit arguments is an instance of zero anaphora detection and resolution (Silberer and Anette, 2012, Tetreault 2002, Whittemore et al., 1991, Palmer et al., 1986). We present an empirical approach for understanding these phenomena in instructions.

10 Conclusion

We presented unsupervised methods for segmenting and identifying latent connections among actions in recipe text. Our model outperformed a strong linear baseline, while learning a variety of domain knowledge, such as verb signatures and probable ingredient components for different composites. Future work includes learning a more comprehensive model of locations (e.g., identifying nested locations such as an oven and a pan in the oven), enriching action graphs with greater semantic coverage (e.g., durations, tools, amounts), and training and evaluating on larger datasets. We also plan to use our techniques to support related tasks, such as instructional recipe generation.

Acknowledgments

We thank the anonymous reviewers, Mike Lewis, Dan Weld, Yoav Artzi, Antoine Bosselut, Kenton Lee, Luheng He, Mark Yatskar, and Gagan Bansal for helpful comments, and Polina Kuznetsova for the preliminary work. This research was supported in part by the Intel Science and Technology Center for Pervasive Computing (ISTC-PC) and the NSF (IIS-1252835 and IIS-1524371).

References

Yoav Artzi and Luke Zettlemoyer. 2013. Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association for Computational Linguistics*, 1(1):49–62.

Niranjan Balasubramanian, Stephen Soderland, Mausam, and Oren Etzioni. 2013. Generating coherent event schemas at scale. In *Proceedings of the 2013 Conference on Empirical Methods on Natural Language Processing*, pages 1721–1731.

Michael Beetz, Ulrich Klank, Ingo Kresse, Alexis Maldonado, Lorenz Mosenlechner, Dejan Pangercic, Thomas Ruhr, and Moritz Tenorth. 2011. Robotic roommates making pancakes. In *Proceedings of the 11th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 529–536.

Mario Bollini, Stefanie Tellex, Tyler Thompson, Nicholas Roy, and Daniela Rus. 2013. Interpreting and executing recipes with a cooking robot. *Experimental Robotics*, 88:481–495.

S.R.K. Branavan, Harr Chen, Luke Zettlemoyer, and Regina Barzilay. 2009. Reinforcement learning for mapping instructions to actions. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1 - Volume 1*, pages 82–90.

S.R.K. Branavan, David Silver, and Regina Barzilay. 2011. Non-linear monte-carlo search in civilization ii. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence*, pages 2404–2410.

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

Nathanael Chambers and Dan Jurafsky. 2009. Unsupervised learning of narrative schemas and their participants. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing*, pages 602–610.

David L. Chen and Raymond J. Mooney. 2011. Learning to interpret natural language navigation instructions from observations. In *Proceedings of the 25th AAAI Conference on Artificial Intelligence (AAAI-2011)*, pages 859–865.

Gregory Druck and Bo Pang. 2012. Spice it up? Mining refinements to online instructions from user generated content. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, pages 545–553.

Charles J. Fillmore, 1982. *Frame semantics*, pages 111–137. Hanshin Publishing Co., Seoul, South Korea.

Lea Frermann, Ivan Titov, and Manfred Pinkal. 2014. A hierarchical bayesian model for unsupervised induction of script knowledge. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 49–57.

Toshiaki Fujiki, Hidetsugu Nanba, and Manabu Okumura. 2003. Automatic acquisition of script knowledge from a text collection. In *Proceedings of the Tenth Conference on European Chapter of the Association for Computational Linguistics - Volume 2*, pages 91–94.

Emmanuelle Gaillard, Emmanuel Nauer, Marie Lefevre, and Amélie Cordier. 2012. Extracting generic cooking adaptation knowledge for the TAAABLE case-based reasoning system. In *Proceedings of the 1st Workshop on Cooking with Computers (CwC)*.

Erica Greene. 2015. Extracting structured data from recipes using conditional random fields. The New York Times Open Blog.

TA Lau, Clemens Drews, and Jeffrey Nichols. 2009. Interpreting written how-to instructions. In *Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence*, pages 1433–1438.

Matt MacMahon, Brian Stankiewicz, and Benjamin Kuipers. 2006. Walk the talk: Connecting language, knowledge, and action in route instructions. In *Proceedings of the 21st National Conference on Artificial Intelligence - Volume 2*, AAAI'06, pages 1475–1482. AAAI Press.

Hirokuni Maeta, Tetsuro Sasada, and Shinsuke Mori. 2015. A framework for procedural text understanding. In *Proceedings of the 14th International Conference on Parsing Technologies*, pages 50–60.

Jon Malmaud, Earl J. Wagner, Nancy Chang, and Kevin Murphy. 2014. Cooking with semantics. In *Proceedings of the ACL 2014 Workshop on Semantic Parsing*, pages 33–38.

Jonathan Malmaud, Jonathan Huang, Vivek Rathod, Nick Johnston, Andrew Rabinovich, and Kevin Murphy. 2015. What's cookin'? Interpreting cooking videos using text, speech and vision. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 143–152.

George A. Miller. 1995. WordNet: A lexical database for english. *Communications of the ACM*, 38(11):39–41.

Shinsuke Mori, Tetsuro Sasada, Yoko Yamakata, and Koichiro Yoshino. 2012. A machine learning approach to recipe text processing. In *Proceedings of the 1st Workshop on Cooking with Computers (CwC)*.

Shinsuke Mori, Hirokuni Maeta, Yoko Yamakata, and Tetsuro Sasada. 2014. Flow graph corpus from recipe texts. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 26–31.

Hidetsugu Nanba, Yoko Doi, Miho Tsujita, Toshiyuki Takezawa, and Kazutoshi Sumiya. 2014. Construction of a cooking ontology from cooking recipes and patents. In *Proceedings the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication*, pages 507–516.

Martha S. Palmer, Deborah A. Dahl, Rebecca J. Schiffman, Lynette Hirschman, Marcia Linebarger, and John Dowding. 1986. Recovering implicit information. In *Proceedings of the 24th Annual Meeting on Association for Computational Linguistics*, pages 10–19.

Karl Pichotta and Raymond Mooney. 2014. Statistical script learning with multi-argument events. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 220–229.

Michaela Regneri, Alexander Koller, and Manfred Pinkal. 2010. Learning script knowledge with web experiments. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 979–988.

Michaela Regneri, Alexander Koller, Josef Ruppenhofer, and Manfred Pinkal. 2011. Learning script participants from unlabeled data. In *Proceedings of the Conference on Recent Advances in Natural Language Processing*, pages 463–470.

Michaela Regneri, Marcus Rohrbach, Dominikus Wetzel, Stefan Thater, Bernt Schiele, and Manfred Pinkal. 2013. Grounding action descriptions in videos. *Transactions of the Association for Computational Linguistics (TACL), Volume 1.*, 1:25–36.

Roger Carl Schank and Robert P. Abelson. 1977. *Scripts, plans, goals and understanding : an inquiry into human knowledge structures*. The Artificial intelligence series. L. Erlbaum, Hillsdale, N.J.

Carina Silberer and Anette Frank. 2012. Casting implicit role linking as an anaphora resolution task. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics - Volume 1: Proceedings of the Main Conference and the Shared Task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*, pages 1–10.

Dan Tasse and Noah A. Smith. 2008. SOUR CREAM: Toward semantic processing of recipes. Technical Report CMU-LTI-08-005, Carnegie Mellon University, Pittsburgh, PA.

Joel R. Tetreault. 2002. Implicit role reference. In *Proceedings of the International Symposium on Reference Resolution for Natural Language Processing*, pages 109–115.

Greg Whittemore, Melissa Macpherson, and Greg Carlson. 1991. Event-building through role-filling and anaphora resolution. In *Proceedings of the 29th Annual Meeting on Association for Computational Linguistics*, pages 17–24.

Yuan Zhang, Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2014. Greed is good if randomized: New inference for dependency parsing. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1013–1024.

Arnold M. Zwicky. 1988. On the subject of bare imperatives in english. In C. Duncan-Rose and T. Vennemann, editors, *On Language: Rhetorica, Phonologica, Syntactica - A Festschrift for Robert P. Stockwell from His Friends and Colleagues*, pages 437–450. Routledge, London.

Semantic Framework for Comparison Structures in Natural Language

Omid Bakhshandeh
University of Rochester
omidb@cs.rochester.edu

James F. Allen
University of Rochester
Institute for Human and Machine Cognition
james@cs.rochester.edu

Abstract

Comparison is one of the most important phenomena in language for expressing objective and subjective facts about various entities. Systems that can understand and reason over comparative structure can play a major role in the applications which require deeper understanding of language. In this paper we present a novel semantic framework for representing the meaning of comparative structures in natural language, which models comparisons as predicate-argument pairs interconnected with semantic roles. Our framework supports not only adjectival, but also adverbial, nominal, and verbal comparatives. With this paper, we provide a novel dataset of gold-standard comparison structures annotated according to our semantic framework.

1 Introduction

Representing the meaning of text has long been a focus in linguistics and deriving computational models of meaning has been pursued by various semantic tasks such as semantic parsing. Deep semantic parsing (as opposed to shallow semantic parsing, such as semantic role labeling) aims to map a sentence in natural language into its corresponding formal meaning representation (Zelle and Mooney, 1996; Berant and Liang, 2014). There has been a renewed interest in deeper semantic representations of natural language (Banarescu et al., 2013) in NLP community. Open-domain semantic representations enable inference and reasoning, which is required for many language understanding tasks such as reading comprehension tests and open-domain question answering. Comparison is a common way for expressing differences in sentiment and other properties towards some entity. Comparison can happen in very simple structures such as 'John is taller than Sam', or more complicated constructions such as 'The table is longer than the sofa is wide'. So far the computational semantics of comparatives and how they affect the meaning of text has not been studied effectively. That is, the difference between the existing semantic and syntactic representation of comparatives is not distinctive enough for enabling deeper understanding of a sentence. For instance, the general logical form representation of the sentence 'John is taller than Susan' using the Boxer system (Bos, 2008) is the following:

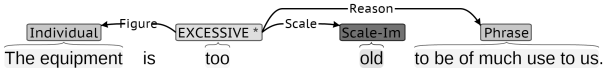
$$named(x0, john, per) \\ \& \; named(x1, susan, nam) \\ \& than(taller(x0), x1) \quad (1)$$

The above meaning representation does not fully capture the underlying semantics of the adjective 'tall' and what it means to be 'taller'. A human reader can easily infer that actually the height of John is greater than the height of Susan. Another example to consider is the sentence 'John is tall', which basically has the typical logical form $tall(john)$ –which is a very superficial representation for the meaning of the predicate 'tall'. Likewise, a human reader can infer that defining someone as 'tall' in some domain of discourse entails that this person is somehow 'taller' than some other population (say their average), however, the earlier typical logical form representation does not enable such inferences.

In this paper we introduce a novel framework for semantic representation and computational analysis of the structure of comparison in natural language. This framework enables deeper representation of semantics of comparatives, including all different types of comparison within *compara-*

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 993–1002,
Lisbon, Portugal, 17-21 September 2015.

tives, superlatives, equatives, excessives, and assetives, and the way they are related to their corresponding semantic roles. Together with this paper, we provide a dataset of gold-annotated comparative structures using our meaning representation, which enables training models on comparison constructions. We propose a new approach for automatic extraction of comparison structures from a given text. A semantic representation of the comparison expressed by the sentence 'The equipment is too old to be much of use to us.' augmented under our representation would be the following:



Throughout this paper we define a comparison to be any statement comparing two or more entities, expressing some kind of measurement on a scale, or indicating some degree of having a measurable property. The details of these variations will be discussed in Section 3.

2 Background

In this section we provide a linguistic background on comparison constructions in language, which provides the basis of our semantic framework (to be presented in Section 3).

2.1 Comparative Structures in Language

Measurement in natural language is mainly expressed in sentences having comparative morphemes such as *more, less, -er, as, too, enough, -est, etc*[1]. Comparatives can be either adjectival, adverbial, nominal, or verbal, i.e., the main component of the sentence carrying out the measurement can have either of these parts of speech.

Adjectival Comparatives: Canonical examples of comparative sentences contain adjectives, e.g., 'tall' or 'pretty'. Even within adjectival comparatives, there is a good deal of structural variety. Consider the following examples:

(1) a. Mary is *taller* than Susan.
b. Mary is 3 inches *taller* than Susan.
c. Mary is *taller* than 5 feet.

[1]These morphemes are counted as the main comparison operators. For easier representation, throughout this paper we specify the smallest constituent containing any of these morphemes as the comparison operator, which is *italicized* in the sentences.

The comparative form of the adjective 'tall' in sentence 1a is viewed as an expression denoting a *greater than ($\succ$)* relation between two individuals, 'Mary' and 'Susan', on the scale of 'tallness'. The degree-theoretic analysis of such adjectives brings up the notion of *Gradable Adjectives*: many adjectives describe qualities that can be measured according to *degrees* on *scales*, such as the scale of 'size', 'beauty', 'age', etc. These adjectives can be used with comparative morphemes, indicating less or more of a particular quality on a scale. Gradable adjectives can express specific relations between individuals on a scale, e.g., in sentence 1b Mary is taller than Susan by a measure of 3 inches.

Comparison on the scale does not always involve two individuals. For example consider sentence 1c which denotes a comparison being made between an individual and a specific point on the scale of 'tallness'. All the earlier examples are among the simplest types of comparative structures using adjectives. Consider the following example:

(2) Mary is *taller* than the bed is long.

In sentence 2 we have a case of *subcomparatives*, where we compare 'Mary' and 'bed' according to two different dimensions: height and length. Each dimension provides a degree, and the degrees are ultimately related by the *greater than ($\succ$)* relation. Scalability is known to be universal in language and a wide variety of linguistic phenomena can be explained in terms of degrees and scales (Solt, 2015).

The Semantics of Scales: A fairly common view (Kennedy, 2007) is that a scale S is a triple of the following form:

$$S = \langle D, \succ, DIM \rangle \quad (2)$$

where D is a set of degrees, $\succ$ is an ordering relation on D, and DIM is the dimension of measurement.[2]

Individuals are linked to degrees by measure functions. A measure function μ_S is the function that maps an individual x to the degree on the scale S that represents x's measure with respect to the dimension DIM. For example, the μ_{HEIGHT} measure function is a function that maps individuals to their respective heights. Under this model, we represent the comparative structure of the sentences 1a-1c as follows:

[2]To know more about the theory of scales and restrictions on dimensions and degrees refer to (Solt, 2015).

(3) a. $\mu_{HEIGHT}(Mary) \succ \mu_{HEIGHT}(Susan)$
b. $\mu_{HEIGHT}(Mary) \succeq \mu_{HEIGHT}(Susan) + 3$"
c. $\mu_{HEIGHT}(Mary) \succ 5\ feet$

A generic comparative interpretation of some degree of tallness under $HEIGHT$ scale is as follows:

$$[\![tall]\!] = \lambda d \lambda x.\mu_{HEIGHT}(x) \succeq d \quad (3)$$

where d is the degree argument which is supplied by some form of degree morpheme: a degree modifier (e.g., too, very), a measure phrase (e.g. 1.7 inches), or simply comparative or superlative morphology. Under this model, we can also represent the comparative structure of the sentence 'Mary is tall'[3], where there is no explicit degree argument. A common assumption is that the degree role is played by a phonologically null degree morpheme called pos, which denotes a context-dependent threshold or standard of comparison (Kennedy, 2007; Heim, 2007). For instance, in a specific context of adult men in north America being 'tall' could be interpreted as being over 6 feet.

Non-canonical Comparatives: Comparative structures can also be verbal, nominal, and adverbial. Consider the following verbal comparatives:

(4) a. The women ate *more* than men did.
b. The lake cooled *more* than 4 degrees.

It has been proposed (Wellwood et al., 2012) that measure functions (μ) can be applied both to individuals and to events, in the latter case measuring either the event or an entity related to the event. The comparative interpretation for the two sentences 4a and 4b is as follows:

(5) a. $\mu_{volume}(eat(women)) \succ \mu_{volume}(eat(men))$
b. $\lambda e.\mu_{coolness}(e)(lake) \succ 4\ degrees$

where cool is a function that takes an event e and an object x (here 'lake') and returns a degree representing the amount to which x changes in coolness as a result of participating in e. The underlying scale of verbal comparatives is sometimes ambiguous, e.g., in sentence 5a it is not clear whether the women ate more in volume or in quantity.

Comparative structures can also be nominal. Consider the following sentences:

(6) a. *More* juniors than seniors came to the ceremony.
b. We bought *more* milk than wine.

The meaning of sentences presented above must be stated with reference to degrees as well (Solt, 2015). Hence, the scale for the comparison sentence 6a is the numerical counting by integers and the scale for sentence 6b is something corresponding to a mass dimension, here perhaps liquid volume. Adverbial comparatives share many of their characteristics with the adjectival and verbal class, which we do not develop further for brevity. For example the sentence 'Mary ran *faster* than Sam' is an example of adverbial comparison, where the implicit 'speed' attribute of the 'running' event associated with Mary and Sam is being compared.

2.2 Categories of Comparison

There are various ways for making comparisons, each indicating different degrees of difference or similarity. Following are the major categories for degrees of comparison together with example sentences[4]:

(7) Comparative
a. Mary is *taller* than Susan.
b. Dogs are *more* intelligent than rabbits.

(8) Superlative
a. Mary is the *tallest* girl in her class.
b. Dogs are the *most* intelligent among pets.

(9) Equative
a. Mary is *as* tall as Bill.
b. Dogs are *as* intelligent as cats.

(10) Excessive
a. Mary is *too* short for basketball.
b. Dogs are *too* intelligent to be fooled.

(11) Assetive
a. Mary is tall *enough* to reach the shelf.
b. Dogs are intelligent *enough* to find their way home.

3 Semantic Framework of Comparison

As discussed earlier, having a deep meaning representation of comparison structures can help us

[3]Such cases are called *positive* usage of the adjective. The *negative* (also called antonym) usage would be 'Mary is short'.

[4]As shown in example sentences of Table 1, there can be non-adjectival comparisons in each of these categories as well.

build computational models of comparison in natural language and perform inferential tasks in various domains. Here we introduce a novel semantic framework of comparison. This framework is based on the linguistic interpretations presented in Section 2, but formalized and adapted to suit our semantic computational framework.

We model comparatives as inter-connected predicate-argument structures, where predicates are the main comparison operators (implicit and explicit comparison morphemes), and arguments are connected to the predicates via semantic roles (relations). Our framework includes not only explicit comparisons, but also implicit ones in the form of an evaluation or a measurement on a scale, which will be explained throughout this section. More detailed and complete list of the predicates, semantic roles, and arguments can be found in the supplementary material.

Predicates: Table 1 lists all the predicate operators under our framework. As the table shows, there are four main types of predicates: *comparatives, extremes, bases, and measurements*. Most of these types can be associated with operators from any of our parts of speech: Adjective (JJ), Adverb (RB), Noun (NN), and Verb (VB). The predicate operator in each of the examples is italicized. The *comparatives* type also includes the operators < and =<, which are the opposite of the operators > and >= presented in the table. It is important to note that the 'base positive' predicate is actually the implicit pos operator (as mentioned in Section 2.1; however, for easier representation we specify it by marking its corresponding adjective or adverb. The same thing happens for measurement predicates. Also, our framework captures the subtle difference between the meaning of 'Mary is $[tall]_{positive}$' and 'Mary is 5 feet $[tall]_{measurement-explicit}$'. The earlier means that Mary is tall according to some standard of tallness in a context, while the latter means that Mary's height equals the degree of 5 feet.

Semantic Roles: Each predicate is characterized by its arguments and each argument is connected to the predicates by a relation (semantic role) type. Table 2 shows the possible semantic role types for a predicate. *Figure* is the core role for a comparison structure, i.e., any comparison should have a role indicating the main entity which is being evaluated/measured/compared on a scale. The simplest form of comparative predicate, e.g., 'John is taller than Sam', involves two main roles: Figure (John) and Ground (Sam). The non-core roles are mainly associated with non-comparative comparisons

Arguments: Last but not least, each role points to an argument, which can have various types, as listed in Table 3. The most frequent argument type is individual, as in 'John is taller than Sam'. The other notable role is Phrase-value, which represents an interesting comparison phenomena. In the corresponding example in the table, the speed of John's driving is explicitly being compared with some point on the scale of speed to which 'he was allowed'. Such *ground* roles are classified as phrase-value, where a verb phrase signifies a point of comparison on scale, not an individual entity. Figure 1 shows an example of predicate-argument structure under the described semantic framework.

4 Predicting Comparison Structures

Given an input sentence, we want to predict the predicate operators, their semantic roles, and arguments. We decompose this problem into three sub-problems:

- Labeling predicate candidates using a multi-class classifier
- For each predicate, considering the set of all possible argument spans:
  - Use a classifier for predicting the role type label
  - Use a classifier for predicting the argument type label

Our overall approach, to be described in this section, is similar to the works on joint inference with global constraints for learning event relations and process structures (Do et al., 2012; Berant et al., 2014).

Predicting Predicates: The first step in comparison structure prediction is to identify and label the predicates. For this purpose we train a multi-class classifier that labels all one-word constituents in the sentence with any of predicate types in Table 1 or *None* (indicating that the constituent is not a predicate). The set of all possible predicate labels is named P.

We used various features for training the predicate classifier: we extract the lemma and POS tag of the word, POS tag of children, siblings, parent and root of the sentence in the dependency

Predicate Type	Subtype	Examples
Comparatives: Comparing against one or more entities.	>	• JJ: The car was *more* modern than I had imagined. • RB: John ran *faster* than Susan. • NN: *More* cookies than cakes were purchased. • VB: Coffee is *less* consumed than tea.
	>=	• JJ: Pizza is *as* expensive as pasta. • RB: The men ran *as* fast as the women did. • NN: That college hires *as* much professors as we do. • VB: Athletes drink as much as others.
	Superlative	• JJ: Mary is the *tallest* among her colleagues. • RB: Mike talked the *most* loudly of the group. • NN: The juniors found the *most* rock of all. • VB: The fire fighters ran the *most* among others.
Extreme: Indicating having enough or too much of a quality or quantity.	Excessive	• JJ: Mary is *too* tall to fit in the chair. • RB: Sam ran *too* fast to get caught. • NN: There are *too* many students at the party. • VB: The kid screamed *too* much.
	Assetive	• JJ: Mary is smart *enough* to accept the offer. • RB: The machine works steadily *enough*. • NN: There are *enough* professors at the party. • VB: Jack passed *enough* interviews to prove himself.
Polarity: Base form expression of +/- quality.	Positive	• JJ: Mary is *tall*. • RB:John talks *beautifully*.
	Negative	• JJ: Susan is *short*. • RB: Philip walks *slowly*.
Measurement: Indicating a measurement on a scale.	Explicit	• JJ: Mary is 5 feet *tall*. • RB: Philip is driving 60mph *fast*.
	Implicit	• JJ: Mary is *5 feet*. • RB: Philip is driving *60mph*.

Table 1: The predicate types defined under our framework.

Figure 1: A full annotation of a sample predicate-argument structure under the described semantic framework.

tree, POS tag and lemma of two adjacent words, similarity features from WordNet (Miller, 1995), word polarity features, and most importantly 'attribute concepts' for words which are adjectives (Bakhshandeh and Allen, 2015). The 'attribute concepts' are the different properties that an adjective can describe, for instance 'height' and 'thickness' are the attributes of the adjective 'gangling'. Last but not least, we include the conjunction of all these features.

Predicting Roles and Arguments: Given the predicates, one should label the predicate-argument role and predict the argument type. Here we take an approach used for semantic role labeling (Punyakanok et al., 2008): given a predicate, we collect all constituents in the sentence to build a set of plausible candidate arguments. As a result, each predicate has a set of candidate arguments which should be labeled with their argument types and be assigned with a semantic role edge. Here we jointly train two logistic regression classifiers for predicting semantic role type and argument type of a predicate-argument pair, using argument identification features from (Punyakanok et al., 2008) and using the structured averaged Perceptron algorithm (Collins, 2002). The role types can be any of the roles from table 2 or *None* (set R), and the argument types can be any

Relation Type	Description	Example
Figure	The main role being compared to something else.	- [Lara] is taller than the tree.
Ground	The main role against which the figure is compared.	- Lara is taller than the [tree].
$\text{Difference}_{\text{degree}}$	The 'plus' and 'times' roles, signifying an amount of difference on a degree.	- Sam is [twice] taller than Jim.
Domain	The explicit expression of the domain/population in which the comparison takes place	- Mary is the most intelligent among [her classmates].
Reason	The reason associated with the excessive and assetive predicates	- John is too lazy [to wake up].
$\text{Measurement}_{\text{degree}}$	The main indication of a measurement	- Henry is [5 feet] tall.
Scale	The scale on which the comparison takes place	- The [height] of the chair equals the [length] of the sofa.

Table 2: The role types under our framework.

Argument Type	Subtype	Example
Individual: An entity being compared against others.	-	- [John] is a better performer than Susie.
Reference: A referring entity, the actual antecedent of which would be resolved in discourse-level.	-	- John is 2" taller than [that].
Phrase: Introduces a degree on scale.	Value	- John was driving faster than [he was allowed].
Amount: The expression of the amount in a measurement.	Value	- Mary is [5 feet] tall. - Mary is [twice] taller than Bill.
	Very Low-Low	- Mary is [a little bit] taller than John.
	High-Very High	- Mary is [a lot] taller than Bill.
Bound: A bound/approximation being set on the amount that is expressed.	Exact	- Mary is [exactly] 5 feet tall.
	Approximate	- Sam was [about] three times faster than others.
	Lower	- John walks [at least] twice faster than you.
	Higher	- Mary is [at most] twice as smart as the others.
Scale: The scale on which the measurement is done.	Explicit	- The [height] of the bridge is too low for the van.
	Implicit	- Sam is more [available] than John is.

Table 3: The argument types under our framework.

of the ones from table 3 or *None* (set G). At the end of this stage we have two scores: $sc_{p,j,r} = log\ Pr_{p,j,r}$ where $p \in P$ is a predicate type, j is a candidate argument, $r \in R$ is a role type; and $sc_{p,j,g} = log\ Pr_{p,j,g}$ where $g \in G$.

Joint Inference: Given a sentence with its extracted predicates[5], for each predicate labeled as p, the goal is the following: find the best assignment for the indicators $y = \{y_{p,j,r} \mid p \in P, 1 < j \leq n, r \in R\}$ and $x = \{x_{p,j,g} \mid p \in P, 1 < j \leq n, g \in G\}$. Here n is the number of candidate arguments for the given predicate. We model the problem as an Integer Linear Programming (ILP). We formulate the problem as follows:

$$\arg\max_{y,x} \sum_{\substack{1<j\leq n \\ r\in R}} sc_{p,j,r}\ y_{p,j,r} + \sum_{\substack{1<j\leq n \\ g\in G}} sc_{p,j,g}\ x_{p,j,g} \quad (4a)$$

$$\mathbf{s.t.} \sum_{r\in R, r\neq None} y_{p,j,r} = 1 \quad (4b)$$

$$\sum_{g\in G} y_{p,j,g} = 1 \quad (4c)$$

$$\sum_{1<j\leq n} y_{p,j,r} = 1 \quad (4d)$$

$$y_{p,j,None} = x_{p,j,None} \quad (4e)$$

$$\sum_{1<j\leq n} y_{p,j,Figure} = 1. \quad (4f)$$

The hard constraints $4b - 4c$ each indicate a restriction on the structure of the predicate-argument relation and labels: each argument can have only

[5] A constituent is a predicate if it is labeled with any $p \in P$ and $p \neq None$.

one role and argument type ($4b - 4c$), each predicate can only have one of each role type ($4d$), a 'None' role type should be matched with a 'None' argument type ($4e$), and each predicate should have exactly one 'Figure' role ($4f$). There are also some other specific constraints such as the fact that a predicate labeled with 'comparative' cannot have a 'Domain' role type and vice verse.

5 Experimental Setup

5.1 Dataset Creation

In order to make our gold-annotated dataset we used OntoNotes (Pradhan et al., 2007) release 5.0 corpus. OntoNotes covers various genres such as conversations, news-wire, and Weblogs, which provides distinctive variations of comparison structures in natural language. Furtheremore, we think our annotations can potentially provide augmentations on OntoNotes, so using the original OntoNotes sentences can be beneficial.

One approach for pinpointing comparison sentences is to mine for some known patterns and train a classifier for distinguishing comparison and non-comparison sentences (Jindal and Liu, 2006b). However, as demonstrated earlier, the variety of comparison structures is so vast that being limited to some specific patterns or syntactic structures will not serve our purpose. In order to address this issue, we randomly selected 2000 sentences from OntoNotes which contained an adjective, an adverb, or any of the comparison morphemes. This set contained some non-comparison sentences, such as 'John admitted to the crime too'.

In order to make the final set of comparison sentences we performed the following task: we define a comparative sentence as a sentence that contains at least one predicate operator as defined in Section 3. Hence, we provided three human experts with a full predicate operator types table and asked each of them to annotate any predicate operator found in the given sentences. Then we retained any sentences with at least one predicate operator which was annotated by at least two of the three judges. We further refined the set to include equal number of predicate types. This resulted in 531 sentences.

After collecting the comparison sentences, we asked the annotators to provide gold-standard annotation of predicate-argument structure of the sentences. This involves the annotator to read the annotation guideline and basically understand the semantic framework for comparison structures that we introduced in Section 3. Initially, we ran a pilot study on a set of 50 sentences where each sentence was annotated by two of the experts. We used pilot results for iterating over the annotation schema and guideline and resolving issues regarding low agreement predicates and argument types[6], until getting to average agreement $\kappa = 0.80$. We split the dataset into 30% and 70% for testing and training respectively.

5.2 Evaluation

Here we evaluate the performance of our proposed predicate-argument structure prediction. We present the following two methods:

- **ILP Method**: Our full approach as described in Section 4. Here we used the Gurobi[7] optimization package for finding an exact solution for our ILP formalization.

- **Baseline**: A simple pattern-based method which uses lexical patterns for predicting predicate type and argument types. This method uses the generic comparative morphemes such as 'er', 'est', 'more' and 'less' for detecting any specific type of predicate. For identifying predicate arguments it relies on rules which use syntactic structure, e.g., for a 'greater' predicate identified by 'er' morpheme, the 'left' argument is always the main subject of the sentence. This method annotates anything not recognized by patterns as 'None'.

Here with compare their predictions on test set to the gold standard annotations and compute micro-averaged precision, recall and F1 score. For this analysis we remove the 'equative' predicate type, given its very low frequency in our training set. Moreover, here we do not include the positive and negative predicate types, as these take only one role argument which is 'figure', making the prediction task trivial.

Table 4 shows the results of predicate type prediction. The final reported average in this table excludes the type 'None'. The best performing category in both methods is 'superlative', which is because of its more typical structure which makes it

[6]The disagreements were mainly on fine-grained predicate types, which were resolved by collapsing some of the types together.

[7]www.gurobi.com

easier to be predicted. In general, the precision of predicate prediction is very high in ILP method, which is due to the fact that our predicates are the comparison operators indicated by the comparison morphemes. The baseline performs considerably weaker than ILP method for predicting *less* and *greater* predicates. This is because predicting these types requires a more complicated analysis where simple morphological and syntactic patterns can result in many false positives.

Table 5 depicts the results of the role type prediction. The weighted average in this table is based on frequency, excluding the type 'None'. The precision on role prediction varies across different types. Overall, the baseline performs weakly on predicting role types, which is dues to the complicated structure of roles.

The best prediction of ILP method is on scales, which has benefited from the attribute concept feature. The weaker performing types have been affected by the low-frequency occurrence in the training set. There are many cases of very long and complex sentences in our dataset. One major reason behind some of the false predictions is incorrect dependency parse for long sentences. One notable issue here is that for easier prediction and analysis, we had asked our annotators to mark only the head words for phrasal arguments. This had often caused lower agreement among annotators and hence worse predictions on the system trained on the dataset. In future, we are going to switch to span-based argument identification.

	ILP Method			Baseline		
Predicate Type	**P**	**R**	**F1**	**P**	**R**	**F1**
Assetive	100	46	63	100	26	41
Greater	90	82	86	54	68	60
Superlative	96	79	87	89	73	80
Excessive	100	43	60	100	24	38
Less	100	86	92	45	71	55
None	96	99	99	78	80	79
Average	97	67	79	77	52	68

Table 4: The evaluation results on predicate type prediction.

6 Related Work

The syntax and semantics of comparison in language have been studied in linguistics for a long time (Bresnan, 1973; Cresswell, 1976; Von Stechow, 1984). However, so far, computational modeling of the semantics of comparison components of natural language has not been developed as elaborately as needed. The main efforts on computational aspects of comparatives have been in the context of sentiment analysis. Jindal and Liu (2006b) introduced the first approach for the identification of sentences containing comparisons. Their system trains a Naive Bayes classifier for labeling sentences as comparative or non-comparative.

	ILP Method			Baseline		
Role Type	**P**	**R**	**F1**	**P**	**R**	**F1**
Plus	67	31	42	11	17	13
Ground	34	56	42	6	23	9
Scale	81	28	41	63	20	30
Figure	25	44	32	3	29	5
Reason	50	12	20	33	7	11
Domain	50	25	33	26	24	25
Times	14	50	22	30	12	17
None	97	96	96	81	78	79
Weighted Average	76	42	54	24	18	20

Table 5: The evaluation result on role type prediction.

Later works progressed into identifying the components of the comparisons: comparative predicates and arguments. For example for the sentence "Canon's optics is better than those of Sony and Nikon.", the extracted relation should be: (better, {optics}, {Canon}, {Sony, Nikon}). Jindal and Liu (2006a) detect such arguments by labeling sequential rules. Xu et al. (2011) use Conditional Random Fields (Lafferty et al., 2001) to extract relations between two entities, an attribute and a predicate phrase. These works all provide a rudimentary basis for computational analysis of comparatives, however, they lack depth and breadth as they are limited to the limited comparison structure *(Entity1, Entity2, aspect)* expressed within some sequential patterns. It is evident that the framework of comparison proposed in this paper goes beyond simple triplet annotation of comparison structures and is more representative of the linguistics literature on comparatives and measurements.

The most recent related work on comparatives (Kessler, 2014) focuses on argument identification task: given a comparative predicate, they find the arguments corresponding to it. They train a classifier for this task emphasizing on syntax information. Most of the entities in their training data are products (cameras, cars, and phones). Another recent work (Kessler and Kuhn, 2014) concentrates on the annotation of what they call multi-word predicates (such as 'more powerful', where

the comparison is not one-word such as 'calmer'). They show that annotating the modifier of comparatives (i.e., the adjectives) gives better results in classification. Both these works share the major shortcoming of the earlier works, as they are very limited to their specific patterns and fail to enable deeper representation and analysis of various complex comparative structures.

7 Conclusion

Systems that can understand and reason over comparatives are crucial for various NLP applications ranging from open-domain question answering to product review analysis. Understanding comparatives requires a semantic framework which can represent their underlying meaning. In this paper we presented a novel semantic framework for representing the meaning of various comparison constructions in natural language. We mainly modeled comparisons as predicate-argument pairs which are connected via semantic roles. Our framework supports all possible parts of speech and variety of measurements and comparisons, hence providing a unique computational representation of the underlying semantics of comparison. Furthermore, we introduced an ILP-based method for predicting the predicate-argument structure of comparison sentences.

With this paper, we provide a novel dataset of gold-standard annotations based on our semantic framework. We are planning on expanding our gold-standard annotations under this framework for having more training data. Our semantic framework on comparison constructions enables us to do logical reasoning and inference over comparatives. In the future, we are planning to design a reading comprehension task where we use this framework for answering comparison questions from a paragraph containing various interrelated comparisons.

Last but not least, the works on broad-coverage semantic parsing (Allen et al., 2008; Bos, 2008) can all benefit from our semantic framework. We will be extending the TRIPS logical form (Allen et al., 2008) according to this framework and will modify the grammar to generate the deeper representations.

Acknowledgments

We would like to thank Alexis Welwood for her invaluable comments and guidelines on this work. Moreover, we thank Ritwik Bose for his help on annotations. This work was funded by the Office of Naval Research (grant N000141110417) and the DARPA Big Mechanism program under ARO contract W911NF-14-1-0391.

References

James F. Allen, Mary Swift, and Will de Beaumont. 2008. Deep semantic analysis of text. In *Proceedings of the 2008 Conference on Semantics in Text Processing*, STEP '08, pages 343–354, Stroudsburg, PA, USA. Association for Computational Linguistics.

Omid Bakhshandeh and James F. Allen. 2015. From adjective glosses to attribute concepts: Learning different aspects that an adjective can describe. In *Proceedings of 11th International Conference on Computational Semantics (IWCS)*.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186, Sofia, Bulgaria, August. Association for Computational Linguistics.

Jonathan Berant and Percy Liang. 2014. Semantic parsing via paraphrasing. In *Association for Computational Linguistics (ACL)*.

Jonathan Berant, Vivek Srikumar, Pei-Chun Chen, Brad Huang, Christopher D. Manning, Abby V. Linden, and Brittany Harding. 2014. Modeling biological processes for reading comprehension. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*.

Johan Bos. 2008. Wide-coverage semantic analysis with boxer. In Johan Bos and Rodolfo Delmonte, editors, *Semantics in Text Processing. STEP 2008 Conference Proceedings*, Research in Computational Semantics, pages 277–286. College Publications.

Joan Bresnan. 1973. Syntax of the comparative clause construction in english. *Linguistic Inquiry*, 4(3):275–343.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing - Volume 10*, EMNLP '02, pages 1–8, Stroudsburg, PA, USA. Association for Computational Linguistics.

Max Cresswell. 1976. The semantics of degree. *Barbara Hall Partee (ed.)*, pages 261–292.

Quang Do, Wei Lu, and Dan Roth. 2012. Joint inference for event timeline construction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, EMNLP-CoNLL 2012, July 12-14, 2012, Jeju Island, Korea*, pages 677–687.

Irene Heim. 2007. Little. In *In Proceedings of 16th Semantics and Linguistic Theory Conference*, Cornell University, Ithaca.

Nitin Jindal and Bing Liu. 2006a. Identifying comparative sentences in text documents. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '06, pages 244–251, New York, NY, USA. ACM.

Nitin Jindal and Bing Liu. 2006b. Mining comparative sentences and relations. In *Proceedings of the 21st National Conference on Artificial Intelligence - Volume 2*, AAAI'06, pages 1331–1336. AAAI Press.

Christopher Kennedy. 2007. Vagueness and grammar: the semantics of relative and absolute gradable adjectives. *Linguistics and Philosophy*, 30(1):1–45, February.

Wiltrud Kessler and Jonas Kuhn. 2014. Detecting comparative sentiment expressions – a case study in annotation design decisions. In *Proceedings of KONVENS*, Hildesheim, Germany.

Wiltrud Kessler. 2014. Improving the detection of comparison arguments in product reviews. In *Proceedings of 44th Jahrestagung der Gesellschaft fur Informatik e.V. (INFORMATIK 2014)*, pages 22–26, Stuttgart, Germany, September.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

George A. Miller. 1995. Wordnet: A lexical database for english. *Commun. ACM*, 38(11):39–41, November.

Sameer Pradhan, Eduard Hovy, Mitch Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2007. Ontonotes: A unified relational semantic representation. In *Proceedings of the International Conference on Semantic Computing*, ICSC '07, pages 517–526, Washington, DC, USA. IEEE Computer Society.

Vasin Punyakanok, Dan Roth, and Wen-tau Yih. 2008. The importance of syntactic parsing and inference in semantic role labeling. *Comput. Linguist.*, 34(2):257–287, June.

Stephanie Solt. 2015. Measurement scales in natural language. *Language and Linguistics Compass*, 9(1):14–32.

Arnim Von Stechow. 1984. Comparing semantic theories of comparison. *Journal of Semantics*, 3(1):1–77.

Alexis Wellwood, V. Hacquard, , and R. Pancheva. 2012. Measuring and comparing individuals and events. *Journal of Semantics*, 29(2):207–228.

Kaiquan Xu, Stephen Shaoyi Liao, Jiexun Li, and Yuxia Song. 2011. Mining comparative opinions from customer reviews for competitive intelligence. *Decision Support Systems*, 50(4):743–754.

John M. Zelle and Raymond J. Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence - Volume 2*, AAAI'96, pages 1050–1055. AAAI Press.

Sarcastic or Not: Word Embeddings to Predict the Literal or Sarcastic Meaning of Words

Debanjan Ghosh[§] and **Weiwei Guo**[†] and **Smaranda Muresan**[‡]
[§]School of Communication and Information, Rutgers University, NJ, USA
[†]Department of Computer Science, Columbia University, NY, USA
[‡]Center for Computational Learning Systems, Columbia University, NY, USA
debanjan.ghosh@rutgers.edu, weiwei@cs.columbia.edu, smara@ccls.columbia.edu

Abstract

Sarcasm is generally characterized as a figure of speech that involves the substitution of a literal by a figurative meaning, which is usually the *opposite* of the original literal meaning. We re-frame the sarcasm detection task as a type of word sense disambiguation problem, where the *sense* of a word is either *literal* or *sarcastic*. We call this the Literal/Sarcastic Sense Disambiguation (LSSD) task. We address two issues: 1) how to collect a set of target words that can have either literal or sarcastic meanings depending on context; and 2) given an utterance and a target word, how to automatically detect whether the target word is used in the literal or the sarcastic sense. For the latter, we investigate several distributional semantics methods and show that a Support Vector Machines (SVM) classifier with a modified kernel using word embeddings achieves a 7-10% F1 improvement over a strong lexical baseline.

1 Introduction

Recognizing sarcasm is important for understanding people's actual sentiments and beliefs. For example, failing to recognize the following message as being sarcastic "I love that I have to go back to the emergency room", will lead a sentiment and opinion analysis system to infer that the author's sentiment is positive towards the event of "going to the emergency room". Current approaches have framed the sarcasm detection task as predicting whether a full utterance is sarcastic or not (Davidov et al., 2010; González-Ibáñez et al., 2011; Riloff et al., 2013; Liebrecht et al., 2013; Maynard and Greenwood, 2014).

We propose a re-framing of sarcasm detection as a type of word sense disambiguation problem: given an utterance and a target word, identify whether the *sense* of the target word is *literal* or *sarcastic*. We call this the Literal/Sarcastic Sense Disambiguation (LSSD) task. In the above utterance, the word "love" is used in a sarcastic, non-literal sense (the author's intended meaning being most likely the opposite of the original literal meaning - a negative sentiment, such as "hate"). Two key challenges need to be addressed: 1) how to collect a set of target words that can have a literal or a sarcastic sense, depending on context; and 2) given an utterance containing a target word, how can we determine whether the target word is used in its literal sense (e.g., "I *love* to take a nice stroll in the park every morning"), or in a sarcastic sense (e.g., "I *love* going to the dentist.").

To address the first challenge, we need to identify a set of words from sarcastic utterances, which have a figurative/sarcastic sense (e.g., "love" in the utterance "I love going to the dentist"). We propose a crowdsourcing task where Turkers in Amazon Mechanical Turk (MTurk) platform are given sarcastic utterances (tweets labeled with #sarcasm or #sarcastic hashtags) and are asked to re-phrase those messages so that they convey the author's intended meaning ("I *love* going to the dentist" can be rephrased as "I *hate* going to the dentist" or "I *don't like* going to the dentist"). [1] Given this parallel dataset, we use unsupervised alignment techniques to identify semantically opposite words (e.g., "love" ↔ "hate", "brilliant" ↔ "stupid", "never" ↔ "always"). The words from these pairs that appear in the original sarcastic utterances are then considered as our collection of target words (e.g., "love", "brilliant", "never") that can have both a sarcastic and a literal sense depending on the context (Section 2).

To address the second challenge, we compare several distributional semantics methods generally used in word sense disambiguation tasks (Sec-

[1] utterances and messages are used interchangeably.

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1003–1012,
Lisbon, Portugal, 17-21 September 2015.

Target	Sense	Utterance
great	S	... starting off the new year great !!!!! sick in bed ...
	L	... you don't need a record label to have great music ...
	L_{sent}	... i'm in love with this song great job justin ...
proud	S	yay something to be proud of 3rd poorest in the NATION ...
	L	im filipino with dark brown eye and forever true and proud ...
	L_{sent}	but i'm proud of all the beliebers AROUND THE WORLD ...

Table 1: Examples of Targets and their Senses

tion 3). We show that using word embeddings in a modified SVM kernel achieves the best results (Section 4). To collect training and test datasets for each of the target words, we use Twitter messages that contain those words. For the *sarcastic sense* (S), we use tweets that contain the target word and are labeled with the #sarcasm or #sarcastic hashtags. For the *literal sense* (L), we collect tweets that contain the target word and are not labeled with the #sarcastic or #sarcasm hashtags. Table 1 shows examples of two targets words ("great" and "proud") and their sarcastic sense (S) and literal sense (L). In addition, for the *literal sense*, we also consider a special case, where the tweets are labeled with either positive or negative hashtags (e.g., #happy, #sad) as proposed by Gonzalez et al. (2011). We denote these sentiment tweets as L_{sent} (Table 1). Gonzalez et al. (2011) argue that it is harder to distinguish sarcastic from non-sarcastic messages where the non-sarcastic messages contain sentiment. Our results support this argument (97% F1 measure for the best result for S vs. L, compared to 84% F1 for the best result for S vs. L_{sent}; Section 4).[2]

2 Collection of Target Words

To collect a set of target words that can have either literal or sarcastic meaning depending on context, we propose a two step approach: 1) a crowdsourcing task to collect a parallel dataset of sarcastic utterances and their re-phrasings that convey the authors' intended meaning; and 2) unsupervised alignment techniques to detect semantically opposite words/phrases.

Crowdsourcing Task. Given a sarcastic message (SM), Turkers were asked to re-phrase the message so that the new message is likely to express the author's intended meaning (IM). Examples of an original sarcastic message (1) and three messages generated by the Turkers (2) is given below:

(1) [SM] I am so happy that I am going back to the emergency room.

(2) a. [IM_1] I don't like that I have to go to the emergency room again.
b. [IM_2] I am so upset I have to return to the emergency room.
c. [IM_3] I'm so unhappy that I am going back to the emergency room.

From the above examples, we can see that aligning the sarcastic message (SM) to the re-phrasings containing the author's intended meaning generated by the Turkers (IM_1, IM_2, IM_3) will allow us to detect that "happy" can be aligned to "don't like", "upset", and "unhappy". Based on this alignment, "happy" will be considered as a target word for the LSSD task.

We used 1,000 sarcastic messages collected from Twitter using the #sarcasm and #sarcastic hashtags. The Turkers were provided with detailed instructions of the task including a definition of *sarcasm*, the task description, and multiple examples. In addition, for messages that contain one or more sentences and where sarcasm is related to only a part of the message, the Turkers were instructed to consider the entire message in their rephrasing. This emphasis was added to avoid high asymmetry in the length between the original sarcastic message and the rephrasing of the intended meaning. For each original sarcastic message (SM), we asked five Turkers to do the rephrasing task. Each HIT contains 1 sarcastic message, and Turkers were paid 5 cents for each HIT. To ensure a high quality level, only qualified workers were allowed to perform the task (i.e., more than 90% approval rate and at least 500 approved HITs). In this way, we obtained a dataset of 5,000 SM-IM pairs.

Unsupervised Techniques to Detect Semantically Opposite Words/Phrases. We use two methods for unsupervised alignment. First, we use the co-training algorithm for paraphrase detection developed by Barzilay and McKeown (2001). This algorithm is used for two specific reasons. First, our dataset is similar in nature to the parallel

[2]The datasets used in the experiments is available at https://github.com/debanjanghosh/sarcasm_wsd.

monolingual dataset used in Barzilay and McKeown (2001), and thus lexical and contextual information from tweets can be used to extract the candidate targets words for LSSD. For instance, we can align the [SM] and [IM_3] (from the above examples), where except for the words happy and unhappy, the majority of the words in the two messages are anchor words and thus happy and unhappy can be extracted as paraphrases via co-training. To model contextual information, such as part of speech tagging for the co-training algorithm, we used Tweet NLP (Gimpel et al., 2011). Second, Bannard and Callison-Burch (2005) noticed that the co-training method proposed by-Barzilay and McKeown (2001) requires identical bounding substrings and has bias towards single words while extracting paraphrases. This apparent limitation, however, is advantageous to us because we are specifically interested in extracting target words. Co-training resulted in 367 extracted pairs of paraphrases.

We also considered a statistical machine translation (SMT) alignment method - IBM Model 4 with HMM alignment implemented in Giza++ (Och and Ney, 2000). We used Moses software(Koehn et al., 2007) to extract lexical translations by aligning the dataset of 5,000 SM-IM pairs. From the set of 367 extracted paraphrases using Barzilay and McKeown (2001)'s approach, we selected only those paraphrases where the lexical translation scores ϕ (resulted after running Moses) are ≥ 0.8. After filtering via translation scores and manual inspection, we obtained a set of 80 semantically opposite paraphrases. Given this set of semantically opposite words, the words that appear in the sarcastic messages were consider our target words for LSSD (70 target words after lemmatization). They range from verbs, such as "love" and "like", adjectives, such as "brilliant", "genius", and adverbs, such as "really".

3 Literal/Sarcastic Sense Disambiguation

Our Literal/Sarcastic Sense Disambiguation (LSSD) task is formulated as follows: given a candidate utterance (i.e., a tweet) that contains a target word t, identify whether the sense of t is *sarcastic* (S) or *literal* (L). In order to be able to solve this problem we need training and test data for each target word that consists of utterances where the target word is used either in the literal sense or the sarcastic sense.

love(26802), like(14995), great(14495), good(11624), really(9825), right(6771), fun(6603), best(6182), better(5960), glad(5748), yeah(5504), nice(4443), awesome(4196), excited(4027), always(3807), happy(3098), cool(2705), amazing(1952), favorite(1883), perfect(1792), wonderful(1749), wonder(1476), lovely(1424), super(1390), fantastic(1369), joy(1176), cute(1007), beautiful(981), sweet(800), hot(729), proud(703), shocked(645), interested(624), brilliant(576), genius(481), attractive(449), mature(427)

Table 2: Target words and # of training instances per sense

3.1 Data Collection

To collect training and test datasets for each of the target words, we use Twitter messages that contain those words. For the *sarcastic sense* (S), we use tweets that contain the target word and are labeled with the #sarcasm or #sarcastic hashtag. For the *literal sense* (L), we collect tweets that contain the target word and are not labeled with the #sarcastic or #sarcasm hashtags. In addition, for the *literal sense* we also consider a special case, where the tweets are labeled with either positive or negative sentiment hashtags (e.g., #happy, #sad). Thus, we consider two LSSD tasks: S vs. L and S vs. L_{sent}, and aim to collect a balanced dataset for each target word.

For the 70 target words (see Section 2), we collected a total of 2,542,249 tweets via Twitter API . We considered a setup where 80% of data is used for training, 10% for development, and 10% for test. We empirically set the number of minimum training instances for each sense of the target word to 400 without any upper restriction. This resulted in 37 target words to be used in the LSSD experiments. Table 2 shows all the target words and their corresponding number of *training* instances for each sense (S and L/L_{sent}). The size of training data ranges from 26,802 for the target word "love" to 427 for the word "mature". As we will see in the results sections, however, the size of the training data is not always the key factor in the LSSD task, especially for the methods that use word embeddings.

3.2 Learning Approaches

We consider two classical approaches used in word sense disambiguation tasks: 1) distributional approaches where each sense of a target word is represented as a context vector derived from the training data; and 2) classification approaches (S

vs. L; S vs. L_{sent}) for each target word.

3.2.1 Distributional Approaches

The Distributional Hypothesis in linguistics is derived from the semantic theory of language usage, i.e., words that are used and occur in the same contexts tend to purport similar meanings (Harris, 1954). Distributional semantic models (DSMs) use vectors that represent the contexts (e.g., co-occurring words) in which target words appear in a corpus, as proxies for meaning representations. Geometric techniques such as cosine similarity are then applied to these vectors to measure the similarity in meaning of corresponding words.

The DSMs are a natural approach to model our LSSD task. For each target word t we build two context-vectors that will represent the two senses of the target word t using the training data: one for the *sarcastic* sense S using the sarcastic training data for t ($\vec{v_s}$) and one for the *literal* sense L using the literal sense training data for t ($\vec{v_l}$).[3] Given a test message u containing a target word t, we first represent the target word as a vector $\vec{v_u}$ using all the context words inside u. To predict whether t is used in a literal or sarcastic sense in the test message u we simply apply geometric techniques (e.g., cosine similarity) between $\vec{v_u}$ and the two sense vectors $\vec{v_s}$ and $\vec{v_l}$, choosing the one with the maximum score.

To create the two sense vectors $\vec{v_s}$ and $\vec{v_l}$ for each of the target words t, we use the positive pointwise mutual information model (PPMI) (Church and Hanks, 1990). Based on t's context words c_k in a window of 10 words, we separately computed PPMI for sarcastic and literal senses using t's training data. The size of the context widow used in DSMs is generally between 5 and 10, and in our experiments we used a window of 10 words since tweets often include meaningful words/tokens at the end of the tweets (e.g., interjections, such as "yay", "ohh"; upper-case words, such as, "GREAT"; novel hashtags, such as "#notreally", "#lolol"; emoticons, such as ":("). We sorted the context words based on the PPMI scores and for each target word t we selected a maximum of 1,000 context words per sense to approximate the two senses of the target word (i.e., the vectors $\vec{v_s}$ and $\vec{v_l}$ for each target word t consist of a maximum of 1,000 words). Table 3 shows some target words and their corresponding context words that were selected based on high PPMI scores.

Targets	Senses	Context Vector
love	S	ignored, being, waking, work, sick, #not
	L	please, follow, ♡, her, :)
	L_{sent}	happy, family, blessed, cute, birthday
fun	S	work, tomorrow, homework, friday, sleep
	L	hope, join, girl, game, friend
	L_{sent}	#friends, #family, weekend, amazing, #christmas
joy	S	working, snow, waking, studying, sick
	L	yesterday, sweet, special, prayer, laughter
	L_{sent}	wishing, warmth, love, christmas, peace

Table 3: Target words and their context words

To predict whether t is used in a literal or sarcastic sense in the test message u we simply apply the cosine similarity to the $\vec{v_u}$ (vector representation of the target word t in the test message u) and the two sense vectors $\vec{v_s}$ and $\vec{v_l}$ of t, choosing the one with the maximum score. All vector elements are given by the tf-idf values of the corresponding words. This approach, denoted as the "PPMI baseline", is the baseline for our DSM experiments.

Context Vectors with Word Embedding: The above method considers that the context vectors $\vec{v_s}$ and $\vec{v_l}$ of each target word t contain the co-occurring words selected by their PPMI values. We enhance the representation of context vectors to represent each word in the context vector by its word embedding. We experiment with three different methods of obtaining word embeddings: Weighted Textual Matrix Factorization (WTMF) (Guo and Diab, 2012b); *word2vec* that implements the skip-gram and continuous bag-of-words models (CBOW) of Mikolov et al. (2013a), and GloVe (Pennington et al., 2014), a log-bilinear regression model based upon global word-word co-occurrence count in the training corpora.

After removing the tweets that are used as test sets, we build the three word embedding models in an unsupervised fashion with the remaining 2,482,763 tweets from our original data collection (Section 3.1). In each of the three models, each word w is represented by its d-dimensional vector $\vec{w}$ of real numbers, where d=100 for all of the embedding algorithms in our experiments. For the size of the embedding vectors, it is common to use

[3] In the remaining of this section we will only mention L and not L_{sent} for clarity and brevity.

100 or 300 dimensions, with larger dimensions for larger datasets. Our current dataset is smaller than the ones used in other applications of word embeddings (e.g., Pennington et al. (2014) have used billion tweets to create word embedding) so we opted for 100 dimensional vectors. Below are the short descriptions of the three word embedding models:

- *Weighted Textual Matrix Factorization (WTMF):* Low-dimensional vectors have been used in WSD tasks, since they are computationally efficient and provide better generalization than surface words. A dimension reduction method is Weighted Textual Matrix Factorization (WTMF) (Guo and Diab, 2012b), which is designed specifically for short texts, and has been successfully applied in WSD tasks (Guo and Diab, 2012a). WTMF models unobserved words, thus providing more robust embeddings for short texts such as tweets.

- *word2vec Representation:* We use both the Skip-gram model and the Continuous Bag-of-Words (CBOW) model (Mikolov et al., 2013a; Mikolov et al., 2013c) as implemented in the word2vec gensim python library. [4] Given a window size of n words around a word w, the skip-gram model predicts the neighboring words given the current word. In contrast, the CBOW model predicts the current word w, given the neighboring words in the window. We considered a context window of 10 words.

- *GloVe Representation:* GloVe (Pennington et al., 2014) is a word embedding model that is based upon weighted least-square model trained on global word-word co-occurrence counts instead of the local context used by word2vec.

Here, the LSSD task is similar to the baseline: to predict whether the target word t in the test message u is used in a literal or sarcastic sense, we simply use a similarity measure between the $\vec{v_u}$ (vector representation of the target word t in the test message u) and the two sense vectors $\vec{v_s}$ and $\vec{v_l}$ of t, choosing the one with the maximum score. The difference from the baseline is twofold: First, all vectors elements are word embeddings (i.e., 100-d vectors). Second, we use the *maximum-valued matrix-element* (MVME) algorithm introduced by Islam and Inkpen (2008), which has been shown to be particularly useful for computing the similarity of short texts. We modify this algorithm to use word embeddings ($MVME_{we}$). The idea behind the MVME algorithm is that it finds a one-to-one "word alignment" between two utterances (i.e., sentences) based on the pairwise word similarity. Only the aligned words contribute to the overall similarity score.

Algorithm 1 $MVME_{we}$

procedure $MVME_{we}(v_s, v_u)$
  $v_{s_{words}} \leftarrow v_s.elements()$
  $v_{u_{words}} \leftarrow v_u.elements()$
  $M[v_{s_{words}}.size(), v_{u_{words}}.size()] \leftarrow 0$
  for $k \leftarrow 0, v_{s_{words}}.size()$ **do**
    $c_k \leftarrow v_{s_{words}}[k]$
    $\vec{c_k} \leftarrow getEmbedding(c_k)$
    for $j \leftarrow 0, v_{u_{words}}.size()$ **do**
      $w_j \leftarrow v_{u_{words}}[j]$
      $\vec{w_j} \leftarrow getEmbedding(w_j)$
      $M[k][j] \leftarrow cosine(\vec{c_k}, \vec{w_j})$
    end for
  end for
  while True **do**
    repeat
      $max \leftarrow getMax(M)$
      $Sim \leftarrow Sim + max$
      $r_m, c_m \leftarrow getRowCol(M, max)$
  ▷ *Remove r_m row and c_m column from M*
      $remove(M, r_m, c_m)$
    until $max > 0$ Or $M.size() > 0$
  end while
  Return Sim
end procedure

procedure GETEMBEDDING(word)
  Return $we_{model}[word]$
end procedure
procedure GETROWCOL(M,max)
  $row, col \leftarrow M.indexOf(max)$
  Return row, col
end procedure

Algorithm 1 presents the pseudocode of our modified algorithm for word embeddings, $MVME_{we}$. Let the total similarity between $\vec{v_s}$ and $\vec{v_u}$ be Sim. For each context word c_k from $\vec{v_s}$ and each word w_j from $\vec{v_u}$, we compute a matrix where the value of the matrix element M_{jk}

[4] https://radimrehurek.com/gensim/models/word2vec.html

denotes the cosine similarity between the embedded vectors $\vec{c_k}$ and $\vec{w_j}$ [lines 5 -13]. Next, we first select the matrix cell that has the highest similarity value in M (max) and add this to the Sim score [lines 16-17]. Let the r_m and c_m be the row and the column of the cell containing max (maximum-valued matrix element), respectively. Next, we remove all the matrix elements of the r_m-th row and the c_m-th column from M [line 20]. We repeat this procedure until we have traversed through all the rows and columns of M or max = 0 [line 21].

3.2.2 Classification Approaches

The second approach for our LSSD task is to treat it as a binary classification task to identify the sarcastic or literal sense of a target word t. We have two classification tasks: S vs. L and S vs. L_{sent} for each of the 37 target words. We use the libSVM toolkit (Chang and Lin, 2011). Development data is used for tuning parameters.

SVM Baseline: The SVM baseline for LSSD tasks uses n-grams and lexicon-based binary-valued features that are commonly used in existing state-of-the-art sarcasm detection approaches (González-Ibáñez et al., 2011; Tchokni et al., 2014). They are derived from i) bag-of-words (BoW) representations of words, ii) LIWC dictionary (Pennebaker et al., 2001), and iii) a list of interjections (e.g., "ah", "oh", "yeah"), punctuations (e.g., "!", "?"), and emoticons collected from Wikipedia. *CMU Tweet Tokenizer* is employed for tokenization. [5] We kept unigrams unchanged when all the characters are uppercase (e.g., "NEVER" in *"A shooting in Oakland? That NEVER happens! #sarcasm"*) but otherwise words are converted to lower case. We also change all numbers to a generic number token "22". To avoid any bias during experiments, we removed the target words from the tweets as well as any hashtag used to determine the sense of the tweet (e.g., #sarcasm, #sarcastic, #happy, #sad).

SVM with $MVME_{we}$ Kernel: We propose a new kernel $kernel_{we}$ to compute the semantic similarity between two tweets u_r and u_s using the $MVME_{we}$ method introduced for the DSM approach, and the three types of word embeddings (WTMF, word2vec, and GloVe). The similarity measure in the kernel is similar to the algorithm $MVME_{we}$ described in Algorithm 1, but instead of measuring the similarity between the sense vectors of t ($\vec{v_s}$, $\vec{v_l}$) and the vector representation of t in test message ($\vec{v_u}$), now we measure the similarity between two tweets u_r and u_s. For each k-th index word w_k in u_r and l-th index word w_l in u_s we compute the cosine similarity between the embedded vectors of the words and fill up a similarity matrix M. We select the matrix cell that has the highest similarity, add this similarity score to the total similarity Sim, remove the row and column from M that has highest similarity score, and repeat the procedure (similar to Algorithm 1). We noticed that $MVME_{we}$ algorithm carefully chooses the best candidate word w_l in u_s for the w_k word in u_r since w_l is the most similar word to w_k. The algorithm continues the same procedure for all the remaining words in u_r and u_s. The final Sim is used as the kernel similarity between u_r and u_s. We augment this kernel $kernel_{we}$ into libSVM and during evaluation we run supervised LSSD classification for each target word t separately.

[5] http://www.ark.cs.cmu.edu/TweetNLP/

4 Results and Discussions

Tables 4 and 5 show the results for the LSSD experiments using distributional approaches and classification-based approaches, respectively. For brevity, we only report the average Precision (P), Recall (R), and F1 scores with their standard deviation (SD) (given by '$\pm$'), and the targets with maximum/minimum F1 scores. $w2v_{sg}$ and $w2v_{cbow}$ represent the skip-gram and CBOW models implemented in word2vec, respectively.

Table 4 presents the results of distributional approaches (Section 3.2.1). We observe that the word embedding methods have better performance than the PPMI baseline for both S vs. L and S vs. L_{sent} disambiguation tasks. Also, the average P/R/F1 scores for S vs. L are much higher than for S vs. L_{sent}. Since all tweets with L_{sent} sense were collected using sentiment hashtags (González-Ibáñez et al., 2011), they might be lexically more similar to the S tweets than the L tweets are and thus identifying the sense of a target word t between S vs. L_{sent} is a harder task. In Table 4 we also observe that the average F1 scores between WTMF, $w2v_{sg}$, $w2v_{cbow}$, and GloVe are comparable and between 84%-86%, with $w2v_{sg}$ and $w2v_{cbow}$ achieving slightly higher F1.

Table 5 outlines the LSSD experiments us-

Expr.	Senses	Avg. P	Avg. R	Avg. F1	Max. F1(Target)	Min. F1(Target)
$PPMI_{bl}$	S	73.5 ± 3.6	84.6 ± 6.0	78.5 ± 3.2	83.9(mature)	68.8(wonder)
	L	83.1 ± 5.0	70.6 ± 5.5	76.1 ± 3.4	82.7(love)	68.3(nice)
	S	67.8 ± 7.0	76.2 ± 13.6	70.4 ± 7.6	81.8(joy)	43.8(like)
	L_{sent}	74.2 ± 7.1	62.7 ± 12.8	66.9 ± 6.6	78.6(joy)	47.1(interested)
WTMF	S	83.0 ± 3.4	87.2 ± 5.4	84.9 ± 2.4	91.4(mature)	78.7(wonder)
	L	87.5 ± 4.4	82.7 ± 4.5	84.9 ± 2.2	90.5(mature)	80.6(nice)
	S	67.4 ± 5.5	86.5 ± 5.1	75.6 ± 3.9	84.4(joy)	65.8(interested)
	L_{sent}	82.1 ± 5.8	58.9 ± 9.7	68.1 ± 7.2	81.5(joy)	50.0(genius)
GloVe	S	83.7 ± 3.6	85.6 ± 5.6	84.5 ± 2.8	90.6(joy)	78.8(sweet)
	L	86.3 ± 4.6	84.0 ± 4.3	85.0 ± 2.5	89.6(joy)	79.2(like)
	S	70.7 ± 5.1	84.3 ± 5.0	76.8 ± 3.9	85.4(joy)	67.1(interested)
	L_{sent}	80.7 ± 5.4	64.7 ± 8.5	71.5 ± 6.1	84.0(joy)	54.7(hot)
$w2v_{sg}$	S	84.9 ± 3.3	87.0 ± 4.8	85.8 ± 2.6	90.9(mature)	80.7(like)
	L	87.5 ± 4.1	85.1 ± 4.0	86.2 ± 2.5	90.7(mature)	80.2(like)
	S	70.8 ± 4.8	85.7 ± 5.1	77.4 ± 4.0	86.7(joy)	68.1(interested)
	L_{sent}	82.2 ± 5.7	64.3 ± 7.8	71.9 ± 5.9	85.4(joy)	57.4(interested)
$w2v_{cbow}$	S	84.9 ± 3.2	86.7 ± 4.7	85.6 ± 2.5	90.9(mature)	80.7(like)
	L	87.3 ± 4.0	85.1 ± 3.8	86.1 ± 2.4	90.7(mature)	80.2(like)
	S	70.7 ± 4.8	85.8 ± 5.0	77.4 ± 4.0	86.4(joy)	68.6(attractive)
	L_{sent}	82.0 ± 5.6	64.0 ± 7.7	71.7 ± 5.8	85.0(joy)	58.7(interested)

Table 4: Evaluation of distributional approaches (PMI and word embedding) for LSSD experiments

Expr.	Senses	Avg. P	Avg. R	Avg. F1	Max. F1(Target)	Min. F1(Target)
SVM_{bl}	S	87.0 ± 3.3	85.6 ± 3.1	86.3 ± 2.7	91.7(yeah)	75.4(sweet)
	L	85.9 ± 2.8	87.1 ± 3.6	86.5 ± 2.8	91.8(yeah)	76.1(sweet)
	S	77.3 ± 4.6	78.2 ± 4.2	77.7 ± 3.8	85.5(love)	68.6(brilliant)
	L_{sent}	77.8 ± 3.7	76.7 ± 6.4	77.1 ± 4.7	85.8(love)	64.6(attractive)
$kernel_{WTMF}$	S	94.1 ± 2.2	94.6 ± 1.8	94.3 ± 1.8	97.3(brilliant)	88.3(joy)
	L	94.6 ± 1.8	94.0 ± 2.3	94.3 ± 1.9	97.2(mature)	87.9(joy)
	S	79.0 ± 4.6	78.8 ± 4.4	78.8 ± 3.8	84.8(mature)	61.0(genius)
	L_{sent}	78.8 ± 3.7	78.9 ± 4.9	78.8 ± 3.6	85.4(mature)	63.5(genius)
$kernel_{GloVe}$	S	95.7 ± 1.6	97.4 ± 1.7	96.5 ± 1.1	99.1(mature)	92.9(glad)
	L	97.4 ± 1.6	95.6 ± 1.7	96.5 ± 1.2	99.1(mature)	92.7(interested)
	S	79.5 ± 3.5	83.1 ± 3.0	81.2 ± 2.8	86.9(joy)	74.2(attractive)
	L_{sent}	82.2 ± 3.0	78.3 ± 4.4	80.2 ± 3.4	86.6(joy)	69.2(attractive)
$kernel_{w2v_{sg}}$	S	96.6 ± 1.1	98.5 ± 0.6	97.5 ± 0.4	99.2(cute)	93.8(interested)
	L	98.5 ± 0.7	96.5 ± 1.2	97.5 ± 0.5	99.2(cute)	93.5(interested)
	S	81.9 ± 3.8	88.1 ± 3.2	84.8 ± 3.0	88.8(love)	74.2(genius)
	L_{sent}	87.0 ± 3.2	80.2 ± 4.7	83.4 ± 3.5	88.8(love)	73.3(genius)
$kernel_{w2v_{cbow}}$	S	96.4 ± 1.0	98.2 ± 1.1	97.3 ± 0.6	99.1(mature)	93.8(interested)
	L	98.2 ± 1.1	96.3 ± 1.1	97.2 ± 0.7	99.1(mature)	93.5(interested)
	S	81.7 ± 3.8	88.6 ± 2.9	84.9 ± 2.8	89.5(love)	74.8(genius)
	L_{sent}	87.4 ± 2.9	79.9 ± 4.8	83.4 ± 3.4	89.2(love)	74.4(genius)

Table 5: Evaluation of classification approaches (SVM_{bl} and $kernel_{we}$) for LSSD experiments

ing the classification approaches (Section 3.2.2): SVM baseline (SVM_{bl}) and SVM using the $kernel_{we}$ with word embeddings ($kernel_{WTMF}$, $kernel_{GloVe}$, $kernel_{w2v_{sg}}$, and $kernel_{w2v_{cbow}}$). The classification approaches give better performance compared to the distributional approaches. The SVM_{bl} is around 7-8 % higher than the $PPMI_{bl}$ and comparable with the word embeddings used in distributional approaches (Table 4). In addition, our new SVM kernel method using word embeddings shows significantly better results when compared to the SVM_{bl} (and distributional approaches). For instance, for the S vs. L task, the average F1 is 96-97%, which is more than 10% higher than SVM_{bl}. Similarly, for S vs. L_{sent} task, F1 scores reported by the kernel using word2vec embeddings are in the range of 83%-84% compared to 77% given by the SVM_{bl}, showing an absolute increase of 7%. As stated earlier, MVME algorithm aligns similar word pairs found in its inputs and this performs well for short texts (i.e., tweets). Thus, the MVME algorithm combined with word embedding in $kernel_{we}$ results in very high F1. Among the word embedding models, word2vec models give marginally better results compared to GloVe and WTMF, and GloVe outperforms marginally WTMF. Similar to Table 4, here, the average F1 scores for S vs. L task are higher than the S vs. L_{sent} results.

In terms of the best and worst performing tar-

gets, SVM_{bl} prefers targets with more training data (e.g., "yeah", "love" vs. "sweet", "attractive"; see Table 2). In contrast, word embedding models for "joy" and "mature", two targets with comparatively low number of training instances have achieved very high F1 using both distributional and classification approaches (Table 4 and 5). This can be explained by the fact that for words, such as "joy", "mature", "cute", and "brilliant", the contexts of their literal and sarcastic sense are quite different, and DSMs and word embeddings are able to capture the difference. For example, observe in the Table 3, negative sentiment words, i.e., "sick", "working", "snow" are the context words for targets "joy" and "love", where as positive sentiment words, such as, "blessed", "family", "christmas", and "peace" are the context words for L or L_{sent} senses. Overall, out of 37 targets, only 5 targets ("mature", "joy", "cute", "love", and "yeah") achieved "maximum" F1 scores in various experimental settings (Tables 4 and 5) whereas targets such as "interested", "genius", and "attractive" achieved low F1 scores.

In terms of variance in results, SVM results show low SD (0-4%). For distributional approaches, SD is slightly higher (5-8%) for several cases.

5 Related Work

Two lines of research are directly relevant to our work: sarcasm detection in Twitter and application of distributional semantics, such as word embedding techniques to various NLP tasks. In contrast to current research on sarcasm and irony detection (Davidov et al., 2010; Riloff et al., 2013; Liebrecht et al., 2013; Maynard and Greenwood, 2014), we have introduced a reframing of this task as a type of word sense disambiguation problem, where the sense of a word is sarcastic or literal. Our SVM baseline uses the lexical features proposed in previous research on sarcasm detection (e.g., LIWC lexicon, interjections, pragmatic features) (Liebrecht et al., 2013; González-Ibáñez et al., 2011; Reyes et al., 2013). Our analysis of target words where the sarcastic sense is the opposite of the literal sense is related to the idea of "positive sentiment toward a negative situation" proposed by Riloff et al. (2013) and recently studied by Joshi et al. (2015). In our approach, we chose distributional semantic approaches that learn contextual information of targets effectively from a large corpus containing both literal and sarcastic uses of words and show that word embedding are highly accurate in predicting the sarcastic or literal sense of a word (Tables 4 and 5). This approach has the potential to capture more nuanced cases of sarcasm, beyond "positive sentiment towards a negative situation" (e.g., one of our target words was "shocked" which is negative). However, our current framing is still inherently limited to cases where sarcasm is characterized as a figure of speech where the author means the opposite of what she says, due to our approach of selecting the target words.

Low-dimensional text representation, such as WTMF, have been successful in WSD disambiguation research and in computing similarity between short texts (Guo and Diab, 2012a; Guo and Diab, 2012b). word2vec and GloVe representations have provided state-of-the-art results on various word similarity and analogy detection task (Mikolov et al., 2013c; Mikolov et al., 2013b; Pennington et al., 2014). Word embedding based models are also used for other NLP tasks such as dependency parsing, semantic role labeling, POS tagging, NER, question-answering (Bansal et al., 2014; Collobert et al., 2011; Weston et al., 2015) and our work on LSSD is a novel application of word embeddings.

6 Conclusion and Future Work

We proposed a reframing of the sarcasm detection task as a type of word sense disambiguation problem, where the sense of a word is its *sarcastic* or *literal* sense. Using a crowdsourcing experiment and unsupervised methods for detecting semantically opposite phrases, we collected a set of target words to be used in the LSSD task. We compared several distributional semantics methods, and showed that using word embeddings in a modified SVM kernel achieves the best results (an increase of 10% F1 and 8% F1 for S vs. L and S vs. L_{sent} disambiguation task, respectively, against a SVM baseline). While the SVM baseline preferred larger amounts of training data (best performance achieved on the targets words with higher number of training examples), the methods using word embeddings seem to perform well on target words where there might be an inherent difference in the contextual sarcastic and literal use of a target word, even if the training data was smaller.

We want to investigate further the nature and

size of training data useful for the LSSD task. For example, to test the effect of larger training dataset, we utilized pre-trained word vectors from GloVe (trained with 2 Billion tweets, using 100 dimensions).[6] For S vs. L disambiguation, the average F1 was 88.9%, which is 7% lower than the result using GloVe on our training set of tweets (much smaller) designed for the LSSD task. This shows the training data utilized to create word embedding models in GloVe probably do not contain enough sarcastic tweets.

Regarding the size of the training data, recall that the unsupervised alignment approach had extracted 70 target words (Section 2), although we have used 37 target words as we did not have enough training data for the remaining targets. Thus, we plan to collect more training data for these targets as well as more target words (especially for the S vs. L_{sent} task). In addition, we plan to improve our unsupervised methods for detecting semantically opposite meaning (e.g., using the IM-IM dataset in addition to the SM-IM dataset).

One common criticism of research based on use of hashtags as gold labels is that the training utterances could be noisy. In other words, tweets might be sarcastic but not have #sarcasm or #sarcastic hashtags. We did a small manual validation on a dataset of 180 tweets from the L_{sent} class using 3 annotators (we asked them to say whether the tweet is sarcastic or not). For cases where all 3 coders agree none of them were considered sarcastic, while when only 2 coders agree 1 tweet out of 180 was considered sarcastic. In future, we plan to perform additional experiments to study the issue of noisy data. We hope that the release of our datasets will stimulate other studies related to the sarcasm detection problem, including addressing the issue of noisy data.

We also plan to study the effect of hyperparameters in designing the DSMs. Recently, Levy et al. (2015) have argued that parameter settings have a large impact on the success of word embedding models. We want to follow their experiments to study whether parameter tuning in PMI based disambiguation can improve its performance.

[6]Downloaded from http://nlp.stanford.edu/projects/glove/

Acknowledgements

This paper is based on work supported by the DARPA-DEFT program. The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government. The authors thank the anonymous reviewers for helpful comments.

References

Colin Bannard and Chris Callison-Burch. 2005. Paraphrasing with bilingual parallel corpora. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 597–604. Association for Computational Linguistics.

Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2014. Tailoring continuous word representations for dependency parsing. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Regina Barzilay and Kathleen R McKeown. 2001. Extracting paraphrases from a parallel corpus. In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, pages 50–57. Association for Computational Linguistics.

Chih-Chung Chang and Chih-Jen Lin. 2011. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

Kenneth Ward Church and Patrick Hanks. 1990. Word association norms, mutual information, and lexicography. *Computational linguistics*, 16(1):22–29.

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

Dmitry Davidov, Oren Tsur, and Ari Rappoport. 2010. Semi-supervised recognition of sarcastic sentences in twitter and amazon. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, CoNLL '10.

Kevin Gimpel, Nathan Schneider, Brendan O'Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A Smith. 2011. Part-of-speech tagging for twitter: Annotation, features, and experiments. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 42–47. Association for Computational Linguistics.

Roberto González-Ibáñez, Smaranda Muresan, and Nina Wacholder. 2011. Identifying sarcasm in twitter: A closer look. In *ACL (Short Papers)*, pages 581–586. Association for Computational Linguistics.

Weiwei Guo and Mona Diab. 2012a. Learning the latent semantics of a concept from its definition. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pages 140–144. Association for Computational Linguistics.

Weiwei Guo and Mona Diab. 2012b. Modeling sentences in the latent space. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 864–872. Association for Computational Linguistics.

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

Aminul Islam and Diana Inkpen. 2008. Semantic text similarity using corpus-based word similarity and string similarity. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 2(2):10.

Aditya Joshi, Vinita Sharma, and Pushpak Bhattacharyya. 2015. Harnessing context incongruity for sarcasm detection. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 757–762, Beijing, China, July. Association for Computational Linguistics.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th annual meeting of the ACL on interactive poster and demonstration sessions*, pages 177–180. Association for Computational Linguistics.

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

CC Liebrecht, FA Kunneman, and APJ van den Bosch. 2013. The perfect solution for detecting sarcasm in tweets# not.

Diana Maynard and Mark A Greenwood. 2014. Who cares about sarcastic tweets? investigating the impact of sarcasm on sentiment analysis. In *Proceedings of LREC*.

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

Tomas Mikolov, Quoc V Le, and Ilya Sutskever. 2013b. Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168*.

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

Franz Josef Och and Hermann Ney. 2000. Giza++: Training of statistical translation models.

James W Pennebaker, Martha E Francis, and Roger J Booth. 2001. Linguistic inquiry and word count: Liwc 2001. *Mahway: Lawrence Erlbaum Associates*, 71:2001.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. *Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014)*, 12.

Antonio Reyes, Paolo Rosso, and Tony Veale. 2013. A multidimensional approach for detecting irony in twitter. *Language resources and evaluation*, 47(1):239–268.

Ellen Riloff, Ashequl Qadir, Prafulla Surve, Lalindra De Silva, Nathan Gilbert, and Ruihong Huang. 2013. Sarcasm as contrast between a positive sentiment and negative situation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 704–714. Association for Computational Linguistics.

Simo Tchokni, Diarmuid O Séaghdha, and Daniele Quercia. 2014. Emoticons and phrases: Status symbols in social media. In *Eighth International AAAI Conference on Weblogs and Social Media*.

Jason Weston, Antoine Bordes, Sumit Chopra, and Tomas Mikolov. 2015. Towards ai-complete question answering: A set of prerequisite toy tasks. *arXiv preprint arXiv:1502.05698*.

Incorporating Trustiness and Collective Synonym/Contrastive Evidence into Taxonomy Construction

Luu Anh Tuan [#1], **Jung-jae Kim** [#2], **Ng See Kiong** [*3]

[#] *School of Computer Engineering, Nanyang Technological University, Singapore*

[1] anhtuan001@e.ntu.edu.sg, [2] jungjae.kim@ntu.edu.sg

[*] *Institute for Infocomm Research, Agency for Science, Technology and Research, Singapore*

[3] skng@i2r.a-star.edu.sg

Abstract

Taxonomy plays an important role in many applications by organizing domain knowledge into a hierarchy of *is-a* relations between terms. Previous works on the taxonomic relation identification from text corpora lack in two aspects: 1) They do not consider the trustiness of individual source texts, which is important to filter out incorrect relations from unreliable sources. 2) They also do not consider collective evidence from synonyms and contrastive terms, where synonyms may provide additional supports to taxonomic relations, while contrastive terms may contradict them. In this paper, we present a method of taxonomic relation identification that incorporates the trustiness of source texts measured with such techniques as PageRank and knowledge-based trust, and the collective evidence of synonyms and contrastive terms identified by linguistic pattern matching and machine learning. The experimental results show that the proposed features can consistently improve performance up to 4%-10% of F-measure.

1 Introduction

Taxonomies which serve as backbone of structured knowledge are useful for many applications such as question answering (Harabagiu et al., 2003) and document clustering (Fodeh et al., 2011). Even though there are many hand-crafted, well-structured taxonomies publicly available, including WordNet (Miller, 1995), OpenCyc (Matuszek et al., 2006), and Freebase (Bollacker et al., 2008), they are incomplete in specific domains, and it is time-consuming to manually extend them or create new ones. There is thus a need for automatically extracting taxonomic relations from text corpora to construct/extend taxonomies.

Previous works on the task of taxonomy construction capture information about potential taxonomic relations between concepts, rank the candidate relations based on the captured information, and integrate the highly ranked relations into a taxonomic structure. They utilize such information as hypernym patterns (e.g. A is a B, A such as B) (Kozareva et al., 2008), syntactic dependency (Drumond and Girardi, 2010), definition sentences (Navigli et al., 2011), co-occurrence (Zhu et al., 2013), syntactic contextual similarity (Tuan et al., 2014), and sibling relations (Bansal et al., 2014).

They, however, lack in the three following aspects: 1) Trustiness: Not all sources are trustworthy (e.g. gossip, forum posts written by non-experts) (Dong et al., 2015). The trustiness of source texts is important in taxonomic relation identification because evidence from unreliable sources can be incorrect. For example, the invalid taxonomic relation between "American chameleon" and "chameleon" is mistakenly more popular in the Web than the valid taxonomic relation between "American chameleon" and "lizard", and statistical methods without considering the trustiness may incorrectly extract the invalid relation instead of the latter. However, to the best of our knowledge, no previous work considered this aspect.

2) Synonyms: A concept may be expressed in multiple ways, for example with synonyms. The previous works mostly assumed that a term represents an independent concept, and did not combine information about a concept, which is expressed with multiple synonyms. The lack of evidence from synonyms may hamper the ranking of candidate taxonomic relations. Navigli and Velardi (2004) combined synonyms into a concept, but only for those from WordNet, called synsets.

3) Contrastive terms: We observe that if two terms are often contrasted (e.g. A but not B, A is different from B) (Kim et al., 2006), they may

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1013–1022, Lisbon, Portugal, 17-21 September 2015.

not have a taxonomic relation.

In this paper, we present a method based on the state-of-the-art method (Tuan et al., 2014), which incorporates the trustiness of source texts and the collective evidence from synonyms/contrastive terms. Tuan et al. (2014) rank candidate taxonomic relations based on such evidence as hypernym patterns, WordNet, and syntactic contextual similarity, where the pattern matches and the syntactic contexts are found from the Web by using a Web search engine. First, we calculate the trustiness score of each data source with the four following weights: importance (if it is linked by many pages), popularity (if it is visited by many users), authority (if it is from a creditable Web site) and accuracy (if it has many facts). We integrate this score to the work of (Tuan et al., 2014) so that evidence for taxonomic relations from unreliable sources is discarded.

Second, we identify synonyms of two terms (t_1, t_2), whose taxonomic relation is being scrutinized, by matching such queries as "t_1 also known as" against the Web to find t_1's synonyms next to the query matches (e.g. t in "t_1 also known as t"). We then collect the evidence for all taxonomic relations between t'_1 and t'_2, where t'_i is either t_i or its synonym ($i \in \{1, 2\}$), and combine them to calculate the evidence score of the candidate taxonomic relation between t_1 and t_2. Similarly, for each pair of two terms (t_1, t_2), we collect their contrastive evidence by matching such queries as "t_1 is not a type of t_2" against the Web, and use them to proportionally decrease the evidence score for taxonomic relation between contrasting terms.

2 Related Work

The previous methods for identifying taxonomic relations from text can be generally classified into two categories: linguistic and statistical approaches. The former approach mostly exploits lexical-syntactic patterns (e.g. *A is a B*, *A such as B*) (Hearst, 1992). Those patterns can be manually created (Kozareva et al., 2008; Wentao et al., 2012) or automatically identified (Navigli et al., 2011; Bansal et al., 2014). The pattern matching methods show high precision when the patterns are carefully defined, but low coverage due to the lack of contextual analysis across sentences.

The latter approach, on the other hand, includes asymmetrical term co-occurrence (Fotzo and Gallinari, 2004), clustering (Wong et al., 2007), syntactic contextual similarity (Tuan et al., 2014), and word embedding (Fu et al., 2014). The main idea behind these techniques is that the terms that are asymmetrically similar to each other with regard to, for example, co-occurrences, syntactic contexts, and latent vector representation may have taxonomic relationships. Such methods, however, usually suffer from low accuracy, though showing relatively high coverage. To get the balance between the two approaches, Yang and Callan (2009), Zhu et al. (2013) and Tuan et al. (2014) combine both statistical and linguistic features in the process of finding taxonomic relations.

Most of these previous methods do not consider if the source text of evidence (e.g. co-occurrences, pattern matches) is trustworthy or not and do not combine evidence from synonyms and contrastive terms as discussed earlier. Related to synonyms, a few previous works utilize siblings for taxonomy construction. Yang and Callan (2009) use siblings as one of the features in the metric-based framework which incrementally clusters terms to form taxonomies. Wentao et al. (2012) also utilize such sibling feature that, for example of the linguistic pattern "A such as B_1, B_2, $\cdots$ and B_n", if the concept at the k-th position (e.g. B_k) from pattern keywords (e.g. such as) is a valid sub-concept (e.g. of A), then most likely its siblings from position 1 to position k-1 (e.g. B_1, $\cdots$, B_{k-1}) are also valid sub-concepts. Bansal et al. (2014) include the sibling factors to a structured probabilistic model over the full space of taxonomy trees, thus helping to add more evidence to taxonomic relations. Navigli and Velardi (2004) utilize the synonym feature (i.e. WordNet synsets) for the process of semantic disambiguation and concept clustering as mentioned above, but not for the process of inducing novel taxonomic relations.

3 Methodology

We briefly introduce (Tuan et al., 2014) in Section 3.1. We then explain how to incorporate trustiness (Section 3.2) and collective evidence from synonyms (Section 3.3) and from contrastive terms (Section 3.4) into the work of (Tuan et al., 2014).

3.1 Overview of baseline (Tuan et al., 2014)

Tuan et al. (2014) follow three steps to construct a taxonomy: term extraction/filtering, taxonomic relation identification and taxonomy induction. Because the focus of this paper is on the second step,

taxonomic relation identification, we use the same methods for the first and third steps as in (Tuan et al., 2014) and will not discuss them here.

Given each ordered pair of two terms t_1 and t_2 from the term extraction/filtering, the taxonomic relation identification of (Tuan et al., 2014) calculates the evidence score that t_1 is a hypernym of t_2 (denoted as $t_1 \gg t_2$) based on the following three measures:

String inclusion with WordNet (SIWN): This measure is to check if t_1 is a substring of t_2, considering synonymy between words using WordNet synsets. $Score_{SIWN}(t_1, t_2)$ is set to 1 if there is such evidence; otherwise, it is set to 0.

Lexical-syntactic pattern (LSP): This measure is to find how much more evidence for $t_1 \gg t_2$ is found in the Web than for $t_2 \gg t_1$. Specifically, a list of manually constructed hypernym patterns (e.g. "t_2 is a t_1") is queried with a Web search engine to estimate the number of evidence for $t_1 \gg t_2$ from the Web. The LSP measure is calculated as follows, where $C_{Web}(t_1, t_2)$ denotes the collection of search results:

$$Score_{LSP}(t_1, t_2) = \frac{log(|C_{Web}(t_1,t_2)|)}{1+log(|C_{Web}(t_2,t_1)|)}$$

Syntactic contextual subsumption (SCS): The idea of this measure is to derive the hypernymy evidence for two terms t_1 and t_2 from their syntactic contexts, particularly from the triples of (subject,verb,object). They observe that if the context set of t_1 mostly contains that of t_2 but not vice versa, then t_1 is likely to be a hypernym of t_2. To implement this idea, they first find the most common relation (or verb) r between t_1 and t_2, and use the queries "t_1 r" and "t_2 r" to construct two corpora $Corpus^{\Gamma}_{t_1}$ and $Corpus^{\Gamma}_{t_2}$ for t_1 and t_2, respectively. Then the syntactic context sets are created from these contextual corpora using a non-taxonomic relation identification method. The details of calculating $Score_{SCS}(t_1, t_2)$ can be found in (Tuan et al., 2014).

They linearly combine the three scores as follows:

$$\begin{aligned} Score(t_1, t_2) = \ & \alpha \times Score_{SIWN}(t_1, t_2) \\ & + \beta \times Score_{LSP}(t_1, t_2) + \gamma \times Score_{SCS}(t_1, t_2) \end{aligned} \quad (1)$$

If $Score(t_1, t_2)$ is greater than a threshold value, then t_1 is regarded as a hypernym of t_2. We use the same values of α, β and γ as in (Tuan et al., 2014).

3.2 Trustiness of the evidence data

We introduce our method of estimating the trustiness of a given source text in Section 3.2.1 and explain how to incorporate it into the work of (Tuan et al., 2014) in Section 3.2.2.

3.2.1 Collecting trustiness score of the evidence data

Given a data source (e.g. Web page), we consider four aspects of trustiness as follows:

- Importance: A data source may be important if it is referenced by many other sources.
- Popularity: If a data source is accessed by many people, it is considered popular.
- Authority: If the data is created by a trusted agency, such as government and education institute, it may be more trustful than others from less trusted sources such as forums and social media.
- Accuracy: If the data contains many pieces of accurate information, it seems trustful.

Importance

To measure the importance of a Web page (d) as data source, we use the Google PageRank score[1] ($Score_{PageRank}(d)$) that is calculated based on the number and quality of links to the page. The PageRank scores have the scale from 0 to 9, where the bigger score means more importance than the lower one. Using this score, the importance of a page is calculated as follows:

$$Trust_{Imp}(d) = \frac{1}{10 - Score_{PageRank}(d)} \quad (2)$$

Note that we use the non-linearity for PageRank score rather than just normalizing PageRank to 0-1. The reason is to vary the gaps between the important sites (which usually have the PageRank score value from 7-10) and majority unimportant site (which usually have the PageRank score value less than 5).

Popularity

We use Alexa's Traffic Rank[2] as the measure of popularity, ($Score_{Alexa}(d)$) which is based on the traffic data provided by users in Alexa's global data panel over a rolling 3 month period. The Traffic Ranks are updated daily. A site's rank is

[1] http://searchengineland.com/what-is-google-pagerank-a-guide-for-searchers-webmasters-11068

[2] http://www.alexa.com/

based on a combined measure of unique visitors and page views. Using this rank, the popularity of a data source is calculated as follows:

$$Trust_{Pop}(d) = \frac{1}{log(Score_{Alexa}(d) + 1)} \quad (3)$$

We use log transform in the popularity score instead of, for example, linear scoring because we want to avoid the bias of the much large gap between the Alexa scores of different sites (e.g. one site has Alexa score 1000, but the other may have score 100,000)

Authority

We rank the authority of a data source based on the internet top-level domain (TLD). We observe that the pages with limited and registered TLD (e.g. *.gov*, *.mil*, *.edu*) are often more credible than those with open domain (e.g. *.com*, *.net*). Therefore, the authority score of a data source is calculated as follows:

$$Trust_{Auth}(d) = \begin{cases} 1 & \text{if TLD of } d \text{ is } .gov, .mil \text{ or } .edu \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

Note that there are some reasons we choose such implementation of Authority in an elementary way. First, we tried finer categorization of various domains, e.g. *.int* has score 1/3, *.com* has score 1/4, etc. However, the experimental results did not show much change of performance. In addition, there is controversy on which open TLD domains are more trustful than the others, e.g. it is difficult to judge whether a *.net* site is more trustful than *.org* or not. Thus, we let all open TLD domains have the same score.

Accuracy

If the data source contains many pieces of accurate information, it is trustful. Inspired by the idea of Dong et al. (2015), we estimate the accuracy of a data source by identifying correct and incorrect information in form of the triples *(Subject, Predicate, Object)* in the source, where *Subject*, *Predicate* and *Object* are normalized with regard to the knowledge base Freebase. The extraction of the triples includes six tasks: named entity recognition, part of speech tagging, dependency parsing, triple extraction, entity linkage (which maps mentions of proper nouns and their co-references to the corresponding entities in Freebase) and relation linkage. We use three information extraction (IE) tools (Angeli et al. (2014), Manning et al. (2014), MITIE[3]) for the first four tasks, and develop a method similar to Hachey et al. (2013) for the last two tasks of entity linkage and relation linkage.

Since the IE tools may produce noisy or unreliable triples, we use a voting scheme for triple extraction as follows: A triple is only considered to be true if it is extracted by at least two extractors. After obtaining all triples in the data source, we use the closed world assumption as follows: Given subject s and predicate p, $O(s, p)$ denotes the set of such objects that a triple (s,p,o) is found in Freebase. Now given a triple (s, p, o) found in the data source, if $o \in O(s, p)$, we conclude that the triple is correct; but if $o \notin O(s, p)$ and $|O(s, p)| > 0$, we conclude that the triple is incorrect. If $|O(s, p)| = 0$, we do not conclude anything about the triple, and the triple is removed from the set of facts found in the data source.

Given a data source d, we define $cf(d)$ as the number of correct facts, and $icf(d)$ as the number of incorrect facts found in d. The accuracy of d is calculated as follows:

$$Trust_{Accu}(d) = \frac{1}{1 + icf(d)^2} - \frac{1}{1 + cf(d)^2} \quad (5)$$

Combining trustiness scores

The final trustiness score of a data source is the linear combination of the four scores as follows:

$$\begin{aligned} Trust(d) = \alpha \times Trust_{Imp}(d) + \beta \times Trust_{Pop}(d) \\ + \gamma \times Trust_{Auth}(d) + \delta \times Trust_{Accu}(d) \end{aligned} \quad (6)$$

To estimate the optimal combination for parameters α, β, γ and δ, we apply linear regression algorithm (Hastie et al., 2009). For parameter learning, we manually list 50 websites as trusted sources (e.g. *stanford.edu*, *bbc.com*, *nasa.gov*), and the top 15 gossip websites listed in a site[4] as untrusted sources. Then we use the scores of their individual pages by the four methods to learn the parameters in Formula 6. The learning results are as follows: α=0.46, β=0.46, γ=2.03, δ=0.61.

3.2.2 Integrating trustiness into taxonomic relation identification methods

Given a data collection C, we define:

$$AvgTrust(C) = \frac{\sum_{d \in C} Trust(d)}{|C|}$$

[3]https://github.com/mit-nlp/MITIE

[4]http://www.ebizmba.com/articles/gossip-websites

as the average trustiness score of all data in C.

We integrate the trustiness score to the three taxonomic relation identification methods described in Section 3.1 as follows:

LSP method:
The LSP evidence score of the taxonomic relation between t_1 and t_2 is recalculated as follows:

$$Score_{LSP}^{Trust}(t_1,t_2) = Score_{LSP}(t_1,t_2)\times (AvgTrust(C_{Web}(t_1,t_2)) + AvgTrust(C_{Web}(t_2,t_1))) \quad (7)$$

The intuition of Formula 7 is that the original LSP evidence score is multiplied with the average trustiness score of all evidence documents for the taxonomic relation from the Web. If the number of Web search results is too large, we use only the first 1,000 results to estimate the average trustiness score.

SCS method:
Similarly, the SCS evidence score is recalculated as follows:

$$Score_{SCS}^{Trust}(t_1,t_2) = Score_{SCS}(t_1,t_2)\times (AvgTrust(Corpus_{t_1}^{\Gamma}) + AvgTrust(Corpus_{t_2}^{\Gamma})) \quad (8)$$

SIWN method:
This method does not use any evidence from the Web, and so its measure does not change as follows:

$$Score_{SIWN}^{Trust}(t_1,t_2) = Score_{SIWN}(t_1,t_2) \quad (9)$$

The three measures of trustiness are also linearly combined as follows:

$$Score^{Trust}(t_1,t_2) = \alpha \times Score_{SIWN}^{Trust}(t_1,t_2) + \beta \times Score_{LSP}^{Trust}(t_1,t_2) + \gamma \times Score_{SCS}^{Trust}(t_1,t_2) \quad (10)$$

The values of α, β, and γ in Formula 10 are identical to those for Formula 1.

3.3 Collective synonym evidence

3.3.1 Synonymy identification

We use the three following methods to collect synonyms: dictionaries, pattern matching, and supervised learning.

Dictionaries: Synonyms can be found in dictionaries like a general-purpose dictionary WordNet and also domain-specific ones. Since our domains of interest include virus, animals, and plants (see the next section for details), we also utilize MeSH[5], a well-known vocabulary in biomedicine.

Pattern matching: Given two terms t_1 and t_2, we use the following patterns to find their synonymy evidence from the Web:

- t_1 also [known|called|named|abbreviated] as t_2
- Other common name[s] of t_1 [is|are|include] t_2
- t_1, or t_2, is a
- t_1 (short for t_2)

, where $[a|b]$ denotes a choice between a and b. If the number of Web search results is greater than a threshold Ψ, t_1 is considered as a synonym of t_2.

Supervised learning: We randomly pick 100 pairs of synonyms in WordNet, and for each pair, we use the Web search engine to collect sample sentences in which both terms of the pair are mentioned. If the number of collected sentences is greater than 2000, we use only the first 2000 sentences for training. After that, we extract the following features from the sentences to train a logistic regression model (Hastie et al., 2009) for the synonymy identification:

- Headwords of the two terms
- Average distance between the terms
- Sequence of words between the terms
- Bag of words between the terms
- Dependency path between the terms (using Stanford parser (Klein and Manning, 2003))
- Bag of words on the dependency path

The average F-measure of the obtained model with 10-fold cross-validation is 81%. We use the learned model to identify more synonym pairs in the next step.

3.3.2 Embedding synonym information

Given a term t, we denote $Syn(t)$ as the set of synonyms of t (including t itself). The evidence scores of the SCS and LSP methods are recalculated with synonyms as follows:

$$Score_X^{Synonym}(t_1,t_2) = \sum_{\substack{t_1' \in Syn(t_1) \\ t_2' \in Syn(t_2)}} Score_X(t_1',t_2') \quad (11)$$

, where the variable X can be replaced with SCS and LSP.

[5] http://www.ncbi.nlm.nih.gov/mesh

The intuition of Formula (11) is that the evidence score of the taxonomic relation between two terms t_1 and t_2 can be boosted by adding all the evidence scores of taxonomic relations between them and their synonyms.

Again, as for the SIWN method, we do not change the evidence score as follows:

$$Score_{SIWN}^{Synonym}(t_1, t_2) = Score_{SIWN}(t_1, t_2)$$

3.4 Contrastive evidence

Given two terms t_1 and t_2, we use the following patterns to find their contrastive (thus negative) evidence from the Web:

- t_1 is not a t_2
- t_1 is not a [type|kind] of t_2
- t_1, unlike the t_2
- t_1 is different [from|with] t_2
- t_1 but not t_2
- t_1, not t_2

$WH(t_1, t_2)$ denotes the total number of Web search results, and the contrastive evidence score between t_1 and t_2 is computed as follows:

$$Contrast(t_1, t_2) = \log(WH(t_1, t_2) + 1) \quad (12)$$

Similar to the collective synonym evidence, the contrastive evidence score of taxonomic relation between t_1 and t_2 is boosted with the contrastive evidence scores of taxonomic relations between the two terms and their synonyms as follows:

$$Score^{Contrast}(t_1, t_2) = \frac{\sum_{\substack{t'_1 \in Syn(t_1) \\ t'_2 \in Syn(t_2)}} Contrast(t'_1, t'_2)}{|Syn(t_1)| * |Syn(t_2)|} \quad (13)$$

3.5 Combining trustiness, synonym and contrastive evidence

We combine all the three features into the system of (Tuan et al., 2014) as follows:

$$\begin{aligned} Score_X^{Final}(t_1, t_2) = & Score_X^{Trust}(t_1, t_2) \\ & + Score_X^{Synonym}(t_1, t_2) \end{aligned} \quad (14)$$

, where the variable X can be replaced with each of the three taxonomic relation evidence measures (i.e. SCS, LSP, SIWN). The final combined score is calculated as follows:

$$\begin{aligned} Score_{Combined}^{Final}(t_1, t_2) = & \ \alpha \times Score_{SIWN}^{Final}(t_1, t_2) \\ & + \beta \times Score_{LSP}^{Final}(t_1, t_2) + \gamma \times Score_{SCS}^{Final}(t_1, t_2) \\ & - \delta \times Score^{Contrast}(t_1, t_2) \end{aligned} \quad (15)$$

For each ordered pair of terms t_1 and t_2, if $Score_{Combined}^{Final}(t_1, t_2)$ is greater than a threshold value, then t_1 is considered as a hypernym of t_2.

We estimate the optimal values of parameters α, β, γ and δ in Formula 15 with ridge regression technique (Hastie et al., 2009) as follows: First, we randomly select 100 taxonomic relations in Animal domain as the training set. For each taxonomic relation $t_1 \gg t_2$, its evidence score is estimated as $\tau + \frac{1}{Dist(t_1,t_2)}$, where τ is the threshold value for $Score_{Combined}^{Final}$, and $Dist(t_1, t_2)$ is the length of the shortest path between t_1 and t_2 found in WordNet. Then we use our system to find evidence scores with taxonomic relation identification methods in Formulas 13 and 14. Finally, we build the training set using Formula 15, and use the ridge regression algorithm to learn that the best value for α is 1.31, β 1.57, γ 1.24 and δ 0.79, where τ=2.3.

4 Experiment

4.1 Datasets

We evaluate our method for taxonomy construction against the following collections of six domains:

- Artificial Intelligence (AI) domain: The corpus consists of 4,976 papers extracted from the IJCAI proceedings from 1969 to 2014 and the ACL archives from year 1979 to 2014.
- Finance domain: The corpus consists of 1,253 papers extracted from the freely available collection of "Journal of Financial Economics" from 1995 to 2012 and from "Review Of Finance" from 1997 to 2012.
- Virus domain: We submit the query "virus" to PUBMED search engine [6] and retrieve the first 20,000 abstracts as the corpus of the virus domain.
- Animals, Plants and Vehicles domains: Collections of Web pages crawled by using the bootstrapping algorithm described by Kozareva et al. (2008).

We report the results of two experiments in this section: (1) Evaluating the construction of new taxonomies for Finance and AI domains (Section 4.2), and (2) comparing with the curated

[6] http://www.ncbi.nlm.nih.gov/pubmed

databases' sub-hierarchies. We compare our approach with other three state-of-the-art methods in the literature, i.e. (Kozareva and Hovy, 2010), (Navigli et al., 2011) and (Tuan et al., 2014) (Section 4.3). In addition, for Animal domain, we also compare with the reported performance of Bansal et al. (2014), a recent work to construct taxonomy using belief propagation.

4.2 Constructing new taxonomies for Finance and AI domains

Referential taxonomy structures such as WordNet and OpenCyc are widely used in semantic analytics applications. However, their coverage is limited to common, well-known areas, and many specific domains like Finance and AI are not well covered in those structures. Therefore, an automatic method which can induce taxonomies for those specific domains can greatly contribute to the process of knowledge discovery.

To estimate the precision of a given method, we randomly choose 100 relations among the results of the method and manually check their correctness. The results summarized in Table 1 show that our method extracts much more relations, though with slightly lower precision, than Kozareva et al. (2008) and Navigli and Velardi (2004). Note that due to the lack of gold standards in these two domains, we do not compare the methods in terms of F-score, which we will measure with curated databases in the next section. Compared to Tuan et al. (2014), which can be considered as the baseline of our approach, our method has significant improvement in both precision and the number of extracted relations. It indicates that the three incorporated features of trustiness, and synonym and contrastive evidence are effective in improving the performance of existing taxonomy construction methods.

	Finance		AI	
	P	N	P	N
Kozareva	**90%**	753	**94%**	950
Navigli	88%	1161	93%	1711
Tuan	85%	1312	90%	1927
Our method	88%	**1570**	92%	**2168**

Table 1: Experiment result for finance and AI domains. P stands for Precision, and N indicates the number of extracted relations.

4.3 Evaluation against curated databases

We evaluate automatically constructed taxonomies for four domains (i.e. Animal, Plant, Vehicle, Virus) against the corresponding sub-hierarchies of curated databases. For Animal, Plant and Vehicle domains, we use WordNet as the gold standards, whereas for Virus domain, we use MeSH sub-hierarchy of virus as the reference.

Note that in this comparison, to be fair, we change our algorithm to avoid using WordNet in identifying taxonomic relations (i.e. SIWN algorithm), and we only use the exact string-matching comparison without WordNet. The evaluation uses the following measures:

$$Precision = \frac{\#relations\ found\ in\ database\ and\ by\ the\ method}{\#relations\ found\ by\ the\ method}$$

$$Recall = \frac{\#relations\ found\ in\ database\ and\ by\ the\ method}{\#relations\ found\ in\ database}$$

To understand the individual contribution of the three introduced features (i.e. trustiness, synonym, contrast), we also evaluate our method only with one of the three features each time, as well as with all the three features (denoted as "Combined").

Tables 2 and 3 summarize the experiment results. Our combined method achieves significantly better performance than the previous state-of-the-art methods in terms of F-measure and Recall (t-test, p-value < 0.05) for all the four domains. For Animal domain, it also shows higher performance than the reported performance of Bansal et al. (2014). In addition, the proposed method improves the baseline (i.e. Tuan et al. (2014)) up to 4%-10% of F-measure.

Furthermore, we find that the three features have different contribution to the performance improvement. The trustiness feature contributes to the improvement on both precision and recall. The synonym feature has the tendency of improving the recall further than the trustiness, whereas the contrastive evidence improves the precision. Note that we discussed these different contributions of the features in the Introduction.

	Animal			Plant		
	P	R	F	P	R	F
Kozareva	**98**	38	55	**97**	39	56
Navigli	97	44	61	**97**	38	55
Bansal	84	55	67			
Tuan	95	56	70	95	53	68
Trustiness	97	61	75	**97**	56	71
Synonym	92	**65**	76	93	58	71
Contrast	97	55	70	**97**	53	69
Combined	97	**65**	**78**	96	**59**	**73**

Table 2: Experiment results for animal and plant domains. P stands for Precision, R Recall, and F F-score. The unit is %.

	Vehicle			Virus		
	P	R	F	P	R	F
Kozareva	**99**	60	75	97	31	47
Navigli	91	49	64	**99**	37	54
Tuan	93	69	79	93	43	59
Trustiness	96	72	82	97	48	64
Synonym	91	72	80	91	53	67
Contrastive	97	68	80	98	42	59
Combined	95	**73**	**83**	96	**54**	**69**

Table 3: Experiment results for vehicle and virus domains. P stands for Precision, R Recall, and F F-score. The unit is %.

4.3.1 Evaluation of individual methods for trustiness and synonymy identification

We evaluate the individual methods for trustiness measurement and synonymy identification described in Sections 3.2.1 and 3.3.1. For this purpose, we evaluate our system only with one of the individual methods at a time (i.e. importance, popularity, authority and accuracy for trustiness measure, and dictionary, pattern matching, and machine learning methods for synonymy identification).

As summarized in Table 4, the "Importance" and "Accuracy" methods for trustiness measurement based on PageRank and IE systems, respectively, have more contribution than the others. Similarly, the experiment results indicate that the "Machine learning" method has the most contribution among the three methods of synonymy identification.

In addition, we also examine the interdependence of the four introduced aspects of trustiness by running the system with the combination of only two aspects, Importance and Accuracy. The results in all domains show that when combining only the Importance and Accuracy, the system almost achieves the same performance to that of the combined system with all four criteria, except for the Plant domain. It can be explained as the Importance aspect (which is expressed as the PageRank score) may subsume the Popularity and Authority aspects. Another interesting point is that the performance of Accuracy, which is solely based on the local information from the website, when applied individually, is almost the same with that of Importance which is based on the distributed information. It shows that the method of ranking of the sites based on the knowledge-based facts can achieve the effectiveness as good as the traditional ranking method using PageRank score.

	Animal	Plant	Vehicle	Virus
Trustiness:				
Importance	74%	**70%**	81%	63%
Popularity	72%	69%	81%	61%
Authority	72%	69%	80%	61%
Accuracy	73%	**70%**	81%	62%
Imp + Accu	**75%**	**70%**	**82%**	**64%**
Synonym:				
Dictionaries	73%	69%	79%	62%
Pattern matching	**74%**	69%	**80%**	64%
Machine learning	**74%**	**70%**	**80%**	**65%**

Table 4: Contribution of individual trustiness measures and collective synonym evidence in terms of F-measure. Imp stands for Important and Accu stands for Accuracy

4.4 Discussion

4.4.1 Case studies

We give two examples to illustrate how the proposed features help to infer correct taxonomic relations and filter out incorrect relations. Our baseline (Tuan et al. (2014)) extracts an incorrect taxonomic relation between 'fox' and 'flying fox' due to the following reasons: (1) 'flying fox' includes 'fox' (SIWN) and (2) untrusted sources such as a public forum[7] support the relation. Using our proposed method, this relation is filtered out because those untrusted sources are discouraged by the trustiness feature, and also because there are contrastive evidence[8] saying that 'flying fox' is NOT a 'fox'. Specifically, the average trustiness score of LSP method of the sources for the invalid relation (i.e. $AvgTrust(C_{Web}(fox, flying\ fox)) + AvgTrust(C_{Web}(flying\ fox, fox))$) is 0.63, which is lower than the average of those scores, 0.90. Also, the collective contrastive evidence score (i.e. $Score^{Contrast}(fox, flying\ fox)$) is 1.10, which is higher than the average collective contrastive score, 0.32.

On the other hand, the true taxonomic relation between 'bat' and 'flying fox' is not identified by the baseline, mainly due to the rare mention of this relation in the Web. However, our proposed method can recognize this relation because of two reasons: (1) 'flying fox' has many synonyms such as 'fruit bat', 'pteropus', 'kalong', and 'megabat', and there are much evidence that these synonyms are kinds of 'bat' (i.e. using the collective synonym evidence). (2) The evidence for the taxonomic relation between 'fly-

[7] http://redwall.wikia.com/wiki/User:Ferretmaiden/Archive3
[8] http://en.cc-english.com/index.php?shownews-1397

ing fox' and 'bat', though rare, is from trusted sites[9] which are maintained by scientists. Thus, the trustiness feature helps to boost the evidence score for this relation over the threshold value. Specifically, the average trustiness score of LSP method (i.e. $(AvgTrust(C_{Web}(bat, flying\ fox)) + AvgTrust(C_{Web}(flying\ fox, bat)))$), 2.84, is higher than the average in total, 0.90.

We further investigate on 256 taxonomic relations that were missed by the baseline but correctly identified by the proposed method. The average $Score_{LSP}$ and the average $Score_{SCS}$ of the relations by the baseline are 0.35 and 0.60, respectively, while those by the proposed method are 1.17 and 0.82, respectively. We thus find that the proposed method is more effective in correctly improving the LSP method than the SCS method.

4.4.2 Empirical comparison with WordNet

By error analysis, we find that our results may complement WordNet. For example, in Animal domain, our method identifies 'wild sheep' as a hyponym of 'sheep', but in WordNet, they are siblings. However, many references [10] [11] consider 'wild sheep' as a species of 'sheep'. Another such example is that our system recognizes 'aquatic vertebrate' as a hypernym of 'aquatic mammal', but WordNet places them in different subtrees incorrectly [12]. Therefore, our results may help restructure and extend WordNet.

4.4.3 Threshold tuning

Our scoring methods utilize several thresholds to select relations of high ranks. Here we discuss them in details below.

The threshold value Ψ for the pattern matching method in Section 3.3.1 controls the number of synonymy relations extracted from text. The threshold value for $Score^{Final}_{Combined}$ of Formula 15 in Section 3.5 controls the number of extracted taxonomic relations. In general, the larger these threshold values are, the higher number of synonyms and taxonomic relations we can get. In our experiments, we found that the threshold values for Ψ between 100 and 120, and those for $Score^{Final}_{Combined}$ between 2.3 and 2.5 generally help the system achieve the best performance.

[9] http://krjsoutheastasianrainforests.weebly.com/animals-in-biome-and-habitat-structures.html

[10] http://en.wikipedia.org/wiki/Ovis

[11] http://www.bjornefabrikken.no/side/norwegian-sheep/

[12] http://en.wikipedia.org/wiki/Aquatic_mammal

5 Conclusion

In this paper, we propose the features of trustiness, and synonym and contrastive collective evidence for the task of taxonomy construction, and show that these features help the system improve the performance significantly. As future work, we will investigate into the task of automatically constructing patterns for the pattern matching methods in Sections 3.3 and 3.4, to improve coverage. We will also enhance the accuracy measure of trustiness, based on the observation that some untrusted sites copy information from other sites to make them look more trustful.

References

Gabor Angeli, Julie Tibshirani, Jean Y. Wu, and Christopher D. Manning. 2014. Combining Distant and Partial Supervision for Relation Extraction. *Proceedings of the Conference on Empirical Methods on Natural Language Processing*, pages 1556–1567.

Mohit Bansal, David Burkett, Gerard De Melo, and Dan Klein. 2014. Structured learning for taxonomy induction with belief propagation. *Proceedings of the 52nd Annual Meeting of the ACL*, pages 1041–1051.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 1247–1250.

Xin L. Dong, Evgeniy Gabrilovich, Kevin Murphy, Van Dang, Wilko Horn, Camillo Lugaresi, Shaohua Sun, and Wei Zhang. 2015. Knowledge-Based Trust: Estimating the Trustworthiness of Web Sources. *Proceedings of the VLDB Endowment*, 8(9).

Lucas Drumond and Rosario Girardi. 2010. Extracting ontology concept hierarchies from text using markov logic. *Proceedings of the ACM Symposium on Applied Computing*, pages 1354–1358.

Samah Fodeh, Bill Punch, and Pang N. Tan. 2011. On ontology-driven document clustering using core semantic features. *Knowledge and information systems*, 28(2):395–421.

Hermine N. Fotzo and Patrick Gallinari. 2004. Learning "generalization/specialization" relations between concepts - application for automatically building thematic document hierarchies. *Proceedings of the 7th International Conference on Computer-Assisted Information Retrieval*.

Ruiji Fu, Jiang Guo, Bing Qin, Wanxiang Che, Haifeng Wang, and Ting Liu. 2014. Learning semantic hierarchies via word embeddings. *Proceedings of the 52nd Annual Meeting of the ACL*, pages 1199–1209.

Ben Hachey, Will Radford, Joel Nothman, Matthew Honnibal, and James R. Curran. 2013. Evaluating entity linking with Wikipedia. *Artificial intelligence*, 194:130–150.

Sanda M. Harabagiu, Steven J. Maiorano, and Marius A. Pasca. 2003. Open-domain textual question answering techniques. *Natural Language Engineering*, 9(3):231–267.

Trevor Hastie, Robert Tibshirani, and Jerome H. Friedman. 2009. *The elements of statistical learning*. Springer-Verlag.

Marti A. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. *Proceedings of the 14th Conference on Computational Linguistics*, pages 539–545.

Jung J. Kim, Zhuo Zhang, Jong C. Park, and See K. Ng. 2006. Biocontrasts: extracting and exploiting protein-protein contrastive relations from biomedical literature. *Bioinformatics*, 22(5):597–605.

Dan Klein and Christopher D. Manning. 2003. Accurate Unlexicalized Parsing. *Proceedings of the 41st Annual Meeting of the ACL*, pages 423–430.

Zornitsa Kozareva and Eduard Hovy. 2010. A Semi-supervised Method to Learn and Construct Taxonomies Using the Web. *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1110–1118.

Zornitsa Kozareva, Ellen Riloff, and Eduard H. Hovy. 2008. Semantic Class Learning from the Web with Hyponym Pattern Linkage Graphs. *Proceedings of the 46th Annual Meeting of the ACL*, pages 1048–1056.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. *Proceedings of the 52nd Annual Meeting of the ACL*, pages 55–60.

Cynthia Matuszek, John Cabral, Michael J. Witbrock, and John DeOliveira. 2006. An introduction to the syntax and content of cyc. *Proceedings of the AAAI Spring Symposium*, pages 44–49.

George A. Miller. 1995. WordNet: a Lexical Database for English. *Communications of the ACM*, 38(11):39–41.

Roberto Navigli and Paola Velardi. 2004. Learning Domain Ontologies from Document Warehouses and Dedicated Web Sites. *Computational Linguistics*, 30(2):151–179.

Roberto Navigli, Paola Velardi, and Stefano Faralli. 2011. A Graph-based Algorithm for Inducing Lexical Taxonomies from Scratch. *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pages 1872–1877.

Luu A. Tuan, Jung J. Kim, and See K. Ng. 2014. Taxonomy Construction using Syntactic Contextual Evidence. *Proceedings of the EMNLP conference*, pages 810–819.

Wu Wentao, Li Hongsong, Wang Haixun, and Kenny. Q. Zhu. 2012. Probase: A probabilistic taxonomy for text understanding. *Proceedings of the ACM SIGMOD conference*, pages 481–492.

Wilson Wong, Wei Liu, and Mohammed Bennamoun. 2007. Tree-traversing ant algorithm for term clustering based on featureless similarities. *Data Mining and Knowledge Discovery*, 15(3):349–381.

Hui Yang and Jamie Callan. 2009. A Metric-based Framework for Automatic Taxonomy Induction. *Proceedings of the 47th Annual Meeting of the ACL*, pages 271–279.

Xingwei Zhu, Zhao Y. Ming, and Tat S. Chua. 2013. Topic hierarchy construction for the organization of multi-source user generated contents. *Proceedings of the 36th ACM SIGIR conference*, pages 233–242.

Learning to Automatically Solve Logic Grid Puzzles

Arindam Mitra
SCIDSE
Arizona State University
amitra7@asu.edu

Chitta Baral
SCIDSE
Arizona State University
chitta@asu.edu

Abstract

Logic grid puzzle is a genre of logic puzzles in which we are given (in a natural language) a scenario, the object to be deduced and certain clues. The reader has to figure out the solution using the clues provided and some generic domain constraints. In this paper, we present a system, LOGICIA, that takes a logic grid puzzle and the set of elements in the puzzle and tries to solve it by translating it to the knowledge representation and reasoning language of Answer Set Programming (ASP) and then using an ASP solver. The translation to ASP involves extraction of entities and their relations from the clues. For that we use a novel learning based approach which uses varied supervision, including the entities present in a clue and the expected representation of a clue in ASP. Our system, LOGICIA, learns to automatically translate a clue with 81.11% accuracy and is able to solve 71% of the problems of a corpus. This is the first learning system that can solve logic grid puzzles described in natural language in a fully automated manner. The code and the data will be made publicly available at http://bioai.lab.asu.edu/logicgridpuzzles.

1 Introduction

Understanding natural language to solve problems be it algebraic word problems (Kushman et al., 2014; Hosseini et al., 2014) or questions from biology texts (Berant et al., 2014; Kim et al., 2011), has attracted a lot of research interest over the past few decades. For NLP, these problems are of particular interest as they are concise, yet rich in information. In this paper, we attempt to solve another problem of this kind, known as Logic Grid Puzzle. Problem.1 shows an example of the same. Puzzle problems in the same spirit as the previously mentioned science problems, do not restrict the vocabulary; they use everyday language and have diverse background stories. The puzzle problems, however, are unique in their requirement of high precision understanding of the text. For a puzzle problem, the solution is never in the text and requires involved reasoning. Moreover, one needs to correctly understand each of the given clues to successfully solve a problem. Another interesting property is that only a small core of the world knowledge, noticeably spatial, temporal and knowledge related to numbers, is crucial to solve these problems.

PROBLEM .1 A LOGIC GRID PUZZLE

Waterford Spa had a full appointment calendar booked today. Help Janice figure out the schedule by matching each masseuse to her client, and determine the total price for each.

1. Hannah paid more than Teri's client.
2. Freda paid 20 dollars more than Lynda's client.
3. Hannah paid 10 dollars less than Nancy's client.
4. Nancy's client, Hannah and Ginger were all different clients.
5. Hannah was either the person who paid $180 or Lynda's client.

Clients: *Aimee, Ginger, Freda, Hannah*.
Prices: *$150, $160, $170, $180*.
Masseuses: *Lynda, Nancy, Tery, Whitney*.

A logic grid puzzle contains a set of categories and an equal number of elements in each category.

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1023–1033,
Lisbon, Portugal, 17-21 September 2015.

And the goal is to find out which elements are linked together based on a series of given clues. Each element is used only once. Each puzzle has a unique solution and can be solved using logical reasoning. A logic grid puzzle is called a (n, m)-puzzle if it contains n categories and each category has m elements. For the example in Problem.1, there are three categories, namely *clients, prices, masseuses* and each category has four elements which are shown in the respective columns. A total of five clues are given in free text and the goal is to find the members of the four tuples, where each tuple shall contain exactly one element from each category such that all the members in a tuple are linked together.

To solve such a puzzle problem, it is crucial to understand the clues (for example, "Hannah paid more than Teri's client."). Each clue talks about a set of entities (for example, "Hannah", "client", "Terry") and their relations ("a greater-than relation between Hannah and the client of Terry on the basis of payment"). Our system, LOGICIA, learns to discover these entities and the underlying semantics of the relations that exist between them. Once the relations are discovered, a pair of Answer Set Programming (ASP) (Baral, 2003) rules are created. The reasoning module takes these ASP rules as input and finds a group configuration that satisfies all the clues. LOGICIA has "knowledge" about a fixed set of predicates which models different relations that hold between entities in a puzzle world. Clues in the puzzle text that are converted into ASP rules, use these predicates as building blocks. In this research, our goal is to build a system which can automatically do this conversion and then reason over it to find the solution. The set of predicates that the reasoning model is aware of is not sufficient to represent all logic grid puzzles. The family of logic grid puzzles is broad and contains variety of clues. Our future work involves dealing with such a diverse set of relations. In this work we assume that the relations in Table 1 are sufficient to represent the clues. Following are some examples of clues that cannot be modeled using the predicates in Table 1.

- *Esther's brother's* seat is at one end of the block of seven.
- The *writer* of *Lifetime Ambition* has a *first name* with more letters than that of the *tennis star*.
- *Edward* was *two places* behind *Salim* in one of the lines, *both* being in odd-numbered positions.
- *Performers* who finished in the top *three* places, in no particular order, are *Tanya* , the *person* who performed the *fox trot*, and the *one* who performed the *waltz*.

The rest of the paper is organized as follows: in section 2, we describe the representation of a puzzle problem in ASP and delineate how it helps in reasoning; in section 3, we present our novel method for learning to automatically translate a logic problem described in natural language to its ASP counterpart. In section 4, we describe the related works. In section 5, we discuss the detailed experimental evaluation of our system. Finally, section 6 concludes our paper.

2 Puzzle Representation

Answer Set Programming (ASP) (Baral, 2003; Lifschitz, 1999; Gelfond and Lifschitz, 1991) has been used to represent a puzzle and reason over it. This choice is facilitated by the two important reasons: 1) non-monotonic reasoning may occur in a puzzle (Nagy and Allwein, 2004) and 2) ASP constructs greatly simplify the reasoning module, as we will see in this section. We now briefly describe a part of ASP. Our discussion is informal. For a detailed account of the language, readers are referred to (Baral, 2003).

2.1 Answer Set Programming

An answer set program is a collection of rules of the form,

$$L_0 \mid ... \mid L_k \text{ :- } L_{k+1}, ..., L_m, \textbf{not } L_{m+1}, ..., \textbf{not } L_n$$

where each of the L_i's is a literal in the sense of a classical logic. Intuitively, the above rule means that if $L_{k+1}, ..., L_m$ are to be true and if $L_{m+1}, ..., L_n$ can be safely assumed to be false then at least one of $L_0, ..., L_k$ must be true. The left-hand side of an ASP rule is called the *head* and the right-hand side is called the *body*. A rule with no *head* is often referred to as a $constraint$. A rule with empty *body* is referred to as a $fact$ and written as,

$$L_0 \mid L_1 \mid ... \mid L_k.$$

Example

fly(X) :- bird(X), not ab(X).

The above program represents the knowledge that "Most birds fly". If we add the following rule ($fact$) to the program,

bird(penguin).

the answer set of the program will contain the belief that penguins can fly, $\{bird(penguin), fly(penguin)\}$. However, adding one more fact, '$ab(penguin)$.', to convey that the penguin is an abnormal bird, will change the belief that the penguin can fly and correspondingly the answer set, $\{bird(penguin), ab(penguin)\}$, will not contain the fact, $fly(penguin)$.

Choice Rule

$$m\ \{p(X) : q(X)\}\ n : -L_1, ..., L_k, ..., \textbf{not}\ L_n.$$

Rules of this type allow inclusion in the program's answer sets of arbitrary collections S of atoms of the form $p(t)$ such that, $m \leq \mid S \mid \leq n$ and if $p(t) \in S$ then $q(t)$ belongs to the corresponding answer set.

2.2 Representing Puzzle Entities

A (m, n)-puzzle problem contains m categories and n elements in each category. The term 'puzzle entity' is used to refer to any of them. Each category is assigned an unique index, denoted by the predicate $cindex/1$ (the number after the '/' denotes the arity of the predicate). The predicate $etype/2$ captures this association. Each element is represented, by the $element/2$ predicate which connects a category index to its element. The predicate $eindex/1$, denotes the tuple indices. The following blocks of code shows the representation of the entities for the puzzle in Problem.1.

```
cindex(1...3).
eindex(1...4).

etype(1,clients).
etype(2,prices).
etype(3,masseuses).

element(1,aimee;;1,ginger).
element(1,freda;;1,hannah).
element(2,150;;2,160).
element(2,170;;2,180).
element(3,lynda;;3,nancy).
element(3,teri;;3,whitney).
```

2.3 Representing Solution

Solution to a logic grid puzzle is a set of tuples containing related elements. The $tuple/3$ predicate captures this tuple membership information of the elements. For example, the fact, $tuple(2, 1, aimee)$, states that the element $aimee$ from the category with index 1 is in the tuple 2. The rel/m predicate captures all the elements in a tuple for a (m, n)-puzzle and is defined using the $tuple/3$ predicate.

2.4 Domain Constraints

In the proposed approach, the logic grid puzzle problem is solved as a constraint satisfaction problem. Given a puzzle problem the goal is to enumerate over all possible configurations of $tuple/3$, and select the one which does not violate the constraints specified in the clues. However, 1) each tuple in a logic grid puzzle will contain exactly one element from each category and 2) an element will belong to exactly one tuple. These constraints come from the specification of a puzzle problem and will hold irrespective of the problem instance. Following blocks of code show an elegant representation of these domain constraints in ASP along with the enumeration.

```
%enumerate over the tuple
%assignments with constraint#1
1 {
    tuple(G,Cat,Elem):
    element(Cat,Elem)
} 1 :- cindex(Cat),
       eindex(G).

%domain constraint#2
:-tuple(G1,Cat,Elem),
  tuple(G2,Cat,Elem),
  G1!=G2.
```

2.5 Representing clues

Each clue describes some entities and the relations that hold between them. In its simplest form, the relations will suggest if the entities are linked together or not. However, the underlying semantics of such relations can be deep such as the one in clue 5 of Problem.1. There are different ways to express the same relation that holds between entities. For example, in Problem.1, the *possessive* relation has been used to express the linking between clients and masseuses; and the word *paid*

expresses the linking between the clients and the prices. Depending on the puzzles the phrases that are used to express the relations will vary and it is crucial to identify their underlying semantics to solve the problems in systematic way.

In the current version, the reasoning module has knowledge of a selected set of relations and the translation module tries to represent the clue as a conjunction of these relations. All these relations and their underlying meanings are described in table 1. In this subsection, we describe the representation of a clue in terms of these relations in ASP and show how it is used by the reasoning module. In the next section, we present our approach to automate this translation.

Let us consider the clues and their representation from Problem.1:

[1] Hannah paid more than Teri's client.

```
clue1 :-
  greaterThan(hannah,1,X,1,2),
  sameTuple(X,1,teri,3).
:- not clue1.
```

The first rule $clue1$ evaluates to true (will be in the answer set) if the element from category 1 with value $hannah$ is linked to some element from category 2 which has a higher value than the element from its own category which is linked to an element from category 1 which is linked to $teri$ from category 3. Since the desired solution must satisfy the relations described in the clue, the second ASP rule is added. A rule of this form that does not have a $head$ is known as a $constraint$ and the program must satisfy it to have an answer set. As the reasoning module enumerates over all possible configurations, in some cases the $clue1$ will not hold and subsequently those branches will be pruned. Similar constraints will be added for all clues. In the below, we show some more examples. A configuration which satisfies all the clue constraints and the domain constraints described in the previous section, will be accepted as the solution to the puzzle.

[2] Nancy's client, Hannah and Ginger were all different clients.

```
clue4 :-
  diffTuple(hannah,1,ginger,1),
  diffTuple(hannah,1,X,1),
  diffTuple(X,1,ginger,1),
  sameTuple(X,1,nancy,3).
:- not clue4.
```

[3] Hannah was either the person who paid $180 or Lynda's client.

```
clue5 :-
   eitherOr(hannah,1,X,1,Y,1),
   sameTuple(X,1,180,2).
   sameTuple(Y,1,lynda,3).
:- clue5.
```

3 Learning Translation

To automate the translation of a clue to the pair of ASP rules, the translation module needs to identify the entities that are present in the clue, their category and their value; and the underlying interpretations of all the relations that hold between them. Once all the relation instances $\{R_1(arg_1, ..., arg_{p_1})$,..., $R_q(arg_1, ..., arg_{p_q})\}$, in the clue are identified, the ASP representation of the clue is generated in the following way:

$$clue : -R_1(arg_1..., arg_{p_1}), ..., R_q(arg_1..., arg_{p_q})$$

The entity classification problem for logic grid puzzles poses several challenges. First, the existence of a wide variety in the set of entities. Entities can be names of objects, time related to some event, numbers, dates, currency, some form of ID etc. And it is not necessary that the entities in puzzles are nouns. It can be verbs, adjectives etc. Second and of paramount important, the "category" of a puzzle "element" is specific to a puzzle problem. Same element may have different category in different problems. Also, a constituent in a clue which refers to an entity in a particular problem may not refer to an entity in another problem. We formalize this problem in this section and propose one approach to solve the problem. Next, we discuss the method that is used to extract relations from clues. To the best of our knowledge, this type of entity classification problem has never been studied before.

Relation	Interpretation
sameTuple(E1, C1, E2, C2)	States that two elements, (C1,E1) and (C2,E2) are in the same tuple. The dictionary also contains the negation of it, *diffTuple*(E1, C1, E2, C2).
referrent(E1, C1, E2, C2)	States that the elements are identical.
posDiff(E1, C1, E2, C2, N1, NC1)	If (C1,E1) is related to (NC1,X1) and (E2,C2) is related to (NC1,X2), then difference(X1,X2)=N1. Similarly the dictionary contains the predicate *negDiff*.
greaterThan(E1, C1, E2, C2, NC1)	Similar to *posDiff* however the difference(X1,X2) $>$ 0. The dictionary also contains its opposite predicate *lessThan*.
members(E1, C1, E2, C2,..., EN, CN)	All the elements are distinct and do not share a tuple.
eitherOr(E1, C1, E2, C2,..., EN, CN)	The first element is related to one of the last $N-1$ elements. The last $N-1$ elements are assumed to be different unless contradicts with other beliefs.
referrent22(E1, C1, E2, C2, E3, C3, E4, C4)	The first two elements are different and referring to the last two elements.

Table 1: Describes the various relations that are part of the reasoning module.

3.1 Entity Classification

The entity classification problem is defined as follows:

Problem description Given m categories C_1, ..., C_m and a text T, each category C_i, $1 \leq i \leq m$, contains a collection of elements E_i and an optional textual description d_i. The goal is to find the class information of all the constituents in the text T. Each category contributes two classes, where one of them represents the category itself and the other represents an instance of that category. Also, a constituent may not refer to any category or any instance of it, in that case the class of that constituent is $null$. So, there are a total $2m+1$ classes and a constituent will take one value from them.

Example In the puzzle of Problem.1, there are 3 categories with, C_1 = {Aimee, Freda, Ginger, Hannah}, C_2 = {\$150, \$160, \$170, \$180}, C_3 = {Lynda, Nancy, Terry, Whiteney} and d_1 = "clients", d_2 = "prices", d_3 = "masseuses". The text T, is the concatenation of all clues. In the last clue, there are a total 5 entities, namely "Hannah", "person", "\$180", "Lydia","client" and the corresponding classes are "Instance of C_1", "Instance of C_1", "Instance of C_2", "Instance of C_3" and "Instance of C_1" respectively. The remaining constituents in that clue have the class value $null$. The constituent "clients" in the fourth clue refers to the category C_1.

Our approach We model the Entity Classification problem as a decoding query on Pairwise Markov Network (Koller and Friedman, 2009; Kindermann et al., 1980; Zhang et al., 2001). A pairwise Markov network over a graph $\mathcal{H}$, is associated with a set of node potentials $\{\phi(X_i) : i = 1, ..., n\}$ and a set of edge potentials $\{\phi(X_i, X_j) : (X_i, X_j) \in \mathcal{H}\}$. Each node $X_i \in \mathcal{H}$, represents a random variable. Here, each X_i can take value from the set $\{1...2m+1\}$, denoting the class of the corresponding constituent in the text T.

In our implementation, the node potential captures the chances of that node to be classified as one of the possible categories without being affected by the given text T. And the edge potentials captures hints from the context in T for classification. After constructing the pairwise Markov network, a decoding query is issued to obtain the configuration that maximizes the joint probability distribution of the pairwise Markov network in consideration. The proposed approach is inspired by the following two observations: 1) to find the class of a constituent one needs some background knowledge; 2) however, background knowledge is not sufficient on its own, one also needs to understand the text to properly identify the class of each constituent. For example, let us consider the word "person" in clue 5 of Problem.1. Just skimming through the categories, one can discover that the word "person" is very unlikely to be a instance of the category "prices", which is from her knowledge about those constituents. However a proper

disambiguation may face an issue here as there are two different categories of human beings. To properly classify the word "person" it is necessary to go through the text.

The following paragraphs describe the construction of the grah $\mathcal{H}$, and the algorithm that is used in the computation of associated set of node potentials and edge potentials.

Construction of the graph While constructing the graph, we assign a label, L, to each edge in $\mathcal{H}$ which will be used in the edge potential computation. Let $\mathcal{D}_\mathcal{G}$ denotes the dependency graph of the text T obtained from the Stanford dependency parser (Chen and Manning, 2014) and $dep(v_1, v_2)$ denotes the grammatical relation between $(v_1, v_2) \in \mathcal{D}_\mathcal{G}$. Then the graph, $\mathcal{H}$, is constructed as follows:

1. Create a node in $\mathcal{H}$ for each constituent w_j in T if $w_j \in \mathcal{D}_\mathcal{G}$.
2. Add an edge (X_i, X_j) to $\mathcal{H}$ if the corresponding edge $(w_p, w_q) \in \mathcal{D}_\mathcal{G}$. $L(X_i, X_j) := dep(w_p, w_q)$.
3. Add an edge between a pair of nodes (X_i, X_j) if the corresponding words are synonyms. $L(X_i, X_j) := synonymy$.
4. Create a node for each element and category specified in the puzzle and add an edge from them to others if the corresponding string descriptions are 'same'. In this case, the edges are labeled as $exact\_match$.
5. If $(X_i, X_j) \in \mathcal{H}$ and $L(X_i, X_j) = exact\_match$ and both of them are referring to a verb, then add more edges (X_i', X_j') to $\mathcal{H}$ with label $spatial\_symmetry$, where $L(X_i, X_i') = L(X_j, X_j')$.

Determining Node potentials For each element in the m category, a set of naive regular-expression based taggers are used to detect it's type (For example, "am-pm time"). Each element type maps to a WordNet (Miller, 1995) representative (For example, "time_unit#n"). For each constituent w a similarity score, $sim(w,c)$, is calculated to each class $c \in \{1...2m+1\}$, in the following way:

•**Class c is denoting instance of some category C_i** Similarity scores are computed between the textual description of the constituent to both the WordNet representative of E_i and the textual description d_i using the HSO WordNet similarity algorithm (Hirst and St-Onge, 1998). The similarity score, $sim(w,c)$, is chosen to be the maximum of them.

•**Class c is denoting a category C_i :** $sim(w,c)$ is assigned the value of HSO Similarity between the textual description and d_i.

•**Class c is $null$:** In this case similarity is calculated using the following formula:

$$sim(w, null) = MAX_{HSO} - \max_{c \neq null} sim(w, c)$$

where MAX_{HSO} denotes the maximum similarity score returned by HSO algorithm, which is 16.

Node potential for each node $X_i \in \mathcal{H}$, corresponding to the constituent w_j, are then calculated by,

$$\phi(X_i = c) = 1 + sim(w_j, c), \forall c$$

Determining Edge potentials For each edge in the graph H, the edge potential, $\phi(X_i, X_j)$ is calculated using the following formula,

$$\phi(X_i = c_1, X_j = c_2) =$$
$$1 + \begin{cases} P(X_i = X_j | L(X_i, X_j)), & \text{if } c_1 = c_2 \\ P(X_i \neq X_j | L(X_i, X_j)), & \text{otherwise} \end{cases}$$

In the training phase, each entity in a clue is tagged with its respective class. The probability values are then calculated from the training dataset using simple count.

3.2 Learning To Extract Relations

The goal here is to identify all the relations $R(arg_1, ..., arg_p)$ that are present in a clue, where each relation belongs to the logical vocabulary described in Table 1 . This problem is known as *Complex relation extraction* (McDonald et al., 2005; Bach and Badaskar, 2007; Fundel et al., 2007; Zhou et al., 2014). The common approach for solving the *Complex relation extraction* problem is to first find the relation between each pair of entities and then discover the complex relations from binary ones using the definition of each relation.

Figure 1 depicts the scenario. The goal is to identify the relation $possDiff(E1, E2, E3)$, where $E1, E2, E3$ are constituents having a non-$null$ class value. However instead of identifying $posDiff(E1, E2, E3)$ directly, first the relation

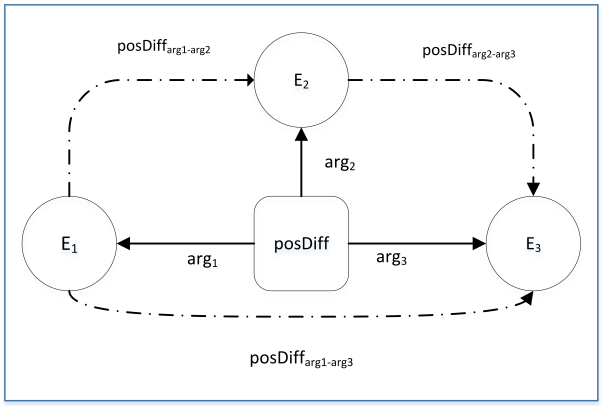


Figure 1: Binary representation of the relation $possDiff$

between each pair of entities will be identified. If the relations $\{posDiff_{arg_1-arg_2}(E_1, E_2)$, $posDiff_{arg_2-arg_3}(E_2, E_3)$, $posDiff_{arg_1-arg_3}$ (E_1,E_3) $\}$ are identified, the extraction module will infer that $posDiff(E1, E2, E3)$ holds. In a similar manner, a set of total 39 binary relations are created for all the relations described in Table 1.

In the training phase, all the relations and their respective arguments in each clue are given. Using this supervision, we have built a Maximum Entropy based model (Berger et al., 1996; Della Pietra et al., 1997) to classify the relation between a pair of entities present in a clue. Maximum entropy classifier has been successfully applied in many natural language processing applications (Charniak, 2000; Chieu and Ng, 2002; Ratnaparkhi and others, 1996) and allows the inclusion of various sources of information without necessarily assuming any independence between the features. In this model, the conditional probability distribution is given by:

$$P(c|d) = \frac{\prod_{j=1...K} e^{\lambda_i f_i(d,c)}}{\sum_{c' \in C} \prod_{j=1...K} e^{\lambda_i f_i(d,c')}} \quad (1)$$

where the denominator is the normalization term and the parameter λ_i correspond to the weight for the feature f_i. Features in Maximum Entropy model are functions from context and classes to the set of real numbers. A detailed description of the model or parameter estimation method used - Generalized Iterative Scaling, can be found at (Darroch and Ratcliff, 1972).

Table 2 describes the features that are used in the classification task. Here, $path(E_1, E_2)$ denotes all the words that occur in the path(s) connecting E_1 and $E2$ in the dependency graph of the clue.

Feature Set
Class of E_1 and E_2
All the grammatical relations between the words in $path(E_1, E_2)$
All the adjectives and adverbs in $path(E_1, E_2)$.
POS tags of all the words in $path(E_1, E_2)$
TypeMatched = [[class of E_1 = class of E_2]]
IsE_1Numeric = [[class of E_1 is Numeric]]
IsE_2Numeric = [[class of E_2 is Numeric]]
All the words that appears in the following grammatical relations $advmod$, $amod$, cop, det with the words in $path(E_1, E_2)$.
hasNegativeWord = [[$\exists$w $\in$ $path(E_1, E_2)$ s.t. w has a neg relation starting with it.]]

Table 2: Features used in the classification task

The relation between each pair of entities in a clue is the one which maximizes the conditional probability in equation (1).

3.2.1 Missing Entity

In the case of comparative relations in Table 1, such as $greaterThan$, the basis of the comparison can be hidden. For example, in clue 1 of the example problem, the two entities, "Hannah" and "client" have been compared on the basis of "price", however there is no constituent in the clue which refers to an element from that category. The basis of comparison is hidden in this case and is implied by the word "paid". In the current implementation, the translation module does not handle this case. For puzzles that contain only one category consisting of numeric elements, the translation module goes with the obvious choice. This is part of our future work.

4 Related Work

There has been a significant amount of work on the representation of puzzle problems in a formal language (Gelfond and Kahl, 2014; Baral, 2003; Celik et al., 2009). However, there has not been any work that can automatically solve a logic grid puzzle. The latest work (Baral and Dzifcak, 2012) on this problem, assumes that the entities in a clue are given and the authors manually simplify the sentences for translation. Furthermore their representation of logic grid puzzles does not consider the category of a variable in the formal representation

i.e. uses $element/1$ and $tuple/2$ predicates and thus cannot solve puzzles containing more than one numeric categories.

In the same work (Baral and Dzifcak, 2012), the authors propose to use a semantic parser to do the translation. This method works well for simple sentences such as "Donna dale does not have green fleece" however it faces several challenges while dealing with real world puzzle sentences. The difficulty arises due to the restrictions enforced in the translation models used by the existing semantic parsers. Traditional semantic parsers (Vo et al., 2015; Zettlemoyer and Collins, 2005) assign meanings to each word in a dictionary and combine the meaning of the words to characterize the complete sentence. A phrase structure grammar formalism such as Combinatory Categorial Grammar (Steedman and Baldridge, 2011; Vo et al., 2015; Zettlemoyer and Collins, 2005), Context Free Grammar (Aho and Ullman, 1972; Wong and Mooney, 2006), is normally used to obtain the way words combine with each other. In the training phase, the semantic parser learns the meanings of words given a corpus of <sentence, meaning> pairs and stores them in a dictionary. During translation, the semantic parser uses those learned meanings to obtain the meaning of the sentence. Firstly, for the puzzle problems the meaning of the words changes drastically depending on the puzzle. A word may be an entity in one puzzle, but, in a different problem it might not be an entity or might belong to a different category altogether. Thus a learned dictionary may not be useful while translating clues in a new puzzle. Secondly, in puzzles relations are normally expressed by phrases. For example, in the clue "The person who played at Eden Gardens played for India", the phrases "played at" and "played for" are used to express two different relations. Thus, using a model that assigns meaning to each word may not be suitable here. Finally, it is difficult to identify the participants of a relation with a parse tree generated following a phrase structure grammar. For example, consider the parse tree of the clue "The person who trekked for 8 miles started at Bull Creek". Even though, the relation "started at" takes the word 'person' and 'Bull Creek' as its input, it receives the entire phrase "the person who trekked for 8 miles" as its argument along with the other input 'Bull Creek'.

The entity classification problem studied in this research shares many similarity with Named Entity Recognition (Nadeau and Sekine, 2007; Zhou and Su, 2002) and the Word Sense disambiguation (Stevenson and Wilks, 2003; Sanderson, 1994) task. However, our work has a major difference; in the entity classification problem, the class of an entity varies with the problem and does not belong to a known closed set, whereas for the other two problems the possible classes are pre-specified.

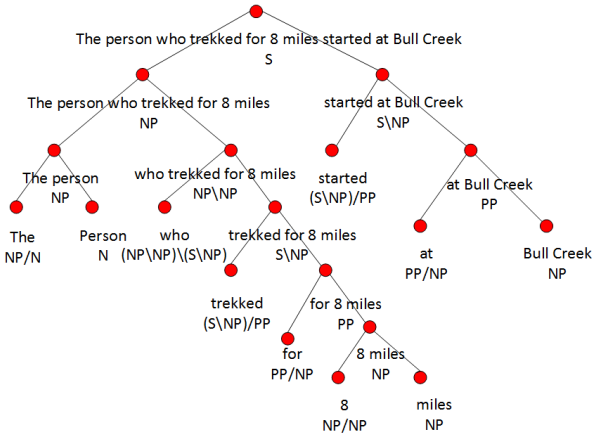


Figure 2: Parse tree of an example sentence in Combinatory categorial grammar

5 Experimental Evaluation

Dataset To evaluate our method we have built a dataset of logic grid puzzles along with their correct solutions. A total of 150 problems are collected from `logic-puzzles.org`. Out of them 100 problems are fully annotated with the entities and the relations information. The remaining 50 puzzles do not have any annotation except their solution. The set of annotated puzzles contain a total of 467 clues, 5687 words, 1681 entities and 862 relations. The set of 50 puzzles contain a total of 222 clues with 2604 words.

Tasks We evaluate LOGICIA on three tasks: 1) puzzle solving; 2) entity classification; and 3) relation extraction. We use the percentage of correct answers as the evaluation metric for all the three tasks. In case of a logic grid puzzle solving, an answer is considered correct if it exactly matches the solution of that puzzle.

Training-Testing Out of the 100 annotated puzzle problems 50 are used as training samples and remaining 50 puzzles are used in testing. The set of 50 unannotated puzzles are used solely for the task of testing puzzle solving.

	Entity classification	Binary relation classification with annotation		Relation extraction with annotation		Solution
		Yes	No	Yes	No	
Total	1766	960		450		50
Correct	1502	922	854	410	365	37
Percentage	85.05%	96.04%	88.95%	90.90%	81.11%	74%

Table 3: Accuracy on 50 annotated puzzle problems in the Test set.

Results Table 3 & 4 shows the efficacy of our approach in solving logic grid puzzles with the selected set of relations. LOGICIA is able to classify the constituents with 85.05% accuracy and is able to solve 71 problems out of the 100 test puzzles. It should be noted that puzzle problems requires precise understanding of the text and to obtain the correct solution of a puzzle problem all the entities and their relations in the puzzle need to be identified. Columns 2 and 3 in Table 3 compares the performance on relation extraction when it is used in conjunction with the entity classification and when it directly uses the annotated entity.

Puzzle Solving		
Total	Correct	Percentage
50	34	68%

Table 4: Accuracy on unannotated 50 test puzzle problems.

Error Analysis The errors in entity classification falls into two major categories. In the first category, more knowledge of similarity is needed than what is currently obtained from the WordNet. Consider for example, the categories are "class number" and "class size" and the constituent is "20 students". Even though the constituent is closer to "class size", standard WordNet based similarity methods are unable to provide such information. In the second category, the WordNet similarity of the constituent to one of the classes is quite high due to their position in the WordNet hierarchy; however with respect to the particular problem the constituent is not an entity. The relation extraction task performs fairly well, however the binary relation classification task does not jointly consider the relation between all the entities and because of that if one of the necessary binary relation of a complex relation is misclassified, the extraction of the entire relation gets affected.

6 Conclusion & Future Work

This paper presents a novel approach for solving logic grid puzzle. To the best of our knowledge, this is a novel work with respect to the fact that that it can automatically solve a given logic grid puzzle.

There are several advantages of our approach. The inclusion of knowledge in terms of a vocabulary of relations makes it scalable. For puzzles which make use of a different set of constraints, such as "Lynda *sat on an even numbered position*", can be easily integrated into the vocabulary and the system can then be trained to identify those relations for new puzzles. Also, the proposed approach separates the representation from reasoning. The translation module only identifies the relation and their arguments; it is not aware of the meaning of those relations. The reasoning module, on the other hand, knows the definition of each relation and subsequently prunes those possibilities when relations appearing in a clue does not hold. This separation of representation from reasoning allows the system to deal with the complex relations that appear in a clue.

There are a few practical and theoretical issues which need to be addressed. One of those is updating the logical vocabulary in a scalable manner. Logic grid puzzle is a wide family of puzzles and it will require more knowledge of relations than what is currently available. Another challenge that needs to be addressed is the computation of similarity between complex concepts such as "size of class" and "20 students". Also, the case of "missing entity" (3.2) needs to be modeled properly. This work is the first step towards further understanding these important issues.

Acknowledgements

We thank NSF for the DataNet Federation Consortium grant OCI-0940841 and ONR for their grant N00014-13-1-0334 for partially supporting this research.

References

Alfred V Aho and Jeffrey D Ullman. 1972. *The theory of parsing, translation, and compiling*. Prentice-Hall, Inc.

Nguyen Bach and Sameer Badaskar. 2007. A review of relation extraction. *Literature review for Language and Statistics II*.

Chitta Baral and Juraj Dzifcak. 2012. Solving puzzles described in english by automated translation to answer set programming and learning how to do that translation. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Thirteenth International Conference, KR 2012, Rome, Italy, June 10-14, 2012*.

Chitta Baral. 2003. *Knowledge representation, reasoning and declarative problem solving*. Cambridge university press.

Jonathan Berant, Vivek Srikumar, Pei-Chun Chen, Brad Huang, Christopher D Manning, Abby Vander Linden, Brittany Harding, and Peter Clark. 2014. Modeling biological processes for reading comprehension. In *Proc. EMNLP*.

Adam L Berger, Vincent J Della Pietra, and Stephen A Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational linguistics*, 22(1):39–71.

Mehmet Celik, Halit Erdogan, Fırat Tahaoglu, Tansel Uras, and Esra Erdem. 2009. Comparing asp and cp on four grid puzzles. In *Proceedings of the Sixteenth RCRA International Workshop on Experimental Evaluation of Algorithms for Solving Problems with Combinatorial Explosion (RCRA09). CEUR Workshop Proceedings*.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*, pages 132–139. Association for Computational Linguistics.

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

Hai Leong Chieu and Hwee Tou Ng. 2002. Named entity recognition: a maximum entropy approach using global information. In *Proceedings of the 19th international conference on Computational linguistics-Volume 1*, pages 1–7. Association for Computational Linguistics.

John N Darroch and Douglas Ratcliff. 1972. Generalized iterative scaling for log-linear models. *The annals of mathematical statistics*, pages 1470–1480.

Stephen Della Pietra, Vincent Della Pietra, and John Lafferty. 1997. Inducing features of random fields. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 19(4):380–393.

Katrin Fundel, Robert Küffner, and Ralf Zimmer. 2007. Relexrelation extraction using dependency parse trees. *Bioinformatics*, 23(3):365–371.

Michael Gelfond and Yulia Kahl. 2014. *Knowledge representation, reasoning, and the design of intelligent agents: The answer-set programming approach*. Cambridge University Press.

Michael Gelfond and Vladimir Lifschitz. 1991. Classical negation in logic programs and disjunctive databases. *New generation computing*, 9(3-4):365–385.

Graeme Hirst and David St-Onge. 1998. Lexical chains as representations of context for the detection and correction of malapropisms. *WordNet: An electronic lexical database*, 305:305–332.

Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. 2014. Learning to solve arithmetic word problems with verb categorization. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Jin-Dong Kim, Sampo Pyysalo, Tomoko Ohta, Robert Bossy, Ngan Nguyen, and Jun'ichi Tsujii. 2011. Overview of bionlp shared task 2011. In *Proceedings of the BioNLP Shared Task 2011 Workshop*, pages 1–6. Association for Computational Linguistics.

Ross Kindermann, James Laurie Snell, et al. 1980. *Markov random fields and their applications*, volume 1. American Mathematical Society Providence, RI.

Daphne Koller and Nir Friedman. 2009. *Probabilistic graphical models: principles and techniques*. MIT press.

Nate Kushman, Yoav Artzi, Luke Zettlemoyer, and Regina Barzilay. 2014. Learning to automatically solve algebra word problems. *ACL (1)*, pages 271–281.

Vladimir Lifschitz. 1999. Answer set planning. In *Logic Programming and Nonmonotonic Reasoning*, pages 373–374. Springer.

Ryan McDonald, Fernando Pereira, Seth Kulick, Scott Winters, Yang Jin, and Pete White. 2005. Simple algorithms for complex relation extraction with applications to biomedical ie. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 491–498. Association for Computational Linguistics.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

David Nadeau and Satoshi Sekine. 2007. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26.

Benedek Nagy and Gerard Allwein. 2004. Diagrams and non-monotonicity in puzzles. In *Diagrammatic Representation and Inference*, pages 82–96. Springer.

Adwait Ratnaparkhi et al. 1996. A maximum entropy model for part-of-speech tagging. In *Proceedings of the conference on empirical methods in natural language processing*, volume 1, pages 133–142. Philadelphia, PA.

Mark Sanderson. 1994. Word sense disambiguation and information retrieval. In *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 142–151. Springer-Verlag New York, Inc.

Mark Steedman and Jason Baldridge. 2011. Combinatory categorial grammar. *Non-Transformational Syntax: Formal and Explicit Models of Grammar. Wiley-Blackwell*.

Mark Stevenson and Yorick Wilks. 2003. Word sense disambiguation. *The Oxford Handbook of Comp. Linguistics*, pages 249–265.

Nguyen Ha Vo, Arindam Mitra, and Chitta Baral. 2015. The NL2KR platform for building natural language translation systems. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 899–908.

Yuk Wah Wong and Raymond J Mooney. 2006. Learning for semantic parsing with statistical machine translation. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 439–446. Association for Computational Linguistics.

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to Map Sentences to Logical Form: Structured Classification with Probabilistic Categorial Grammars. In *UAI*, pages 658–666. AUAI Press.

Yongyue Zhang, Michael Brady, and Stephen Smith. 2001. Segmentation of brain mr images through a hidden markov random field model and the expectation-maximization algorithm. *Medical Imaging, IEEE Transactions on*, 20(1):45–57.

GuoDong Zhou and Jian Su. 2002. Named entity recognition using an hmm-based chunk tagger. In *proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 473–480. Association for Computational Linguistics.

Deyu Zhou, Dayou Zhong, and Yulan He. 2014. Biomedical relation extraction: From binary to complex. *Computational and mathematical methods in medicine*, 2014.

Improving fast_align by Reordering

Chenchen Ding, Masao Utiyama, Eiichiro Sumita
Multilingual Translation Laboratory
National Institute of Information and Communications Technology
3-5 Hikaridai, Seikacho, Sorakugun, Kyoto, 619-0289, Japan
{chenchen.ding, mutiyama, eiichiro.sumita}@nict.go.jp

Abstract

fast_align is a simple, fast, and efficient approach for word alignment based on the IBM model 2. fast_align performs well for language pairs with relatively similar word orders; however, it does not perform well for language pairs with drastically different word orders. We propose a *segmenting-reversing reordering* process to solve this problem by alternately applying fast_align and reordering source sentences during training. Experimental results with Japanese-English translation demonstrate that the proposed approach improves the performance of fast_align significantly without the loss of efficiency. Experiments using other languages are also reported.

1 Introduction

Aligning words in a parallel corpus is a basic task for almost all state-of-the-art statistical machine translation (SMT) systems. Word alignment is used to extract translation rules in various way, such as the phrase pairs used in a phrase-based (PB) SMT system (Koehn et al., 2003), the hierarchical rules used in a HIERO system (Chiang, 2007), and the sophisticated translation templates used in tree-based SMT systems (Liu et al., 2006).

Among different approaches, GIZA++[1] (Och and Ney, 2003), which is based on the IBM translation models, is the most widely used word alignment tool. Other well-known tools are the BerkeleyAligner[2], Nile[3] (Riesa et al., 2011), and pialign[4] (Neubig et al., 2011).

fast_align[5] (Dyer et al., 2013) is a recently proposed word alignment approach based on the reparameterization of the IBM model 2, which is usually referred to as a *zero-order* alignment model (Och and Ney, 2003). Taking advantage of the simplicity of the IBM model 2, fast_align introduces a "*tension*" parameter to model the overall accordance of word orders and an efficient parameter re-estimation algorithm is devised. It has been reported that the fast_align approach is more than 10 times faster than baseline GIZA++, with comparable results in end-to-end French-, Chinese-, and Arabic-to-English translation experiments.

However, the simplicity of the IBM model 2 also leads to a limitation. As demonstrated in this study, fast_align does not perform well when applied to language pairs with drastically different word orders, e.g., Japanese and English. The problem is because of the IBM model 2's intrinsic inability to handle complex distortions. In this study, we propose a simple and efficient reordering approach to improve the fast_align's performance in such situations, referred to as *segmenting-reversing* (seg_rev). Our motivation is to apply a rough but robust reordering to make the source and target sentences have more similar word orders, where fast_align can show its power. Specifically, seg_rev first segments a source-target sentence pair into a sequence of *minimal monotone* chunk pairs[6] based on the automatically generated word alignment. Within the chunk pairs, source word sequences are examined to determine whether they should be *completely reversed* or the original order should be retained. The objective of this step is to convert the source sentence to a roughly target-like word order. The seg_rev process is applied recursively but not deeply (only twice in our ex-

[1] http://www.statmt.org/moses/giza/GIZA++.html
[2] https://code.google.com/p/berkeleyaligner/
[3] http://jasonriesa.github.io/nile/
[4] http://www.phontron.com/pialign/
[5] https://github.com/clab/fast_align
[6] same as the "*tuple*" used in Mariño et al. (2006)

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1034–1039,
Lisbon, Portugal, 17-21 September 2015.

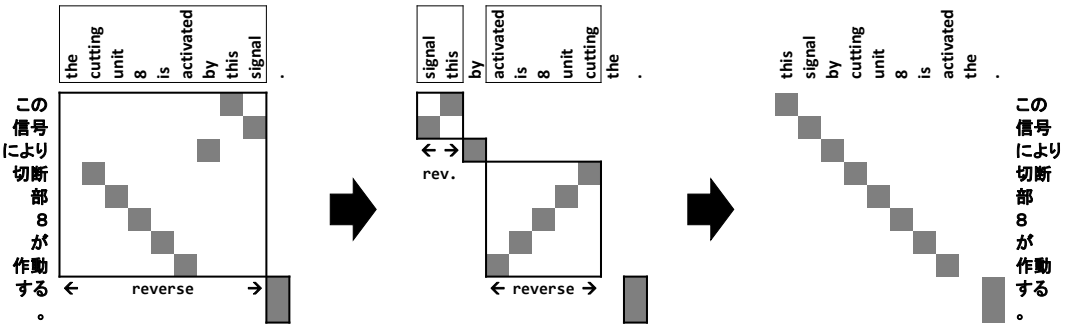


Figure 1: Example of `seg_rev` applied to a word-aligned English-Japanese sentence pair. Based on the word alignment, the source sentence is reordered in a target-like order after applying `seg_rev` **twice**.

periments) for each source sentence in the training data. Consequently, the `seg_rev` process is lightweight and shallow. Local word sequences, except those at chunk boundaries, are not scrambled, while global word orders are re-arranged if there are *large* chunks.

Our primary experimental results for Japanese-English translation show that applying `seg_rev` significantly improves `fast_align`'s performance to a level comparable to `GIZA++`. The training time becomes 2–4 times that of a baseline `fast_align`, which is still at least $2-4$ times faster than the training time required by baseline `GIZA++`. Results for German-, French-, and Chinese-English translations are also reported.

2 Segmenting-Reversing Reordering

The `seg_rev` is inspired by the "*REV preorder*" (Katz-Brown and Collins, 2008), which is a simple pre-reordering approach originally designed for the Japanese-to-English translation task. More efficient pre-reordering approaches usually require trained parsers and sophisticated machine learning frameworks (de Gispert et al., 2015; Hoshino et al., 2015). We adopt the `REV` method in Katz-Brown and Collins (2008) considering it is the simplest and lightest pre-reordering approach (to our knowledge), which may bring a minimal effect on the efficiency of `fast_align`.

An example `seg_rev` process, where the word alignment is generated by `fast_align`, is illustrated in Fig. 1. The example we selected has relatively correct word alignment and `seg_rev` performs well. In general cases, the alignment has significant noise and the reordering is rougher .

Algorithm 1 describes the repeated (δ times) application of the `seg_rev` process, and Algorithm 2 describes a single application. Specifically, Algorithm 1 applies Algorithm 2 δ times. For each application of Algorithm 2, source sentence S and source indices in alignment A are reordered, and the overall permutation R_I is updated and recorded. In Fig. 1, the original English sentence had 10 words (including the period), being indexed as $[0,1,2,3,4,5,6,7,8,9]$. After the first application of `seg_rev`, R_I was $[\boxed{8,7,6,5,4,3,2,1,0},9]$, and after the second application, R_I was $[\boxed{7,8},6,\boxed{1,2,3,4,5},0,9]$ (reversed parts are boxed).

In Algorithm 2, the main **for** loop (line 3) scans the source sentence from the beginning to the end to obtain monotone segmentation. The **foreach** (line 5) and **if** (line 11) are general phrase pair extraction process. The **if** (line 13) guarantees that the chunk is monotone on the target side. The `rev` function (line 16), which is described in Algorithm 3, determines whether the sub-sequence from s_{start} to s_{end} should be reversed by examining the related alignment A_{sub}. For example, in the first application shown in Fig.1, two sub-sequences $[0:8]$ and $[9:9]$ are processed by `rev` and $[0:8]$ is reversed. Four sub-sequences $[0:1]$, $[2:2]$, $[3:7]$, and $[9:9]$ are processed in the second application and $[0:1]$ and $[3:7]$ are reversed.[7] Finally, source sentence S and source indices in alignment A are reordered (lines 19 – 20).[8]

Algorithm 3 performs the reversal. We count the concordant and discordant pairs[9] and reverse

[7] The sub-sequences are based on the input, not the original sentence, e.g., sub-sequence $[0:1]$ contains the 8th and 7th word of the original source sentence in the 2nd application.

[8] Unaligned words between chunks on the source side are problematic. They are not touched by line 18. Although they can be attached to preceding or succeeding chunks, we do not use further heuristics to handle them. An example is the drifting "*the*" in the English sentence in Fig. 1, which our approach cannot handle properly.

[9] As used in Kendall's τ or Goodman and Kruskal's γ.

Algorithm 1: `seg_rev`$^{\delta}$

input : *same as Algorithm 2 and a depth δ;*
output : *same as Algorithm 2 except R_A;*

$R_I \leftarrow [0, \cdots, I]$; $R_S \leftarrow S$; $R_A \leftarrow A$;
for $i \leftarrow 1$ **to** δ **do**
  $R_S, R_A, R_{I'} \leftarrow$ `seg_rev` (R_S, T, R_A);
  permute R_I with $R_{I'}$;
return R_S, R_I;

Algorithm 2: `seg_rev`

input : source sentence $S = s_0, \cdots, s_I$;
target sentence $T = t_0 \cdots, t_J$;
word alignment $A = \{(i,j)|$
s_i, t_j are aligned, $i \in [0, I], j \in [0, J]\}$;
output : reordered source sentence R_S;
source index reordered alignment R_A;
reordered indices $R_I = \pi(0, \cdots, I)$,
which is a permutation from 0 to I;

1 $R_I \leftarrow [0, \cdots, I]$;
2 $s_{start} \leftarrow 0$; $t_{pre\_end} \leftarrow -1$;
3 **for** $s_{end} \leftarrow s_{start}$ **to** I **do**
4 $t_{start} \leftarrow J+1$; $t_{end} \leftarrow -1$;
5 **foreach** $(i,j) \in A$ **do**
6 **if** $i \in [s_{start}, s_{end}]$ **then**
7 **if** $j < t_{start}$ **then**
8 $t_{start} \leftarrow j$;
9 **if** $j > t_{end}$ **then**
10 $t_{end} \leftarrow j$;
11 **if** $\exists (i,j) \in A :$
$j \in [t_{start}, t_{end}] \wedge i \notin [s_{start}, s_{end}]$ **then**
12 **continue**;
13 **if** $\exists (i,j) \in A : j \in (t_{pre\_end}, t_{start})$ **then**
14 **continue**;
15 $A_{sub} \leftarrow \{(i,j)|(i,j) \in A,$
$i \in [s_{start}, s_{end}] \wedge j \in [t_{start}, t_{end}]\}$;
16 `rev` $(R_I[s_{start} : s_{end}], A_{sub})$;
17 $t_{pre\_end} \leftarrow t_{end}$;
18 $s_{start} \leftarrow min(\{i | i > s_{end} \wedge \exists (i,j) \in A\})$;
19 $R_S \leftarrow$ permute S according to R_I;
20 $R_A \leftarrow$ permute i in A according to R_I;
21 **return** R_S, R_A, R_I;

Algorithm 3: `rev`

input : index sequence I_{sub}; word alignment A_{sub};

$con \leftarrow 0$; $dis \leftarrow 0$;
foreach unordered tuple $((i_0, j_0), (i_1, j_1)) :$
$(i_0, j_0) \in A_{sub} \wedge (i_1, j_1) \in A_{sub}$ **do**
  $x \leftarrow (i_1 - i_0) \times (j_1 - j_0)$;
  if $x > 0$ **then**
  $con \leftarrow con + 1$;
  else if $x < 0$ **then**
  $dis \leftarrow dis + 1$;
  else
  continue;
if $dis > con$ **then**
  $I_{sub} \leftarrow$ reverse I_{sub};

Algorithm 4: `fast_align` with `seg_rev`

input : parallel corpus $\mathcal{C}$ with N sentence pairs
$\mathcal{C} = \{(S^1, T^1) \cdots, (S^N, T^N)\}$;
maximum iteration M; depth δ for `seg_rev`;
output : word alignment $\mathcal{A} = \{A^1, \cdots, A^N\}$;

$\mathcal{A} \leftarrow \varnothing$;
for $iter \leftarrow 1$ **to** M **do**
  $\mathcal{I} \leftarrow \varnothing$; $\mathcal{C}' \leftarrow \varnothing$;
  if $\mathcal{A} \neq \varnothing$ **then**
  for $n \leftarrow 1$ **to** N **do**
  $R_S, R_I \leftarrow$ `seg_rev`$^{\delta}(S^n, T^n, A^n)$;
  append (R_S, T^n) to $\mathcal{C}'$;
  append R_I to $\mathcal{I}$;
  else
  $\mathcal{C}' \leftarrow \mathcal{C}$;
  $\mathcal{A} \leftarrow$ `sym` $\circ$ `fast_align` $(\mathcal{C}')$;
  if $\mathcal{I} \neq \varnothing$ **then**
  for $n \leftarrow 1$ **to** N **do**
  recover A^n by $\mathcal{I}[n]$;
return $\mathcal{A}$;

the sub-sequence if and only if there are more discordant pairs than the concordant pairs. In Fig.1, the sub-sequence $[0:8]$ in the first application has $C_8^2 = 28$ pairs of aligned word pair (i.e., 28 gray block pairs for eight gray blocks); however, only 11 pairs are concordant ($C_5^2 = 10$ pairs in $[1:5]$ and one pair in $[7:8]$), Consequently, the sub-sequence $[0:8]$ is reversed because there are more discordant pairs ($17 = 28 - 11$). The two reversed subsequences in the second application are obvious.

Algorithm 4 describes the training framework, where `fast_align` and `seg_rev` are applied alternately. To generate word alignment, `fast_align` is run bi-directionally and symmetrization heuristics are applied to reduce noise (line 11). In each iteration, the source sentences for `seg_rev` are the original sentences, and `fast_align` uses the reordered sentences with the exception of the first iteration. The word alignment generated is thus based on the reordered source sentences; consequently, the recorded permutation (line 14) is used to recover word alignment before the next iteration. The permutation is a one-to-one mapping; therefore, recovering is realized by the inverse mapping of the permutation, which transfers the source-side word alignment indices to match the original source sentences.

The time complexity of Algorithm 3 is $O(l^2)$, where l is the size of A_{sub} that is related to the chunk size. If the average chunk size is a constant C depending on languages pairs or data sets, then the time complexity of Algorithm 2 is $O(C \cdot I^2)$ assuming J and the size of A are both linear against I. The average chunk size will be reduced when `seg_rev` is applied successively; therefore, the time required for subsequent `seg_rev` processes will decrease. In practice, compared with the training time required by `fast_align`,

seg_rev processing time is negligible. Note that seg_rev processes are accelerated easily by parallel processing.

3 Experiments and Discussion

We applied the proposed approach to Japanese-English translation, a language pair with dramatically different word orders. In addition, we applied the approach to German-English translation, a language pair with relatively different word orders among European languages.

For Japanese-English translation, we used the **NTCIR-7 PAT-MT** data (Fujii et al., 2008). For German-English translation, we used the **Europarl v7** corpus[10] (Koehn, 2005) for training, the **WMT 08**[11] **/ WMT 09**[12] test sets for development / testing, respectively. Default settings for the PB SMT in MOSES[13] (Koehn et al., 2007) were used, except for Japanese-English translations where the *distortion-limit* was set to 12 to reach a recently reported baseline (Isozaki et al., 2012). MERT (Och, 2003) was used to tune development set parameter weights and BLEU (Papineni et al., 2002) was used on test sets to evaluate the translation performance. *Bootstrap sampling* (Koehn, 2004) was employed to test statistical significance using bleu_kit[14].

We compared GIZA++ and fast_align with default settings. GIZA++ was used as a module of MOSES. The bi-directional outputs of fast_align were symmetrized by atools in cdec[15] (Dyer et al., 2010), and further training steps were conducted using MOSES. *grow-diag-final-and* symmetrization was used consistently in the experiments. For the the proposed approach, we set $\delta = 2$ and $M = 4$ in Algorithm 4. Note that δ can be set to a larger value and seg_rev could be applied repeatedly until no additional reordering is possible. As mentioned, the word alignment is noisy and our intention is a robust and rough process; therefore, we restricted seg_rev to two applications and did not consider the difference in sentence lengths or different languages during training. Within each iteration, fast_align was run with default settings, except *initial diagonal-tension* (λ^{ini}) was set to 0.1 in the first iteration, to avoid overly strong monotone preference at the beginning of training.

	ja-en	**en-ja**	**de-en**	**en-de**
GIZA++	**28.8**	**30.8**	**18.2**	**12.9**
$\text{FA}_{\lambda^{ini}=4.0}$	$28.1^{\ddagger}$	$29.5^{\ddagger}$	$18.0^{\dagger}$	$12.7^{\dagger}$
$\text{FA}_{\lambda^{ini}=0.1}$	$28.0^{\ddagger}$	$29.8^{\ddagger}$	$17.5^{\ddagger}$	$12.5^{\ddagger}$
iteration 2	$28.3^{\dagger}$	**30.9**	$17.9^{\ddagger}$	**12.8**
iteration 3	$28.4^{\dagger}$	$30.1^{\ddagger}$	**18.1**	$12.7^{\dagger}$
iteration 4	**28.8**	**30.7**	**18.1**	$12.7^{\dagger}$

Table 1: Test set BLEU scores for Japanese-English and German-English translations. (‡, statistical significance at $p < 0.01$; †, at $p < 0.05$; boldface, no significance; all compared with GIZA++)

Experimental results for Japanese-English and German-English translations in both directions are listed in Table 1. The first two rows show the baseline performance. fast_align (using a default $\lambda^{ini} = 4.0$) performance was statistically significantly lower than GIZA++, particularly for Japanese-English translation. The following four rows show the results of the proposed approach. For the first iteration, λ^{ini} was set to 0.1, and the performance did not change significantly. The translations **from** English improved (equal to GIZA++) at the second iteration. However, translations **to** English improved more slowly. We attribute the difference in improvement rates between translation to and from English to the relatively fixed word order of English, whereby the reordering process is easier and more consistent. Note that once translations **from** English improved in the second iteration, performance decreased in the following iterations. The results in Table 1 were obtained using *predictable-seed* for tuning, which generated determinate results. Another attempt using random seeds to tune returned test set BLEU scores of **30.5**, **30.4** on **en-ja** and **12.8**, **12.8** on **en-de**, for iterations 3 and 4, respectively. These four scores had no statistical significance against GIZA++. The instability is largely due to the alignment of function words, which affects translation performance (Riesa et al., 2011). The alignment does not change significantly after the second iteration; however, it is unstable around function words,[16] because seg_rev does not process

[10] http://www.statmt.org/europarl/
[11] http://www.statmt.org/wmt08/
[12] http://www.statmt.org/wmt09/
[13] http://www.statmt.org/moses/
[14] http://www.nlp.mibel.cs.tsukuba.ac.jp/bleu_kit/
[15] http://www.cdec-decoder.org/

[16] Specifically, *to*, articles, and prepositions in English-Japanese; *of*, *have*, and relative pronouns in English-German.

	fr-en	zh-en
GIZA++	23.3	31.4
$\text{FA}_{\lambda^{ini}=4.0}$	23.1	31.7
iteration 2	23.1	31.7

Table 2: Test set BLEU scores for French-to-English and Chinese-to-English translations. For **fr-en**, the data sets were the same as for **de-en**. For **zh-en**, NIST 2006 OpenMT data were used for training and test; test data from 2002 to 2005 OpenMT were used for tuning.

unaligned function words between chunks. Our approach is too rough to handle function words precisely. We plan to address this in future.

We also tested our approach on French- and Chinese-to-English translations. The results are listed in Table 2. GIZA++ and fast_align showed no statistically significant difference in performance, which is consistent with Dyer et al. (2013). The proposed approach did not affect performance for French- and Chinese-to-English translations. These results are expected as these language pairs have similar word orders.

With regard to processing time, a naïve, single-thread implementation of seg_rev in C++ took approximately $60s$ / $40s$ in the first / second application on the entire Japanese-English corpus[17]. The recover process took less than $30s$ in each iteration. In contrast, fast_align, although very fast, took approximately one hour for one round of training (using five iterations for its log-linear model) on the same corpus. Therefore, the additional time required in our approach is quite small and can be ignored compared with the training time of fast_align.[18]

4 Conclusion and Future Work

We have proposed a simple and efficient approach to improve the performance of fast_align on language pairs with drastically different word orders. With the proposed approach, fast_align obtained results comparable with GIZA++, and its efficiency is retained. We are investigating further properties of seg_rev and plan to extend it to achieve greater stability and efficiency.[19]

[17] 1.8 *M* sentence pairs with an average length of 35 words.

[18] GIZA++ actually took around 18 hours to align the Japanese-English corpus (parallel processes for two directions, including mkcls, five iterations for the model 1 and the HMM model, three iteration for the model 3 and 4).

[19] Ongoing experiments using seg_rev with GIZA++ returned negative results. BLEU decreases by approx. 1 point.

Acknowledgments

We thank Dr. Atsushi Fujita for his helpful discussions on this work.

References

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.

Adrià de Gispert, Gonzalo Iglesias, and Bill Byrne. 2015. Fast and accurate preordering for SMT using neural networks. In *Proc. of NAACL-HLT*, pages 1012–1017.

Chris Dyer, Adam Lopez, Juri Ganitkevitch, Johnathan Weese, Ferhan Ture, Phil Blunsom, Hendra Setiawan, Vladimir Eidelman, and Philip Resnik. 2010. cdec: A decoder, alignment, and learning framework for finite-state and context-free translation models. In *Proc. of ACL (System Demonstrations)*, pages 7–12.

Chris Dyer, Victor Chahuneau, and Noah A Smith. 2013. A simple, fast, and effective reparameterization of IBM model 2. In *Proc. of NAACL-HLT*, pages 644–648.

Atsushi Fujii, Masao Utiyama, Mikio Yamamoto, Takehito Utsuro, Terumasa Ehara, Hiroshi Echizen-ya, and Sayori Shimohata. 2008. Overview of the patent translation task at the NTCIR-7 workshop. In *Proc. of NTCIR*, pages 389–400.

Sho Hoshino, Yusuke Miyao, Katsuhito Sudoh, Katsuhiko Hayashi, and Masaaki Nagata. 2015. Discriminative preordering meets Kendall's Tau maximization. In *Proc. of ACL (Short Papers)*, pages 139–144.

Hideki Isozaki, Katsuhito Sudoh, Hajime Tsukada, and Kevin Duh. 2012. HPSG-based preprocessing for English-to-Japanese translation. *ACM Transactions on Asian Language Information Processing*, 11(3):8.

Jason Katz-Brown and Michael Collins. 2008. Syntactic reordering in preprocessing for Japanese → English translation: MIT system description for NTCIR-7 patent translation task. In *Proc. of NTCIR*, pages 409–414.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proc. of HLT-NAACL*, pages 48–54.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proc. of ACL (Demo and Poster Sessions)*, pages 177–180.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proc. of EMNLP*, pages 388–395.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *Proc. of MT summit*, pages 79–86.

Yang Liu, Qun Liu, and Shouxun Lin. 2006. Tree-to-string alignment template for statistical machine translation. In *Proc. of ACL*, pages 609–616.

José B. Mariño, Rafael E. Banchs, Josep M. Crego, Adrià de Gispert, Patrik Lambert, José A. R. Fonollosa, and Marta R. Costa-Jussà. 2006. N-gram-based machine translation. *Computational Linguistics*, 32(4):527–549.

Graham Neubig, Taro Watanabe, Eiichiro Sumita, Shinsuke Mori, and Tatsuya Kawahara. 2011. An unsupervised model for joint phrase alignment and extraction. In *Proc. of ACL-HLT*, pages 632–641.

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

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proc. of ACL*, pages 160–167.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proc. of ACL*, pages 311–318.

Jason Riesa, Ann Irvine, and Daniel Marcu. 2011. Feature-rich language-independent syntax-based alignment for statistical machine translation. In *Proc. of EMNLP*, pages 497–507.

Touch-Based Pre-Post-Editing of Machine Translation Output

Benjamin Marie
LIMSI-CNRS, Orsay, France
Lingua et Machina, Le Chesnay, France
benjamin.marie@limsi.fr

Aurélien Max
LIMSI-CNRS, Orsay, France
Univ. Paris Sud, Orsay, France
aurelien.max@limsi.fr

Abstract

We introduce *pre-post-editing*, possibly the most basic form of interactive translation, as a touch-based interaction with iteratively improved translation hypotheses prior to classical post-editing. We report simulated experiments that yield very large improvements on classical evaluation metrics (up to 21 BLEU) as well as on a parameterized variant of the TER metric that takes into account the cost of matching/touching tokens, confirming the promising prospects of the novel translation scenarios offered by our approach.

1 Introduction

As shown by oracle studies (Wisniewski et al., 2010; Turchi et al., 2012; Marie and Max, 2013), Statistical Machine Translation (SMT) systems produce results that are of significantly lower quality than what could be produced from their available resources. As a pragmatic solution, human intervention is commonly used for improving automatic draft translations, in so-called *post-editing* (PE), but is also studied earlier in the translation process in a variety of interactive strategies, including e.g. completion assistance and local translation choices (e.g. (Foster et al., 2002; Koehn and Haddow, 2009; González-Rubio et al., 2013)). Although *interactive machine translation* does facilitate the work of the SMT system in certain situations by allowing it to make efficient use of knowledge contributed by the human translator, post-editing has been shown to remain a faster alternative (Green et al., 2014). Nevertheless, this activity usually requires complex intervention from an expert translator (Carl et al., 2011).

In this work we reduce interaction with an SMT system to its most basic form: similarly to what a human translator is likely to do when first reading a draft translation to post-edit, we require a user to simply *spot* those segments of a draft translation that can participate in an acceptable translation. The corresponding information is then used by a SMT system in a soft way to improve the draft translation. This process may be iteratively repeated as long as enough improvements are obtained, and terminates with classical post-editing on the obtained translation, hence we dub it *pre-post-editing* (PPE). We resort to simulated pre-post-editing and post-editing, as in other works (Carl et al., 2011; Denkowski et al., 2014), to measure translation performance on some available reference translation using both classical metrics and a variant of the TER metric (Snover et al., 2006), where, essentially, the cost of a token *matching* operation is a parameterized fraction of the cost of the other token edit operations. With the implementation of appropriate strategies in the SMT system, we show under reasonable assumptions that this approach has the potential to significantly reduce the amount of human effort required to obtain a final translation.

In the remainder of this article, we describe the technical details of pre-post-editing (Section 2), report experiments conducted on two translation directions and two domains (Section 3), and finally discuss our proposal and introduce our future work (Section 4).

2 Touch-based pre-post-editing

In our PPE framework, the human *pre-post-editor* has to mark n-grams from a translation hypothesis that can take part in a correct translation.[1] The annotated n-grams are counted, as an n-gram can appear more than once in the same sentence, and a "positive" 6-gram language model (LM)

[1] A touch-based interface when a keyboard is not available or typing is inconvenient lends itself particularly well to PPE.

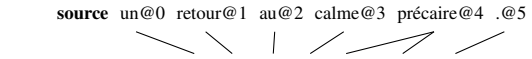
Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1040–1045,
Lisbon, Portugal, 17-21 September 2015.

(positive-lm) is trained on these counts[2]. A "negative" LM (negative-lm) is also trained on the counted n-grams left unannotated. Then, all bi-phrases from the SMT system's phrase table that match an annotated n-gram, according to the source token alignments provided by the decoder, are removed from the main phrase table and stored in a separate "positive" phrase table (positive-pt). Conversely, n-grams containing at least one token left unannotated are considered as incorrect, and the set of bi-phrases matching these n-grams are removed and stored in a "negative" phrase table (negative-pt).

As source tokens can appear more than once in a source text, they are *located*: an identifier is concatenated to each token to make it unique in the source text. Tokens of the source phrases in the phrase table are accordingly also located, so each bi-phrase is duplicated as needed to cover all located tokens. Using located tokens allows our PPE framework to treat differently source tokens that are correctly translated from incorrectly translated ones in the same sentence or text. Figure 1 shows an example of phrase table extraction, using located source tokens[3], for one iteration of PPE.

If an n-gram is annotated as correct, all its inner n-grams of lower order are also deemed correct. Although annotating translations of high quality may be less expensive by explicitely annotating *incorrect* n-grams instead of correct ones, such annotations would not permit to identify correct n-grams inside incorrect ones, as illustrated in Figure 2. PPE can thus be worded as a simple problem for the pre-post-editor: *which sequences of tokens should appear in the final translation?*

The newly extracted phrase tables and LMs[4], along with the remainder of the original phrase table and the original LM, are used to re-decode the source text in a first iteration of PPE. A new PPE annotation can then be performed on the new translations. The newly extracted "positive" and "negative" phrase tables are merged with the corresponding phrase table of the previous iteration. The extracted n-gram counts from the current iteration and the counts of the previous iterations are summed, and the LMs are re-trained with the updated counts. A new iteration of PPE is then performed with the updated models. The weights for all, old or new, models in the log-linear combination are found by tuning on a development set for each PPE iteration.[5] Figure 3 illustrates 4 iterations of PPE from an initial translation hypothesis assuming a given target reference translation.



positive-pt source	positive-pt target	negative-pt source	negative-pt target
retour@1 au@2 précaire@4 .@5	return to precarious .	précaire@4 calme@3 précaire@4 précaire@4 .@5	is precarious calm is precarious is precarious .

positive-lm n-gram	positive-lm count	negative-lm n-gram	negative-lm count
return	1	a	1
return to	1	a return to	1
to	1	to precarious	1
calm	1	to calm is precarious .	1

Figure 1: Examples of some of the bi-phrases and n-grams extracted for phrase tables and language models according to a reference translation.

source	son impopularité semble être en grande partie due au chômage
PPE#0	his unpopularity seems to be / owing / largely / to unemployment
PPE#1	his unpopularity seems to be largely owing to unemployment
target ref	his unpopularity seems to be largely owing to unemployment

Figure 2: Annotation example for two correct tokens forming an incorrect n-gram. At the first PPE iteration a reordering is performed and the new hypothesis now matches the reference translation.

3 Experiments

3.1 Data and systems

We ran experiments on two translation tasks of different domains: the WMT'14 Medical translation task (medical) and the WMT'11 news translation task (news) for the language pair en-fr on both translation directions. For both tasks we trained two competitive phrase-based SMT systems using Moses (Koehn et al., 2007) and WMT data[6] (see Table 1). The tuning for all systems, including our iteration-specific PPE systems, was performed with kb-mira (Cherry and Foster, 2012).

3.2 An adapted evaluation metric: TER_{PPE}

Classical MT evaluation metrics cannot take into account the interactive cost of PPE, and thus do

[2] We used SRILM (Stolcke, 2002) to train the LMs with Witten-Bell smoothing.

[3] Subsequent examples do not use located tokens.

[4] The extracted LMs are sentence-level, and are only used on their specific sentence during PPE.

[5] In this work, we did not exceed 5 iterations.

[6] http://www.statmt.org/wmt14

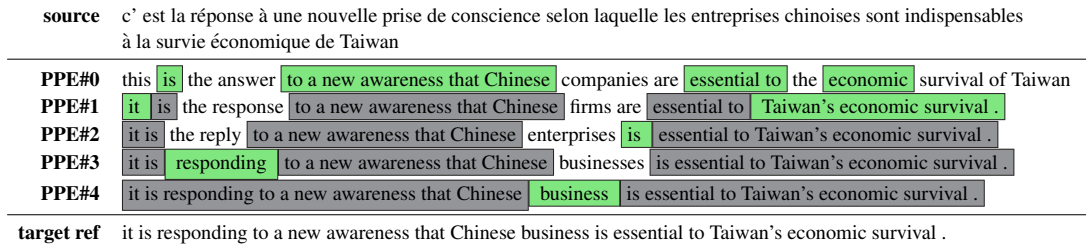


Figure 3: Example of a pre-post-edition trace for French to English translation (using the `news` task, cf. Section 3) using a given implicit target reference translation for simulating pre-post-editing and post-editing. Each newly touched phrase is indicated with a green background. Phrases with a gray background indicate previously touched phrases but their tokens remain individually touchable by the user.

Tasks	Corpus	Sentences	Tokens (fr-en)
`news`	train	12M	383M - 318M
	dev	2,525	73k - 65k
	test	3,003	85k - 74k
`medical`	train	4.9M	91M - 78M
	dev	500	12k - 10k
	test	1,000	26k - 21k
	specialized LM		146M - 78M
for both tasks	LM		2.5B - 6B

Table 1: Data used in this work.

not allow us to make direct comparisons with PE. We thus adapt the TER (Snover et al., 2006) metric, which typically uses 4 types of token edits: `substitution` (s), `insertion` (i), `deletion` (d) and `shift` (f). While these edit types all have a (debatable) uniform cost of 1, the operation of `matching` (m) a correct token is ignored. We posit that this operation is in fact performed by a human translator during PE (at the minimum, by recognizing and skipping tokens), and that it can be directly compared to our touch-based selection of tokens for PPE. However, we cannot at this stage of our work provide a realistic cost value for this operation, and so we introduce a match cost parameter α, and use the following as our PPE-aware metric:

$$\text{TER}_{\text{PPE}} = \frac{\#s + \#d + \#i + \#f + \alpha\#m}{r + \alpha r} \quad (1)$$

where r is the number of tokens in the reference translation. Note that a null value for α makes TER_{PPE} correspond to TER, while a value of 1 would indicate that a token matching/touch (m) is e.g. as costly as a token rewriting (s). We anticipate that a realistic value for α given a reasonably skilled user should be rather small, but we will provide TER_{PPE} results for the full range $[0, 1]$.

3.3 Experimental results

To validate our approach, we initially used a simulated post-editing paradigm (Carl et al., 2011; Denkowski et al., 2014) in which non-post-edited reference translations are used in lieu of human post-editions. Results on TER (Snover et al., 2006) and BLEU (Papineni et al., 2002), tuning on both metrics, are provided in Tables 2 (`news`) and 3 (`medical`).

First, we observe that whatever the metric and the task, the first iteration of PPE always yields a significant improvement over the `Moses` initial system (e.g. up to +9.8 BLEU and -8.2 TER for `news` fr→en). Unsurprisingly, tuning on a metric yields better results for the same metric for the first iteration; however, we note that this is not always true for the TER metric at later iterations (cf. `news` en→fr). More generally, tuning on the TER metric results in lower improvements for `news`, which are mostly concentrated on the first iterations; as systems tuned on BLEU have been found to produce better translations than systems tuned on TER (Cer et al., 2010), only BLEU tuning was used for `medical`.[7]

Improvements follow an interesting pattern over PPE iterations: for instance, on `news` fr→en, BLEU scores steadily increase after each new touch-based iteration and reach a gain of +21.1 BLEU and -12.3 TER over the initial `Moses` translation after 5 PPE iterations. Results are very comparable on both language pairs and both domains, e.g. gains of +12.1 BLEU and -9.7 TER are obtained on fr→en `medical`. The lesser amplitude of the gains obtained after 5 iterations may be attributed to the higher ini-

[7] We have observed a tendency of the TER tuning to shrink the size of hypotheses, resulting in higher brevity penalty values for BLEU and a higher number of insertions for TER.

Iteration	fr→en				en→fr			
	tuned with TER		tuned with BLEU		tuned with TER		tuned with BLEU	
	TER	BLEU	TER	BLEU	TER	BLEU	TER	BLEU
Moses	51.1	28.2	52.7	28.6	52.3	29.7	51.8	31.1
PPE iteration 1	42.9	35.4	46.7	38.4	44.4	35.0	47.3	39.6
PPE iteration 2	40.8	37.3	43.7	43.4	43.0	36.3	44.6	43.9
PPE iteration 3	40.8	37.8	42.2	46.2	42.5	36.4	43.5	46.6
PPE iteration 4	39.9	37.9	40.9	48.3	42.3	36.5	42.3	48.2
PPE iteration 5	39.9	37.9	40.4	49.7	42.2	36.6	41.0	49.5

Table 2: PPE results on the news task.

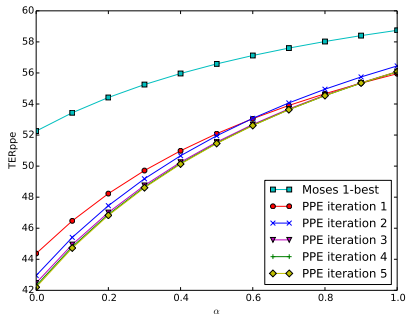
tial quality of the translations in the medical task (e.g. 37.1 BLEU vs 28.6 BLEU in fr→en for Moses with BLEU tuning).

Iteration	fr→en		en→fr	
	tuned with BLEU		tuned with BLEU	
	TER	BLEU	TER	BLEU
Moses	42.2	37.1	44.0	38.8
PPE iteration 1	36.9	44.9	37.2	48.3
PPE iteration 2	34.8	47.5	35.3	51.1
PPE iteration 3	34.1	48.5	33.5	52.9
PPE iteration 4	32.9	49.2	32.4	54.0
PPE iteration 5	32.5	49.2	32.1	54.8

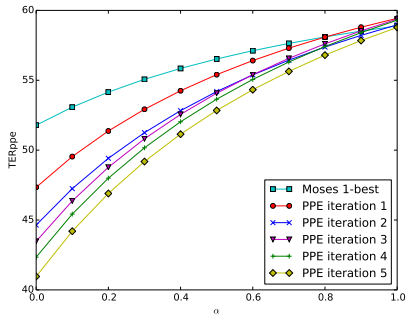
Table 3: PPE results on the medical task.

Figures 4 and 5 show how our TER_{PPE} metric varies for different values of our α parameter (recall that $\alpha = 0$ corresponds to TER). Essentially, whatever the value of α, we observe that any iteration of PPE dominates PE (Moses 1-best), but with a tendency to become as costly as PE for high, but probably unrealistic values of α. Tuning with BLEU allows us to bring regular improvements as the number of iteration increases, while tuning with TER makes the amplitude of the gains decrease faster.

Furthermore, results shown in Table 4 point out the complementarity between negative models (negative-lm and negative-pt) and positive models (positive-lm and positive-pt), with a drop of almost 10 BLEU points compared to the corresponding configuration using all models when removing one type of models on both translation directions. The language models (negative-lm and positive-lm) seem to play a more important role during PPE than the phrase tables (negative-pt and positive-pt), with a drop of 9.6 BLEU points on news fr→en when removing the language models against a significantly lower drop of 4.4 BLEU points when removing the phrase tables.



(a) Tuned with TER



(b) Tuned with BLEU

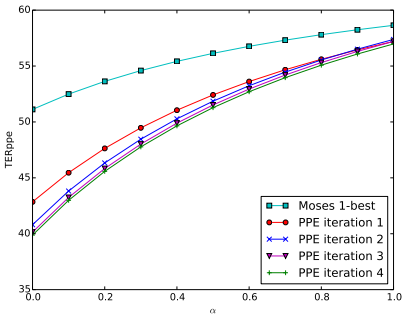
Figure 4: PPE results on the en→fr news task.

4 Discussion and future work

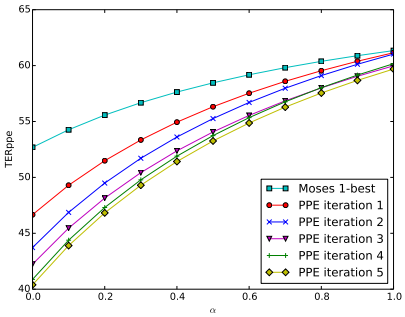
We have introduced *pre-post-editing*, a minimalist interactive machine translation paradigm where a user is only asked to spot text fragments that may be used in the final translation. Our approach is quite comparable to the two-pass procedure described by Luong et al. (2014) using word-level confidence estimation (e.g. (Bach et al., 2011)) to update the cost of the search graph hypotheses. However, contrarily to Luong et al.'s work, our PPE framework is efficiently multi-pass, updates the models over iterations and relies on more informative annotations made at n-gram-level. Our evaluation based on simulated post-editing has revealed a large potential for translation improvement. Interestingly, the type of interaction defined

Configuration	fr→en tuned with BLEU TER	fr→en tuned with BLEU BLEU	en→fr tuned with BLEU TER	en→fr tuned with BLEU BLEU
`Moses`	52.7	28.6	51.8	31.1
PPE w/ all models	40.4	49.7	41.0	49.5
PPE w/o `negative-pt` and `negative-lm`	45.2	39.4	47.1	39.0
PPE w/o `positive-pt` and `positive-lm`	46.7	39.8	48.3	39.8
PPE w/o `negative-pt` and `positive-pt`	45.0	45.3	46.5	44.9
PPE w/o `negative-lm` and `positive-lm`	42.7	40.1	43.2	42.0

Table 4: PPE results for the `news` task after 5 iterations using various configurations.



(a) Tuned with TER



(b) Tuned with BLEU

Figure 5: PPE results on the fr→en `news` task.

is very different from that expected of a post-editor or in existing interactive translation modes, and lends itself nicely to touch-based interaction. Furthermore, our proposal may in fact define a new *role* in Computer-Assisted Translation, with PPE being performed on-the-go on mobile devices by more people than available human translators, and even possibly by monolinguals of the target language whose contribution may be more efficiently exploited than that of monolinguals of the source language (e.g. (Resnik et al., 2010)).

In terms of usability, our future work will focus on two important questions: (*a*) study the actual use of PPE in an interactive setting and tune the α parameter for our TER_{PPE} metric on HTER (Snover et al., 2006) traces, and (*b*) study whether PPE alters in any positive way the work of the human translator performing the residual post-editing, hoping that PE *could become a less tedious task by nature*. We further anticipate that some additions would improve our approach, including dealing early with out-of-vocabulary phrases, proposing local drop-down options (e.g. (Koehn and Haddow, 2009)), possibly clustered by senses, allowing the user to easily fix reordering issues, and adapting PPE to be discourse-aware (e.g. (Ture et al., 2012)).

5 Acknowledgements

The authors would like to thank the anonymous reviewers for their helpful comments and suggestions. The work of the first author is supported by a CIFRE grant from French ANRT.

References

Nguyen Bach, Fei Huang, and Yaser Al-Onaizan. 2011. Goodness: A Method for Measuring Machine Translation Confidence. In *Proceedings of ACL*, Portland, USA.

Michael Carl, Dragsted Barbara, Jakob Elming, Hardt Daniel, and Jakobsen Arnt Lykke. 2011. The Process of Post-Editing: A pilot study. In *Copenhagen Studies in Language*.

Daniel Cer, Christopher D. Manning, and Daniel Jurafsky. 2010. The best lexical metric for phrase-based statistical mt system optimization. In *Proceedings of NAACL*, Los Angeles, USA.

Colin Cherry and George Foster. 2012. Batch Tuning Strategies for Statistical Machine Translation. In *Proceedings of NAACL*, Montréal, Canada.

Michael Denkowski, Chris Dyer, and Alon Lavie. 2014. Learning from Post-Editing : Online Model Adaptation for Statistical Machine Translation. In *Proceedings of EACL*, Gothenburg, Sweden.

George Foster, Philippe Langlais, and Guy Lapalme. 2002. User-Friendly Text Prediction for Translators. In *Proceedings of EMNLP*, Philadelphia, USA.

Jesús González-Rubio, Daniel Ortiz-Martínez, José-Miguel Benedi, and Francisco Casacuberta. 2013. Interactive Machine Translation using Hierarchical Translation Models. In *Proceedings of EMNLP*, Seattle, USA.

Spence Green, Sida Wang, Jason Chuang, Jeffrey Heer, Sebastian Schuster, and Christopher D. Manning. 2014. Human Effort and Machine Learnability in Computer Aided Translation. In *Proceedings of EMNLP*, Doha, Qatar.

Philipp Koehn and Barry Haddow. 2009. Interactive assistance to human translators using statistical machine translation methods. In *Proceedings of MT Summit*, Ottawa, Canada.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open Source Toolkit for Statistical Machine Translation. In *Proceedings of ACL, demo session*, Prague, Czech Republic.

Benjamin Marie and Aurélien Max. 2013. A Study in Greedy Oracle Improvement of Translation Hypotheses. In *Proceedings of IWSLT*, Heidelberg, Germany.

Luong Ngoc Quang, Laurent Besacier, and Lecouteux Benjamin. 2014. An Efficient Two-Pass Decoder for SMT Using Word Confidence Estimation. In *Proceedings of EAMT*, Dubrovnik, Croatia.

Kishore Papineni, Salim Roukos, Todd Ward, and Weijing Zhu. 2002. BLEU: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of ACL*, Philadelphia, USA.

Philip Resnik, Olivia Buzek, Chang Hu, Yakov Kronrod, Alex Quinn, and Benjamin B. Bederson. 2010. Improving Translation via Targeted Paraphrasing. In *Proceedings of EMNLP*, Cambridge, USA.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A Study of Translation Edit Rate with Targeted Human Annotation. In *Proceedings of AMTA*, Cambridge, USA.

Andreas Stolcke. 2002. SRILM - an extensible language modeling toolkit. In *Proceedings of ICSLP*, Denver, USA.

Marco Turchi, Tijl De Bie, Cyril Goutte, and Nello Cristianini. 2012. Learning to Translate: A Statistical and Computational Analysis. *Advances in Artificial Intelligence*.

Ferhan Ture, Douglas W. Oard, and Philip Resnik. 2012. Encouraging Consistent Translation Choices. In *Proceedings of NAACL*, Montréal, Canada.

Guillaume Wisniewski, Alexandre Allauzen, and François Yvon. 2010. Assessing Phrase-Based Translation Models with Oracle Decoding. In *Proceedings of EMNLP*, Cambridge, USA.

A Discriminative Training Procedure for Continuous Translation Models

† *Quoc-Khanh Do, † *Alexandre Allauzen and *François Yvon
† Université Paris-Sud, Orsay, France
* LIMSI/CNRS, Orsay, France
firstname.surname@limsi.fr

Abstract

Continuous-space translation models have recently emerged as extremely powerful ways to boost the performance of existing translation systems. A simple, yet effective way to integrate such models in inference is to use them in an N-best rescoring step. In this paper, we focus on this scenario and show that the performance gains in rescoring can be greatly increased when the neural network is trained jointly with all the other model parameters, using an appropriate objective function. Our approach is validated on two domains, where it outperforms strong baselines.

1 Introduction

Over the past few years, research on neural networks (NN) architectures for Natural Language Processing has been rejuvenated. Boosted by early successes in language modelling for speech recognition (Schwenk, 2007; Le et al., 2011), NNs have since been successufully applied to many other tasks (Socher et al., 2013; Huang et al., 2012; Yang et al., 2013). In particular, these techniques have been applied to Statistical Machine Translation (SMT), first to estimate continuous-space translation models (CTMs) (Schwenk et al., 2007; Le et al., 2012; Devlin et al., 2014), and more recently to implement end-to-end translation systems (Cho et al., 2014; Sutskever et al., 2014).

In most SMT settings, CTMs are used as an additional feature function in the log-linear model, and are conventionally trained by maximizing the regularized log-likelihood on some parallel training corpora. Since this objective function requires to normalize scores, several alternative training objectives have recently been proposed to speed up training and inference, a popular and effective choice being the Noise Contrastive Estimation (NCE) introduced in (Gutmann and Hyvärinen, 2010). In any case, NN training is typically performed (a) in isolation from the other components of the SMT system and (b) using a criterion that is unrelated to the actual performance of the SMT system (as measured for instance by BLEU). It is therefore likely that the resulting NN parameters are sub-optimal with respect to their intended use.

In this paper, we study an alternative training regime aimed at addressing problems (a) and (b). To this end, we propose a new objective function used to discriminatively train or adapt CTMs, along with a training procedure that enables to take the other components of the system into account. Our starting point is a non-normalized extension of the n-gram CTM of (Le et al., 2012) that we briefly restate in section 2. We then introduce our objective function and the associated optimization procedure in section 3. As will be discussed, our new training criterion is inspired both from max-margin methods (Watanabe et al., 2007) and from pair-wise ranking (PRO) (Hopkins and May, 2011; Simianer et al., 2012). This proposal is evaluated in an N-best rescoring step, using the framework of n-gram-based systems, within which they integrate seamlessly. Note, however that it could be used with any phrase-based system. Experimental results for two translation tasks (section 4) clearly demonstrate the benefits of using discriminative training on top of an NCE-trained model, as it almost doubles the performance improvements of the rescoring step in all settings.

2 n-gram-based CTMs

The n-gram-based approach in Machine Translation is a variant of the phrase-based approach (Zens et al., 2002). Introduced in (Casacuberta and Vidal, 2004), and extended in (Mariño et al., 2006; Crego and Mariño, 2006), this approach is based on a specific factorization of the joint probability of parallel sentence pairs, where the

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1046–1052,
Lisbon, Portugal, 17-21 September 2015.

source sentence has been reordered beforehand.

2.1 n-gram-based Machine Translation

Let $(\mathbf{s}, \mathbf{t})$ denote a sentence pair made of a source $\mathbf{s}$ and target $\mathbf{t}$ sides. This sentence pair is decomposed into a sequence of L bilingual units called *tuples* defining a joint segmentation. In this framework, tuples constitute the basic translation units: like phrase pairs, they represent a matching between a source and a target chunk . The joint probability of a *synchronized* and *segmented* sentence pair can be estimated using the n-gram assumption. During training, the segmentation is obtained as a by-product of source reordering, (see (Crego and Mariño, 2006) for details). During the inference step, the SMT decoder will compute and output the best derivation in a small set of pre-defined reorderings.

Note that the n-gram translation model manipulates bilingual tuples. The underlying set of events is thus much larger than for word-based models, while the training data (parallel corpora) are typically order of magnitude smaller than monolingual resources. As a consequence, data sparsity issues for such models are particularly severe. Effective workarounds consist in factorizing the conditional probabitily of tuples into terms involving smaller units: the resulting model thus splits bilingual phrases in two sequences of respectively source and target words, synchronised by the tuple segmentation. Such bilingual word-based n-gram models were initially described in (Le et al., 2012). We assume here a similar decomposition.

2.2 Neural Architectures

The estimation of n-gram probabilities can be performed via multi-layer NN structures, as described in (Bengio et al., 2003; Schwenk, 2007) for a monolingual language model. The standard *feed-forward* structure is used to estimate the translation models sketched in the previous section. We give here a brief description, more details are in (Le et al., 2012): first, each context word is projected into language dependent continuous spaces, using two projection matrices for the source and target languages. The continuous representations are then concatenated to form the representation of the context, which is used as input for a feed-forward NN predicting a target word.

In such architecture, the size of output vocabulary is a bottleneck when normalized distributions are expected. Various workarounds have been proposed, relying for instance on a structured output layer using word-classes (Mnih and Hinton, 2008; Le et al., 2011). A more effective alternative, which however only delivers *quasi-normalized* scores, is to train the network using the *Noise Contrastive Estimation* or NCE (Gutmann and Hyvärinen, 2010; Mnih and Teh, 2012). This technique is readily applicable for CTMs and has been adopted here. We therefore assume that the NN outputs a positive score $\mathbf{b}_{\boldsymbol{\theta}}(w, \mathbf{c})$ for each word w given its context $\mathbf{c}$; this score is simply computed as $\mathbf{b}_{\boldsymbol{\theta}}(w, \mathbf{c}) = \exp(\mathbf{a}_{\boldsymbol{\theta}}(w, \mathbf{c}))$, where $\mathbf{a}_{\boldsymbol{\theta}}(w, \mathbf{c})$ is the activation at the output layer; $\boldsymbol{\theta}$ denotes all the network free parameters.

3 Discriminative Training of CTMs

In SMT, the primary role of CTMs is to help the system in ranking a set of hypotheses so that the top scoring hypotheses correspond to the best translations, where quality is measured using automatic metrics such as BLEU (Papineni et al., 2002). Given the computational burden of continuous models, the prefered use of CTMs is to rescore a list of N-best hypotheses, a scenario we favor here; note that their integration in a first pass search is also possible (Niehues and Waibel, 2012; Vaswani et al., 2013; Devlin et al., 2014). The important point is to realize that the CTM score will in any case be composed with several scores computed by other components: reordering model(s), monolingual language model(s), etc. In this section, we propose a discriminative training framework which implements a tight integration of the CTM with the rest of the system.

3.1 A Discriminative Training Framework

The decoder generates a list of N hypotheses for each source sentence $\mathbf{s}$. Each hypothesis $\mathbf{h}$ is composed of a target sentence $\mathbf{t}$ along with its associated derivation and is evaluated as follows:

$$G_{\boldsymbol{\lambda},\boldsymbol{\theta}}(\mathbf{s}, \mathbf{h}) = \sum_{k=1}^{M} \lambda_k f_k(\mathbf{s}, \mathbf{h}) + \lambda_{M+1} f_{\boldsymbol{\theta}}(\mathbf{s}, \mathbf{h}),$$

where M conventional feature functions[1] $f_1 ... f_M$, estimated during the training phase, are scaled by coefficients $\lambda_1 ... \lambda_M$. The introduction of a continuous model during the rescoring step is implemented by adding the feature $f_{\boldsymbol{\theta}}(\mathbf{s}, \mathbf{h})$, which ac-

[1]The functions used in our experiments are similar to the ones used in other phrase-based systems (Crego et al., 2011).

Algorithm 1 Joint optimization of $\boldsymbol{\theta}$ and $\boldsymbol{\lambda}$

```
Init. of θ and λ
for each iteration do
    for P mini-batch do                    ▷ λ is fixed
        Compute the sub-gradient of L(θ) for each sentence s in the mini-batch
        Update θ
    end for
    Update λ on development set            ▷ θ is fixed
end for
```

cumulates, over all contexts $\mathbf{c}$ and word w, the CTM log-score $\log \mathbf{b}_{\boldsymbol{\theta}}(w, \mathbf{c})$.

$G_{\boldsymbol{\lambda},\boldsymbol{\theta}}$ depends both on the NN parameters $\boldsymbol{\theta}$ and on the log-linear coefficients $\boldsymbol{\lambda}$. We propose to train these two sets of parameters, by alternatively updating $\boldsymbol{\theta}$ through SGD on the training corpus, and updating $\boldsymbol{\lambda}$ using conventional algorithms on the development data. This procedure, which has also been adopted in recent studies (e.g. (He and Deng, 2012; Gao and He, 2013)) is sketched in algorithm 1. In practice, the training data is successively divided into mini-batches of 128 sentences. Each mini-batch is used to compute the sub-gradient of the training criterion (see section 3.2) and to update $\boldsymbol{\theta}$. After each training iteration of the CTM, $\boldsymbol{\lambda}$s are retuned on the development set; we use here the *K-Best Mira* algorithm of Cherry and Foster (2012) as implemented in MOSES.[2]

3.2 Loss function

The training criterion considered here draws inspiration both from max-margin methods (Watanabe et al., 2007) and from the pair-wise ranking (PRO) (Hopkins and May, 2011; Simianer et al., 2012). The choice of a ranking loss seems to be the most appropriate in our setting; as in many recent studies on discriminative training for MT (e.g. (Chiang, 2012; Flanigan et al., 2013)), the integration of the translation metric into the loss function is critical to obtain parameters that will yield good translation performance.

Translation hypotheses $\mathbf{h}_i$ are scored using a sentence-level approximation of BLEU denoted $SBLEU(\mathbf{h}_i)$. Let r_i be the rank of hypothesis $\mathbf{h}_i$ when hypotheses are sorted according to their sentence-level BLEU. *Critical hypotheses* are defined as follows:[3]

$$\mathcal{C}^{\alpha}_{\delta}(\mathbf{s}) = \{(i,k) : 1 \leq k, i \leq N, r_k - r_i \geq \delta, \\ \Delta_{i,k} G_{\boldsymbol{\lambda},\boldsymbol{\theta}}(\mathbf{s}, \mathbf{h}) < \alpha \Delta_{i,k} SBLEU(\mathbf{h}).$$

A pair of hypotheses is thus deemed critical when a large difference in $SBLEU$ is not reflected by the difference of scores, which falls below a threshold. This threshold is defined by the difference between their sentence-level BLEU, multiplied by α. Our loss function $\mathcal{L}(\boldsymbol{\theta})$ is defined with respect to this critical set and can be written as:[4]

$$\sum_{(i,k) \in \mathcal{C}^{\alpha}_{\delta}(\mathbf{s})} \alpha \Delta_{i,k} SBLEU(\mathbf{h}) - \Delta_{i,k} G_{\boldsymbol{\lambda},\boldsymbol{\theta}}(\mathbf{s}, \mathbf{h}_i)$$

Initialization is an important issue when optimizing NN. Moreover, our training procedure depends heavily on the log-linear coefficients $\boldsymbol{\lambda}$. To initialize $\boldsymbol{\theta}$, preliminary experiments (Do et al., 2014; Do et al., 2015) show that it is more efficient to start from a NN pre-trained using NCE, while the discriminative loss is used only in a fine-tuning phase. Given the pre-trained CTM's scores, we initialize $\boldsymbol{\lambda}$ by optimizing it on the development set. This strategy forces the training of $\boldsymbol{\theta}$ to focus on errors made by the system as a whole.

4 Experiments

4.1 Tasks and Corpora

The discriminative optimization framework is evaluated both in a training and in an adaptation scenario. In the *training* scenario, the CTM is trained on the same parallel data as the one used for the baseline system. In the *adaptation* scenario, large out-of-domain corpora are used to train the baseline SMT system, while the CTM is trained on a much smaller, in-domain corpus and only serves for rescoring. An intermediate situation (*partial training*) is when only a fraction of the training data is re-used to estimate the CTM: this situation is interesting because it allows us to train the CTM much faster than in the training scenario.[5]

Two domains are investigated. For the TED Talkstask[6] the only parallel in-domain data contains 180K sentence pairs; the out-of-domain

[2] http://www.statmt.org/moses/

[3] $\Delta_{i,k}$ denotes the difference of values (for SBLEU or $G_{\boldsymbol{\lambda},\boldsymbol{\theta}}$) between hypthoses h_i and h_k.

[4] This is for one single training sample.

[5] The discriminative training step also uses the development data.

[6] http://workshop2014.iwslt.org/

	dev	test	train
Training scenario			
Baseline Ncode on TED	28.1	32.3	**65.6**
Baseline + CTM NCE	28.9	33.1	64.1
Baseline + CTM discriminative	**29.0**	**33.5**	64.9
Adaptation scenario			
Baseline Ncode on WMT	28.5	32.0	33.3
Baseline + CTM NCE	29.2	33.0	34.9
Baseline + CTM discriminative	**29.8**	**33.9**	**35.8**

Table 1: BLEU scores for the TED Talkstasks.

	dev	test	train
Partial training scenario			
Baseline Ncode	40.4	37.4	45.8
Baseline + CTM NCE	40.8	38.1	45.2
Baseline + CTM discriminative	**41.8**	**38.8**	**46.0**
"Adaptation" scenario			
Baseline Ncode	39.8	37.2	39.4
Baseline + CTM NCE	41.2	38.2	40.4
Baseline + CTM discriminative	**41.8**	**38.9**	**41.5**

Table 2: BLEU scores for the medical tasks.

data is much larger and contains all corpora allowed in the translation shared task of WMT'14 (English-French), amounting to 12M parallel sentences. The second task is the medical translation task of WMT'14[7] (English to French) for which we use all authorized corpora. The Patent-Abstract corpus, made of 200K parallel sentence pairs, is used either for adaptation or partial training for the CTM. Experimental results are reported on official evaluation sets, as well as on the CTM training set.

All translation systems are based on the open source implementation[8] of the bilingual n-gram approach to MT. For the NN structure, each vocabulary's word is projected into a 500-dimension space followed by two hidden layers of 1000 and 500 units. For the discriminative training and adaptation tasks, baseline SMT systems are used to generate respectively 600 and 300 best hypotheses for each sentence of the in-domain corpus. [9]

4.2 Experimental results

Results in Table 1 measure the impact of discriminative training on top of an NCE-trained model for the two TED Talks conditions. In the adaptation task, the discriminative training of the CTM gives a large improvement of 0.9 BLEU score over the CTM only trained with NCE and 1.9 over the baseline system. However, for the training scenario, these gains are reduced respectively to 0.4 and 1.2 BLEU points. The BLEU scores (in the **train** column) measured on the N-best lists used to train the CTM provide an explanation for this difference: in training, the N-best lists contain hypotheses with an overoptimistic BLEU score, to be compared with the ones observed on unseen data. As a result, adding the CTM significantly worsens the performance on the discriminative training data, contrarily to what is observed on the development and test sets. Even if the results of these two conditions cannot be directly compared (the baselines are different), it seems that the proposed discriminative training has a greater impact on performance in the adaptation scenario, even though the out-of-domain system initially yields lower BLEU scores.

The medical translation task represents a different situation, in which a large-scale system is built from multiples but domain-related corpora, among which, one is used to train the CTM. Nevertheless, results reported in Table 2 exhibit a similar trend. For both conditions, the discriminative training gives a significant improvement, up to 0.7 BLEU score over the one only trained with NCE and up to 1.7 over the baseline system. Arguably, the difference between the two conditions is much smaller than what was observed with the TED Talks task, due to the fact that the Patent-Abstract corpus used to discriminatively train the CTM only corresponds to a small subset of the parallel data. However, the best strategy seems, here again, to exclude the data used for the CTM from the data used to train the baseline system.

5 Related work

It is important to notice that similar discriminative methods have been used to train phrase table's scores (He and Deng, 2012; Gao and He, 2013; Gao et al., 2014), or a recurrent NNLM (Auli and Gao, 2014). In recent studies, the authors tend to limit the number of iterations to 1 (Gao et al., 2014; Auli and Gao, 2014), while we still advocate the general iterative procedure sketched in Algo. 1. Initialization is also an important issue when optimizing NN. In this work, we initialize CTM's parameters by using a pre-training procedure based on the model's probabilistic in-

[7] www.statmt.org/wmt14/medical-task/

[8] ncode.limsi.fr/

[9] The threshold δ is set to 250 for 300-best and to 500 for 600-best lists, while α is set empirically.

terpretation and NCE algorithm to produce *quasi-normalized scores*, while similar work in (Auli and Gao, 2014) only uses *un-normalized scores*. The initial values of $\boldsymbol{\lambda}$ also needs some investigation. Gao et al. (2014) and Auli and Gao (2014) initialize λ_{M+1} to 1, and normalize all other coefficients; here we initialize $\boldsymbol{\lambda}$ by optimizing it on the development set using the pre-trained CTM's scores. This strategy forces the training of $\boldsymbol{\theta}$ to focus on errors made by the system as a whole. The fundamental difference of this work hence lays in the use of the ranking loss described in Section 3.2, whereas previous works use expected BLEU loss. We plan a systematic comparison between these two criteria, along with some other discriminative losses in a future work.

About the CTM's structure, our used model is based on the feed-forward CTM described in (Le et al., 2012) and extended in (Devlin et al., 2014). This structure, though simple, have been shown to achieve impressive results, and with which efficient tricks are available to speed up both training and inference. While models in (Le et al., 2012) employ a structured output layer to reduce *softmax* operation's cost, we prefer the NCE *self-normalized* output which is very efficient both in training and inference. Another form of self-normalization is presented in (Devlin et al., 2014) but does not seem to have fast training. Finally, although N-best rescoring is used in this work to facilitate the discriminative training, other CTM's integration into SMT systems exist, such as lattice reranking (Auli et al., 2013) or direct decoding with CTM (Niehues and Waibel, 2012; Devlin et al., 2014; Auli and Gao, 2014).

6 Conclusions

In this paper, we have proposed a new discriminative training procedure for continuous-space translation models, which correlates better with translation quality than conventional training methods. This procedure has been validated using an n-gram-based CTM, but the general idea could be applied to other continuous models which compute a score for each translation hypothesis. The core of the method lays in the definition of a new objective function inspired both from max-margin and Pairwise Ranking approach in MT, which enables us to effectively integrate the CTM into the SMT system through N-best rescoring. A major difference with most past efforts along these lines is the joint training of the CTM and the log-linear parameters. In all our experiments, discriminative training, when applied on a CTM initially trained with NCE, yields substantial performance gains.

Acknowledgments

This work has been partly funded by the European Union's Horizon 2020 research and innovation programme under grant agreement No. 645452 (QT21).

References

Michael Auli and Jianfeng Gao. 2014. Decoder integration and expected bleu training for recurrent neural network language models. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 136–142.

Michael Auli, Michel Galley, Chris Quirk, and Geoffrey Zweig. 2013. Joint language and translation modeling with recurrent neural networks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1044–1054.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155.

Francesco Casacuberta and Enrique Vidal. 2004. Machine translation with inferred stochastic finite-state transducers. *Computational Linguistics*, 30(3):205–225.

Colin Cherry and George Foster. 2012. Batch tuning strategies for statistical machine translation. In *Proceedings of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 427–436.

David Chiang. 2012. Hope and fear for discriminative training of statistical translation models. *Journal of Machine Learning Research*, 13(1):1159–1187.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar.

Josep M. Crego and José B. Mariño. 2006. Improving statistical MT by coupling reordering and decoding. *Machine Translation*, 20(3):199–215.

Josep M. Crego, François Yvon, and José B. Mariño. 2011. N-code: an open-source bilingual N-gram SMT toolkit. *Prague Bulletin of Mathematical Linguistics*, 96:49–58.

Jacob Devlin, Rabih Zbib, Zhongqiang Huang, Thomas Lamar, Richard Schwartz, and John Makhoul. 2014. Fast and robust neural network joint models for statistical machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1370–1380, Baltimore, MD.

Quoc-Khanh Do, Alexandre Allauzen, and François Yvon. 2014. Discriminative adaptation of continuous space translation models. In *International Workshop on Spoken Language Translation (IWSLT 2014)*, Lake Tahoe, USA.

Quoc-Khanh Do, Alexandre Allauzen, and François Yvon. 2015. Apprentissage discriminant des modèles continus de traduction. In *Actes de la 22e conférence sur le Traitement Automatique des Langues Naturelles*, pages 267–278, Caen, France, June. Association pour le Traitement Automatique des Langues.

Jeffrey Flanigan, Chris Dyer, and Jaime Carbonell. 2013. Large-scale discriminative training for statistical machine translation using held-out line search. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 248–258, Atlanta, Georgia.

Jianfeng Gao and Xiaodong He. 2013. Training mrf-based phrase translation models using gradient ascent. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 450–459, Atlanta, Georgia.

Jianfeng Gao, Xiaodong He, Wen-tau Yih, and Li Deng. 2014. Learning continuous phrase representations for translation modeling. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, Baltimore, MD.

Michael Gutmann and Aapo Hyvärinen. 2010. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In Yeh Whye Teh and Mike Titterington, editors, *Proceedings of th International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 9, pages 297–304.

Xiaodong He and Li Deng. 2012. Maximum expected bleu training of phrase and lexicon translation models. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 292–301.

Mark Hopkins and Jonathan May. 2011. Tuning as ranking. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1352–1362, Edinburgh, Scotland, UK.

Eric Huang, Richard Socher, Christopher Manning, and Andrew Ng. 2012. Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 873–882, Jeju Island, Korea.

Hai-Son Le, Ilya Oparin, Alexandre Allauzen, Jean-Luc Gauvain, and François Yvon. 2011. Structured output layer neural network language model. In *Proceedings of the International Conference on Audio, Speech and Signal Processing*, pages 5524–5527.

Hai-Son Le, Alexandre Allauzen, and François Yvon. 2012. Continuous space translation models with neural networks. In *Proceedings of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 39–48, Montréal, Canada.

José B. Mariño, Rafael E. Banchs, Josep M. Crego, Adrià de Gispert, Patrick Lambert, José A.R. Fonollosa, and Marta R. Costa-Jussà. 2006. N-gram-based machine translation. *Computational Linguistics*, 32(4):527–549.

Andriy Mnih and Geoffrey E Hinton. 2008. A scalable hierarchical distributed language model. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, volume 21, pages 1081–1088.

Andriy Mnih and Yeh Whye Teh. 2012. A fast and simple algorithm for training neural probabilistic language models. In *Proceedings of the International Conference of Machine Learning (ICML)*.

Jan Niehues and Alex Waibel. 2012. Continuous space language models using restricted Boltzmann machines. In *Proceedings of International Workshop on Spoken Language Translation (IWSLT)*, pages 164–170, Hong-Kong, China.

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

Holger Schwenk, Marta R. Costa-jussa, and Jose A. R. Fonollosa. 2007. Smooth bilingual n-gram translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 430–438, Prague, Czech Republic.

Holger Schwenk. 2007. Continuous space language models. *Computer Speech and Language*, 21(3):492–518.

Patrick Simianer, Stefan Riezler, and Chris Dyer. 2012. Joint feature selection in distributed stochastic learning for large-scale discriminative training in SMT. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 11–21.

Richard Socher, John Bauer, Christopher D. Manning, and Ng Andrew Y. 2013. Parsing with compositional vector grammars. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 455–465, Sofia, Bulgaria.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, NIPS*27, pages 3104–3112, Montréal, Canada.

Ashish Vaswani, Yinggong Zhao, Victoria Fossum, and David Chiang. 2013. Decoding with large-scale neural language models improves translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1387–1392, Seattle, Washington, USA.

Taro Watanabe, Jun Suzuki, Hajime Tsukada, and Hideki Isozaki. 2007. Online large-margin training for statistical machine translation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 764–773, Prague, Czech Republic.

Nan Yang, Shujie Liu, Mu Li, Ming Zhou, and Nenghai Yu. 2013. Word alignment modeling with context dependent deep neural networks. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 166–175, Sofia, Bulgaria.

Richard Zens, Franz Josef Och, and Hermann Ney. 2002. Phrase-based statistical machine translation. In *KI '02: Proceedings of the 25th Annual German Conference on AI*, pages 18–32, London, UK. Springer-Verlag.

System Combination for Machine Translation through Paraphrasing

Wei-Yun Ma
Institute of Information science
Academia Sinica
Taipei 115, Taiwan
ma@iis.sinica.edu.tw

Kathleen McKeown
Department of Computer Science
Columbia University
New York, NY 10027, USA
kathy@cs.columbia.edu

Abstract

In this paper, we propose a paraphrasing model to address the task of system combination for machine translation. We dynamically learn hierarchical paraphrases from target hypotheses and form a synchronous context-free grammar to guide a series of transformations of target hypotheses into fused translations. The model is able to exploit phrasal and structural system-weighted consensus and also to utilize existing information about word ordering present in the target hypotheses. In addition, to consider a diverse set of plausible fused translations, we develop a hybrid combination architecture, where we paraphrase every target hypothesis using different fusing techniques to obtain fused translations for each target, and then make the final selection among all fused translations. Our experimental results show that our approach can achieve a significant improvement over combination baselines.

1 Introduction

In the past several years, many machine translation (MT) combination approaches have been developed. Word-level combination approaches, such as the confusion network decoding model, have been quite successful (Matusov et al., 2006; Rosti et al., 2007a; He et al. 2008; Karakos et al. 2008; Chen et al. 2009a; Narsale 2010; Leusch 2011; Freitag et al. 2014).

In addition to word-level combination approaches, some phrase-level combination approaches have also recently been developed; the goal is to retain coherence and consistency between the words in a phrase. The most common phrase-level combination approaches are re-decoding methods: by constructing a new phrase table from each MT system's source-to-target phrase alignments, the source sentence can also be re-decoded using the new translation table (Rosti et al., 2007b; Huang and Papineni, 2007; Chen et al., 2007; Chen et al., 2009b). One problem with these approaches is that, just with a new phrase table, existing information about word ordering present in the target hypotheses is not utilized; thus the approaches are likely to make new mistakes of word reordering which do not appear in the target hypotheses of MT engines. Huang and Papineni (2007) attacked this issue through a reordering cost function that encourages search along with decoding paths from all MT engines' decoders.

Another phrase-level combination approach relies on a lattice decoding model to carry out the combination (Feng et al 2009; Du and Way 2010; Ma and McKeown 2012). In a lattice, each edge is associated with a phrase (a single word or a sequence of words) rather than a single word. The construction of the lattice is based on the extraction of phrase pairs from word alignments between a selected best MT system hypothesis (the backbone) and the other translation hypotheses. One challenge of the lattice decoding model is that it is difficult to consider structural consensus among target hypotheses from multiple MT engines, i.e, the consensus among occurrences of discontinuous words.

In this paper, we propose another phrase-level combination approach – a paraphrasing model using hierarchical paraphrases (paraphrases contain subparaphrases), to fuse target hypotheses. We dynamically learn hierarchical paraphrases from target hypotheses without any syntactic annotations and form a synchronous context-free grammar (SCFG) (Aho and Ullman 1969) to

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1053–1058,
Lisbon, Portugal, 17-21 September 2015.

guide a series of transformations of target hypotheses into fused translations. Through these structural transformations, the paraphrasing model is able to exploit phrasal and structural system-weighted consensus and also able to utilize existing information about word ordering present in the target hypotheses. In addition, to consider a diverse set of plausible fused translations, we develop a hybrid combination architecture, where we paraphrase every target hypothesis using different fusing techniques to obtain fused translations for each target, and then make the final selection among all fused translations through a sentence-level selection-based model.

In short, compared with other related work, our approach features the following advantages:

1. It can consider structural system-weighted consensus among target hypotheses from multiple MT engines through its hierarchical paraphrases, which non-hierarchical paraphrases are not able to do.

2. It can utilize existing information about word ordering present in the target hypotheses.

3. It can retain coherence and consistency between the words in a phrase.

4. The hybrid combination architecture enables us to consider a diverse set of plausible fused translations produced by different fusing techniques.

2 Hybrid Combination Architecture

In the context of system combination, discriminative reranking or post editing, MT researchers (Rosti et al., 2007a; Huang and Papineni, 2007; Devlin and Matsoukas, 2012, Matusov et al., 2008; Gimpel et al., 2013) have recently shown many positive results if more diverse translations are considered. Inspired by them, we develop a hybrid combination architecture in order to consider more diverse fused translations. We paraphrase every target hypothesis to obtain the corresponding fused translation, and then make the final selection among all fused translations through a sentence-level selection-based model, shown in Figure 1. In the architecture, different fusing techniques can be used to generate fused translations for the further sentence-level selection, enabling us to exploit more sophisticated information of the whole sentence.

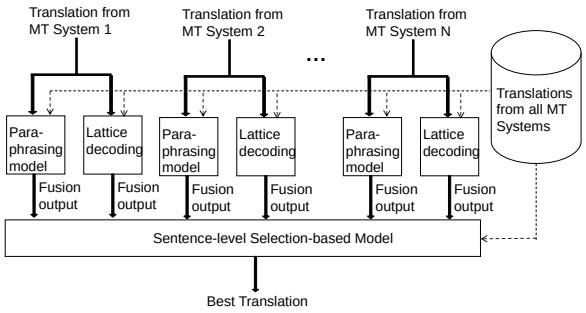


Figure 1. An example of *hybrid combination architecture*

3 Paraphrasing Model

In this section, we introduce our paraphrasing model. For each single target hypothesis, we extract a set of hierarchical paraphrases from monolingual word alignments between the hypothesis and other hypotheses. Each set of hierarchical paraphrases forms a synchronous context-free grammar to guide a series of transformations of that target hypothesis into a fused translation.

Any monolingual word aligner can be used to produce the monolingual word alignments. In our system, we adopt TERp (Snover et al. 2009), one of the state-of-the-art alignment tools, to serve this purpose. TERp is an extension of TER (Snover et al. 2006). Both TERp and TER are automatic evaluation metrics for MT, based on measuring the ratio of the number of edit operations between the reference sentence and the MT system hypothesis. The edit operations of TERp include TER's Matches, Insertions, Deletions, Substitutions and Shifts—as well as three new edit operations: Stem Matches, Synonym Matches and Paraphrases. A valuable side product of TERp is the monolingual word alignment. A constructed example is shown in Figure 2.

3.1 Hierarchical Paraphrase Extraction

We first introduce our notation. For a given sentence *i*, we use E_h^i to denote the target hypothesis from MT system *h*, use EP_h^i to denote E_h^i attached with related word positions, use e_h^i to denote a phrase within E_h^i, and use ep_h^i to denote e_h^i attached with related word positions. For instance, If E_h^i is "you buy the book", then EP_h^i would be "you[1] buy[2] the[3] book[4]". If e_h^i is "the book", then ep_h^i is "the[3] book[4]".

For a given sentence *i*, a MT system *h* and a MT system *k*, we use a SCFG denoted by $Q_{h,k}^i$ to represent the set of hierarchical paraphrases learned from EP_h^i and EP_k^i. Adapting (Chiang

2007), we design the following rules to obtain $Q^i_{h,k}$, based on the monolingual word alignment, obtained by a aligner, such as TERp.

- If $< ep^i_h , ep^i_k >$ is consistent[1] with the monolingual word alignment, then $X \rightarrow < ep^i_h , e^i_k >$ is added to $Q^i_{h,k}$.
- If $X \rightarrow < \gamma , \alpha >$ is in $Q^i_{h,k}$, and $< ep^i_h , ep^i_k >$ is consistent with monolingual word alignment such that $\gamma = \gamma_1 ep^i_h \gamma_2$ and $\alpha = \alpha_1 e^i_k \alpha_2$, then $X \rightarrow < \gamma_1 X_a \gamma_2 , \alpha_1 X_a \alpha_2 >$ is added to $Q^i_{h,k}$, where *a* is an index.

Please note that for each extracted hierarchical paraphrase - $X \rightarrow < \gamma , \alpha >$, γ would include information of word positions while α would not.

For a certain target hypothesis - EP^i_h, our goal is to paraphrase it to get the fusion output by using a set of hierarchical paraphrases, denoted by Q^i_h. Thus we create the union of all related hierarchical paraphrases learned from EP^i_h and other target hypotheses. Two special "glue" rules - $S \rightarrow < S_1 X_2 , S_1 X_2 >$ and $S \rightarrow < X_1 , X_1 >$ are also added to Q^i_h. The process can be represented formally in the following:

$$Q^i_h = \bigcup_{k=1}^{N} Q^i_{h,k} \bigcup \{ S \rightarrow < S_1 X_2 , S_1 X_2 > , S \rightarrow < X_1 , X_1 > \}$$

where *N* is the total number of MT systems.

3.1.1 An Example

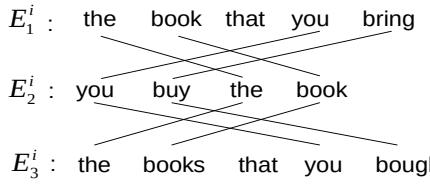


Figure 2. A constructed example of a sentence - "你買的書 (the book that you bought)" and its translations from three MT systems – E^i_1, E^i_2 and E^i_3, and word alignments between E^i_2 and E^i_1, and between E^i_2 and E^i_3, obtained through TERp.

We use a Chinese-to-English example in Figure 2 to illustrate the extraction process. The extracted hierarchical paraphrases to paraphrase EP^i_2 - "you[1] buy[2] the[3] book[4]" are shown in Table 1. Because of limited space, only part of the paraphrases, i.e, part of the rules of Q^i_2, are shown.

part of rules of Q^i_2		in $Q^i_{2,1}$?	in $Q^i_{2,2}$?	in $Q^i_{2,3}$?
X →< you[1] , you>	(a)	✔	✔	✔
X →< you[1] buy[2] , you buy >	(b)		✔	
X →< you[1] buy[2] , you bought >	(c)			✔
X →< you[1] buy[2] , you bring >	(d)	✔		
X →< book[4] , book >	(e)	✔	✔	
X →< book[4] , books >	(f)			✔
X →< the[3] book[4] , the book >	(g)	✔	✔	
X →< the[3] book[4] , the books >	(h)			✔
X →< you[1] buy[2] the[3] book[4] , the books that you bought >	(i)			✔
X →< X_1 the[3] book[4] , the books that X_1 >	(j)			✔
X →< you[1] buy[2] the[3] X_1 , the X_1 that you bought >	(k)			✔
X →< X_1 the[3] X_2 , the X_2 that X_1 >	(l)	✔		✔

Table 1. Part of extracted hierarchical paraphrases to paraphrase EP^i_2, i.e, part of the rules of Q^i_2.

Note that, in Table 1, the rules (j), (k) and (l) can be regarded as structural paraphrases, and they utilize existing information about word ordering present in the target hypotheses. Since rule (l) is included in both $Q^i_{2,1}$ and $Q^i_{2,3}$, we can say that rule (l) has more structural consensus than rule (j) and (k). And rule (l) also models the word reordering through reversing the order of X_1 and X_2. By the example, we can see the reason why our model is able to exploit structural consensus and also to utilize existing information about word ordering present in the target hypotheses.

3.2 Decoding

Given a certain target hypothesis - EP^i_h, and its set of hierarchical paraphrases - Q^i_h, the decoder aims to paraphrase EP^i_h using Q^i_h by performing a search for the single most probable derivation via the CKY algorithm with a Viterbi approximation. The derivation is the paraphrased result, i.e, the fusion result indicated in Figure 1. The single most probable derivation can be represented as

$$\arg\max_{\overline{E}^i_h} \log p(\overline{E}^i_h \mid EP^i_h) = \arg\max_{\overline{E}^i_h} \sum_{j=1}^{J} \left(\sum_{k=1}^{N} \lambda_k * f(q^{i,j}_h , k) \right) + \lambda_p * J + \lambda_l * \log(LM(\overline{E}^i_h)) + \lambda_w * length(\overline{E}^i_h)$$

$$f(q^{i,j}_h , k) = \begin{cases} 1 \text{ if } q^{i,j}_h \in Q^i_{h,k} \\ 0 \text{ otherwise} \end{cases}$$

[1] This means that words in a legal paraphrase are not aligned to words outside of the paraphrase, and should include at least one pair of words aligned with each other.

where $q_h^{i,j}$ is the jth paraphrase in Q_h^i used to generate $\overline{E}_h^i$, J is the number of paraphrases used to generate $\overline{E}_h^i$. N is the total number of MT systems. λ_k is the weight of MT system k, in charge of the system-weighted consensus. λ_p is phrase penalty. λ_l is the LM weight and λ_w is word penalty. All weights are trained discriminatively for Bleu score using Minimum Error Rate Training (MERT) procedure (Och 2004).

The ideal result of paraphrasing EP_2^i is shown in the following, which is supposed to be generated with a higher chance if, regardless of system weights. That is because of the use of the rules with higher degree of structural consensus, such as (l) and (e).

$< S_1 >$
$\Rightarrow < X_2, X_2 >$ using glue rule
$\Rightarrow < X_3 \text{ the}^3 X_4, \text{the } X_4 \text{ that } X_3 >$ using rule (l)
$\Rightarrow < \text{you}^1 \text{ buy}^2 \text{ the}^3 X_4, \text{the } X_4 \text{ that you bought} >$ using rule (c)
$\Rightarrow < \text{you}^1 \text{ buy}^2 \text{ the}^3 \text{ book}^4, \text{the book that you bought} >$ using rule (e)

4 Sentence-Level Selection-based Model

For a given sentence i and its M multiple fusion outputs - $\overline{E}_f^i$, $1 \le f \le M$ generated by the paraphrasing model or the lattice decoding model, the goal here is to select the best one among them, as shown in Figure 1 (For the case shown in the figure, M is $2N$). The idea is to compare system-weighted consensus among all fusion outputs and translations from all MT systems, and then select the one with the highest consensus. We adopt Minimum Bayes Risk (MBR) decoding (Kumar and Byrne, 2004; Sim et al., 2007) to serve our purpose and develop the following TER-based MBR:

$$\arg\max_f \log p(\overline{E}_f^i) = \arg\max_f \omega_l * \log(LM(\overline{E}_f^i)) + \sum_{m=1}^{M}\left(\omega_m * \log(1 - TER(\overline{E}_f^i, \overline{E}_m^i))\right) + \sum_{k=1}^{N}\left(\omega_k * \log(1 - TER(\overline{E}_f^i, E_k^i))\right)$$

where *TER* is Translation Tdit Ratio. ω_m is the fusion weight specific to a certain MT system and a certain fusion model, ω_k is the weight of MT system k and ω_l is the LM weight. All weights are trained discriminatively for Bleu score using MERT.

5 Experiments

Our experiments are conducted and reported on three datasets: The first dataset includes Chinese-English system translations and reference translations from DARPA GALE 2008 (GALE Chi-Eng). The second dataset includes Chinese-English system translations and reference translations and from NIST 2008 (NIST Chi-Eng). And the third dataset includes Arabic-English system translations and reference translations and from NIST 2008 (NIST Ara-Eng).

	MTSystem#	TuneSent#	TestSent#
GALE Chi-Eng	5	422	422
NIST Chi-Eng	5	524	788
NIST Ara-Eng	5	592	717

Table 2. Experimental setting

MT System	Approach	Bleu
nrc	phrase-based SMT	30.95
rwth-pbt-aml	phrase-based SMT + source reordering	31.83
rwth-pbt-jx	phrase-based SMT + word segmentation	31.78
rwth-pbt-sh	phrase-based SMT + source reordering + rescoring	32.63
sri-hpbt	hierarchical phrase-based SMT	32.00

Table 3: Techniques of top five MT of GALE Chi-Eng Dataset

Table 3 lists distinguishing machine translation approaches of top five MT of GALE Chi-Eng Dataset. And "rwth-pbt-sh" performs the best in Bleu score.

Two combination baselines are implemented for comparison: one is an implementation based on confusion network decoding, and the other is Lattice Decoding from (Ma and McKeown 2012), both of which are using TERp to obtain word alignments between a selected backbone hypothesis and other target hypotheses. The former uses these word alignments to construct a confusion network while the latter extracts phrases which are consistent with these word alignments to construct a lattice. For both baselines, backbone hypotheses are selected sentence by sentence based on system-weighted consensus among translation of all MT systems.

5.1 Results

In Table 4, CN represents confusion network; LD represents Lattice Decoding (Ma and McKeown 2012); PARA represents paraphrasing model proposed in this paper; Backbone_* represents that * is carried out on selected backbones, in contrast with the hybrid combination architecture. Arch_LD represents that only lattice decoding is carried out using hybrid combination architecture. Arch_PARA represents that only paraphrasing model is carried out using hybrid combination

architecture. Arch_LD_PARA represents that LD and PARA are both carried out using hybrid combination architecture, which is the example shown in Figure 2.

	GALE Chi-Eng	NIST Chi-Eng	NIST Ara-Eng
Best MT system	32.63	30.16	48.40
Backbone_CN (baseline)	33.04	31.21	48.56
Backbone_LD (baseline)	33.16	32.65	49.33
Backbone_PARA	33.09	32.59	49.46
Arch_LD	33.24	32.66	50.48
Arch_PARA	33.32	32.90	50.20
Arch_LD_PARA	33.72	33.42	50.44

Table 4. Experimental results in Bleu score

From Table 4, we can first observe that, for the three datasets, Backbone_PARA and Backbone_LD outperform Backbone_CN, which shows the advantage of using phrases over words in combination. However, Backbone_PARA does not show improvement over Backbone_LD. The reason could be that selected backbones already have a high level of quality and fewer words need to be replaced or re-ordered in contrast with other target hypotheses.

We find that Arch_PARA performs better than Backbone_PARA, and Arch_LD performs better than Backbone_LD. This observation supports our claim that it is beneficial to consider more diverse sets of plausible fused translations.

Arch_LD_PARA achieves the best performance among all techniques used in this paper. It not only supports our claim, but also brings a conclusion that the paraphrasing model and lattice decoding can compensate for the weaknesses of the other in our architecture.

Since the paraphrasing model uses hierarchical paraphrases to carry out the fusion, it is able to make a bigger degree of word-reordering or structural change on the input hypothesis in comparison with lattice decoding. We suppose that when more word-reordering and structural changes are needed, paraphrasing model can bring more benefits than lattice decoding. Because the quality of a given translation hypothesis is highly related to word reordering and structural change, it can be expected that when a poorly translated hypothesis is paraphrased, paraphrasing model can bring more benefits than lattice decoding. In order to obtain the evidence to support this hypothesis, we carried out the following experiment on NIST Chi-Eng Dataset.

For each MT system from the selected top 5 system A-E, we paraphrase its translations using the paraphrasing model and lattice decoding separately, aiming to compare the performances of the two models on each MT system. In other words, we do not first do backbone selection. Every MT system's translation is regarded as a backbone. The results are shown in Table 5.

	MT	Lattice Decoding	Paraphrasing model
Sys A	30.16	32.17	31.76
Sys B	30.06	31.93	31.72
Sys C	**28.15**	**30.66**	**31.00**
Sys D	29.94	31.86	31.46
Sys E	**29.52**	**31.52**	**31.92**

Table 5. The Bleu score of each MT system, the Bleu score of paraphrasing each MT system using lattice decoding and the Bleu score of paraphrasing each MT system using paraphrasing model.

Among the five MT systems, "Sys C" and "Sys D" perform poorer than the other three MT systems. When we paraphrase the two systems, we find that paraphrasing model outperforms lattice decoding. These results support our hypothesis that when more word-reordering and structural changes are needed, paraphrasing model can bring more benefits than lattice decoding.

6 Conclusion

We view MT combination as a paraphrasing process using a set of hierarchical paraphrases, in which more complicated paraphrasing phenomena are able to be modeled, such as phrasal and structural consensus. Existing information about word ordering present in the target hypotheses are also considered. The experimental results show that our approach can achieve a significant improvement over combination baselines.

There are many possibilities for enriching the simple framework. Many ideas from recent translation developments can be borrowed and modified for combination. Our future work aims to incorporate syntactic or semantic information into our paraphrasing framework.

Acknowledgments

We would like to thank the anonymous reviewers for their valuable comments and suggestions. This work is supported by the National Science Foundation via Grant No. 0910778 entitled "Richer Representations for Machine Translation".

Reference

A. V. Aho and J. D. Ullman. 1969. Syntax directed translations and the pushdown assembler. Journal of Com-puter and System Sciences, 3:37–56.

Yu Chen, Andreas Eisele, Christian Federmann, Eva Hasler, Michael Jellinghaus, and Silke Theison. 2007. Multi-engine machine translation with an open-source SMT decoder. In Proceedings of WMT07

Boxing Chen, Min Zhang and Aiti Aw. 2009a. A Comparative Study of Hypothesis Alignment and its Improvement for Machine Translation System Combination. In: Proceedings of ACL-IJCNLP. pp. 1067-1074. Singapore. August.

Yu Chen, Michael Jellinghaus, Andreas Eisele, Yi Zhang, Sabine Hunsicker, Silke Theison, Christian Federmann, Hans Uszkoreit. 2009b. Combining Multi-Engine Translations with Moses. In Proceedings of the Fourth Workshop on Statistical Machine Translation

David Chiang. Hierarchical phrase-based translation. 2007. Computational Linguistics, 33(2):201–228.

J. Devlin and S. Matsoukas. 2012. Trait-based hypothesis selection for machine translation. In Proc. of NAACL

Jinhua Du and Andy Way. 2010. Using TERp to Augment the System Combination for SMT. In Proceedings of the Ninth Conference of the Association for Machine Translation (AMTA2010)

Yang Feng, Yang Liu, Haitao Mi, Qun Liu, and Yajuan Lu. 2009 Lattice-based System Combination for Statistical Machine Translation. In Proceedings of ACL

Markus Freitag, Matthias Huck, and Hermann Ney. 2014. Jane: Open source machine translation system combination. In Conference of the European Chapter of the Association for Computational Linguistics, pages 29–32, Gothenburg, Sweden, April. Association for Computational Linguistics.

Kevin Gimpel, Dhruv Batra, Chris Dyer, and Gregory Shakhnarovich. 2013. A Systematic Exploration of Diversity in Machine Translation. In Proc. of EMNLP

Xiaodong He, Mei Yang, Jianfeng Gao, Patrick Nguyen, and Robert Moore. 2008. Indirect-hmm-based hypothesis alignment for computing outputs from machine translation systems. In Proceedings of EMNLP

Fei Huang and Kishore Papineni. 2007. Hierarchical System Combination for Machine Translation. In Proceed-ings of EMNLP-CoNLL

Damianos Karakos, Jason Eisner, Sanjeev Khudanpur, and Markus Dreyer. 2008. Machine translation system combination using ITG-based alignments. In Proceedings of ACL-HLT

S. Kumar and W. Byrne. 2004. Minimum Bayes-risk decoding for statistical machine translation. In Proceedings of HLT

Wei-Yun Ma and Kathleen McKeown. 2012. Phrase-level System Combination for Machine Translation Based on Target-to-Target Decoding. In Proceedings of the 10th Biennial Conference of the Association for Ma-chine Translation in the Americas (AMTA), San Diego, CA.

Gregor Leusch, Markus Freitag, and Hermann Ney. The RWTH System Combination System for WMT 2011. 2011. In Proceedings of the Sixth Workshop on Statistical Machine Translation

Evgeny Matusov, Nicola Ueffing, and Hermann Ney 2006. Computing consensus translation from multiple machine translation systems using enhanced hypotheses alignment. In Proceedings of EACL

E. Matusov, G. Leusch, R. E. Banchs, N. Bertoldi, D. Dechelotte, M. Federico, M. Kolss, Y. S. Lee, J. B. Marino, M. Paulik, S. Roukos, H. Schwenk, and H. Ney. 2008. System combination for machine translation of spoken and written language. IEEE Transactions on Audio, Speech and Language Processing, 16(7):1222–1237, September.

Sushant Narsale. JHU System Combination Scheme for WMT 2010. 2010. In Proceedings of the Fifth Workshop on Statistical Machine Translation

Franz Josef Och. 2004. Minimum Error Rate Training in Statistical Machine Translation. In Proceedings of ACL

Antti-Veikko I. Rosti, Spyros Matsoukas, and Richard Schwartz. 2007a. Improved word-level system combination for machine translation. In Proceedings of ACL

Antti-Veikko I. Rosti, Necip F. Ayan, Bing Xiang, Spyros Matsoukas, Richard Schwartz, and Bonnie J. Dorr. 2007b. Combining outputs from multiple machine translation systems. In Proceedings of NAACL-HLT

Khe Chai Sim, William J. Byrne, Mark J.F. Gales, Hichem Sahbi, and Phil C. Woodland. 2007. Consensus network decoding for statistical machine translation system combination. In Proceedings of ICASSP

M. Snover, B. Dorr, R. Schwartz, L. Micciulla, and J. Makhoul. 2006. A Study of Translation Edit Rate with Targeted Human Annotation. In Proceedings of AMTA.

M. Snover, N. Madnani, B. Dorr, and R. Schwartz. 2009. TER-Plus: Paraphrase, Semantic, and Alignment Enhancements to Translation Edit Rate. Machine Translation, 23(2–3):117–127.

Hierarchical Incremental Adaptation for Statistical Machine Translation

Joern Wuebker
Lilt Inc.
joern@lilt.com

Spence Green
Lilt Inc.
spence@lilt.com

John DeNero
Lilt Inc.
john@lilt.com

Abstract

We present an incremental adaptation approach for statistical machine translation that maintains a flexible hierarchical domain structure within a single consistent model. Both weights and rules are updated incrementally on a stream of post-edits. Our multi-level domain hierarchy allows the system to adapt simultaneously towards local context at different levels of granularity, including genres and individual documents. Our experiments show consistent improvements in translation quality from all components of our approach.

1 Introduction

Suggestions from a machine translation system can increase the speed and quality of professional human translators (Guerberof, 2009; Plitt and Masselot, 2010; Green et al., 2013a, *inter alia*). However, querying a single fixed model for all different documents fails to incorporate contextual information that can potentially improve suggestion quality. We describe a model architecture that adapts simultaneously to multiple genres and individual documents, so that translation suggestions are informed by two levels of contextual information.

Our primary technical contribution is a hierarchical adaptation technique for a post-editing scenario with incremental adaptation, in which users request translations of sentences in corpus order and provide corrected translations of each sentence back to the system (Ortiz-Martínez et al., 2010). Our learning approach resembles Hierarchical Bayesian Domain Adaptation (Finkel and Manning, 2009), but updates both the model weights and translation rules in real time based on these corrected translations (Mathur et al., 2013; Denkowski et al., 2014). Our adapted system can provide on-demand translations for any genre and document to which it has ever been exposed, using weights and rules for domains associated with each translation request.

Our weight adaptation is performed using a hierarchical extension to fast and adaptive online training (Green et al., 2013b), a technique based on AdaGrad (Duchi et al., 2011) and forward-backward splitting (Duchi and Singer, 2009) that can accurately set weights for both dense and sparse features (Green et al., 2014b). Rather than adjusting all weights based on each example, our extension adjusts offsets to a fixed baseline system. In this way, the system can adapt to multiple genres while preventing cross-genre contamination.

In large-scale experiments, we adapt a multi-genre baseline system to patents, lectures, and news articles. Our experiments show that sparse models, hierarchical updates, and rule adaptation all contribute consistent improvements. We observe quality gains in all genres, validating our hypothesis that document and genre context are important additional inputs to a machine translation system used for post-editing.

2 Background

The log-linear appoach to statistical machine translation models the predictive translation distribution $p(e|f; w)$ directly in log-linear form (Och and Ney, 2004):

$$p(e|f;w) = \sum_{\substack{r: \\ src(r)=f \\ tgt(r)=e}} \frac{1}{Z(f)} \exp\left[w^\top \phi(r;c)\right] \quad (1)$$

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1059–1065,
Lisbon, Portugal, 17-21 September 2015.

where $f \in \mathcal{F}$ is a string in the set of all source language strings $\mathcal{F}$, $e \in \mathcal{E}$ is a string in the set of all target language strings $\mathcal{E}$, r is a phrasal derivation with source and target projections $src(r)$ and $tgt(r)$, $w \in \mathbb{R}^d$ is the vector of model parameters, $\phi(\cdot) \in \mathbb{R}^d$ is a feature map computed using corpus c, and $Z(f)$ is an appropriate normalizing constant. During search, the maximum approximation is applied rather than summing over the derivations r.

Model. We extend a phrase-based system for which $\phi(r; c)$ includes 16 dense features:

- Two phrasal channel models and two lexical channel models (Koehn et al., 2003), the (log) count of the rule in the training corpus c, and an indicator for singleton rules in c.
- Six orientation models that score ordering configurations in r by their frequency in c (Koehn et al., 2007).
- A linear distortion penalty that promotes monotonic translation.
- An n-gram language model score, $p(e)$, which scores the target language projection of r using statistics from a monolingual corpus.
- Fixed-value phrase and word penalties.

The elements of $\phi(r; c)$ may also include sparse features that have non-zero values for only a subset of rules, but typically do not depend on c (Liang et al., 2006). In this paper, we use four types of sparse features: rule indicators, discriminative lexicalized reordering indicators, rule shape indicators and alignment features (Green et al., 2014b).

The model parameters w are chosen to maximize translation quality on a tuning set.

Adaptation. Domain adaptation for machine translation has improved quality using a variety of approaches, including data selection (Ceauşfu et al., 2011), regularized online learning (Simianer et al., 2012; Green et al., 2013b), and input classification (Xu et al., 2007; Banerjee et al., 2010; Wang et al., 2012) and has also been investigated for multi-domain tasks (Sennrich et al., 2013; Cui et al., 2013; Simianer and Riezler, 2013). Even without domain labels at either training or test time, multi-task learning can boost translation quality in a batch setting (Duh et al., 2010; Song et al., 2011).

Post-editing with incremental adaptation describes a particular mixed-initiative setting (Ortiz-Martínez et al., 2010; Hardt and Elming, 2010). For each f in a corpus, the machine generates a hypothesis e, then a human provides a corrected translation e^* to the machine. Observing e^* can affect both the model weights w and corpus c used for rule extraction and dense feature estimation.[1] To translate the ith sentence f_i, the system uses weights w_{i-1} and corpus c_{i-1}. The new corpus c_i results from adding (f_i, e_i^*) to c_{i-1}. For incremental adaptation, speed is essential, and so w_i is typically computed with a single online update from w_{i-1} using (f_i, e_i^*) as the tuning example.

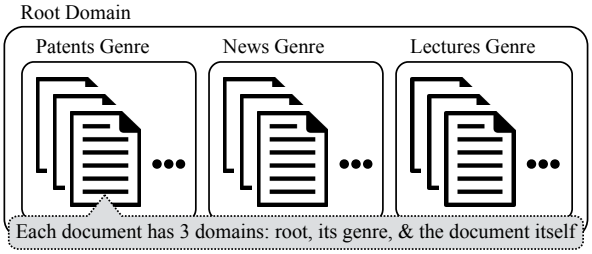


Figure 1: The weights used to translate a document in the patent genre include three domains.

To alleviate the need for human intervention in the experiment cycle, *simulated post-editing* (Hardt and Elming, 2010; Denkowski et al., 2014) replaces each e^* with a reference that is not a corrected variant of e. Thus, a standard test corpus can be used as an adaptation corpus. Prior work on online learning from post-edits has demonstrated the benefit of adjusting only c (Ortiz-Martínez et al., 2010; Hardt and Elming, 2010) and further benefit from adjusting both c and w (Mathur et al., 2013; Denkowski et al., 2014). Incremental adaptation of both c and the weights w for sparse features is reported to yield large quality gains by Wäschle et al. (2013).[2]

3 Hierarchical Incremental Adaptation

Our hierachical approach to incremental adaptation uses document and genre information to adapt appropriately to multiple contexts. We assume that each sentence f_i has a known set D_i of domains, which identify the genre and individual document origin of the sentence. This set could be extended to include topics, individual translators, etc.

Figure 1 shows the domains that we apply in experiments. All sentences in the baseline training corpus, the tuning corpus, and the adaptation corpus share a ROOT domain.

[1] For the purpose of our description, the corpus c is equivalent to the set of phrases and their scores in the rule table. We prefer this notation because it is consistent with our stream-based rule table, where the models are computed on-the-fly from the indexed training corpus c.

[2] Language model adaptation also has a rich literature, but it is beyond the scope of this paper.

Our adaptation is conceptually similar to hierarchical Bayesian domain adaptation (Finkel and Manning, 2009), but both weights and feature values depend on D_i, and we use L_1 regularization.

Weight Updates. Model tuning and adaptation are performed with AdaGrad, an online subgradient method with an *adaptive learning rate* that comes with good theoretical guarantees. AdaGrad makes the following update:

$$w_t = w_{t-1} - \eta \Sigma_t^{1/2} \nabla \ell_t(w_{t-1}) \quad (2)$$

$$\Sigma_t^{-1} = \Sigma_{t-1}^{-1} + \nabla \ell_t(w_{t-1}) \nabla \ell_t(w_{t-1})^\top$$

$$= \sum_{i=1}^{t} \nabla \ell_i(w_{i-1}) \nabla \ell_i(w_{i-1})^\top \quad (3)$$

The loss function ℓ reflects the pairwise ordering between hypotheses. For feature selection, we apply an L_1 penalty via forward-backward splitting (Duchi and Singer, 2009). η is the initial learning rate. See (Green et al., 2013b) for details.

Our adaptation schema is an extension of *frustratingly easy domain adaptation* (FEDA) (Daumé III, 2007) to multiple domains with different regularization parameters, similar to (Finkel and Manning, 2009). Each feature value is replicated for each domain. Let $\mathcal{D}$ denote the set of all domains present in the adaptation set. Given an original feature vector $\phi(r; c)$ for derivation r of sentence f_i with $D_i \subseteq \mathcal{D}$, the replicated feature vector includes $|\mathcal{D}|$ copies of $\phi(r; c)$, one for each $d \in |\mathcal{D}|$, such that

$$\phi_d(r; c) = \begin{cases} \phi(r; c), & d \in D_i \\ 0, & \text{otherwise.} \end{cases} \quad (4)$$

The weights of this replicated feature space are initialized using the weights w tuned for the baseline $\phi(r; c)$.

$$w_d = \begin{cases} w, & d \text{ is ROOT} \\ 0, & \text{otherwise.} \end{cases} \quad (5)$$

In this way, the ROOT domain corresponds to the unadapted baseline weights, denoted as Θ_* in (Finkel and Manning, 2009). The idea is that we simultaneously maintain a generic set of weights that applies to all domains as well as their domain-specific "offsets", describing how a domain differs from the generic case. Model updates during adaptation are performed according to the same procedure as tuning updates, but now in the replicated space.

Different from (Finkel and Manning, 2009), this generalized FEDA model does not restrict the domains to be strictly hierarchically structured. We could, for example, include a domain for each translator that crossed different genres. However, all of our experimental evaluations maintain a hierarchical domain structure, leaving more general setups to future work.

Rules and Feature Values. A derivation r of sentence f_i has features that are computed from the combination of the baseline training corpus c_0 and a genre-specific corpus that includes all sentence pairs from the tuning corpus as well as from the adaptation corpus (f_j, e_j^*) with $j < i$ sharing f_i's genre. We refer to this combined corpus as c_i. The tuning corpus is the same that is used for parameter tuning in the baseline system. The adaptation corpus is our test set. Note that in our evaluation, each sentence is translated before it is used for adaptation, so that there is no contamination of results.

In order to extend the model efficiently within a streaming data environment, we make use of a suffix-array implementation for our phrase table (Levenberg et al., 2010).

Rather than combining corpus counts across these different sources, separate rules extracted from the baseline corpus and the genre-specific corpus exist independently in the derivation space, and features of each are computed only with one corpus. In this configuration, a large amount of out-of-domain evidence from the baseline model will not dampen the feature value adaptation effects of adding new sentence pairs from the adaptation corpus. The genre-specific phrases are distinguished by an additional binary provenance feature.

In order to extract features from the genre-specific corpus, a word-level alignment must be computed for each (f_j, e_j^*). We force decode using the adapted translation model for f_j. In order to avoid decoding failures, we insert high-cost single-word translation rules that allow any word in f_j to align to any word in e_j^*.

Sparse Features. Applying a large number of sparse features would compromise responsiveness of our translation system and is thus a poor fit for real-time adaptive computer-assisted translation. However, features that can be learned on a single document are limited in number and can be discarded after the document has been processed. Therefore, document-level sparse features are a powerful means to fit our model to local context with a comparatively small impact on efficiency.

4 Experiments

We performed two sets of German→English experiments; Table 1 contains the results for both. Our first set of experiments was performed on the *PatTR* corpus (Wäschle and Riezler, 2012). We divided the corpus into training and development data by date and selected 2.4M parallel segments dated before 2000 from the "claims" section as bilingual training data, taking equal parts from each of the eight patent types A–H as classified by the Cooperative Patent Classification (CPC). From each type we further drew separate test sets and a single tune set, selecting documents with at least 10 segments and a maximum of 150 source words per segment, with around 2,100 sentences per test set and 400 sentences per type for the tune set. The "claims" section of this corpus is highly repetitive, which makes it ideal for observing the effects of incremental adaptation techniques.

To train the language and translation model we additionally leveraged all available bilingual and monolingual data provided for the *EMNLP 2015 Tenth Workshop on Machine Translation*[3]. The total size of the bitext used for rule extraction and feature estimation was 6.4M sentence pairs. We trained a standard 5-gram language model with modified Kneser-Ney smoothing (Kneser and Ney, 1995; Chen and Goodman, 1998) using the *KenLM* toolkit (Heafield et al., 2013) on 4 billion running words. The bitext was word-aligned with *mgiza* (Och and Ney, 2003), and we used the *phrasal* decoder (Green et al., 2014a) with standard German-English settings for experimentation.

Our second set of experiments was performed on a mixed-genre corpus containing lectures, patents, and news articles. The standard dev and test sets of the IWSLT 2014 shared task[4] were used for the *lecture* genre. Each document corresponded to an entire lecture. For the *news* genre, we used *newstest2012* for tuning, *newstest2013* for meta-parameter optimization, and *newstest2014* for testing. The tune set for the *patent* genre is identical to the first set of experiments, while the test set consists of the first 300 sentence pairs of each of the patent type specific test sets of the previous experiment. The documents in the news and patent genres contain around 20 segments on average.

Our evaluation proceeded in multiple stages. We first trained a set of background weights on the concatenated tune sets (*baseline*). Keeping these weights fixed, we performed an additional tuning run to estimate genre-level weights (+ *genre weights*).[5] In the patent-only setup, we used patent CPC type as genre. Next, we trained a genre-specific translation model for each genre by first feeding the tune set and then the test set into our incremental adaptation learning method as a continuous stream of simulated post edits (+ *genre TM*). After each sentence, we performed an update on the genre-specific weights. In separate experiments, we also included document-level weights as an additional domain (+ *doc. weights*) and included sparse features at the document level (+ *sparse features*).[6]

	PatTR	**heterogeneous data**		
	avg	lecture	news	patent
repetition rate	*27.80*	*5.46*	*3.13*	*27.42*
baseline	48.89	25.82	24.92	48.97
+ genre weights	49.05	26.64	25.12	49.39
+ genre TM	53.25	27.67	25.66	53.22
+ doc. weights	53.56	27.98	25.71	53.40
+ sparse features	54.53	28.09	25.89	54.30

Table 1: Results in uncased BLEU [%]. Each component is added on top of the previous line. All results in line + *genre TM* and below are statistically significant improvements over the baseline with 95% confidence. We also report the repetition rate of the test corpora as propsed by Bertoldi et al. (2013).

Table 1 demonstrates that each component of this approach offered consistent incremental quality gains, but with varying magnitudes. For the patent experiments we report the average over our eight test sets (A-H) due to lack of space, but total improvement varied from +4.92 to +6.46 BLEU. In the mixed-genre experiments, BLEU increased by +2.27 on *lectures*, +0.97 on *news*, and +5.33 on *patents*. On all tasks, we observed statistically significant improvements over the baseline (95% confidence level) in the + *genre TM*, + *doc. weights* and + *sparse features* experiments using bootstrap resampling (Koehn, 2004).

These results demonstrate the efficacy of hierarchical incremental adaptation, although we would like to stress that the patent data was selected specifically for its high level of repetitiveness, and the

[3] http://www.statmt.org/wmt15/

[4] http://workshop2014.iwslt.org/

[5] Learning rates and regularization weights for this step were selected on *newstest2013*.

[6] Learning rates and regularization weights for each genre were selected on the genre-specific tune sets.

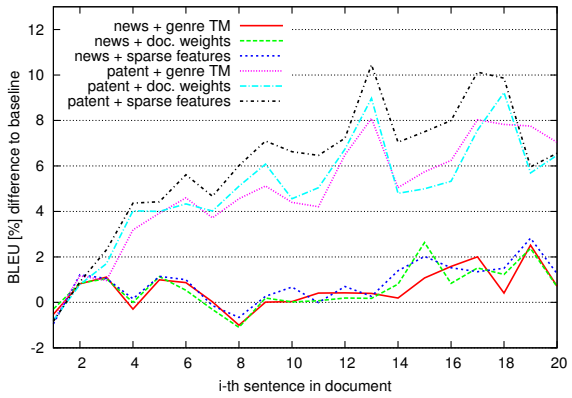


Figure 2: BLEU difference between *baseline + genre weights* and our incremental adaptation approach, computed on a single segment from each document according to their order, i.e. the first segment from each document, then the second segment from each document, etc.

large improvement in this genre would only be expected to arise in similarly structured domains. This property is quantified by the *repetition rate* measure (RR) (Bertoldi et al., 2013) reported in Table 1, which confirms the finding by Cettolo et al. (2014) that RR correlates with the effectiveness of adaptation.

Analysis. Figure 2 shows BLEU score differences to the *baseline + genre weights* system for different subsets of the news and patent test sets. Each point is computed by document slicing, i.e. on a single segment from each document. The rightmost data point is the BLEU score we obtain by evaluating on the 20th segment of each document, grouped into a pseudo-corpus. Note that this group does not correspond to any number in Table 1, which reports BLEU on the entire test sets. Thus, we evaluate on all sentences that have learned from exactly $(i-1)$ segments of the same document, with $i = 1, \ldots, 19$. Although the graph is naturally very noisy (each score is computed on roughly 150 segments), we can clearly see that incremental adaptation learns on the document level: on average, the improvement over the baseline increases when proceeding further into the document.

Decoding speed. In our real-time computer-assisted translation scenario, a certain translation speed is required to allow for responsive user interaction. Table 2 reports the speed in words per second on the lecture data. Adding a genre-specific translation model results in a speed reduction by a factor of 12.6 due to the additional (forced) decoding run and weight updates. Sparse features slows the system down further by a factor of 2.4. However, the largest part of the computation time incurs only when the user has finalized collaborative translation of one sentence and is busy reading the next source sentence. Further, the speed/quality tradeoff can be adjusted with pruning parameters.

	words / sec
baseline	177.6
+ genre weights	58.5
+ genre TM	14.1
+ doc. weights	9.8
+ sparse features	5.8

Table 2: Decoding speed on the lecture data.

5 Conclusion

We have presented an incremental learning approach for MT that maintains a flexible hierarchical domain structure within a single consistent model. In our experiments, we define a three-level hierarchy with a global root domain as well as genre- and document-level domains. Further, we perform incremental adaptation by training a genre-specific translation model on the stream of incoming post-edits and adding document-level sparse features that do not significantly compromise efficiency. Our results show consistent contributions from each level of adaptation across multiple genres.

References

Pratyush Banerjee, Jinhua Du, Baoli Li, Sudip Kr. Naskar, Andy Way, and Josef van Genabith. 2010. Combining multi-domain statistical machine translation models using automatic classifiers. In *Proceedings of the Association for Machine Translation in the Americas*, Denver, Colorado, October.

Nicola Bertoldi, Mauro Cettolo, and Marcello Federico. 2013. Cache-based online adaptation for machine translation enhanced computer assisted translation. In *Proceedings of the XIV Machine Translation Summit*, pages 36–42, Nice, France, September.

Alexandru Ceauşfu, John Tinsley, Jian Zhang, and Andy Way. 2011. Experiments on domain adaptation for patent machine translation in the PLuTO project. In Mikel L. Forcada, Heidi Depraetere, and Vincent Vandeghinste, editors, *Proceedings of the 15th International Conference of the European Association for Machine Translation (EAMT)*, pages 21–28, Leuven, Belgium.

Mauro Cettolo, Nicola Bertoldi, and Marcello Federico. 2014. The repetition rate of text as a predictor of the effectiveness of machine translation adaptation. In *Conference of the Association for Machine Translation in the Americas (AMTA)*, pages 166–179, Vancouver, Canada, October.

Stanley F. Chen and Joshuo Goodman. 1998. An Empirical Study of Smoothing Techniques for Language Modeling. Technical Report TR-10-98, Computer Science Group, Harvard University, Cambridge, MA, August.

Lei Cui, Xilun Chen, Dongdong Zhang, Shujie Liu, Mu Li, and Ming Zhou. 2013. Multi-domain adaptation for SMT using multi-task learning. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1055–1065, Seattle, Washington, USA, October.

Hal Daumé III. 2007. Frustratingly easy domain adaptation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 256–263, Prague, Czech Republic, June.

Michael Denkowski, Chris Dyer, and Alon Lavie. 2014. Learning from post-editing: Online model adaptation for statistical machine translation. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 395–404, Gothenburg, Sweden, April.

John Duchi and Yoram Singer. 2009. Efficient online and batch learning using forward backward splitting. *Journal of Machine Learning Research*, 10:2899–2934, December.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159, July.

Kevin Duh, Katsuhito Sudoh, Hajime Tsukada, Hideki Isozaki, and Masaaki Nagata. 2010. N-best reranking by multitask learning. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, pages 375–383, Uppsala, Sweden, July.

Jenny Rose Finkel and Christopher D. Manning. 2009. Hierarchical bayesian domain adaptation. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 602–610, Boulder, Colorado.

Spence Green, Sida Wang, Daniel Cer, and Christopher D. Manning. 2013a. The efficacy of human post-editing for language translation. In *ACM CHI Conference on Human Factors in Computing Systems*, Paris, France, April.

Spence Green, Sida Wang, Daniel Cer, and Christopher D. Manning. 2013b. Fast and adaptive online training of feature-rich translation models. In *51st Annual Meeting of the Association for Computational Linguistics*, pages 311–321, Sofia, Bulgaria, August.

Spence Green, Daniel Cer, , and Christopher D. Manning. 2014a. Phrasal: A Toolkit for New Directions in Statistical Machine Translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 114–121, Baltimore, Maryland USA, June.

Spence Green, Daniel Cer, and Christopher D. Manning. 2014b. An empirical comparison of features and tuning for phrase-based machine translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 466–476, Baltimore, Maryland, USA, June.

A. Guerberof. 2009. Productivity and quality in the post-editing of outputs from translation memories and machine translation. *International Journal of Localization*, 7(1):11–21.

Daniel Hardt and Jakob Elming. 2010. Incremental re-training for post-editing SMT. In *Proceedings of the Ninth Conference of the Association for Machine Translation in the Americas*, Denver, Colorado, October.

Kenneth Heafield, Ivan Pouzyrevsky, Jonathan H. Clark, and Philipp Koehn. 2013. Scalable modified Kneser-Ney language model estimation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 690–696, Sofia, Bulgaria, August.

Reinerd Kneser and Hermann Ney. 1995. Improved backing-off for M-gram language modeling. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 181–184, May.

Philipp Koehn, Franz Joseph Och, and Daniel Marcu. 2003. Statistical Phrase-Based Translation. In *Proc. of the Human Language Technology Conf. (HLT-NAACL)*, pages 127–133, Edmonton, Canada, May/June.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantine, and Evan Herbst. 2007. Moses: Open Source Toolkit for Statistical Machine Translation. pages 177–180, Prague, Czech Republic, June.

Philipp Koehn. 2004. Statistical Significance Tests for Machine Translation Evaluation. In *Proc. of the Conf. on Empirical Methods for Natural Language Processing (EMNLP)*, pages 388–395, Barcelona, Spain, July.

Abby Levenberg, Chris Callison-Burch, and Miles Osborne. 2010. Stream-based translation models for statistical machine translation. In *Human Language Technologies: The 2010 Annual Conference of the*

North American Chapter of the Association for Computational Linguistics, pages 394–402, Los Angeles, California, June.

Percy Liang, Alexandre Buchard-Côté, Dan Klein, and Ben Taskar. 2006. An End-to-End Discriminative Approach to Machine Translation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 761–768, Sydney, Australia.

Prashant Mathur, Cettolo Mauro, and Marcello Federico. 2013. Online learning approaches in computer assisted translation. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 301–308, Sofia, Bulgaria, August.

Franz Josef Och and Hermann Ney. 2003. A Systematic Comparison of Various Statistical Alignment Models. *Computational Linguistics*, 29(1):19–51, March.

F. J. Och and H. Ney. 2004. The Alignment Template Approach to Statistical Machine Translation. *Computational Linguistics,*, 30(4):417–450.

Daniel Ortiz-Martínez, Ismael García-Varea, and Francisco Casacuberta. 2010. Online learning for interactive statistical machine translation. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 546–554, Los Angeles, California, June.

M. Plitt and F. Masselot. 2010. A productivity test of statistical machine translation post-editing in a typical localisation context. *The Prague Bulletin of Mathematical Linguistics*, 93:7–16.

Rico Sennrich, Holger Schwenk, and Walid Aransa. 2013. A multi-domain translation model framework for statistical machine translation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 832–840, Sofia, Bulgaria, August.

Patrick Simianer and Stefan Riezler. 2013. Multi-task learning for improved discriminative training in SMT. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 292–300, Sofia, Bulgaria, August.

Patrick Simianer, Stefan Riezler, and Chris Dyer. 2012. Joint Feature Selection in Distributed Stochastic Learning for Large-Scale Discriminative Training in SMT. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, pages 11–21, Jeju Island, Korea, July.

Linfeng Song, Haitao Mi, Yajuan Lü, and Qun Liu. 2011. Bagging-based system combination for domain adaption. In *Proceedings of the 13th Machine Translation Summit (MT Summit XIII)*, pages 293–299, Xiamen, China.

Wei Wang, Klaus Macherey, Wolfgang Macherey, Franz Och, and Peng Xu. 2012. Improved domain adaptation for statistical machine translation. In *Proceedings of the Tenth Conference of the Association for Machine Translation in the Americas (AMTA)*, San Diego, California.

Katharina Wäschle and Stefan Riezler. 2012. Analyzing Parallelism and Domain Similarities in the MAREC Patent Corpus. *Multidisciplinary Information Retrieval*, pages 12–27.

Katharina Wäschle, Patrick Simianer, Nicola Bertoldi, Stefan Riezler, and Marcello Federico. 2013. Generative and Discriminative Methods for Online Adaptation in SMT. In *Proceedings of Machine Translation Summit XIV*, Nice, France.

Jia Xu, Yonggang Deng, Yuqing Gao, and Hermann Ney. 2007. Domain dependent statistical machine translation. In *Proceedings of the MT Summit*, pages 515–520, Copenhagen, Denmark, September.

ReVal: A Simple and Effective Machine Translation Evaluation Metric Based on Recurrent Neural Networks

Rohit Gupta[1], **Constantin Orăsan**[1], **Josef van Genabith**[2]
[1]Research Group in Computational Linguistics, University of Wolverhampton, UK
[2]Saarland University and German Research Center for Artificial Intelligence (DFKI), Germany
{r.gupta, c.orasan}@wlv.ac.uk
josef.van_genabith@dfki.de

Abstract

Many state-of-the-art Machine Translation (MT) evaluation metrics are complex, involve extensive external resources (e.g. for paraphrasing) and require tuning to achieve best results. We present a simple alternative approach based on dense vector spaces and recurrent neural networks (RNNs), in particular Long Short Term Memory (LSTM) networks. For WMT-14, our new metric scores best for two out of five language pairs, and overall best and second best on all language pairs, using Spearman and Pearson correlation, respectively. We also show how training data is computed automatically from WMT ranks data.

1 Introduction

Deep learning approaches have turned out to be successful in many NLP applications such as paraphrasing (Mikolov et al., 2013b; Socher et al., 2011), sentiment analysis (Socher et al., 2013b), parsing (Socher et al., 2013a) and machine translation (Mikolov et al., 2013a). While dense vector space representations such as those obtained through Deep Neural Networks (DNNs) or RNNs are able to capture semantic similarity for words (Mikolov et al., 2013b), segments (Socher et al., 2011) and documents (Le and Mikolov, 2014) naturally, traditional MT evaluation metrics can only achieve this using resources like WordNet and paraphrase databases. This paper presents a novel, efficient and compact MT evaluation measure based on RNNs. Our metric is simple in the sense that it does not require much machinery and resources apart from the dense word vectors. This cannot be said of most of the state-of-the-art MT evaluation metrics, which tend to be complex and require extensive feature engineering. Our metric is based on RNNs and particularly on Tree Long Short Term Memory (Tree-LSTM) networks (Tai et al., 2015). LSTM (Hochreiter and Schmidhuber, 1997) is a sequence learning technique which uses a memory cell to preserve a state over a long period of time. This enables distributed representations of sentences using distributed representations of words. Tree-LSTM is a recent approach, which is an extension of the simple LSTM framework (Zaremba and Sutskever, 2014). To provide the required training data, we also show how to automatically convert the WMT-13 (Bojar et al., 2013) human evaluation rankings into similarity scores between the reference and the translation. Our metric including training data is available at https://github.com/rohitguptacs/ReVal.

2 Related Work

Many metrics have been proposed for MT evaluation. Earlier popular metrics are based on n-gram counts (e.g. BLEU (Papineni et al., 2002) and NIST (Doddington, 2002)) or word error rate. Other popular metrics like METEOR (Denkowski and Lavie, 2014) and TERp (Snover et al., 2008) also use external resources like WordNet and paraphrase databases. However, system-level correlation with human judgements for these metrics remains below 0.90 Pearson correlation coefficient (as per WMT-14 results, BLEU-0.888, NIST-0.867, METEOR-0.829, TER-0.826, WER-0.821).

Recent best-performing metrics in the WMT-14 metric shared task (Machácek and Bojar, 2014) used a combination of different metrics. The top performing system DISKOTK-PARTY-TUNED (Joty et al., 2014) in the WMT-14 task uses five different discourse metrics and twelve different metrics from the ASIYA MT evaluation toolkit (Giménez and Màrquez, 2010). The metric computes the number of common sub-trees between a reference and a translation using a convolution

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1066–1072,
Lisbon, Portugal, 17-21 September 2015.

tree kernel (Collins and Duffy, 2001). The basic version of the metric does not perform well but in combination with the other 12 metrics from the ASIYA toolkit obtained the best results for the WMT-14 metric shared task. Another top performing metric LAYERED (Gautam and Bhattacharyya, 2014), uses linear interpolation of different metrics. LAYERED uses BLEU and TER to capture lexical similarity, Hamming score and Kendall Tau Distance (Birch and Osborne, 2011) to identify syntactic similarity, and dependency parsing (De Marneffe et al., 2006) and the Universal Networking Language[1] for semantic similarity. Recently, Guzmán et al. (2015) presented a metric based on word embeddings and neural networks. However, this metric is limited to ranking the available systems and does not provide an absolute score.

In this paper we propose a compact MT evaluation metric. We hypothesize that our model learns different notions of similarity (which other metrics tend to capture using different metrics) using input, output and forget gates of an LSTM architecture.

3 LSTMs and Tree-LSTMs

Recurrent Neural Networks allow processing of arbitrary length sequences, but early RNNs had the problem of vanishing and exploding gradients (Bengio et al., 1994). RNNs with LSTM (Hochreiter and Schmidhuber, 1997) tackle this problem by introducing a memory cell composed of a unit called constant error carousel (CEC) with multiplicative input and output gate units. Input gates protect against irrelevant inputs and output gates against current irrelevant memory contents. This architecture is capable of capturing important pieces of information seen in a bigger context. Tree-LSTM is an extension of simple LSTM. A typical LSTM processes the information sequentially whereas Tree-LSTM architectures enable sentence representation through a syntactic structure. Equation (1) represents the composition of a hidden state vector for an LSTM architecture. For a simple LSTM, c_t represents the memory cell and o_t the output gate at time step t in a sequence. For Tree-LSTM, c_t represents the memory cell and o_t represents the output gate corresponding to node t in a tree. The structural processing of Tree-LSTM makes it better suited to representing sentences. For example, dependency tree structure captures syntactic features and model parameters the importance of words (content vs. function words).

$$h_t = o_t \odot \tanh c_t \tag{1}$$

Figure 1 shows simple LSTM and Tree-LSTM architectures.

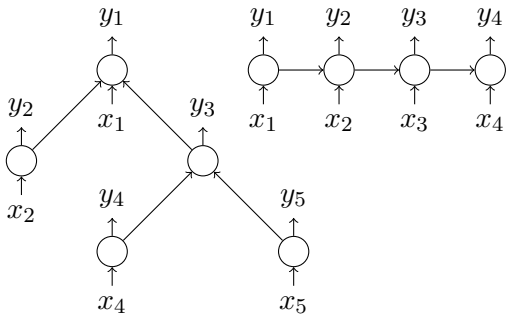


Figure 1: Tree-LSTM (left) and simple LSTM (right)

4 Evaluation Metric

We represent both the reference (h_{ref}) and the translation (h_{tra}) using an LSTM and predict the similarity score $\hat{y}$ based on a neural network which considers both distance and angle between h_{ref} and h_{tra}:

$$\begin{aligned} h_\times &= h_{ref} \odot h_{tra} \\ h_+ &= |h_{ref} - h_{tra}| \\ h_s &= \sigma\left(W^{(\times)} h_\times + W^{(+)} h_+ + b^{(h)}\right) \\ \hat{p}_\theta &= \text{softmax}\left(W^{(p)} h_s + b^{(p)}\right) \\ \hat{y} &= r^T \hat{p}_\theta \end{aligned} \tag{2}$$

where, σ is a sigmoid function, $\hat{p}_\theta$ is the estimated probability distribution vector and $r^T = [1\ 2 ... K]$. The cost function $J(\theta)$ is defined over probability distributions p and $\hat{p}_\theta$ using regularised Kullback-Leibler (KL) divergence.

$$J(\theta) = \frac{1}{n} \sum_{i=1}^{n} \text{KL}\left(p^{(i)} \middle|\middle| \hat{p}_\theta^{(i)}\right) + \frac{\lambda}{2} ||\theta||_2^2 \tag{3}$$

In Equation 3, i represents the index of each training pair, n is the number of training pairs and p is the sparse target distribution such that $y = r^T p$ is defined as follows:

$$p_j = \begin{cases} y - \lfloor y \rfloor, & j = \lfloor y \rfloor + 1 \\ \lfloor y \rfloor - y + 1, & j = \lfloor y \rfloor \\ 0 & \text{otherwise} \end{cases}$$

[1] http://www.undl.org/unlsys/unl/unl2005/UW.htm

for $1 \leq j \leq K$, where, $y \in [1, K]$ is the similarity score of a training pair. For example, for $y = 2.7$, $p^T = [0\ 0.3\ 0.7\ 0\ 0]$. In our case, the similarity score y is a value between 1 and 5.

For our work, we use *glove* word vectors (Pennington et al., 2014) and the simple LSTM, the dependency Tree-LSTM and neural network implementations by Tai et al. (2015). [2] The system uses the scientific computing framework Torch[3]. Training is performed on the data computed in Section 5. The system uses a mini batch size of 25 with learning rate 0.05 and regularization strength 0.0001. The compositional parameters for our Tree-LSTM systems with memory dimensions 150 and 300 are 203,400 and 541,800, respectively. The training is performed for 10 epochs. System-level scores are computed by aggregating and normalising segment-level scores.

5 Computing Similarity Scores from WMT Rankings

As we do not have access to any dataset which provides scores to segments on the basis of translation quality, we used the WMT-13 ranks corpus to automatically derive training data. This corpus is a by-product of the manual systems evaluation carried out in the WMT-13 evaluation. In the evaluation, the annotators are presented with a source segment, the output of five systems and a reference translation. The annotators are given the following instructions: "*You are shown a source sentence followed by several candidate translations. Your task is to rank the translations from best to worst (ties are allowed)*". Using the WMT-13 ranked corpus, we derived a corpus where the reference and corresponding translations are assigned similarity scores. The fact that *ties are allowed* makes it more suitable to generate similarity scores. If all translations are bad, annotators can mark all as rank 5 and if all translations are accurate, annotators can mark all as rank 1. The selection of the WMT-13 corpus over other WMT workshops is motivated by the fact that it is the largest among them. It contains ten times more ranks than WMT-12 and three to four times more than WMT-14. This also makes it possible to obtain enough reference translation pairs which are evaluated several times.

[2]The adapted code for MT evaluation scenarios is available at https://github.com/rohitguptacs/ReVal.

[3]http://torch.ch

Our hypothesis is that if a translation is given a certain rank many times, this reflects its similarity score with the reference. A better ranked translation among many systems will be close to the reference whereas a worse ranked translation among many systems will be dissimilar from the reference. To remove noisy pairs, we collect reference translation pairs below a certain variance only. We determined appropriate variance values using Algorithm 1 below for $n = 3, 4, 5, 6, 7$ and ≥ 8, separately. The computed variance values are given in Table 1.

n	3	4	5	6	7	≥ 8
Var	0.65	1.0	1.2	1.2	1.3	0.85

Table 1: Variances computed using Algorithm 1

Algorithm 1 Variance Computation

```
procedure GETVARIANCE(judgements)
    V, v ← −1, 0.25                          ▷ Initialise N
    for v ≤ max do
        prs ← pairs with variance below v
        score ← kendall(prs, judgements)
        if score ≥ 0.78 then
            V ← v
            v ← v + 0.05
        else
            break
    Return V                                 ▷ Return variance
```

In Algorithm 1, the $kendall$ function calculates Kendall tau correlation using the WMT-13 human judgements. We select a set for which the correlation coefficient is greater than 0.78.[4] The correlation is computed using the annotations for which scores are available in the corpus (prs). In other words, the corpus acts as a scoring function for the available reference translation pairs, which gives a similarity score between a reference and a translation. We selected pairs below the variance values obtained for $n = 4, 5, 6, 7$ and ≥ 8. Finally, all the pairs are merged to obtain a set (L). Apart from this set, we created three other sets for our experiments. The last two also use the SICK data (Marelli et al., 2014) which was developed for evaluating semantic similarity. All four sets are described below:

L: contains the set generated by selecting the pairs ranked *four or more times* and filtering the segments based on the variance

LNF: contains the set generated by selecting the pairs ranked *four or more times* without any filtering depending on the variance

[4]The score was decided so that we obtain around 10K pairs which are annotated at least four times.

L+Sick: Added 4500 sentence pairs from the SICK training set to Set L in the training set and 500 pairs in the development set.

XL+Sick: Added also the pairs ranked *three times* to Set L+Sick.

	Train	Dev	Test
L	9559	1000	1000
LNF	17855	1000	1000
L+Sick	14059	1500	1000
XL+Sick	21356	1500	1000

Table 2: Derived Corpus statistics

Table 2 shows the number of pairs extracted for each set to train our LSTM based models.[5]

6 Results

We evaluate our approach trained on the four different datasets obtained from WMT-13 (as given in Table 2) on WMT-14. Table 3 shows system-level Pearson correlation obtained on different language pairs as well as average Pearson correlation (PAvg) over all language pairs. The last column of the table also shows average Spearman correlation (SAvg). The 95% confidence level scores are obtained using bootstrap resampling as used in the WMT-2014 metric task evaluation. The scores in bold show best scores overall and the scores in bold italic show best scores in our variants.

In Table 3 and Table 4, the first section (L+Sick(lstm)) shows the results obtained using simple LSTM (layer 1, hidden dimension 50, memory dimension 150, compositional parameters 203400). The second section shows the scores of our Tree-LSTM metric trained on different training sets and dimensions. Dimensions are shown in brackets, e.g L(50,150) shows the results on set 'L' with the hidden dimension 50 and the memory dimension 150. L+Sick(mix) shows results of combining the two systems: L+Sick(50,150) and L+Sick(100,150). For the sentences longer than 20 words, the system uses scores of L+Sick(100,150) and scores of L+Sick(50,150) for the rest. The third section shows the best three overall systems from the WMT-14 metric task. The fourth section in Table 3 shows the systems from the WMT-14 task which obtained best results for certain languages but do not preform well overall. The last section in Tables 3 and 4 shows systems implementing BLEU (or variants for the segment level) and METEOR in the WMT-14 metric task.

Tables 3 and 4 contain a deluge of evaluation data, mainly to explore the effect of different training data and model parameter settings for our models. The main messages can be summarised as follows: 1. Tree LSTM models significantly outperform the LSTM model (L+Sick(lstm) and L+Sick(50,150) have the same data and parameter settings). 2. For Tree-LSTM models different parameter settings have only a minor impact on performance (in fact only for a few language pairs (e.g. hi-en at system-level, L+Sick(100, 300) and L+Sick(100,150)) results are statistically significantly different). This is reassuring as it indicates that the metric is not overly sensitive to extensive and delicate parameter tuning. 3. For the system level evaluation Tree-LSTM models are fully competitive with the best of the current complex models that combine many different metrics, substantial external resources and may require a significant amount of feature engineering and tuning. 4. For the segment level evaluation our metric outperforms BLEU based approaches and the other three systems[6] but lags behind some other approaches. We investigate this further below.

Tables 3 and 4 show that set L is able to obtain similar results compared to set LNF even though we filter out almost half of the pairs. Table 3 shows that for L+Sick(50, 150) and L+Sick(mix), we obtained an average second best Pearson correlation and best Spearman correlation coefficient. We also obtained better results for the Russian-English and Czech-English language pairs compared to any other systems in the WMT-14 task.

We also evaluate our setting L-Sick(50,150) on the WMT-12 task dataset. Our metric performs best for two out of four language pairs and best overall at the system level with 0.950 and 0.926 Pearson and Spearman correlation coefficient, respectively. At the segment level, we obtained 0.222 Kendall tau correlation which was better than seven out of the total ten metrics in the WMT-12 task.

One of the reasons for the difference in segment-level and system-level correlations is that Kendall Tau segment-level correlation is calcu-

[5]For testing our approach we use WMT-12 and WMT-14 rankings instead of the test sets in this table.

[6]These three systems are not given in this paper. Please refer (Machácek and Bojar, 2014) for results of these systems.

Test	cs-en	de-en	fr-en	hi-en	ru-en	PAvg	SAvg
L+Sick(lstm)	.922 ± .051	.882 ± .028	.974 ± .009	.898 ± .011	.863 ± .023	.908 ± .024	.872 ± .060
LNF(50,150)	.972 ± .032	.900 ± .026	.974 ± .009	.900 ± .011	.882 ± .021	.925 ± .020	.913 ± .045
L(50,150)	.988 ± .022	.897 ± .027	.978 ± .008	.905 ± .010	.875 ± .022	.929 ± .018	.904 ± .042
L+Sick(50,150)	.993 ± .017	.904 ± .025	.978 ± .008	.908 ± .010	.881 ± .022	.933 ± .016	.915 ± .042
L+Sick(100,300)	.993 ± .018	.907 ± .025	.973 ± .009	.866 ± .012	**.890** ± .020	.926 ± .017	.902 ± .050
XL+Sick(100,300)	.913 ± .054	***.917*** ± .024	.978 ± .008	.904 ± .010	.884 ± .022	.919 ± .024	.889 ± .055
L+Sick(100,150)	**.994** ± .016	.911 ± .025	.975 ± .009	***.923*** ± .010	.870 ± .022	***.935*** ± .016	.904 ± .049
L+Sick(mix)	**.994** ± .017	.906 ± .025	***.979*** ± .008	.918 ± .010	.881 ± .022	***.935*** ± .016	**.919** ± .045
DISCOTK-PARTY-TUNED	.975 ± .031	**.943** ± .020	.977 ± .009	.956 ± .007	.870 ± .022	**.944** ± .018	.912 ± .043
LAYERED	.941 ± .045	.893 ± .026	.973 ± .009	**.976** ± .006	.854 ± .023	.927 ± .022	.894 ± .047
DISCOTK-PARTY	.983 ± .025	.921 ± .024	.970 ± .010	.862 ± .015	.856 ± .023	.918 ± .019	.856 ± .046
REDSYS	.989 ± .021	.898 ± .026	**.981** ± .008	.676 ± .022	.814 ± .026	.872 ± .021	.786 ± .047
REDSYSSENT	.993 ± .018	.910 ± .024	.980 ± .008	.644 ± .023	.807 ± .027	.867 ± .020	.771 ± .043
BLEU	.909 ± 0.54	.832 ± .034	.952 ± .012	.956 ± .007	.789 ± .027	.888 ± .027	.833 ± .058
METEOR	.980 ± .029	.927 ± .022	.975 ± .009	.457 ± .027	.805 ± .026	.829 ± .023	.788 ± .046

Table 3: Results: System-Level Correlations on WMT-14

Test	cs-en	de-en	fr-en	hi-en	ru-en	Average	Avg wmt12
L+Sick(lstm)	.204 ± .015	.232 ± .014	.289 ± .013	.319 ± .013	.236 ± .012	.256 ± .013	.254 ± .013
NFL(50,150)	.228 ± .015	***.288*** ± .014	.318 ± .014	.341 ± .014	.271 ± .012	.289 ± .014	.287 ± .014
L(50,150)	.225 ± .015	.272 ± .014	.328 ± .013	.346 ± .013	.280 ± .011	.290 ± .013	.287 ± .013
L+Sick(50,150)	.243 ± .016	.274 ± .013	.333 ± .013	.360 ± .014	.278 ± .011	.298 ± .013	.295 ± .014
L+Sick(100,300)	.233 ± .014	.286 ± .014	.343 ± .014	.358 ± .013	***.281*** ± .011	.300 ± .013	.297 ± .013
XL+Sick(100,300)	***.252*** ± .014	.279 ± .014	***.347*** ± .013	.367 ± .013	.274 ± .011	***.304*** ± .013	***.301*** ± .013
L+Sick(100,150)	.243 ± .016	.274 ± .014	.329 ± .013	***.368*** ± .012	.276 ± .011	.298 ± .013	.295 ± .013
L+Sick(mix)	.243 ± .016	.276 ± .013	.338 ± .013	.358 ± .013	.273 ± .011	.298 ± .013	.295 ± .013
DISCOTK-PARTY-TUNED	**.328** ± .014	**.380** ± .014	**.433** ± .013	.434 ± .013	**.355** ± .010	**.386** ± .013	**.386** ± .013
BEER	.284 ± .015	.337 ± .014	.417 ± .013	**.438** ± .014	.333 ± .011	.362 ± .013	.358 ± .013
REDCOMBSENT	.284 ± .015	.338 ± .013	.406 ± .012	.417 ± .014	.336 ± .011	.356 ± .013	.346 ± .013
METEOR	.282 ± .015	.334 ± .014	.406 ± .012	..420 ± .013	.329 ± .010	.354 ± .013	.341 ± .013
BLEU_NRC	.226 ± .014	.272 ± .014	.382 ± .013	.322 ± .013	.269 ± .011	.294 ± .013	.267 ± .013
SENTBLEU	.213 ± .016	.271 ± .014	.378 ± .013	.300 ± .013	.263 ± .011	.285 ± .013	.258 ± .014

Table 4: Results: Segment-Level Correlations on WMT-14

lated based on rankings and does not consider the amount of difference between scores. Here is an example similar to that given in (Hopkins and May, 2013). Suppose four systems produce the translations T0, T1, T2 and T3. Suppose we have two metrics M1 and M2 and they produce scores and rankings as follows. GS represents the correct ranking and scores; Scores are in a scale [0, 1] with a higher score indicating a better translation:

M1: T0 (0.10), T3 (0.71), T1 (0.72), T2 (0.73)

M2: T1 (0.71), T0 (0.72), T2 (0.73), T3 (0.74)

GS: T0 (0.10), T1 (0.71), T2 (0.72), T3 (0.73)

Certainly, M1 produces better scores and ranking than M2. But, Kendall Tau segment-level correlation is higher for M2. (There are four concordant pairs in the M1 rank and five in the M2 rank.) Therefore, if a metric does not scale well as per the quality of translations, it may still obtain a good Kendall Tau segment-level correlation and a better metric may end up getting a low correlation. Another reason for the discrepancy between segment and system-level scores may be a low agreement on annotations. For the WMT-14 dataset, inter-annotator and intra-annotator agreement were 0.367 and 0.522. These problems should not occur with Pearson correlation at the system level because system-level scores are calculated using more sophisticated approaches (Koehn, 2012; Hopkins and May, 2013; Sakaguchi et al., 2014). For example, Hopkins and May (2013) model the differences among annotators by adding random Gaussian noise.

7 Conclusion

We conclude that our dense-vector-space-based ReVal metric is simple, elegant and effective with state-of-the-art results. ReVal is fully competitive with the best of the current complex alternative approaches that involve system combination, extensive external resources, feature engineering and tuning.

Acknowledgement

The research leading to these results has received funding from the People Programme (Marie Curie Actions) of the European Unions Seventh Framework Programme FP7/2007-2013/ under REA grant agreement no. 317471 and the EC- funded project QT21 under Horizon 2020, ICT 17, grant agreement no. 645452.

References

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166.

Alexandra Birch and Miles Osborne. 2011. Reordering metrics for MT. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 1027–1035. Association for Computational Linguistics.

Ondřej Bojar, Christian Buck, Chris Callison-Burch, Christian Federmann, Barry Haddow, Philipp Koehn, Christof Monz, Matt Post, Radu Soricut, and Lucia Specia. 2013. Findings of the 2013 Workshop on Statistical Machine Translation. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 1–44, Sofia, Bulgaria, August. Association for Computational Linguistics.

Michael Collins and Nigel Duffy. 2001. Convolution kernels for natural language. In *Advances in Neural Information Processing Systems*, pages 625–632.

Marie-Catherine De Marneffe, Bill MacCartney, Christopher D Manning, et al. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC*, volume 6, pages 449–454.

Michael Denkowski and Alon Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the EACL 2014 Workshop on Statistical Machine Translation*.

George Doddington. 2002. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Proceedings of the second international conference on Human Language Technology Research*, pages 138–145. Morgan Kaufmann Publishers Inc.

Shubham Gautam and Pushpak Bhattacharyya. 2014. Layered: Metric for machine translation evaluation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*.

Jesús Giménez and Lluís Màrquez. 2010. Linguistic measures for automatic machine translation evaluation. *Machine Translation*, 24(3-4):209–240.

Francisco Guzmán, Shafiq Joty, Lluís Màrquez, and Preslav Nakov. 2015. Pairwise neural machine translation evaluation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 805–814, Beijing, China. Association for Computational Linguistics.

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

Mark Hopkins and Jonathan May. 2013. Models of translation competitions. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1416–1424, Sofia, Bulgaria.

Shafiq Joty, Francisco Guzmán, Lluís Màrquez, and Preslav Nakov. 2014. DiscoTK: Using Discourse Structure for Machine Translation Evaluation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*.

Philipp Koehn. 2012. Simulating human judgment in machine translation evaluation campaigns. In *Proceedings of the Ninth International Workshop on Spoken Language Translation*, pages 179–184, Hong Kong.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of The 31st International Conference on Machine Learning*, pages 1188–1196.

Matouš Machácek and Ondrej Bojar. 2014. Results of the WMT-14 metrics shared task. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*.

Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, and Roberto Zamparelli. 2014. A sick cure for the evaluation of compositional distributional semantic models. In *Proceedings of LREC 2014*.

Tomas Mikolov, Quoc V Le, and Ilya Sutskever. 2013a. Exploiting Similarities among Languages for Machine Translation. *CoRR*, pages 1–10.

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

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the ACL*, pages 311–318.

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

Keisuke Sakaguchi, Matt Post, and Benjamin Van Durme. 2014. Efficient elicitation of annotations for human evaluation of machine translation. In *In Proceedings of the Ninth Workshop on Statistical Machine Translation*, Baltimore, Maryland.

Matthew Snover, Nitin Madnani, Bonnie Dorr, and Richard Schwartz. 2008. TERp system description. In *MetricsMATR workshop at AMTA*. Citeseer.

Richard Socher, Eh Huang, and Jeffrey Pennington. 2011. Dynamic Pooling and Unfolding Recursive Autoencoders for Paraphrase Detection. In *Advances in Neural Information Processing Systems*, pages 801–809.

Richard Socher, John Bauer, Christopher D. Manning, and Andrew Y. Ng. 2013a. Parsing With Compositional Vector Grammars. *In Proceedings of the ACL*, pages 455–465.

Richard Socher, Alex Perelygin, and Jy Wu. 2013b. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of EMNLP*, pages 1631–1642.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1556–1566, Beijing, China. Association for Computational Linguistics.

Wojciech Zaremba and Ilya Sutskever. 2014. Learning to execute. *arXiv preprint arXiv:1410.4615*.

Investigating Continuous Space Language Models for Machine Translation Quality Estimation

Kashif Shah[§], **Raymond W. M. Ng**[§], **Fethi Bougares**[†], **Lucia Specia**[§]
§Department of Computer Science, University of Sheffield, UK
{kashif.shah, wm.ng, l.specia}@sheffield.ac.uk

†LIUM, University of Le Mans, France
fethi.bougares@lium.univ-lemans.fr

Abstract

We present novel features designed with a deep neural network for Machine Translation (MT) Quality Estimation (QE). The features are learned with a Continuous Space Language Model to estimate the probabilities of the source and target segments. These new features, along with standard MT system-independent features, are benchmarked on a series of datasets with various quality labels, including post-editing effort, human translation edit rate, post-editing time and METEOR. Results show significant improvements in prediction over the baseline, as well as over systems trained on state of the art feature sets for all datasets. More notably, the addition of the newly proposed features improves over the best QE systems in WMT12 and WMT14 by a significant margin.

1 Introduction

Quality Estimation (QE) is concerned with predicting the quality of Machine Translation (MT) output without reference translations. QE is addressed with various features indicating fluency, adequacy and complexity of the translation pair. These features are used by a machine learning algorithm along with quality labels given by humans to learn models to predict the quality of unseen translations.

A variety of features play a key role in QE. A wide range of features from source segments and their translated segments, extracted with the help of external resources and tools, have been proposed. These go from simple, language-independent features, to advanced, linguistically motivated features. They include features that summarise how the MT systems generate translations, as well as features that are oblivious to the systems. The majority of the features in the literature are extracted from each sentence pair in isolation, ignoring the context of the text. QE performance usually differs depending on the language pair, the specific quality score being optimised (e.g., post-editing time vs translation adequacy) and the feature set. Features based on n-gram language models, despite their simplicity, are among those with the best performance in most QE tasks (Shah et al., 2013b). However, they may not generalise well due to the underlying discrete nature of words in n-gram modelling.

Continuous Space Language Models (CSLM), on the other hand, have shown their potential to capture long distance dependencies among words (Schwenk, 2012; Mikolov et al., 2013). The assumption of these models is that semantically or grammatically related words are mapped to similar geometric locations in a high-dimensional continuous space. The probability distribution is thus much smoother and therefore the model has a better generalisation power on unseen events. The representations are learned in a continuous space to estimate the probabilities using neural networks with single (called shallow networks) or multiple (called deep networks) hidden layers. Deep neural networks have been shown to perform better than shallow ones due to their capability to learn higher-level, abstract representations of the input (Arisoy et al., 2012). In this paper, we explore the potential of these models in context of QE for MT. We obtain more robust features with CSLM and improve the overall prediction power for translation quality.

The paper is organised as follows: In Section 2 we briefly present the related work. Section 3 describes the CSLM model training and its various settings. In Section 4 we propose the use of CSLM features for QE. In Section 5 we present our experiments along with their results.

2 Related Work

For a detailed overview of various features and algorithms for QE, we refer the reader to the

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1073–1078,
Lisbon, Portugal, 17-21 September 2015.

WMT12-14 shared tasks on QE (Callison-Burch et al., 2012; Bojar et al., 2013; Ling et al., 2014). Most of the research work lies on deciding which aspects of quality are more relevant for a given task and designing feature extractors for them. While simple features such as counts of tokens and language model scores can be easily extracted, feature engineering for more advanced and useful information can be quite labour-intensive.

Since their introduction in (Bengio et al., 2003), neural network language models have been successfully exploited in many speech and language processing problems, including automatic speech recognition (Schwenk and Gauvain, 2005; Schwenk, 2007) and machine translation (Schwenk, 2012).

Recently, (Banchs et al., 2015) used a Latent Semantic Indexing approach to model sentences as bag-of-words in a continuous space to measure cross language adequacy. (Tan et al., 2015) proposed to train models with deep regression for machine translation evaluation in a task to measure semantic similarity between sentences. They reported positive results on simple features; larger feature sets did not improve these results.

In this paper, we propose to estimate the probabilities of source and target segments with continuous space language models based on a deep architecture and to use these estimated probabilities as features along with standard feature sets in a supervised learning framework. To the best of our knowledge, such approach has not been studied before in the context of QE for MT. The result shows significant improvements in many prediction tasks, despite its simplicity. Monolingual data for source and target language is the only resource required to extract these features.

3 Continuous Space Language Models

A key factor for quality inference of a translated text is to determine the fluency of such a text and how well it conforms to the linguistic regularities of the target language. It involves grammatical correctness, idiomatic and stylistic word choices that can be derived by using n-gram language models. However, in high-order n-grams, the parameter space is sparse and conventional modelling is inefficient. Neural networks model the non-linear relationship between the input features and target outputs. They often outperform conventional techniques in difficult machine learning tasks. Neural network language models (CSLM) alleviate the curse of dimensionality by projecting words into a continuous space, and modelling and estimating probabilities in this space.

The architecture of a deep CSLM is illustrated in Figure 1. The inputs to a CSLM model are the $(K-1)$ left-context words $(w_{i-K+1}, \ldots, w_{i-2}, w_{i-1})$ to predict w_i. A one-hot vector encoding scheme is used to represent the input w_{i-k} with an N-dimensional vector. The output of CSLM is a vector of posterior probabilities for all words in vocabulary, $P(w_i|w_{i-1}, w_{i-2}, \ldots, w_{i-K+1})$. Due to the large output layer (vocabulary size), the complexity of a basic neural network language model is very high. Schwenk (2007) proposed efficient training strategies in order to reduce the computational complexity and speed up the training time. They process several examples at once and use a *short-list* vocabulary V with only the most frequent words.

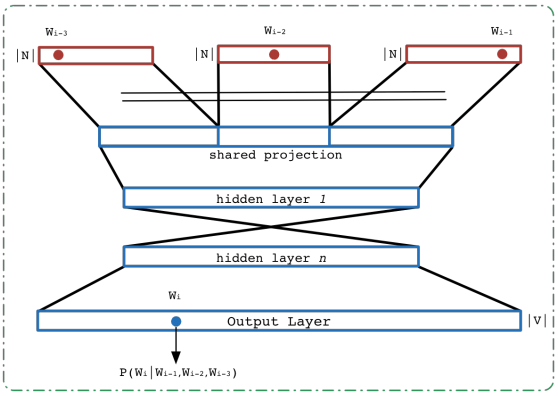


Figure 1: Deep CSLM architecture.

Following the settings mentioned in (Schwenk et al., 2014), all CSLM experiments described in this paper are performed using deep networks with four hidden layers: first layer for the projection (320 units for each context word) and three hidden layers of 1024 units with *tanh* activation. At the output layer, we use a *softmax* activation function applied to a *short-list* of the 32k most frequent words. The probabilities of the out-of-vocabulary words are obtained from a standard back-off n-gram language model. The projection of the words onto the continuous space and the training of the neural network is done by the standard back-propagation algorithm and outputs are the converged posterior probabilities. The model parameters are optimised on a development set.

4 CSLM and Quality Estimation

In the context of MT, CSLMs are generally trained on the target side of a given language pair to ex-

press the probability that the generated sentence is "correct" or "likely", without looking at the source sentence. However, QE is also concerned with how well the source segments can be translated. Therefore, we trained two models, one for each side of a given language pair. We extracted the probabilities for QE training and test sets for both source and its translation with their respective models and used them as features, along with other features, in a supervised learning setting.

Finally, we also used CSLM in a spoken language translation (SLT) task. In SLT, an automatic speech recogniser (ASR) is used to decode the source language text from audio. This creates an extra source of variability, where different ASR models and configurations give different outputs. In this paper, we use QE to exploit different ASR outputs (i.e. MT inputs) which in turn can lead to different MT outputs.

5 Experiments

We focus on experiments with sentence level QE tasks. Our English-Spanish experiments are based on the WMT QE shared task data from 2012 to 2015.[1] These tasks are diverse in nature, with different sizes and labels such as post-editing effort (PEE), post-editing time (PET) and human translation error rate (HTER). The results reported in Section 5.5 are directly comparable with the official systems submitted for each of the respective tasks. We also performed experiments on the IWSLT 2014 English-French SLT task [2] to study the applicability of our models on n-best ASR (MT inputs) comparison.

5.1 QE Datasets

In Table 1 we summarise the data and tasks for our experiments. We refer readers to the WMT and IWSLT websites for detailed descriptions of these datasets. All datasets are publicly available.

WMT12: English-Spanish news sentence translations produced by a Moses "baseline" statistical MT (SMT) system, and judged for perceived post-editing effort in 1–5 (highest-lowest), taking a weighted average of three annotators (Callison-Burch et al., 2012).

WMT13 (Task-1): English-Spanish sentence translations of news texts produced by a Moses "baseline" SMT system. These were then post-edited by a professional translator and labelled using HTER. This is a superset of the WMT12 dataset, with 500 additional sentences for test, and a different quality label (Bojar et al., 2013).

WMT14 (Task-1.1): English-Spanish news sentence translations. The dataset contains source sentences and their human translations, as well as three versions of machine translations: by an SMT system, a rule-based system system and a hybrid system. Each translation was labelled by professional translators with 1-3 (lowest-highest) scores for perceived post-editing effort.

WMT14 (Task-1.3): English-Spanish news sentence translations post-edited by a professional translator, with the post-editing time collected on a sentence-basis and used as label (in milliseconds).

WMT15 (Task-1): Large English-Spanish news dataset containing source sentences, their machine translations by an online SMT system, and the post-editions of the translation by crowdsourced translators, with HTER used as label.

IWSLT14: English-French dataset containing source language data from the 10-best (sentences) ASR system output. On the target side, the 1-best MT translation is used. The ASR system leads to different source segments, which in turn lead to different translations. METEOR (Banerjee and Lavie, 2005) is used to label these alternative translations against a reference (human) translation. Both ASR and MT outputs come from a system submission in IWSLT 2014 (Ng et al., 2014). The ASR system is a multi-pass deep neural network tandem system with feature and model adaptation and rescoring. The MT system is a phrase-based SMT system produced using Moses.

Dataset	Lang.	Train	Test	Label
WMT12	en-es	1,832	422	PEE 1-5
WMT13	en-es	2,254	500	HTER 0-1
WMT14$_{task1.1}$	en-es	3,816	600	PEE 1-3
WMT14$_{task1.3}$	en-es	650	208	PET (ms)
WMT15	en-es	11,271	1,817	HTER 0-1
IWSLT14	en-fr	8,180	11,240	MET. 0-1

Table 1: QE datasets: # sentences and labels.

5.2 CSLM Dataset

The dataset used for CSLM training consists of Europarl, News-commentary and News-crawl corpus. We used a data selection method (Moore

[1] http://www.statmt.org/wmt[12,13,14,15]/quality-estimation-task.html

[2] https://sites.google.com/site/iwsltevaluation2014/slt-track

and Lewis, 2010) to select the most relevant training data with respect to a development set. For English-Spanish, the development data is the concatenation of newstest2012 and newstest2013 of the WMT translation track. For English-French, the development set is the concatenation of the IWSLT dev2010 and eval2010. In Table 2 we show statistics on the selected monolingual data used to train back-off LM and CSLM.

Lang.	Train	Dev	LM ppl	CSLM ppl
en	4.3G	137.7k	164.63	116.58 (29.18%)
fr	464.7M	54K	99.34	64.88 (34.68%)
es	21.2M	149.4k	145.49	87.14 (40.10%)

Table 2: Training data size (number of tokens) and language models perplexity (ppl). The values in parentheses in last column shows percentage decrease in perplexity.

5.3 Feature Sets

We use the `QuEst` [3] toolkit (Specia et al., 2013; Shah et al., 2013a) to extract two feature sets for each dataset:

- **BL**: 17 features used as baseline in the WMT shared tasks on QE.
- **AF**: 80 augmented MT system-independent features[4] (superset of **BL**). For the En-Fr SLT task, we have additional 36 features (21 ASR + 15 MT-dependent features)

The resources used to extract these features (corpora, etc.) are also available as part of the WMT shared tasks on QE. The CSLM features for each of the source and target segments are extracted using the procedure described in Section 3 with the CSLM toolkit. [5]

We trained QE models with following combination of features:

- **BL + CSLM**$_{src,tgt}$: CSLM features for source and target segments, plus the baseline features.
- **AF + CSLM**$_{src,tgt}$: CSLM features for source and target segments, plus all available features.

For the WMT12 task, we performed further experiments to analyse the improvements with CSLM:

- **CSLM**$_{src}$: Source side CSLM feature only.
- **CSLM**$_{tgt}$: Target side CSLM feature only.
- **CSLM**$_{src,tgt}$: Source and target CSLM features by themselves.
- **FS(AF) + CSLM**$_{src,tgt}$: CSLM features in addition to the best performing feature set (**FS(AF)**) selected as described in (Shah et al., 2013b; Shah et al., 2015).

5.4 Learning algorithms

We use the Support Vector Machines implementation of the `scikit-learn` toolkit to perform regression (SVR) with either Radial Basis Function (RBF) or linear kernel and parameters optimised via grid search. To evaluate the prediction models we use Mean Absolute Error (MAE), its squared version – Root Mean Squared Error (RMSE), and Pearson's correlation (r) score.

Task	System	#feats	MAE	RMSE	r
WMT12	BL	17	0.6821	0.8117	0.5595
	AF	80	0.6717	0.8103	0.5645
	BL + CSLM$_{src,tgt}$	19	0.6463	0.7977	0.5805
	AF + CSLM$_{src,tgt}$	82	0.6462	0.7946	0.5825
WMT13	BL	17	0.1411	0.1812	0.4612
	AF	80	0.1399	0.1789	0.4751
	BL + CSLM$_{src,tgt}$	19	0.1401	0.1791	0.4771
	AF + CSLM$_{src,tgt}$	82	0.1371	0.1750	0.4820
WMT14 Task 1.1	BL	17	0.5241	0.6591	0.2502
	AF	80	0.4896	0.6349	0.3310
	BL + CSLM$_{src,tgt}$	19	0.4931	0.6351	0.3545
	AF + CSLM$_{src,tgt}$	82	0.4628*	0.6165*	0.3824*
WMT14 Task 1.3	BL	17	0.1798	0.2865	0.5661
	AF	80	0.1753	0.2815	0.5871
	BL + CSLM$_{src,tgt}$	19	0.1740	0.2758	0.6243
	AF + CSLM$_{src,tgt}$	82	0.1701**	0.2734	0.6201
WMT15	BL	17	0.1562	0.2036	0.1382
	AF	80	0.1541	0.1995	0.2205
	BL + CSLM$_{src,tgt}$	19	0.1501	0.1971	0.2611
	AF + CSLM$_{src,tgt}$	82	0.1471	0.1934	0.2862
IWSLT14	BL	17	0.1390	0.1791	0.5012
	AF	116	0.1361	0.1775	0.5211
	BL + CSLM$_{src,tgt}$	19	0.1358	0.1750	0.5321
	AF + CSLM$_{src,tgt}$	118	0.1337	0.1728	0.5445

Table 3: Results for datasets with various feature sets. Figures with * beat the official best systems, and with ** are second best. Results with CSLM features are significantly better than BL and AF on all tasks (paired t-test with $p \leq 0.05$).

Task	System	#feats	MAE	RMSE	r
WMT12	BL + CSLM$_{src}$	18	0.6751	0.8125	0.5626
	BL + CSLM$_{tgt}$	18	0.6694	0.8023	0.5815
	CSLM$_{src,tgt}$	2	0.6882	0.8430	0.5314
	FS(AF)	19	0.6131	0.7598	0.6296
	FS(AF) + CSLM$_{src,tgt}$	21	0.5950*	0.7442*	0.6482*

Table 4: Impact of different combinations of CSLM features on the WMT12 task. Figures with * beat the official best system. Results with CSLM features are significantly better than BL and AF on all tasks (paired t-test with $p \leq 0.05$).

[3] `http://www.quest.dcs.shef.ac.uk/`

[4] 80 features `http://www.quest.dcs.shef.ac.uk/quest_files/features_blackbox`

[5] `http://www-lium.univ-lemans.fr/cslm/`

5.5 Results

Table 3 presents the results with different feature sets for data from various shared tasks. It can be noted that CSLM features always bring significant improvements whenever added to either baseline or augmented feature set. A reduction in both error scores (MAE and RMSE) as well as an increase in Pearson's correlation with human labels can be observed on all tasks. It is also worth noticing that the CSLM features bring improvements over all tasks with different labels, evidencing that different optimisation objectives and language pairs can benefit from these features. However, the improvements are more visible when predicting post-editing effort for WMT12 and WMT14's Task 1.1. For these two tasks, we are able to achieve state-of-the-art performance by adding the two CSLM features to all available or selected feature sets.

For WMT12, we performed another set of experiments to study the effect of CSLM features by themselves and in combination. The results in Table 4 show that the target side CSLM feature bring larger improvements than its source side counterpart. We believe that it is because the target side feature directly reflects the fluency of the translation, whereas the source side feature (regarded as a translation complexity feature) only has indirect effect on quality. Interestingly, the two CSLM features alone give comparable results (slightly worse) than the BL feature set [6] despite the fact that these 17 features cover many complexity, adequacy and fluency quality aspects. CSLM features bring further improvements on pre-selected feature sets, as shown in Table 3. We also performed feature selection over the full feature set along with CSLM features, following the procedure in (Shah et al., 2013b). Interestingly, both CSLM features were selected among the top ranked features, confirming their relevance.

In order to investigate whether our CSLM features results hold for other feature sets, we experimented with the feature sets provided by most teams participating in the WMT12 QE shared task. These feature sets are very diverse in terms of the types of features, resources used, and their sizes. Table 5 shows the official results from the shared task (Off.) (Callison-Burch et al., 2012), those from training an SVR on these features with and without CSLM features. Note that the official scores are often different from the results obtained with our SVR models because of differences in the learning algorithms. As shown in Table 5, we observed similar improvements with additional CSLM features over all of these feature sets.

System	#feats	Off.	SVR without CSLM	SVR with CSLM
SDL	15	0.61	0.6115	0.5993
UU	82	0.64	0.6513	0.6371
Loria	49	0.68	0.6978	0.6729
UEdin	56	0.68	0.6879	0.6724
TCD	43	0.68	0.6972	0.6715
WL-SH	147	0.69	0.6791	0.6678
UPC	57	0.84	0.8419	0.8310
DCU	308	0.75	0.6825	0.6812
PRHLT	497	0.70	0.6699	0.6649

Table 5: MAE score on official WMT12 feature sets using SVR with and without CSLM features.

6 Conclusions

We proposed novel features for machine translation quality estimation obtained using a deep continuous space language models. The proposed features led to significant improvements over standard feature sets for a variety of datasets, outperforming the state-of-art on two official WMT QE tasks. These results showed that different optimisation objectives and language pairs can benefit from the proposed features. The proposed features have been shown to also perform well on QE within a spoken language translation task.

Both source and target CSLM features improve prediction quality, either when used separately or in combination. They proved complementary when used together with other feature sets and produce comparable results to high performing baseline features when used alone for prediction. Finally, results comparing all official WMT12 QE feature sets showed significant improvements in the predictions when CSLM features were added to those submitted by participating teams. These findings provide evidence that the proposed features bring valuable information into prediction models, despite their simplicity and the fact that they require only monolingual data as resource, which is available in abundance for many languages.

As future work, it would be interesting to explore various distributed word representations for quality estimation and joint models that look at both the source and the target sentences simultaneously.

Acknowledgements

This work was supported by the QT21 (H2020 No. 645452), Cracker (H2020 No. 645357) and DARPA Bolt projects.

[6] We compare results in terms of MAE scores only.

References

Ebru Arisoy, Tara N. Sainath, Brian Kingsbury, and Bhuvana Ramabhadran. 2012. Deep neural network language models. In *NAACL-HLT 2012 Workshop: Will We Ever Really Replace the N-gram Model? On the Future of Language Modeling for HLT*, pages 20–28, Montreal, Canada.

Rafael E Banchs, Luis F D'Haro, and Haizhou Li. 2015. Adequacy–fluency metrics: Evaluating mt in the continuous space model framework. *Audio, Speech, and Language Processing, IEEE/ACM Transactions on*, 23(3):472–482.

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *ACL workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, volume 29, pages 65–72.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *The Journal of Machine Learning Research*, 3:1137–1155.

Ondrej Bojar, Christian Buck, Chris Callison-Burch, Christian Federmann, Barry Haddow, Philipp Koehn, Christof Monz, Matt Post, Radu Soricut, and Lucia Specia. 2013. Findings of the 2013 Workshop on Statistical Machine Translation. In *Eighth Workshop on Statistical Machine Translation*, pages 1–44, Sofia, Bulgaria.

Chris Callison-Burch, Philipp Koehn, Christof Monz, Matt Post, Radu Soricut, and Lucia Specia. 2012. Findings of the 2012 WMT. In *Seventh Workshop on Statistical Machine Translation*, pages 10–51, Montréal, Canada.

Wang Ling, Luis Marujo, Chris Dyer, Alan Black, and Isabel Trancoso. 2014. Crowdsourcing high-quality parallel data extraction from twitter. In *Ninth Workshop on Statistical Machine Translation*, WMT14, pages 426–436, Baltimore, USA.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013. Linguistic regularities in continuous space word representations. In *HLT-NAACL*, pages 746–751.

Robert C. Moore and William Lewis. 2010. Intelligent selection of language model training data. In *Proceedings of the ACL 2010 Conference Short Papers*, ACLShort '10, pages 220–224, Stroudsburg, PA, USA. Association for Computational Linguistics.

Raymond W. N. Ng, Mortaza Doulaty, Rama Doddipatla, Oscar Saz, Madina Hasan, Thomas Hain, Wilker Aziz, Kashif Shaf, and Lucia Specia. 2014. The USFD spoken language translation system for IWSLT 2014. *Proc. IWSLT*, pages 86–91.

Holger Schwenk and Jean-Luc Gauvain. 2005. Training neural network language models on very large corpora. In *Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 201–208.

Holger Schwenk, Fethi Bougares, and Loic Barrault. 2014. Efficient training strategies for deep neural network language models. In *NIPS workshop on Deep Learning and Representation Learning*.

Holger Schwenk. 2007. Continuous space language models. *Computer Speech & Language*, 21(3):492–518.

Holger Schwenk. 2012. Continuous space translation models for phrase-based statistical machine translation. In *COLING (Posters)*, pages 1071–1080.

Kashif Shah, Eleftherios Avramidis, Ergun Biçicic, and Lucia Specia. 2013a. Quest - design, implementation and extensions of a framework for machine translation quality estimation. *The Prague Bulletin of Mathematical Linguistics*, 100:19–30.

Kashif Shah, Trevor Cohn, and Lucia Specia. 2013b. An investigation on the effectiveness of features for translation quality estimation. In *Machine Translation Summit*, volume 14, pages 167–174.

Kashif Shah, Trevor Cohn, and Lucia Specia. 2015. A bayesian non-linear method for feature selection in machine translation quality estimation. *Machine Translation*, 29(2):101–125.

Lucia Specia, Kashif Shah, José G. C. de Souza, and Trevor Cohn. 2013. QuEst - A translation quality estimation framework. In *51st Annual Meeting of the Association for Computational Linguistics: Demo Session*, pages 79–84, Sofia, Bulgaria.

Liling Tan, Carolina Scarton, Lucia Specia, and Josef van Genabith. 2015. Usaar-sheffield: Semantic textual similarity with deep regression and machine translation evaluation metrics. In *Proceedings of the 9th International Workshop on Semantic Evaluation*, pages 85–89, Denver, Colorado.

Supervised Phrase Table Triangulation with Neural Word Embeddings for Low-Resource Languages

Tomer Levinboim and **David Chiang**
Department of Computer Science and Engineering
University of Notre Dame
{levinboim.1,dchiang}@nd.edu

Abstract

In this paper, we develop a supervised learning technique that improves noisy phrase translation scores obtained by phrase table triangulation. In particular, we extract word translation distributions from small amounts of source-target bilingual data (a dictionary or a parallel corpus) with which we learn to assign better scores to translation candidates obtained by triangulation. Our method is able to gain improvement in translation quality on two tasks: (1) On Malagasy-to-French translation via English, we use only 1k dictionary entries to gain +0.5 B over triangulation. (2) On Spanish-to-French via English we use only 4k sentence pairs to gain +0.7 B over triangulation interpolated with a phrase table extracted from the same 4k sentence pairs.

1 Introduction

Phrase-based statistical machine translation systems require considerable amounts of source-target parallel data to produce good quality translation. However, large amounts of parallel data are available for only a fraction of language pairs, and mostly when one of the languages is English.

Phrase table triangulation (Utiyama and Isahara, 2007; Cohn and Lapata, 2007; Wu and Wang, 2007) is a method for generating source-target phrase tables without having access to any source-target parallel data. The intuition behind triangulation (and pivoting techniques in general) is the transitivity of translation: if a source language phrase **s** translates to a pivot language phrase **p** which in turn translates to a target language phrase **t**, then **s** should likely translate to **t**. Following this intuition, a triangulated source-target phrase table $\hat{T}$ can be composed from a source-pivot and pivot-target phrase table (§2).

However, the resulting triangulated phrase table $\hat{T}$ contains many spurious phrase pairs and noisy probability estimates. Therefore, early triangulation work (Wu and Wang, 2007) already realistically assumed access to a limited source-target parallel data from which a relatively high-quality source-target phrase table T can be directly estimated. The two phrase tables were then combined, resulting in a higher quality phrase table that proposes translations for many source phrases not found in T. Wu and Wang (2007) report that interpolation of the two phrase tables T and $\hat{T}$ leads to higher quality translations. However, the triangulated phrase table $\hat{T}$ is obtained without using the source-target bilingual data, which suggests that the source-target data is not used as fully as it could be.

In this paper, we develop a supervised learning algorithm that corrects triangulated word translation probabilities by relying on word translation distributions w^{sup} derived from the limited source-target data. In particular, we represent source and target words using word embeddings (Mikolov et al., 2013) and learn a transformation between the two embedding spaces in order to approximate w^{sup}, thus down-weighting incorrect translation candidates proposed by triangulation (§3). By representing words as embeddings, our model can generalize the information contained in the source-target data (as encoded in the distributions w^{sup}) to a much larger vocabulary, and can assign lexical-weighting probabilities to most of the phrase pairs in $\hat{T}$.

Fixing English as the pivot language (the most realistic pivot language choice), on a low-resource Spanish-to-French translation task our model gains +0.7 B on top of standard phrase table interpolation. On Malagasy-to-French translation, our model gains +0.5 B on top of triangulation when using only 1k Malagasy-French dictionary entries (§4).

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1079–1083,
Lisbon, Portugal, 17-21 September 2015.

2 Preliminaries

Let s, p, t denote words and $\mathbf{s}, \mathbf{p}, \mathbf{t}$ denote phrases in the source, pivot, and target languages, respectively. Also, let T denote a phrase table estimated over a parallel corpus and $\hat{T}$ denote a triangulated phrase table. We use similar notation for their respective phrase translation features ϕ, lexical-weighting features *lex*, and the word translation probabilities w.

2.1 Triangulation (weak baseline)

In phrase table triangulation, a source-target phrase table T_{st} is constructed by combining a source-pivot and pivot-target phrase table $T_{\text{sp}}, T_{\text{pt}}$, each estimated on its respective parallel data. For each resulting phrase pair $(\mathbf{s}, \mathbf{t})$, we can also compute an alignment $\hat{\mathbf{a}}$ as the most frequent alignment obtained by combining source-pivot and pivot-target alignments $\mathbf{a_{sp}}$ and $\mathbf{a_{pt}}$ across all pivot phrases $\mathbf{p}$ as follows: $\{(s,t) \mid \exists p : (s,p) \in \mathbf{a_{sp}} \wedge (p,t) \in \mathbf{a_{pt}}\}$.

The triangulated source-to-target lexical weights, denoted $\widehat{lex}_{\text{st}}$, are approximated in two steps: First, word translation scores $\hat{w}_{\text{st}}$ are approximated by marginalizing over the pivot words:

$$\hat{w}_{\text{st}}(t \mid s) = \sum_p w_{\text{sp}}(p \mid s) \cdot w_{\text{pt}}(t \mid p). \quad (1)$$

Next, given a (triangulated) phrase pair $(\mathbf{s}, \mathbf{t})$ with alignment $\hat{\mathbf{a}}$, let $\hat{\mathbf{a}}_{s,:} = \{t \mid (s,t) \in \hat{\mathbf{a}}\}$; the lexical-weighting probability is (Koehn et al., 2003):

$$\widehat{lex}_{\text{st}}(\mathbf{t} \mid \mathbf{s}, \hat{\mathbf{a}}) = \prod_{s \in \mathbf{s}} \frac{1}{|\hat{\mathbf{a}}_{s,:}|} \sum_{t \in \hat{\mathbf{a}}_{s,:}} \hat{w}_{\text{st}}(t \mid s). \quad (2)$$

The triangulated phrase translation scores, denoted $\hat{\phi}_{\text{st}}$, are computed by analogy with Eq. 1.

We also compute these scores in the reverse direction by swapping the source and target languages.

2.2 Interpolation (strong baseline)

Given access to source-target data, an ordinary source-target phrase table T_{st} can be estimated directly. Wu and Wang (2007) suggest interpolating phrase pairs entries that occur in both tables:

$$T_{\text{interp}} = \alpha T_{\text{st}} + (1 - \alpha)\hat{T}_{\text{st}}. \quad (3)$$

Phrase pairs appearing in only one phrase table are added as-is. We refer to the resulting table as the interpolated phrase table.

3 Supervised Word Translations

While interpolation (Eq. 3) may help correct some of the noisy triangulated scores, its effect is limited to phrase pairs appearing in both phrase tables. Here, we suggest a discriminative supervised learning method that can affect all phrase pairs.

Our idea is to regard word translation distributions derived from source-target bilingual data (through word alignments or dictionary entries) as the correct translation distributions, and use them to learn discriminately: correct target words should become likely translations, and incorrect ones should be down-weighted. To generalize beyond the vocabulary of the source-target data, we appeal to word embeddings.

We present our formulation in the source-to-target direction. The target-to-source direction is obtained simply by swapping the source and target languages.

3.1 Model

Let c^{sup}_{st} denote the number of times source word s was aligned to target word t (in word alignment, or in the dictionary). We define the word translation distributions $w^{\text{sup}}(t \mid s) = c^{\text{sup}}_{st}/c^{\text{sup}}_{s}$, where $c^{\text{sup}}_{s} = \sum_t c^{\text{sup}}_{st}$. Furthermore, let $q(t \mid s)$ denote the word translation probabilities we wish to learn and consider maximizing the log-likelihood function:

$$\arg\max_q L(q) = \arg\max_q \sum_{(s,t)} c^{\text{sup}}_{st} \log q(t \mid s).$$

Clearly, the solution $q(\cdot \mid s) := w^{\text{sup}}(\cdot \mid s)$ maximizes L. However, we would like a solution that generalizes to source words s beyond those observed in the source-target corpus – in particular, those source words that appear in the triangulated phrase table $\hat{T}$, but not in T.

In order to generalize, we abstract from words to vector representations of words. Specifically, we constrain q to the following parameterization:

$$q(t \mid s) = \frac{1}{Z_s} \exp\left(v_s^T A v_t + f_{st}^T h\right)$$
$$Z_s = \sum_{t \in \mathcal{T}(s)} \exp\left(v_s^T A v_t + f_{st}^T h\right).$$

Here, the vectors v_s and v_t represent monolingual features and the vector f_{st} represents bilingual features. The parameters A and h are to be learned.

In this work, we use monolingual word embeddings for v_s and v_t, and set the vector f_{st} to contain only the value of the triangulated score, such

that $f_{st} := \hat{w}_{\text{st}}$. Therefore, the matrix A is a linear transformation between the source and target embedding spaces, and h (now a scalar) quantifies how the triangulated scores $\hat{w}$ are to be trusted.

In the normalization factor Z_s, we let t range only over possible translations of s suggested by either w^{sup} or the triangulated word probabilities. That is:

$$\mathcal{T}(s) = \{t \mid w^{\text{sup}}(t \mid s) > 0 \ \vee \ \hat{w}(t \mid s) > 0\}.$$

This restriction makes efficient computation possible, as otherwise the normalization term would have to be computed over the entire target vocabulary.

Under this parameterization, our goal is to solve the following maximization problem:

$$\max_{A,h} L(A, h) = \max_{A,h} \sum_{s,t} c_{st}^{\text{sup}} \log q(t \mid s). \quad (4)$$

3.2 Optimization

The objective function in Eq. 4 is concave in both A and h. This is because after taking the log, we are left with a weighted sum of linear and concave (negative log-sum-exp) terms in A and h. We can therefore reach the global solution of the problem using gradient descent.

Taking derivatives, the gradient is

$$\frac{\partial L}{\partial A} = \sum_{s,t} m_{st} v_s v_t^{\text{T}} \qquad \frac{\partial L}{\partial h} = \sum_{s,t} m_{st} f_{st}$$

where the scalar $m_{st} = c_{st}^{\text{sup}} - c_s^{\text{sup}} q(t \mid s)$ for the current value of q.

For quick results, we limited the number of gradient steps to 200 and selected the iteration that minimized the total variation distance to w^{sup} over a held out dev set:

$$\sum_s \|q(\cdot \mid s) - w^{\text{sup}}(\cdot \mid s)\|_1. \quad (5)$$

We obtained better convergence rate by using a batch version of the effective and easy-to-implement Adagrad technique (Duchi et al., 2011). See Figure 1.

3.3 Re-estimating lexical weights

Having learned the model (A and h), we can now use $q(t \mid s)$ to estimate the lexical weights (Eq. 2) of any aligned phrase pairs $(\mathbf{s}, \mathbf{t}, \hat{\mathbf{a}})$, assuming it is composed of embeddable words.

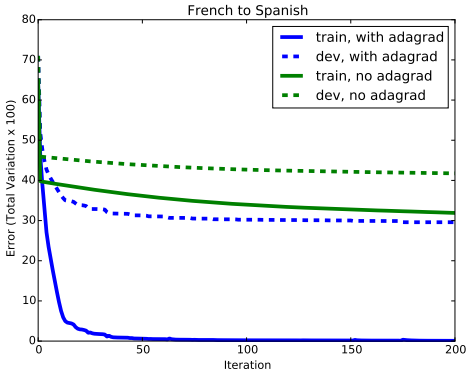


Figure 1: The (target-to-source) objective function per iteration. Applying batch Adagrad (blue) significantly accelerates convergence.

However, we found the supervised word translation scores q to be too sharp, sometimes assigning all probability mass to a single target word. We therefore interpolated q with the triangulated word translation scores $\hat{w}$:

$$q_\beta = \beta q + (1 - \beta)\hat{w}. \quad (6)$$

To integrate the lexical weights induced by q_β (Eq. 2), we simply appended them as new features in the phrase table in addition to the existing lexical weights. Following this, we can search for a β value that maximizes B on a tuning set.

3.4 Summary of method

In summary, to improve upon a triangulated or interpolated phrase table, we:

1. Learn word translation distributions q by supervision against distributions w^{sup} derived from the source-target bilingual data (§3.1).

2. Smooth the learned distributions q by interpolating with triangulated word translation scores $\hat{w}$ (§3.3).

3. Compute new lexical weights and append them to the phrase table (§3.3).

4 Experiments

To test our method, we conducted two low-resource translation experiments using the phrase-based MT system Moses (Koehn et al., 2007).

4.1 Data

Fixing the pivot language to English, we applied our method on two data scenarios:

1. **Spanish-to-French:** two related languages used to simulate a low-resource setting. The baseline is phrase table interpolation (Eq. 3).

2. **Malagasy-to-French:** two unrelated languages for which we have a small dictionary, but no parallel corpus (aside from tuning and testing data). The baseline is triangulation alone (there is no source-target model to interpolate with).

Table 1 lists some statistics of the bilingual data we used. European-language bitexts were extracted from Europarl (Koehn, 2005). For Malagasy-English, we used the Global Voices parallel data available online.[1] The Malagasy-French dictionary was extracted from online resources[2] and the small Malagasy-French tune/test sets were extracted[3] from Global Voices.

	lines of data		
language pair	train	tune	test
sp-fr	4k	1.5k	1.5k
mg-fr	1.1k	1.2k	1.2k
sp-en	50k	–	–
mg-en	100k	–	–
en-fr	50k	–	–

Table 1: Bilingual datasets. Legend: sp=Spanish, fr=French, en=English, mg=Malagasy.

Table 2 lists token statistics of the monolingual data used. We used word2vec[4] to generate French, Spanish and Malagasy word embeddings. The French and Spanish embeddings were (independently) estimated over their combined tokenized and lowercased Gigaword[5] and Leipzig news corpora.[6] The Malagasy embeddings were similarly estimated over data form Global Voices,[7] the Malagasy Wikipedia and the Malagasy Common Crawl.[8] In addition, we estimated a 5-gram French language model over the French monolingual data.

[1]http://www.ark.cs.cmu.edu/global-voices
[2]http://motmalgache.org/bins/homePage
[3]https://github.com/vchahun/gv-crawl
[4]https://radimrehurek.com/gensim/models/word2vec.html
[5]http://catalog.ldc.upenn.edu
[6]http://corpora.uni-leipzig.de/download.html
[7]http://www.isi.edu/~qdou/downloads.html
[8]https://commoncrawl.org/the-data/

language	words
French	1.5G
Spanish	1.4G
Malagasy	58M

Table 2: Size of monolingual corpus per language as measured in number of tokens.

4.2 Spanish-French Results

To produce w^{sup}, we aligned the small Spanish-French parallel corpus in both directions, and symmetrized using the intersection heuristic. This was done to obtain high precision alignments (the often-used *grow-diag-final-and* heuristic is optimized for phrase extraction, not precision).

We used the skip-gram model to estimate the Spanish and French word embeddings and set the dimension to $d = 200$ and context window to $w = 5$ (default). Subsequently, to run our method, we filtered out source and target words that either did not appear in the triangulation, or, did not have an embedding. We took words that appeared more than 10 times in the parallel corpus for the training set (~690 words), and between 5–9 times for the held out dev set (~530 words). This was done in both source-target and target-source directions.

In Table 3 we show that the distributions learned by our method are much better approximations of w^{sup} compared to those obtained by triangulation.

Method	source→target	target→source
triangulation	71.6%	72.0%
our scores	30.2%	33.8%

Table 3: Average total variation distance (Eq. 5) to the dev set portion of w^{sup} (computed only over words whose translations in w^{sup} appear in the triangulation). Using word embeddings, our method is able to better generalize on the dev set.

We then examined the effect of appending our supervised lexical weights. We fixed the word level interpolation $\beta := 0.95$ (effectively assigning very little mass to triangulated word translations $\hat{w}$) and searched for $\alpha \in \{0.9, 0.8, 0.7, 0.6\}$ in Eq. 3 to maximize B on the tuning set.

Our MT results are reported in Table 4. While interpolation improves over triangulation alone by +0.8 B , our method adds another +0.7 B on top of interpolation, a statistically significant gain ($p < 0.01$) according to a bootstrap resampling significance test (Koehn, 2004).

Method	α	tune	test
source-target	–	26.8	25.3
triangulation	–	29.2	28.4
interpolation	0.7	30.2	29.2
interpolation+our scores	0.6	30.8	**29.9**

Table 4: Spanish-French B scores. Appending lexical weights obtained by supervision over a small source-target corpus significantly outperforms phrase table interpolation (Eq. 3) by +0.7 B .

4.3 Malagasy-French Results

For Malagasy-French, the w^{sup} distributions used for supervision were taken to be uniform distributions over the dictionary translations. For each training direction, we used a 70%/30% split of the dictionary to form the train and dev sets.

Having significantly less Malagasy monolingual data, we used $d = 100$ dimensional embeddings and a $w = 3$ context window to estimate both Malagasy and French words.

As before, we added our supervised lexical weights as new features in the phrase table. However, instead of fixing $\beta = 0.95$ as above, we searched for $\beta \in \{0.9, 0.8, 0.7, 0.6\}$ in Eq. 6 to maximize B on a small tune set. We report our results in Table 5. Using only a dictionary, we are able to improve over triangulation by +0.5 B , a statistically significant difference ($p < 0.01$).

Method	β	tune	test
triangulation	–	12.2	11.1
triangulation+our scores	0.6	12.4	**11.6**

Table 5: Malagasy-French B . Supervision with a dictionary significantly improves upon simple triangulation by +0.5 B .

5 Conclusion

In this paper, we argued that constructing a triangulated phrase table independently from even very limited source-target data (a small dictionary or parallel corpus) underutilizes that parallel data.

Following this argument, we designed a supervised learning algorithm that relies on word translation distributions derived from the parallel data as well as a distributed representation of words (embeddings). The latter enables our algorithm to assign translation probabilities to word pairs that do not appear in the source-target bilingual data.

We then used our model to generate new lexical weights for phrase pairs appearing in a triangulated or interpolated phrase table and demonstrated improvements in MT quality on two tasks. This is despite the fact that the distributions (w^{sup}) we fit our model to were estimated automatically, or even naïvely as uniform distributions.

Acknowledgements

The authors would like to thank Daniel Marcu and Kevin Knight for initial discussions and a supportive research environment at ISI, as well as the anonymous reviewers for their helpful comments. This research was supported in part by a Google Faculty Research Award to Chiang.

References

Trevor Cohn and Mirella Lapata. 2007. Machine translation by triangulation: Making effective use of multi-parallel corpora. In *Proc. ACL*, pages 728–735.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *J. Machine Learning Research*, 12:2121–2159, July.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proc. NAACL HLT*, pages 48–54.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proc. ACL, Interactive Poster and Demonstration Sessions*, pages 177–180.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proc. EMNLP*, pages 388–395.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *Proc. MT Summit*, pages 79–86.

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

Masao Utiyama and Hitoshi Isahara. 2007. A comparison of pivot methods for phrase-based statistical machine translation. In *Proc. HLT-NAACL*, pages 484–491.

Hua Wu and Haifeng Wang. 2007. Pivot language approach for phrase-based statistical machine translation. In *Proc. ACL*, pages 856–863.

Translation Invariant Word Embeddings

Matt Gardner*
Carnegie Mellon University
mg1@cs.cmu.edu

Kejun Huang*
University of Minnesota
huang663@umn.edu

Evangelos Papalexakis
Carnegie Mellon University
epapalex@cs.cmu.edu

Xiao Fu
University of Minnesota
xfu@umn.edu

Partha Talukdar
Indian Institute of Science
ppt@serc.iisc.in

Christos Faloutsos
Carnegie Mellon University
christos@cs.cmu.edu

Nicholas Sidiropoulos
University of Minnesota
nikos@umn.edu

Tom Mitchell
Carnegie Mellon University
tom.mitchell@cmu.edu

Abstract

This work focuses on the task of finding latent vector representations of the words in a corpus. In particular, we address the issue of what to do when there are multiple languages in the corpus. Prior work has, among other techniques, used canonical correlation analysis to project pre-trained vectors in two languages into a common space. We propose a simple and scalable method that is inspired by the notion that the learned vector representations should be invariant to translation between languages. We show empirically that our method outperforms prior work on multilingual tasks, matches the performance of prior work on monolingual tasks, and scales linearly with the size of the input data (and thus the number of languages being embedded).

1 Introduction

Representing words as vectors in some latent space has long been a central idea in natural language processing. The *distributional hypothesis*, perhaps best stated as "You shall know a word by the company it keeps" (Firth, 1957), has had a long and productive history, as well as a recent revival in neural-network-based models (Mikolov et al., 2013). These methods generally construct a word by context matrix, then either use the vectors directly (often weighted by term frequency and inverse document frequency), perform some factorization of the matrix, or use it as input to a neural network which produces vectors for each word. The resultant vectors can be used in a wide array of tasks, from information retrieval to part-of-speech tagging and parsing.

There has also been some recent work addressing how to create these vectors when information from multiple languages is available. Two recent attempts involve using canonical correlation analysis (CCA) to project pre-trained vectors from each of two languages into a common space (Faruqui and Dyer, 2014b) and using an alignment matrix to heuristically project the vectors from one language onto the words in another language (Guo et al., 2015). These methods generally only work with two languages at a time, however.

In this paper, we introduce a technique for constructing multilingual word embeddings that is inspired by the notion of translational invariance. CCA and the heuristic projection mentioned above both attempt to construct vectors such that words that are translations of each other are close in the vector space, but the method we introduce formalizes this as part of the objective function of the *original decomposition*. We further show how to optimize this objective function with a method that scales linearly in the size of the input data. This results in a scalable, single-step method that is informed by both the monolingual corpus statistics and the multilingual alignment data. We show experimentally that this results in vectors that outperform prior work on multilingual tasks and match the performance of prior work on monolingual tasks.

*These authors contributed equally.

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1084–1088,
Lisbon, Portugal, 17-21 September 2015.

The contributions of this paper are the following:

- **Problem formulation:** we formalize the notion of translation-invariance, *regardless of the number of languages*, as part of the objective function of a standard matrix decomposition;
- **Scalable algorithm:** we introduce scalable means of optimizing this augmented objective functions; and
- **Effectiveness:** we present state-of-the-art results on a multilingual task using the vectors obtained by these methods.

The code and data used in this paper are publicly available at `https://sites.google.com/a/umn.edu/huang663/`.

2 Problem Definition

The informal problem definition is the following:

Informal Problem. Given *a set of cooccurrence statistics between words in each of several languages, and a translation table containing alignment counts between words in each of these languages,* **Find** *a latent representation for each word in each language that (1) captures information from the cooccurrence statistics and (2) is* invariant *to translations of the cooccurrence statistics between languages.*

More formally, suppose we have M_1 words and N_1 contexts in the first language ("English"), and M_2 and N_2 for the second language ("Spanish"). Then, we are given two matrices of cooccurrence statistics (one for each language), with dimensions $M_1 \times N_1$ and $M_2 \times N_2$, and two dictionary matrices containing translations from English to Spanish, and from Spanish to English, repesctively. A more detailed description on how the data is obtained can be found in (Faruqui and Dyer, 2014b). For simplicity in what follows, we denote these matrices as

- $\mathbf{X}$: a single multilingual cooccurrence matrix (with all the $M_1 + M_2$ words as the rows, and $N_1 + N_2$ contexts as columns). Entries in this matrix specify the cooccurrence between a word in any language and a context in any language.
- $\mathbf{D}_1$: a word dictionary matrix (with all the $M_1 + M_2$ English and Spanish words as both rows and columns). Entries in this matrix specify which words are translations of which other words, and is generally block-normalized, so that (e.g.) each Spanish word has a probability distribution over English words.
- $\mathbf{D}_2$: a context dictionary matrix (with all the N_1+N_2 English and Spanish contexts as both rows and columns). This is similar to $\mathbf{D}_1$ in its construction.

We seek decompositions of $\mathbf{X}$ that are invariant to multiplications along each mode by its respective $\mathbf{D}$ matrix. Note that, while we only described the case where we have two languages, it is straightforward to extend this to having many languages in the combined $\mathbf{X}$, $\mathbf{D}_1$ and $\mathbf{D}_2$ matrices, and we do this in some of the experiments described below.

3 Translation-invariant LSA

Without the side information provided by the dictionary matrices, the classic method for generating word vectors finds a low-rank decomposition of the data matrix $\mathbf{X}$:

$$\min_{\mathbf{U},\mathbf{V}} \|\mathbf{X} - \mathbf{U}\mathbf{V}^T\|_F^2.$$

With proper scaling (see our discussion in §4.2), the rows of $\mathbf{U}$ (or rows of $\mathbf{V}$) are the word embeddings (or "context embeddings"). It is well-known that the solution is given by the principal components of the singular value decomposition (SVD) of $\mathbf{X}$. Generating word embeddings in this way is known as latent semantic analysis (LSA) (Deerwester et al., 1990).

Our method extends LSA to incorporate information from many languages at a time, with the constraint that the decomposition should be invariant to translation between these languages. We call this method *translation-invariant LSA* (TI-LSA).

In order to take the dictionary matrices $\mathbf{D}_1$ and $\mathbf{D}_2$ into consideration, we propose to seek a decomposition that can simultaneously explain the original matrix $\mathbf{X}$ and various translations of it. We can formalize this in the following objective function:

$$\min_{\mathbf{U},\mathbf{V}} \|\mathbf{X} - \mathbf{U}\mathbf{V}^T\|_F^2 + \|\mathbf{D}_1\mathbf{X} - \mathbf{U}\mathbf{V}^T\|_F^2 + \|\mathbf{X}\mathbf{D}_2^T - \mathbf{U}\mathbf{V}^T\|_F^2 + \|\mathbf{D}_1\mathbf{X}\mathbf{D}_2^T - \mathbf{U}\mathbf{V}^T\|_F^2. \quad (1)$$

By expanding and combining all four quadratic terms, we can see that the above problem is equivalent to (up to a constant difference)

$$\min_{\mathbf{U},\mathbf{V}} \|\tilde{\mathbf{X}} - \mathbf{U}\mathbf{V}^T\|_F^2, \quad (2)$$

where

$$\tilde{\mathbf{X}} = \frac{1}{4}\left(\mathbf{X} + \mathbf{D}_1\mathbf{X} + \mathbf{X}\mathbf{D}_2^T + \mathbf{D}_1\mathbf{X}\mathbf{D}_2^T\right)$$
$$= \frac{1}{4}(\mathbf{I} + \mathbf{D}_1)\mathbf{X}(\mathbf{I} + \mathbf{D}_2)^T.$$

Taking the SVD of $\tilde{\mathbf{X}}$ does not seem numerically appealing at first glance: even though $\mathbf{D}_1$, $\mathbf{D}_2$, and $\mathbf{X}$ are all very sparse, forming $\tilde{\mathbf{X}}$ explicitly will introduce a significant amount of nonzeros. However, as we will explain below, it is not necessary to explicitly form $\tilde{\mathbf{X}}$ in order to find a few principal components of it.

We propose to use the Lanczos algorithm (Golub and Van Loan, 1996, Chapter 9) to calculate the SVD of $\tilde{\mathbf{X}}$. The Lanczos method can be viewed as a generalization of the power method for computing an arbitrary number of principal components, and the basic operation required is only matrix-vector multiplication. For our problem specifically, the required matrix-vector multiplications $\tilde{\mathbf{X}}\boldsymbol{\mu}$ and $\tilde{\mathbf{X}}^T\boldsymbol{\nu}$ can be carried out very efficiently with three sparse matrix-vector multiplications, each with complexity linear in the number of nonzeros in the sparse matrix involved, so that any dense intermediate matrix is avoided. As a result, by using our implementation of the Lanczos method, the time required for calculating the SVD of $\tilde{\mathbf{X}}$ is not much more than that of $\mathbf{X}$, even though $\tilde{\mathbf{X}}$ is significantly denser than $\mathbf{X}$.

4 Experiments

We present three experiments to evaluate the method introduced in this paper. The first experiment uses our word embeddings in a cross-lingual dependency parsing task; the second experiment looks at monolingual (English) performance on a series of word-similarity tasks; and the final experiment shows the scalability of our method by applying it to multiple languages.

4.1 Cross-lingual evaluation

Guo et al. (2015) recently introduced a method for using multilingual word embeddings to perform cross-lingual dependency parsing. They train a neural-network-based dependency parsing model using word vectors from one language, and then test the model using data and word vectors from another language. They used the embeddings obtained by Faruqui and Dyer (2014b), along with a heuristic projection. Because we used the same data to obtain our embeddings, our method is directly comparable to the CCA method of Faruqui and Dyer, and the projection method of Guo et al.

Embedding method	LAS	UAS
CCA (Faruqui & Dyer)	60.7	69.8
Projection (Guo et al.)	61.3	71.1
TI-LSA	**62.8**	**72.5**

Table 1: Labeled and unlabeled attachment score (LAS/UAS) on a cross-lingual dependency task. TI-LSA outperforms prior work on this task.

We used code and data graciously provided by Guo to run experiments, training a dependency parsing model on their English treebank, and testing it on the Spanish treebank. We report the results below for the methods used by Guo et al. and the method introduced in this paper. We could not exactly reproduce Guo's result with the code we were provided, so we report all results from our use of the provided code, in case some parameter settings are different from those used in Guo's paper. The results are shown in Table 1. As can be seen in the table, our first method for obtaining multilingual embeddings outperforms both the CCA method of Faruqui and Dyer, and the heuristic projection used by Guo et al.

4.2 Monolingual evaluation

While our focus is on generating embeddings that are invariant to translations (and thus most suited to multi- or cross-lingual tasks), we would hope that the addition of multiple languages would not hurt performance on monolingual tasks. We used wordvectors.org (Faruqui and Dyer, 2014a) to evaluate our learned vectors on a variety of English-language word similarity tasks. The tasks are mostly all variations on performing word similarity judgments, finding the correlation between the system's output and human responses. We used the same data as that used by Faruqui and Dyer (2014b) (English-Spanish only), and thus our method for obtaining multilingual embeddings is directly comparable to their technique for doing the same (CCA). We used the first 11 tasks on wordvectors.org, and obtained Faruqui and Dyer's results from that website. Due to space constraints, we only report the average performance across these 11 tasks for each of the methods we tested. The results are shown in Table 2. To test statistical significance, we performed a paired permutation test, treating performance on each task as

Method	Average Correlation
CCA (Faruqui & Dyer)	0.638
LSA	0.626
TI-LSA	0.628

Table 2: Average correlation with human similarity judgments on 11 word-similarity tasks. The differences between these methods are not statistically significant, showing that the gains we see in cross-lingual tasks are not at the expense of monolingual tasks.

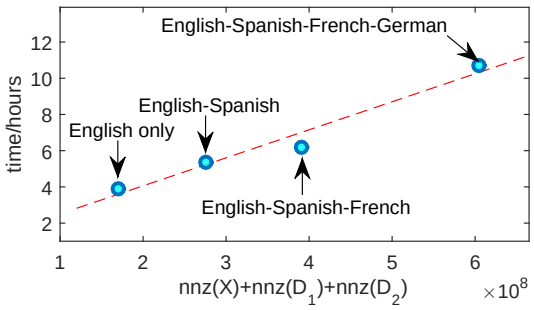


Figure 1: TI-LSA is linear in the number of nonzeros in the data matrices, and can easily scale to many languages.

paired data. The important thing to note from the table is that the differences between the methods are all quite small, and none of them are statistically significant.

Note that LSA on just the *English* data performs on par with all of the other methods presented; we have not found a way to improve performance on this monolingual task from using multilingual data.[1] However, it is also important to note that our multilingual methods do not *hurt* performance on these monolingual tasks, either—we get the benefits described in our other evaluations without losing performance on English-only tasks.

4.3 Scalability

We mentioned in Section 3 that our method is linear in the number of nonzeros in the data, as we are simply using the Lanczos algorithm to compute a sparse SVD. To show this in practice, we briefly present how the running time of our algorithm scales with the number of languages used. Each additional language adds roughly the same amount of data to the $\mathbf{X}$ matrix. Figure 1 shows that our method does indeed scale linearly with the number of nonzeros in the matrix, and thus also with the number of languages used (assuming each language has roughly the same amount of data). All the experiments are performed in MATLAB 2013a on a Linux server with 32 Xeon 2.00GHz cores and 128GB memory.

5 Discussion

We discuss here two points on the flexibility of the method we have introduced. First, note that the dictionary matrices we used contained information about translations between languages. It is also possible to include information about *paraphrases* in this dictionary. For instance, a resource such as the Paraphrase Database (Ganitkevitch et al., 2013) could be used to further constrain the embeddings obtained; this could be useful if the resource used to obtain a paraphrase dictionary contained more or different information than the corpus statistics used in the decomposition.

Second, note that we have two dictionaries, one for the words and one for the contexts. These dictionaries correspond to the modes of the matrix; we have one dictionary matrix per mode, and we always multiply the dictionary along its corresponding mode. It would be easy to extend this method to a setting where the data is a 3-mode tensor instead of a matrix, e.g., if the data were (subject, verb, object) triples, or relation triples in some knowledge base. In these settings, the dictionaries used for each mode might be more different; in the subject-verb-object example, one of the dictionaries would only have verbs, while the other two would only have nouns, for instance. Standard tensor decompositions could be augmented with a translation-invariance term, similar to what we have done with matrices in this work.

6 Related Work

The most closely related work is that of Faruqui and Dyer (2014b), whose CCA-based method we have already mentioned; however, it is not obvious how CCA-based methods can be applied to more than two languages at a time. Our work is also similar to prior work on multilingual latent semantic analysis; Bader and Chew (2008) also include a translation dictionary when decomposing the $\mathbf{X}$ matrix, though their formulation uses a term-document matrix instead of a word-context

[1] This is in contrast to the results reported by Faruqui and Dyer, who by our evaluation also do not improve performance using multilingual data. To obtain word vectors from our decomposition, we used only the U component of the SVD; including the singular values, as Faruqui and Dyer did, gives worse performance. We confirmed this with the authors, and replicated their result for English-only LSA when using the singular values.

matrix, and the way they use the translation dictionary is quite different.

7 Conclusions

We have presented a new technique for generating word embeddings from multilingual corpora. This technique formalizes the notion of translation invariance into the objective function of the matrix decomposition and provides flexible and scalable means for obtaining word vectors where words that are translations of each other are close in the learned vector space. Through three separate evaluations, we showed that our technique gives superior performance on multilingual tasks, matches prior work on monolingual tasks, and scales linearly in the size of the input data. The code and data used in this paper are available at `https://sites.google.com/a/umn.edu/huang663/`.

Acknowledgements

Research was supported by the National Science Foundation Grant No. IIS-1247489, IIS-1247632, and a gift from Google. This work was also supported in part by a fellowship to Kejun Huang from the University of Minnesota Informatics Institute. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the funding parties.

References

[Bader and Chew2008] Brett W Bader and Peter A Chew. 2008. Enhancing multilingual latent semantic analysis with term alignment information. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, pages 49–56. Association for Computational Linguistics.

[Deerwester et al.1990] Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407.

[Faruqui and Dyer2014a] Manaal Faruqui and Chris Dyer. 2014a. Community evaluation and exchange of word vectors at wordvectors.org. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, Baltimore, USA, June. Association for Computational Linguistics.

[Faruqui and Dyer2014b] Manaal Faruqui and Chris Dyer. 2014b. Improving vector space word representations using multilingual correlation. In *Proceedings of EACL*, volume 2014.

[Firth1957] John R. Firth. 1957. A synopsis of linguistic theory 1930-1955. *Studies in Linguistic Analysis*, pages 1–32.

[Ganitkevitch et al.2013] Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. Ppdb: The paraphrase database. In *HLT-NAACL*, pages 758–764.

[Golub and Van Loan1996] Gene H. Golub and Charles F. Van Loan. 1996. *Matrix Computations*. Johns Hopkins University Press, 3rd edition.

[Guo et al.2015] Jiang Guo, Wanxiang Che, David Yarowsky, Haifeng Wang, and Ting Liu. 2015. Cross-lingual dependency parsing based on distributed representations. In *Proceedings of ACL*.

[Mikolov et al.2013] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *ICLR workshop*.

Hierarchical Phrase-based Stream Decoding

Andrew Finch and **Xiaolin Wang** and **Masao Utiyama** and **Eiichiro Sumita**
Advanced Speech Translation Research and Development Promotion Center
Advanced Translation Technology Laboratory
National Institute of Information and Communications Technology
Kyoto, Japan
{andrew.finch,xiaolin.wang,mutiyama,eiichiro.sumita}@nict.go.jp

Abstract

This paper proposes a method for hierarchical phrase-based stream decoding. A stream decoder is able to take a continuous stream of tokens as input, and segments this stream into word sequences that are translated and output as a stream of target word sequences. Phrase-based stream decoding techniques have been shown to be effective as a means of simultaneous interpretation. In this paper we transfer the essence of this idea into the framework of hierarchical machine translation. The hierarchical decoding framework organizes the decoding process into a chart; this structure is naturally suited to the process of stream decoding, leading to an efficient stream decoding algorithm that searches a restricted subspace containing only relevant hypotheses. Furthermore, the decoder allows more explicit access to the word re-ordering process that is of critical importance in decoding while interpreting. The decoder was evaluated on TED talk data for English-Spanish and English-Chinese. Our results show that like the phrase-based stream decoder, the hierarchical is capable of approaching the performance of the underlying hierarchical phrase-based machine translation decoder, at useful levels of latency. In addition the hierarchical approach appeared to be robust to the difficulties presented by the more challenging English-Chinese task.

1 Introduction

Statistical machine translation traditionally operates on sentence segmented input. This technology has advanced to the point where it is becoming capable enough to be useful for many applications. However, this approach may be unsuitable for simultaneous interpretation where the machine translation system is required to provide translations within a reasonably short space of time after words have been spoken. Under this type of constraint, it may not be possible to wait for the end of the sentence before translating, and segmentation at the sub-sentential level may be required as a consequence. This segmentation process is difficult, even for skilled human interpreters, and presents a major challenge to a machine since in addition to the translation process, decisions need to be made about when to commit to outputting a partial translation. Such decisions are critical since once such an output is made it can be difficult and highly undesirable to correct it later if it is in error.

2 Related Work

In order to automatically perform segmentation for interpretation, two types of strategy have be proposed. In the first, which we will call presegmentation, the stream is segmented prior to the start of the machine translation decoding process, and the machine translation system is constrained to translate using the given segmentation. This approach has the advantage that it can be implemented without the need to modify the machine translation decoding software. In the second type of strategy, which we will call incremental decoding, the segmentation process is performed during the decoding of the input stream. In this approach the segmentation process is able to exploit segmentation cues arising from the decoding process itself. That is to say, the order in which the decoder would prefer to generate the target sequence is taken into account.

A number of diverse strategies for presegmentation were studied in (Sridhar et al., 2013). They studied both non-linguistic techniques, that included fixed-length segments, and a "hold-output" method which identifies contiguous blocks of text that do not contain alignments to words outside them, and linguistically-motivated segmentation techniques beased on segmenting on

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1089–1094,
Lisbon, Portugal, 17-21 September 2015.

conjunctions, sentence boundaries and commas. Commas were the most effective segmentation cue in their investigation.

In (Oda et al., 2014) a strategy for segmentation prior to decoding based on searching for segmentation points while optimizing the BLEU score was presented. An attractive characteristic of this approach is that the granularity of the segmentation could be controlled by choosing the number of segmentation boundaries to be inserted, prior to the segmentation process. In (Matusov et al., 2007) it was shown that the prediction and use of soft boundaries in the source language text, when used as re-ordering constraints can improve the quality of a speech translation system.

(Siahbani et al., 2014) used a pre-segmenter in combination with a left-to-right hierarchical decoder (Watanabe et al., 2006) to achieve a considerably faster decoder in return for a small cost in terms of BLEU score.

A phrase-based incremental decoder called the stream decoder was introduced in (Kolss et al., 2008b), and further studied in (Finch et al., 2014). Their results, conducted on translation between European languages, and also on English-Chinese, showed that this approach was able to maintain a high level of translation quality for practically useful levels of latency. The hierarchical decoding strategy proposed here is based on this work.

2.1 Stream Decoding

The reader is referred to the original paper (Kolss et al., 2008a) for a complete description of the stream decoding process; in this section we provide a brief summary.

Figure 1 depicts a stream decoding process, and the figure applies to both the original phrase-based technique, and the proposed hierarchical method. The input to the stream decoder is a stream of tokens (it is also possible for the decoder to operate on tuples of confusable token sequences from a speech recognition decoder). As new tokens arrive, states in the search graph are extended with the new possible translation options arising from the new tokens. Periodically the stream decoder will commit to outputting a sequence of target tokens. At this point a state from the search graph is selected, the search graph leading from this state is kept, and the remainder discarded. The search then continues using the pruned search graph. The language model context is preserved at this state for use during the subsequent decoding. In this manner the stream decoder is able to jointly segment and translate a continuous stream of tokens that contains no segment boundary information; the segmentation occurs as a natural by-product of the decoding process. Re-ordering occurs in exactly the same manner as the sentence-by-sentence hierarchical decoder, and word re-ordering within segments is possible.

2.1.1 Latency Parameters

The stream decoding process is governed by two parameters L_{max} and L_{min}. These parameters are illustrated in Figure 1. The L_{max} parameter controls the maximum latency of the system. That is, the maximum number of tokens the system is permitted to fall behind the current position. If interpreting from speech, the parameter represents the number of words the system is allowed to fall behind the speaker, before being required to provide an output translation. This parameter is a hard constraint that guarantees the system will always be within L_{max} tokens of the current last token in the stream of input tokens. The parameter L_{min} represents the minimum number of words the system will lag behind the last word spoken. It serves as a means of preventing the decoder from committing to a translation too early.

Both the phrase-based and hierarchical phrase-based stream decoders maintain a sequence of tokens that represent the sequence of untranslated tokens from the input stream (see Figure 1). As new tokens arrive from the input stream, they are added to the end of the sequence. When the length of this sequence reaches L_{max}, the decoder is forced to provide an output.

2.1.2 Phrase-based Segmentation

When forced to commit to a translation, the phrase-based decoder rolls back the best hypothesis state by state, until the remaining state sequence translates a contiguous sequence of source words starting from beginning of the sequence of untranslated words, and the number of words that would remain in the sequence of untranslated words after the translation is made, is at least L_{min}. It is possible that no such state exists, in which case since the stream decoder is required to make an output, it must use an alternative strategy.

In this alternative strategy, the stream decoder will undertake a new decoding pass in which it is forced to make a monotonic step as the first step in the decoding process. Then, a state is selected from the best hypothesis using the roll-back strategy above. This process may also fail if the monotonic step would lead to the violation of L_{min}. In the implementation of (Finch et al., 2014), the decoder is permitted to violate L_{min} only in this case.

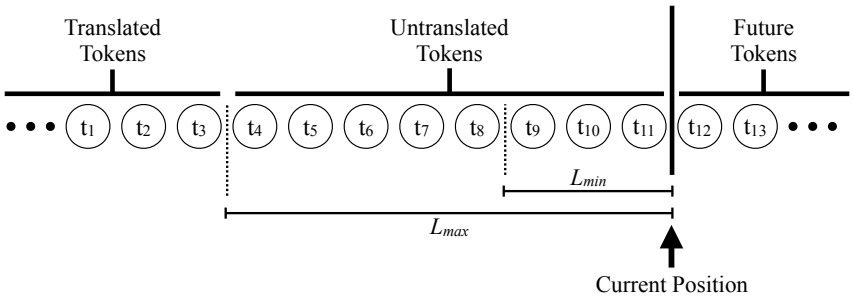


Figure 1: The stream decoding process.

2.1.3 The Proposed Method

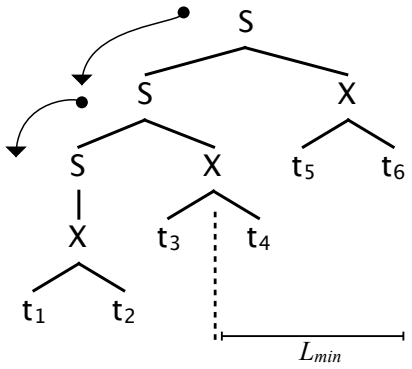


Figure 2: Selecting a segmentation point during hierarchical decoding.

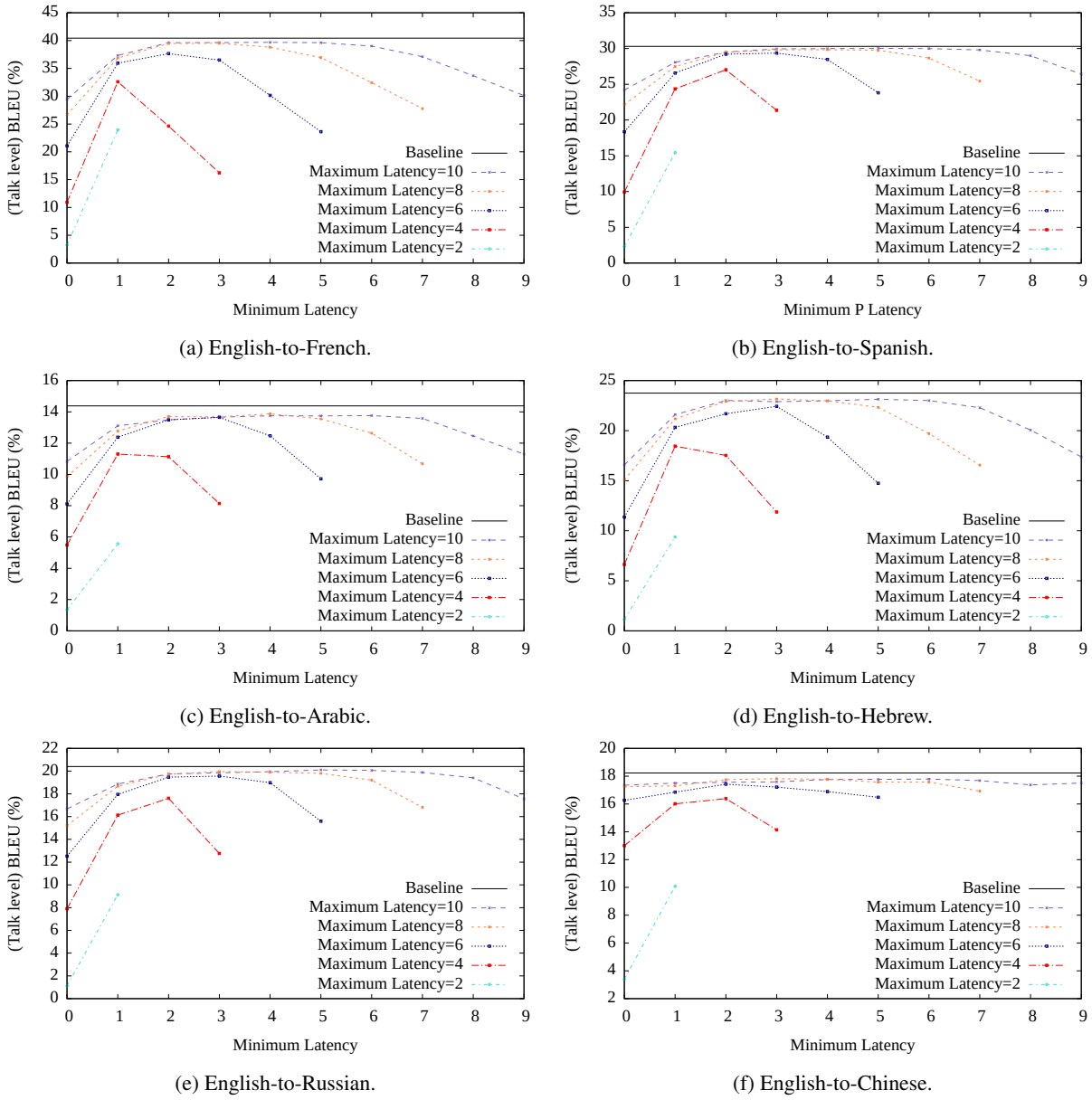
The proposed hierarchical method attempts to capture the spirit of the phrase-based method. When forced to commit to a translation of a sequence of n words, the segmentation process is simple and guided direcly by the chart.

As in the phrase-based approach, the best hypothesis at the top of the chart is used to provide the partial translation and segmentation point. This hypothesis has a span of $[1, n]$ over the source words. The left child of the rule (defined in accordance with the binarized grammar used by the decoder) that was applied to create this hypothesis is examined; let its span be $[1, k]$. If $n - k \geq L_{min}$, then this partial hypothesis represents a translation of the first k words of the sentence that leaves at least L_{min} words untranslated, and therefore the target word sequence from this partial hypothesis is output, and the associated source words are removed from the sequence of untranslated words. If this hypothesis is not able to meet the constraint, the parse tree traversal continues in the same manner: depth first along the left children until either a translation can be made, or no further traversal is possible.

Following the translation of of word sequence, similar to the phrase-based stream decoder of (Finch et al., 2014), the hierarchical stream decoder proceeds from an initial state in which the language model context is preserved. The decoding process relies on an implicit application of the glue grammar to connect the past and future nodes. An visual example of this selection process is given in Figure 2. In this example, the neither the root of the tree (spanning $t_1t_2t_3t_4t_5t_6$) nor its left child (spanning $t_1t_2t_3t_4$) are not able to generate an output since they both span sequences of words that would violate L_{min}, which is 3 in this example. The left child two levels down from the root node spans only t_1t_2 and would leave 4 words untranslated, therefore it defines an acceptable segmentation point.

Instead of forcing a monotonic decoding step in the event of a failure to find a segmentation point during the decoding, the hierarchical stream decoder directly eliminates hypotheses that would lead to such a failure. The search process is constrained such that all parse trees that cover the first word of the source sentence, must contain a subtree that can give rise to a translation that does not violate L_{min} (constituents that can produce translations cannot span more than $L_{max} - L_{min}$ words). Any search state that would violate this constraint is not allowed to enter the chart. This property is recursively propagated up the chart during the parsing process ensuring that each entry placed into the first column of the chart contains a constituent that could be used to produce a translation.

This approach is more appealing than the forced monotonic step in that it will also allow non-monotonic translations that are guaranteed to be usable. Similar to the phrase-based approach, in some circumstances it may not be possible to produce a parse that does not violate L_{min}, and only in this rare case is the decoder allowed to violate L_{min} in order to guarantee maximum latency.



(a) English-to-French.

(b) English-to-Spanish.

(c) English-to-Arabic.

(d) English-to-Hebrew.

(e) English-to-Russian.

(f) English-to-Chinese.

Figure 3: Stream decoding performance for several language pairs. The baseline was the same hierarchical phrase-based decoder, but decoded in the usual manner sentence-by-sentence without the stream decoding process. The baseline used the sentence segmentation provided by the corpus.

3 Experiments

3.1 Corpora

In all experiments, we used the TED[1] talks data sets from the IWSLT2014 campaign. We evaluated on English-to-Spanish, and English-to-Chinese translation using the same data sets that were used in (Finch et al., 2014). These pairs were chosen to include language pairs with a relatively monotonic translation process (English-Spanish) and (English-French), and also language pairs that required a greater amount of word re-ordering for example (English-Chinese). The Chinese corpus was segmented using the Stanford Chinese word segmenter (Tseng et al., 2005) according to the Chinese Penn Treebank standard.

3.2 Experimental Methodology

Our stream decoder was implemented within the framework of the AUGUSTUS decoder, a hierarchical statistical machine translation decoder (Chiang, 2007) that operates in a similar manner to the moses-chart decoder provided in the Moses machine translation toolkit (Koehn et al., 2007). The training procedure was quite typical: 5-gram language models were used, trained with modified

[1] http://www.ted.com

English input stream:

```
... we want to encourage a world of creators of inventors
of contributors because this world that we live in this
interactive world is ours ...
```

Sequence of translated segments:

Segment 1:	queremos	[we want to]
Segment 2:	animar a un mundo de	[encourage a world of]
Segment 3:	creadores de inventores	[creators of inventors]
Segment 4:	de colaboradores	[of collaborators]
Segment 5:	porque este mundo	[because this world]
Segment 6:	en el que vivimos	[in which we live]
Segment 7:	este interactiva mundo	[this interactive world]
Segment 8:	es la nuestra	[is ours]

Figure 4: Example translation segmentation from the English-Spanish task ($L_{max} = 8$ and $L_{min} = 4$).

Kneser-Ney smoothing; MERT (Och, 2003) was used to train the log-linear weights of the models; the decoding was performed with a distortion limit of 20 words.

To allow the results to be directly comparable to those in (Finch et al., 2014), the talk level BLEU score (Papineni et al., 2001) was used to evaluate the machine translation quality in all experiments.

3.3 Results

The results for decoding with various values of the latency parameters are shown in Figure 3 for English-French, English-Spanish, English-Arabic, English-Hebrew, English-Russian and English-Chinese. Overall the behavior of the system was quite similar in character to the published results for phrase-based stream decoding for English-Spanish (Kolss et al., 2008b; Finch et al., 2014). The hierarchical system seemed to be more sensitive to small values of minimum latency, and less sensitive to larger values. The results for the more challenging English-Chinese pair were more surprising. In (Finch et al., 2014), the performance of the phrase-based decoder suffered as expected in comparison to pairs of European languages. This was in line with the increase in difficulty of the task due to word order differences. However, in comparison to prior results published on the phrase-based stream decoder, the hierarchical stream decoder seems less affected by the differences between these languages; the curves are higher at the optimal values of minimum latency, and seem less sensitive to its value. The character of the results appears to be very similar to those from English-Spanish. This result is encouraging and suggests that the hierarchical method may be better suited to interpreting between the more difficult language pairs. Figure 4 shows the segmentation given by the system with $L_{max} = 8$ and $L_{min} = 4$, on a sequence of English words which is a subsequence of an unseen test stream of words being decoded.

4 Conclusion

In this paper we propose and evaluate the first hierarchical phrase-based steam decoder. The standard hierarchical phrase-based decoding process generates from the source in left-to-right order, making it naturally suited for incremental decoding. The hierarchical decoder organizes the search process in a chart which can be directly exploited to perform stream decoding. The proposed hierarchical stream decoding process only searches a subset of the search space that is capable of generating useful partial translation hypothesis. This eliminates the necessity for the forced monotonic step necessary in the phrase-based counterpart. Hypotheses that are not useful are discarded, and are therefore not able to compete with useful hypotheses in the search. Additionally, a beneficial side-effect of the pruning of the search space is that decoding speed increased by a factor of approximately 8 over the baseline sentence-by-sentence decoder. Looking to the future, one important benefit of taking a hierarchical approach is that the re-ordering process is made explicit, and in further research we wish to explore the possibility of introducing of new interpretation-oriented rules into the stream decoding process.

Acknowledgements

We would like to thank the reviewers for their valuable comments.

References

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.

Andrew Finch, Xiaolin Wang, and Eiichiro Sumita. 2014. An Exploration of Segmentation Strategies in Stream Decoding. In *Proceedings of the International Workshop on Spoken Language Translation (IWSLT)*, pages 139–142, South Lake Tahoe, USA.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowa, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL 2007): demo and poster sessions*, pages 177–180, Prague, Czeck Republic, June.

Muntsin Kolss, Stephan Vogel, and Alex Waibel. 2008a. Stream decoding for simultaneous spoken language translation. In *Proceedings of Interspeech*, pages 2735–2738, Brisbane, Australia.

Muntsin Kolss, Matthias Wölfel, Florian Kraft, Jan Niehues, Matthias Paulik, and Alex Waibel. 2008b. Simultaneous German-English lecture translation. In *Proceedings of the International Workshop on Spoken Language Translation (IWSLT)*, pages 174–181, Waikiki , Hawai'i, USA.

Evgeny Matusov, Dustin Hillard, Mathew Magimai-Doss, Dilek Hakkani-Tur, Mari Ostendorf, and Hermann Ney. 2007. Improving speech translation with automatic boundary prediction. In *Proceedings of Interspeech*, pages 2449–2452, Antwerp.

Franz J. Och. 2003. Minimum error rate training for statistical machine translation. In *Proceedings of the 41st Meeting of the Association for Computational Linguistics (ACL 2003)*, volume 1, pages 160–167, Sapporo, Japan.

Yusuke Oda, Graham Neubig, Sakriani Sakti Tomoki Toda, and Satoshi Nakamura. 2014. Optimizing segmentation strategies for simultaneous speech translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, Baltimore, USA, June. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2001. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 311–318, Morristown, NJ, USA. Association for Computational Linguistics.

Maryam Siahbani, Ramtin Mehdizadeh Seraj, Baskaran Sankaran, and Anoop Sarkar. 2014. Incremental translation using hierarchichal phrase-based translation system. In *Spoken Language Technology Workshop (SLT), 2014 IEEE*, pages 71–76. IEEE.

Vivek Kumar Rangarajan Sridhar, John Chen, Srinivas Bangalore, Andrej Ljolje, and Rathinavelu Chengalvarayan. 2013. Segmentation strategies for streaming speech translation. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics – Human Language Technologies (HLT-NAACL)*, pages 230–238, Atlanta, USA.

Huihsin Tseng, Pichuan Chang, Galen Andrew, Daniel Jurafsky, and Christopher Manning. 2005. A conditional random field word segmenter for sighan bakeoff 2005. In *Proceedings of the Fourth SIGHAN Workshop on Chinese Language Processing*, volume 171. Jeju Island, Korea.

Taro Watanabe, Hajime Tsukada, and Hideki Isozaki. 2006. Left-to-right target generation for hierarchical phrase-based translation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, ACL-44, pages 777–784, Stroudsburg, PA, USA. Association for Computational Linguistics.

Rule Selection with Soft Syntactic Features for String-to-Tree Statistical Machine Translation

Fabienne Braune and **Nina Seemann** and **Alexander Fraser**
CIS, Ludwig-Maximilians-Universität München
Oettingenstraße 67, 80538 München, Germany
[braunefe|seemanna]@ims.uni-stuttgart.de
fraser@cis.uni-muenchen.de

Abstract

In syntax-based machine translation, rule selection is the task of choosing the correct target side of a translation rule among rules with the same source side. We define a discriminative rule selection model for systems that have syntactic annotation on the target language side (string-to-tree). This is a new and clean way to integrate soft source syntactic constraints into string-to-tree systems as features of the rule selection model. We release our implementation as part of Moses.

1 Introduction

Syntax-based machine translation is well known for its ability to handle non-local reordering. Syntax-based models either use linguistic annotation on the source language side (Huang, 2006; Liu et al., 2006), target language side (Galley et al., 2004; Galley et al., 2006) or are syntactic in a structural sense only (Chiang, 2005). Recent shared tasks have shown that systems integrating information on the target language side, also called *string-to-tree* systems, achieve the best performance on several language pairs (Bojar et al., 2014). At the same time, soft syntactic features significantly improve the translation quality of hierarchical systems (Hiero) as shown in (Marton et al., 2012; Chiang, 2010; Liu et al., 2011; Cui et al., 2010). Improving the performance of string-to-tree systems through the integration of soft syntactic constraints on the source language side is therefore an interesting task.

So far, all approaches on this topic include soft syntactic constraints into the rules of string-to-tree (Zhang et al., 2011; Huck et al., 2014) or string-to-dependency (Huang et al., 2013) systems and define heuristics to determine to what extent these constituents match the syntactic structure of the source sentence. We propose a novel way to integrate soft syntactic constraints into a string-to-tree system. We define a discriminative rule selection model for string-to-tree machine translation. We consider rule selection as a multi-class classification problem where the task is to select the correct target side of a rule given its source side as well as contextual information about the source sentence and the considered rule. So far, such models have been applied to systems without syntactic annotation on the target language side. He et al. (2008), He et al. (2010) and Cui et al. (2010) apply such rule selection models to hierarchical machine translation, Liu et al. (2008) to tree-to-string systems and Zhai et al. (2013) to systems based on predicate argument structures. When target side syntactic annotations are taken into account, the task of rule selection has to be reformulated (see Section 2) while the same type of model can be used in approaches without target annotations. This work is the first attempt to define a rule selection model for a string-to-tree system. We make our implementation publicly available as part of Moses.[1]

We show in Section 2 that string-to-tree rule selection is different from the hierarchical case addressed by previous work and define our rule selection model. In Section 3 we present the training procedure before providing a proof-of-concept evaluation in Section 4.

2 Rule selection for string-to-tree SMT

2.1 String-to-tree machine translation

We present string-to-tree machine translation as implemented in Moses (which is the framework that we use). String-to-tree rules have the form $X/A \rightarrow \langle \alpha, \gamma, \sim \rangle$. On the source language side,

[1] We use the string-to-tree component of Moses (Williams and Koehn, 2012; Hoang et al., 2009) in which we integrate the high-speed classifier Vowpal Wabbit http://hunch.net/~vw/.

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1095–1101,
Lisbon, Portugal, 17-21 September 2015.

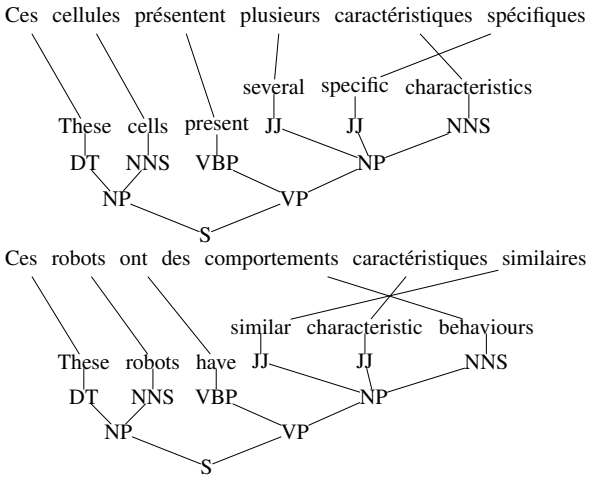


Figure 1: Word-aligned sentence pairs with target-side parse.

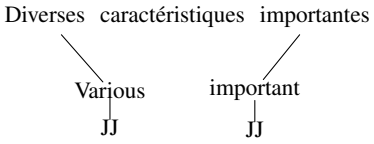


Figure 2: Partial translation during decoding.

all non-terminals have the unique label X while on the target language side non-terminals are annotated with syntactic labels $n_t \in N_t$. The left-hand side X/A consists of source and target non-terminals. In the right hand side (rhs), α is a string of source terminal symbols and the non-terminal X. The string γ consists of target terminals and non-terminals $n_t \in N_t$. The alignment $\sim$ is a one-to-one correspondence between source and target non-terminal symbols. String-to-tree rules are extracted from pairs of strings and trees as exemplified in Figure 1. Rules r_1 and r_2 are example rules extracted from this data.

(r_1) $X/NP \rightarrow \langle\ X_1$ caractéristiques X_2, $JJ_1\ JJ_2$ characteristics $\rangle$
(r_2) $X/NP \rightarrow \langle\ X_1$ caractéristiques X_2, NNS_1 characteristic $JJ_2\ \rangle$

During decoding, CYK+ chart parsing (Chappelier et al., 1998) with cube pruning and language model scoring is performed on an input sentence such as F below. Each time a rule is applied to the input sentence, candidate target trees are built. Figure 2 shows the partial translations built after the segments *Diverses* and *importantes* have been decoded. Given these partial translations, rule r_1 can be applied in a further decoding step.

F (Diverses)$_{X1}$ caractéristiques (importantes)$_{X2}$ n'ont pas été prises en compte.
(Various)$_{X_1}$ ***characteristics*** *(important)*$_{X_2}$ *were not considered.*

2.2 String-to-tree rule selection

Rule selection is the problem of selecting the rule with the correct target side among rules with the same source side. For hierarchical machine translation (Hiero), the rule selection problem consists of choosing, among r_3 and r_4, the rule that correctly applies to F (r_3 in our example).

(r_3) $X/X \rightarrow \langle\ X_1$ caractéristiques X_2, $X_1\ X_2$ characteristics $\rangle$
(r_4) $X/X \rightarrow \langle\ X_1$ caractéristiques X_2, X_1 characteristic $X_2\ \rangle$

Rule selection models disambiguate between these rules using context information about the source sentence and the shape of the rules.

In string-to-tree machine translation, the rule selection problem is different. Because the decoding process is guided by target side syntactic annotation, partial trees built during decoding must be considered when new rules are applied. For instance, when a rule is selected to translate sentence F given the partial translations in Figure 2, then the non-terminals in the target side of this rule must match the constituents selected so far. Consequently, rules r_1 and r_2 (Section 2.1) are not competing during rule selection.[2] Competing rules for r_1 would be r_5 and r_6 below.

(r_5) $X/NP \rightarrow \langle\ X_1$ caractéristiques X_2, JJ_1 properties $JJ_2\ \rangle$
(r_6) $X/NP \rightarrow \langle\ X_1$ caractéristiques X_2, $JJ_1\ JJ_2$ features $\rangle$

For consistency with decoding, we redefine the rule selection problem for the string-to-tree case. In this setup, it is the task of disambiguating rules with the same source side *and aligned target non-terminals*. As a consequence, our rule selection model (presented next) is not only normalized over the source rhs of the rules but also takes target non-terminals into account. The default rule scoring procedure for string-to-tree rules implemented in Moses uses the same normalization as we do. However, Williams and Koehn (2012) propose to normalize string-to-tree rules over the source rhs only.

[2]This is because their target side non-terminals are different.

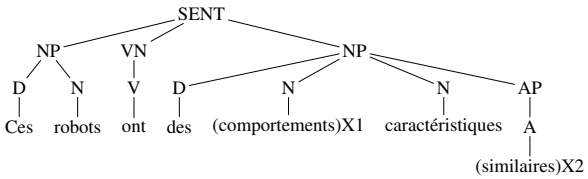


Figure 3: French sentence with input parse tree.

2.3 Rule selection model

We denote string-to-tree rules with $X/A \rightarrow \langle \alpha, \gamma, \sim \rangle$, as in Section 2.1. By $\tilde{N}t_t$, we denote target non-terminals with their alignment to source non-terminals.[3] $C(f, \alpha)$ is context information in the source sentence f and the source side α. $R(\alpha, \gamma)$ represents features on string-to-tree rules. The rule selection model estimates $P(\gamma \mid C(f, \alpha), R(\alpha, \gamma), \alpha, \tilde{N}t_t)$ and is normalized over the set G' of candidate target sides γ' for a given α *and* $\tilde{N}t_t$. The function $GTO : \alpha \rightarrow G'$ generates, given the source side α and target non-terminals $\tilde{N}t_t$, the set G' of all corresponding target sides γ'. The estimated distribution can be written as:

$$P(\gamma \mid C(f,\alpha), R(\alpha,\gamma), \alpha, \tilde{N}t_t) = \frac{\exp(\sum_i \lambda_i h_i(C(f,\alpha), R(\alpha,\gamma), \alpha, \tilde{N}t_t)))}{\sum_{\gamma' \in GTO(\alpha, \tilde{N}t_t)} \exp(\sum_i \lambda_i h_i(C(f,\alpha), R(\alpha,\gamma'), \alpha, \tilde{N}t_t))}$$

In the same fashion as (Cui et al., 2010) do for the hierarchical case, we define a global rule selection model instead of a model that is local to the source side of each rule.

To illustrate the feature templates $C(f, \alpha)$ and $R(\alpha, \gamma)$ of our rule selection model, we suppose that rule r_1 has been extracted from the French sentence in Figure 3. The syntactic features are:

- Does α match a constituent: *no_match*
- Type of matched constituent: *None*
- Lowest parent of unmatched constituent: *NP*
- Span width covered by α: *3*

The rule internal features are:

- Source side α: *X1_caractéristiques_X2* (one feature)
- Target side γ: *JJ1_JJ2_characteristics*
- Aligned terminals in α and γ: *caractéristiques↔characteristics*
- Aligned non-terminals in α and γ: *X1↔JJ1_X2↔JJ2*
- Best baseline translation probability: *Most_Frequent*

Our rule selection model is integrated in the Moses string-to-tree system as an additional feature of the log-linear model.

[3]For rule r_1,r_5 and r_6, $\tilde{N}t_t$ would be JJ_1 and JJ_2.

3 Model Training

We create training examples using the rule extraction procedure in (Williams and Koehn, 2012).[4] We begin by generating a rule-table using this procedure. Then, each time a rule $r : X/A \rightarrow \langle \alpha, \gamma, \sim \rangle$ can be extracted from the training data, we generate a new training example. The target side γ of the extracted rule is a positive instance and gets a loss of 0. To generate negative samples, we collect all rules $r_2, \ldots, r_n$ that have the same source language side as r as well as the same aligned target non-terminals $\tilde{N}t_t$. Each of these rules is a negative example and gets a cost of 1. As an example, suppose that rule r_1 introduced in Section 2.1 has been extracted from the training example in Figure 1. The target side "JJ_1 JJ_2 characteristics" is a correct class and gets a cost of 0. The target side of all other rules having the same source side *and aligned target non-terminals*, such as rule r_5 and r_6, are incorrect classes.

For model training, we use the cost-sensitive one-against-all-reduction (Beygelzimer et al., 2005) of Vowpal Wabbit (VW).[5] We avoid overfitting to training data by employing early stopping once classifier accuracy decreases on a held-out dataset.[6]

4 Experiments

4.1 Experimental Setup

Our baseline system is a syntax-based system with linguistic annotation on the target language side (*string-to-tree*). We use the version implemented in the Moses open source toolkit (Hoang et al., 2009; Williams and Koehn, 2012) with standard parameters. Rule extraction is performed as in (Galley et al., 2004) with rule composition (Galley et al., 2006; DeNeefe et al., 2007). Non-lexical unary rules are removed (Chung et al., 2011) and scope-3 pruning (Hopkins and Langmead, 2010) is performed. Rule scoring is done using relative frequencies normalized over the source rhs and aligned non-terminals in the target rhs. The contrastive system is the same string-to-tree system but augmented with our rule selection model as a feature of the log-linear model.

[4]Which is based on (Galley et al., 2004; Galley et al., 2006; DeNeefe et al., 2007).

[5]Specifically, the label dependent version of *Cost Sensitive One Against All* which uses classification.

[6]We use the development set which is also used for MIRA tuning.

System	science	medical	news
Baseline	34.06	49.87	18.35
Contrastive	34.36	49.57	18.59

Table 2: String-to-tree system evaluation results.

We evaluate the baseline and our global model on three domains: (1) news, (2) medical, and (3) science. The training data for news is taken from Europarl-v4. Development and test sets are from the news translation task of WMT 2009 (Callison-Burch et al., 2009). For medical we use the biomedical data from EMEA (Tiedemann, 2009). Since this is a parallel corpus only, we first removed duplicate sentences and then constructed development and test sets by randomly selecting sentence pairs. As training data for science we use the scientific abstracts data provided by Carpuat et al. (2013). Table 1 gives an overview of the corpora sizes.

Berkeley parser (Petrov et al., 2006) is used to parse the English side of each parallel corpus (for string-to-tree rule extraction) as well as for parsing the French source side (for feature extraction). We trained a 5-gram language model on the English side of each training corpus using the SRI Language Modeling Toolkit (Stolcke, 2002). We train the model in the standard way and generate word alignments using GIZA++. After training, we reduced the number of translation rules by only keeping the 30-best rules with the same source side according to the direct rule translation rule probability. Our rule selection model was trained with VW. All systems were tuned using batch MIRA (Cherry and Foster, 2012). We measured the overall translation quality with 4-gram BLEU (Papineni et al., 2002), which was computed on tokenized and lowercased data for all systems. Statistical significance is computed with the pairwise bootstrap resampling technique of Koehn (2004).

4.2 Results

Table 2 displays the BLEU scores for our experiments. On science and news, small improvements are achieved while for medical a small decrease is observed. None of these differences is statistically significant.

An analysis of the system outputs for each domain showed that the small improvements are due to the fact that in string-to-tree systems there is not enough ambiguity between competing rules during decoding. To support this conjecture, we first analyzed rule diversity by looking at the negative samples collected during training example acquisition. In a second step, we compared the results of the string-to-tree systems in Table 2 with a system where the translation rules are much more ambiguous. To this aim, we applied our approach to a hierarchical system in the same line as (Cui et al., 2010). Finally, we further tested the ability of our system to disambiguate between competing rules by training a model on the concatenation of all domains.

4.3 Analysis of Rule Diversity

The amount of competing rules during decoding can be estimated by looking at the negative samples collected for each training example. This analysis showed that the diversity of rules containing non-terminal symbols is limited. We present rules q_1 to q_3 (taken from science) to illustrate the poor diversity observed in our training examples.

(q_1) $X/PP \rightarrow \langle$ à X_1 X_2 éventail X_3, to DT_1 JJ_2 variety $PP_3 \rangle$
(q_2) $X/PP \rightarrow \langle$ à X_1 X_2 éventail X_3, to DT_1 JJ_2 range $PP_3 \rangle$
(q_3) $X/PP \rightarrow \langle$ à X_1 X_2 éventail X_3, to DT_1 JJ_2 array $PP_3 \rangle$

Rules q_1 to q_3 are the only rules with source side *à* X_1 X_2 *éventail* X_3. This number is very low given that the source side contains three non-terminal symbols out of which two are adjacent. Moreover, the difference between these rules is limited to the lexical translation of *éventail*. This lack of diversity is due to the constraint that competing string-to-tree rules must have the same aligned non-terminal symbols, which is taken into account when collecting negative samples. In other words, the ambiguity between translation rules in a string-to-tree system is heavily restricted by the target side syntax.

The observed lack of diversity could be minimized by allowing rules with the same source rhs to have different aligned target non-terminals. In this perspective, rule scoring should be done by normalizing over the source rhs only as in Williams and Koehn (2012). The rule selection model in Section 2.3 should then be redefined and normalized over all rules with the same source rhs. Another way to improve rule diversity would be to remove target non-terminals and use preference

	news	medical	science
training data	4th EuroParl corpus	(Tiedemann, 2009)	(Carpuat et al., 2013)
training data size	149,986 sentence pairs	111,081 sentence pairs	139,199 sentence pairs
development size	1,025 sentences	2,000 sentences	2,907 sentences
test size	1,026 sentences	1,999 sentences	3,915 sentences

Table 1: Overview of the sizes of the three domains.

System	science	medical	news
Baseline	31.22	48.67	17.28
Contrastive	**32.27**	**49.66**	17.38

Table 3: Hierarchical system evaluation results. The results in bold are statistically significant improvements over the Baseline (at confidence $p < 0.05$).

grammars as in Huck et al. (2014).

4.4 Comparison with Hierarchical Rule Selection

We applied our approach in a hierarchical phrase-based setting (Hiero). To this end, we trained 3 Hiero baseline systems and 3 Hiero systems augmented with our rule selection model on the data given in Section 4.1. The results of these experiments are shown in Table 3. Our augmented system largely outperforms the baselines. Interestingly, hierarchical rule selection significantly helps on the medical and scientific domain but still yields results that are significantly lower than those of the string-to-tree systems. This indicates that systems with target side syntax better disambiguate than hierarchical models with improved rule selection. Overall, we find the results of both types of systems promising and we will consider how to introduce more diversity into the rules of string-to-tree systems.

4.5 Concatenation of Training Data

In order to further evaluate the ability of our model to disambiguate string-to-tree rules, we trained a system using the concatenated training data of all 3 domains as presented in Section 4.1. This global model was then used to tune and decode using the development and test data of each domain. The results in Table 4 show that even on concatenated data our rule selection model does not improve over the baseline.

System	science	medical	news
Baseline	33.78	49.48	19.12
Contrastive	33.87	49.14	19.00

Table 4: String-to-tree system evaluation results with concatenated training data.

5 Conclusion and future work

We presented the first attempt to define a rule selection model with syntactic features for string-to-tree machine translation. We have shown that in order to be applied to the string-to-tree case, the rule selection problem must be redefined. An extensive evaluation on French-English translation tasks for different domains has shown that rule selection cannot significantly improve string-to-tree systems. An analysis of rule diversity and an empirical comparison with hierarchical rule selection indicate that the low improvements are due to the fact that the ambiguity between string-to-tree rules is too small to be improved with a rule selection model. In future work, we will use different techniques to improve the diversity of the string-to-tree rules considered during decoding in our system.

Acknowledgements

We thank all members of the DAMT team of the 2012 JHU Summer Workshop. We are especially grateful to Hal Daumé III and Ales Tamchyna for their ongoing support in the implementation of our system. We also thank Andreas Maletti for his shared expertise on tree grammars. This project has received funding from the European Unions Horizon 2020 research and innovation programme under grant agreement No 644402 (HimL) and the DFG grant *Models of Morphosyntax for Statistical Machine Translation (Phase 2)*, which we gratefully acknowledge.

References

Alina Beygelzimer, John Langford, and Bianca Zadrozny. 2005. Weighted one-against-all. In

AAAI, pages 720–725.

Ondrej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Johannes Leveling, Christof Monz, Pavel Pecina, Matt Post, Herve Saint-Amand, Radu Soricut, Lucia Specia, and Aleš Tamchyna. 2014. Findings of the 2014 workshop on statistical machine translation. In *Ninth Workshop on Statistical Machine Translation*, WMT, pages 12–58, Baltimore, Maryland.

Chris Callison-Burch, Philipp Koehn, Christof Monz, and Josh Schroeder. 2009. Findings of the 2009 Workshop on Statistical Machine Translation. In *Proc. 4th Workshop on Statistical Machine Translation*, pages 1–28.

Marine Carpuat, Hal Daumé III, Katharine Henry, Ann Irvine, Jagadeesh Jagarlamudi, and Rachel Rudinger. 2013. Sensespotting: Never let your parallel data tie you to an old domain. In *Proc. ACL*.

Jean-Cédric Chappelier, Martin Rajman, et al. 1998. A generalized cyk algorithm for parsing stochastic cfg. *TAPD*, 98(133-137):5.

Colin Cherry and George Foster. 2012. Batch tuning strategies for statistical machine translation. In *Proc. NAACL*.

David Chiang. 2005. Hierarchical phrase-based translation. In *Proc. ACL*.

David Chiang. 2010. Learning to translate with source and target syntax. In *Proc. ACL*.

Tagyoung Chung, Licheng Fang, and Daniel Gildea. 2011. Issues concerning decoding with synchronous context-free grammars. In *Proc. ACL*.

Lei Cui, Dongdong Zhang, Mu Li, Ming Zhou, and Tiejun Zhao. 2010. A joint rule selection model for hierarchical phrase-based translation. In *Proc. ACL*.

Steve DeNeefe, Kevin Knight, Wei Wang, and Daniel Marcu. 2007. What can syntax-based mt learn from phrase-based mt. In *Proc. EMNLP*.

Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. 2004. What's in a translation rule? In *Proc. HLT-NAACL*.

Michel Galley, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve Deneefe, Wei Wang, and Ignacio Thayer. 2006. Scalable inference and training of context-rich syntactic translation models. In *Proc. ACL*.

Zhongjun He, Qun Liu, and Shouxun Lin. 2008. Improving statistical machine translation using lexicalized rule selection. In *Proc. COLING*.

Zhongjun He, Yao Meng, and Hao Yu. 2010. Maximum entropy based phrase reordering for hierarchical phrase-based translation. In *Proc. EMNLP*.

Hieu Hoang, Philipp Koehn, and Adam Lopez. 2009. A unified framework for phrase-based, hierarchical, and syntax-based statistical machine translation. In *In Proceedings of the International Workshop on Spoken Language Translation (IWSLT*.

Mark Hopkins and Greg Langmead. 2010. Scfg decoding without binarization. In *Proc. EMNLP*.

Zhongqiang Huang, Jacob Devlin, and Rabih Zbib. 2013. Factored soft source syntactic constraints for hierarchical machine translation. In *Proc. EMNLP*.

Liang Huang. 2006. Statistical syntax-directed translation with extended domain of locality. In *In Proc. AMTA 2006*.

Mathias Huck, Hieu Hoang, and Philipp Koehn. 2014. Preference grammars and soft syntactic constraints for ghkm syntax-based statistical machine translation. In *Proc. SSST-8*.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proc. EMNLP*.

Yang Liu, Qun Liu, and Shouxun Lin. 2006. Tree-to-string alignment template for statistical machine translation. In *Proc. ACL*.

Qun Liu, Zhongjun He, Yang Liu, and Shouxun Lin. 2008. Maximum entropy based rule selection model for syntax-based statistical machine translation. In *Proc. EMNLP*.

Lemao Liu, Tiejun Zhao, Chao Wang, and Hailong Cao. 2011. A unified and discriminative soft syntactic constraint model for hierarchical phrase-based translation. In *Proceedings of the 13th Machine Translation Summit*, pages 253–261.

Yuval Marton, David Chiang, and Philip Resnik. 2012. Soft syntactic constraints for arabic—english hierarchical phrase-based translation. *Machine Translation*, 26:137–157.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proc. ACL*.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proc. ACL*.

Andreas Stolcke. 2002. Srilm - an extensible language modeling toolkit. In *Proceedings of the International Conference on Spoken language Processing*.

Jörg Tiedemann. 2009. News from opus : A collection of multilingual parallel corpora with tools and interfaces. In *Recent Advances in Natural Language Processing V*, volume V, pages 237–248. John Benjamins.

Philip Williams and Philipp Koehn. 2012. Ghkm rule extraction and scope-3 parsing in moses. In *Proc. WMT*.

Feifei Zhai, Jiajun Zhang, Yu Zhou, and Chengqing Zong. 2013. Handling ambiguities of bilingual predicate-argument structures for statistical machine translation. In *Proc. ACL*.

Jiajun Zhang, Feifei Zhai, and Chengqing Zong. 2011. Augmenting string-to-tree translation models with fuzzy use of source-side syntax. In *Proc. EMNLP*.

Motivating Personality-aware Machine Translation

Shachar Mirkin*
IBM Research - Haifa
Mount Carmel, Haifa
31905, Israel
shacharm@il.ibm.com

Scott Nowson, Caroline Brun, Julien Perez
Xerox Research Centre Europe
6 chemin de Maupertuis
38240 Meylan, France
firstname.surname@xrce.xerox.com

Abstract

Language use is known to be influenced by personality traits as well as by socio-demographic characteristics such as age or mother tongue. As a result, it is possible to automatically identify these traits of the author from her texts. It has recently been shown that knowledge of such dimensions can improve performance in NLP tasks such as topic and sentiment modeling. We posit that machine translation is another application that should be personalized. In order to motivate this, we explore whether translation preserves demographic and psychometric traits. We show that, largely, both translation of the source training data into the target language, and the target test data into the source language has a detrimental effect on the accuracy of predicting author traits. We argue that this supports the need for personal and personality-aware machine translation models.

1 Introduction

Computational personality recognition is garnering increasing interest with a number of recent workshops exploring the topic (Celli et al., 2014; Tkalčič et al., 2014). The addition of personality as target traits in the PAN Author Profiling challenge in 2015 (Rangel et al., 2015) is further evidence. Such user modeling – when performed on text – is built on a long-standing understanding that language use is influenced by socio-demographic characteristics such as age, gender, education level or mother tongue and personality traits like agreeableness or openness (Tannen, 1990; Pennebaker et al., 2003).

In this work we explore *multilingual* user modelling. The motivation is not only to enable modeling in multiple languages, but also to enable modeling multilingual users who may express different sides of their personality in each language. One way to address multilinguality in this context is to create models separately in each language, and then fuse the resulting models. However, labelled data of this nature, particularly in non-English languages, is often not available. Personality labelling is time consuming, requiring the completion of psychometric questionnaires which may be considered invasive by many. An alternative is the use of machine translation (MT) to bootstrap corpora in resource poor languages, and to translate the user's content into a single language before modeling. Translated text, either manually or automatically generated, is known to have different characteristics than native text. Yet, MT was shown to be of use within traditional NLP tasks such as sentiment analysis (Balahur and Turchi, 2012). We explore the utility of MT for classification of demographic and personality traits.

MT models, even domain-specific, are user-generic. Thus, the linguistic signals of user traits which are conveyed in the original language may not be preserved over translation. In other words, the attributes on which we wish to rely for modelling may be lost. This concern is perhaps most observable with gender, a trait of the speaker that is encoded in the morphology of many languages, though not in English. Gendered translation was the topic of research for many years. However, the gender of the author is largely ignored by MT systems, and specifically statistical ones, that would often arbitrarily (or rather statistically-based) translate into one gender form or another. Other demographic and personality traits have not yet been investigated.

One way to address this concern is *personalized* translation, or *author-aware* translation.[1] The first step toward this goal would be to consider the author traits in the model. Such an approach has already shown to be useful for several NLP tasks (Volkova et al., 2013; Hovy, 2015). However, before embarking on this challenging task, we explore if the above concerns are founded by addressing the research question: does MT has an impact on the classification of demographic and personality traits?

2 Background

Oberlander and Nowson (2006) motivated their study of computational personality recognition by arguing that automatically understanding an author's personality would permit the personalization of sentiment analysis. Such personalized NLP has recently been

*This work was mostly done while the first author was at Xerox Research Centre Europe.

[1] In this work we investigate MT awareness of the author; in (Mirkin and Meunier, 2015) we address the task of reader-aware MT.

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1102–1108,
Lisbon, Portugal, 17-21 September 2015.

explored by Volkova et al. (2013). They incorporated age and gender features for sentiment analysis, and show improvements in three different languages. Hovy (2015) extends this work to other languages and NLP tasks. Using demographically-informed word embeddings, they show improvements in sentiment analysis, topic classification and trait detection. None of these works, however, addressed cross-lingual issues.

Yet, personality projection goes beyond automatic detection of traits – there is also human perception to be considered. The casual reader may not be aware of personality related linguistic cues. Yet, studies have shown that traits can be reliably detected following cold readings of texts from unknown authors (Mehl et al., 2006; Gill et al., 2012) without such explicit knowledge. Although personality projection in different languages is under-explored, it has been shown that the relationship between language use and personality traits varies between domains (Nowson and Gill, 2014). Thus, while it would seem that there are cues which translate directly between languages, this may not always be the case. In English, for example, women tend to use first-person pronouns such as "*I*" more than men (Newman et al., 2008); but this does not guarantee a gender-based usage difference for, say, "*je*" in French. Furthermore, what happens with more subtle, language-specific indicators of personality? For instance, Nowson (2006) showed that use of contractions (e.g. *don't* vs. *do not*) is a marker for the Agreeableness trait. These different forms do not naturally translate into other languages. It is doubtful that even a human translator would always pay attention to such subtleties. In investigating whether these cues are preserved when a text is translated, we are also beginning to address the question of consistency in cues between languages.

MT systems do not explicitly consider demographic or personality traits. Instead, they often exploit "in-domain" data to create translation models that are adapted for the domain of interest (Lu et al., 2007; Foster et al., 2010; Axelrod et al., 2011; Gong et al., 2012; Mirkin and Besacier, 2014). The term "domain" has a wide interpretation in the MT literature and may refer to topic, dialect, genre or style (Chen et al., 2013). However, to the best of our knowledge, MT domain adaptation does not extend to consider demographic or personality traits of the author. Gender in translation has been researched extensively; in human translation studies, it has been shown that the gender of translators impact the translation. In SMT, phrase-based models (Koehn et al., 2003) can correctly pick-up translations of gender-inflected words, and rule-based MT systems and factored models (Koehn and Hoang, 2007) provide more explicit ways for gender translation. Yet, most SMT systems are unaware of the gender of the author, neither in the training nor in the test data, and are therefore unable to adapt their translation beyond the local inflectional level; in particular when no such evidence exists, as in English. To a much greater extent, this is the case with other demographics, such as age, and with personality traits.

3 Methodology

3.1 Hypothesis

The hypothesis of our broader vision is that personalized MT or author-aware translation is an important necessity. We believe the human understanding of translated text (of its explicit and implicit meanings, of its author and of the full context) would be improved if author traits are better conveyed.

In order to motivate this future work, this paper explores a supporting hypothesis: that author traits are not conveyed accurately under machine translation. We assess this by investigating whether trait detection performs as well on translated data as on native text.

3.2 Experimental Framework

To explore our hypothesis, we require data in multiple languages which is labelled with socio-demographic or personality traits. Using English as the base language (as typically the most resource-rich language in NLP studies), we perform three comparative experiments on several non-English ("foreign") corpora. In these experiments we train a classification model:

1. Using only foreign language data. This provides a baseline, as no translated data is used.
2. Augmenting the foreign training data with English data translated into the foreign language. Here, the goal is to assess a scenario where translations from a resource-rich language supplement scarce training data in the foreign language, under the assumption that more training data can be beneficial.
3. Translating the foreign test data into English and classifying it using a model trained on the English data. This allows us to explore another practical scenario, where an English model already exists and we wish to use it to classify data from another language for which we do not have a robust model.

For this task we use the data from the 2015 PAN workshop (Rangel et al., 2015) which is labelled for author gender and personality traits. For more details see Section 4.1. We also wish to explore if any affect was due strictly to the use of MT or to translation (or language change) in general. The PAN corpus is multi-lingual but does not contain parallel data. Such parallel corpora, however, are not typically labelled with the type of author information we wish to investigate. Therefore we required such a corpus to which we could easily add labels. For these we used a selection of TED talks which we labelled for gender (see Section 4.2). The full details of our approach to text processing, translation and classification can be found in our technical paper at the PAN workshop (Nowson et al., 2015); in the interests of space a compressed version is presented here.

3.3 Preprocessing and feature extraction

We use the multilingual parser described by Ait-Mokhtar et al. (2001) to preprocess the texts and extract a wide range of features. The parser has been customized to handle social media data, e.g. by detecting hashtags, mentions, and emoticons. For English, we have integrated a normalization dictionary by Han et al. (2012) in the preprocessing. The English and French grammars also include a polarity lexicon to recognize sentiment bearing words or expressions. The features we extract include: 1-, 2-, 3-grams of surface, normalized and lemmatized forms; part-of-speech tagged forms, and n-grams of POS; named entities (places, persons, organization, dates, time expressions), emoticons, hashtags, mentions and URLs.

3.4 Learning framework

To train classification models we first prune features with a frequency threshold. Next, the remaining set of features is compressed using truncated singular value decomposition (SVD). SVD (Golub and Reinsch, 1970) is a widely used technique in sparse dataset situations. This method copes with noise present in the data by extracting the principal dimensions describing the data and projecting the data to a latent space. In the truncated version, a low-rank approximation, all but the top-k dimensions are removed. The result is a dense, low-dimension representation of the data. Finally, ensemble models (Schapire, 1990; Dietterich, 2000) are used to predict trait values: for gender, we use the majority vote of 10 classifiers; for each personality trait, we use the mean of 10 regression estimators.

3.5 Machine translation models

We created standard machine translation models between English and each one of Spanish, Italian, French and Dutch. The details are described below.

Parallel corpora We wished to use the same setting for all language pairs. To that end, we chose parallel corpora that are available for all of them, namely Europarl (Koehn, 2005)[2] and WIT3 (Cettolo et al., 2012), from the IWSLT 2014 evaluation campaign (Cettolo et al., 2014). WIT3, consisting of spoken-language transcripts, represents an in-domain corpus for the TED dataset and a "near-domain" for PAN. The data consisted of approximately 2 million parallel sentences for each language pair, with 50 million tokens for each language. The Europarl corpus comprised more than 90% of that data. The two corpora were concatenated to create the training data for the MT models.

Translation System Moses (Koehn et al., 2007), an open-source phrase-based MT system,[3] was used to train translation models and translate the data.

Preprocessing We used the standard Moses tools to preprocess the data, including tokenization, lowercasing and removal of sentence pairs where at least one of the sentences is empty or longer than 80 tokens.

Recasing and Language models We used SRILM (Stolcke, 2002) version 1.7.1 to train 5-gram language models on the target side of the parallel corpus, with modified Kneser-Ney discounting (Chen and Goodman, 1996). A recasing model was trained from the same corpus, with a 3-gram KenLM language model (Heafield, 2011).

Tuning We tuned the translation models using MERT (Och, 2003), using the development set of the above mentioned campaign (*dev2010*), consisting of 887 sentence pairs for each language pair.

Translation and post-processing Each of the tweets of the PAN training set was preprocessed in the same fashion as the training data. It was then translated with the trained model of the corresponding language pair, and finally underwent quick post-processing, namely recasing and detokenization.

4 Data

Personality-tagged datasets in multiple languages are scarce. We used two datasets, with content from twitter and TED talks, as described in this section.

4.1 PAN

The first corpus we used was the data of the PAN 2015 Author Profiling task (Rangel et al., 2015), drawn from Twitter (PAN15). For each user, the data consists of tweets (average $n = 100$) and gold standard labels: gender (Male or Female), and personality. The labels are provided by the author, with scores on five traits being calculated via self-assessment responses to the short Big 5 test, BFI-10 (Rammstedt and John, 2007)), then normalized between -0.5 and +0.5. Table 1 shows the volume of data per language for the training set.

Language	Authors	Tweets
English (en)	152	14166
Spanish (es)	100	9879
Italian (it)	38	3687
Dutch (nl)	34	3350

Table 1: Number of authors and tweets across the four languages of the PAN dataset.

4.2 TED

The PAN15 data allows us to assess personality projection in multilingual data. In addition to exploring automatic translation, we wish to compare with manual translation. We turned to TED talks[4] for such comparative evaluation. We chose the English-French lan-

[2] Version 7, www.statmt.org/europarl

[3] We used version 3.0, downloaded on 16 Feb 2015 from www.statmt.org/moses.

[4] www.ted.com

guage pair, because French is not a language in PAN15, but also due to the difficulties in obtaining such data, as described below.

4.2.1 TED English-French

We use data of the MT track of the IWSLT 2014 Evaluation Campaign, which includes parallel corpora from transcripts of TED talks. The English-French (*en-fr*) corpus consists of 1415 talks, with approximately 190k sentence pairs and 3 million tokens for each of the source and target sides (before preprocessing). We annotated the gender of each speaker with a simple web interface. Any talk with multiple speakers or where the majority is not a speech (e.g. a performance) is discarded. After discarding 59 talks, 1012 (75%) were annotated as male and 344 (25%) as female.[5]

4.2.2 TED French-English

The WIT3 data seems to also include data in the *fr-en* direction. However, in practice, TED hosts only talks in English and all foreign to English corpora were collected from the translated versions of the site. We therefore turned to TEDx[6] for *fr-en* data. TEDx are independent TED-style events, often including talks in languages other than English. Unlike *en-fr*, there is no easily accessible parallel data available for *fr-en*, where the source is native French. We applied the following procedure to collect the necessary data. We used the Google YouTube Analytics API[7] to search for videos of talks in French. We have extracted the list of TEDx events in France and their dates via `www.tedxenfrance.fr`.[8] Each event-name and year is used as a query in YouTube, e.g. "TEDx Paris 2011". For each talk, we download the **manual** French and English subtitles, i.e. the transcript and the translation, respectively. These files were annotated using the same process and criteria described above. This resulted with a small corpus (TED61$_{fr-en}$) of 61 talks of which 32 are annotated as male and 29 as female.

TED61$_{en}$ In order to account for any potential effect of length, we created a subset of the *en-fr* corpus, that is of the same size of the *fr-en* dataset. We matched files from the French side of the *en-fr* corpus to each of those in the *fr-en* for gender and length (in tokens). The French *en-fr* files were truncated after the nearest line break to the desired size; the corresponding English *en-fr* files were truncated at the same point.

5 Experiments

It has been shown that standard approaches to gender classification on English texts can be sub-optimal for non-English language data (Ciot et al., 2013). However, state-of-the-art classification results are not our focus; rather, our intention is to understand the impact of translation on classification of socio-demographic and personality traits. Therefore, we fix our models with a set of parameters selected via cross-validation (CV) on the native language: the occurrence threshold is set to 5 and the SVD dimensionality to 500.

5.1 PAN

For each of the three non-English languages of PAN15 we train classification models as explained in Section 3: using the original training data, adding training data translated from English, and translating the test data into English to use the English-trained model.

Results can be seen in Table 2. For the majority of the traits, the native results outperform both translation settings, in some cases by considerable margin. The assumption posited earlier that more training data is be beneficial appears not to have held up in this context. The alternative scenario seems to be doing even worse.

The most distinct results are perhaps the accuracy of gender prediction (for which each corpora is balanced, thus a baseline of 50%). One explanation may be that the translation is done from and into English, which does not express gender via morphology, in contrast to Italian and Spanish. An interesting exception is that adding translated English texts into Dutch considerably improves performance. This may be explained by the lesser expression of gender in Dutch morphology, much like English. In this instance it appears that adding more data – when translation is between two gender-agnostic languages – does indeed help. For both Italian and Dutch, English adds a very substantial amount of data; the outcomes, however, are opposite.

5.2 TED

Though the TED data is currently only labelled for gender, it allows us to make comparisons between manual and machine translation. First we explored if there were gender signals in the English corpus which a classifier could uncover. For this, we performed leave-one-out CV on each of the following three versions: native English, manually and machine translated into French. The results, presented in Table 3, show that some gender signal is lost between manual and machine translation. In the manual translation, the translator, who is aware of the speaker's gender is able to reflect that through morphological and lexical cues, that exist in French much more than in English. The MT's ability to project these features properly was more limited. Note that the results between English and French are not directly comparable, since any text classification on different languages may yield different results.

One interesting observation is the low performance relative to the baseline and that of PAN15. Though we do not discuss this in detail here, we suspect this may be an effect of genre muting (Argamon et al., 2003; Herring and Paolillo, 2006).

Next, we explore another setting: The English corpus is modified to exclude the speakers of the

[5] Annotation data is available at `cm.xrce.xerox.com`

[6] `www.ted.com/watch/tedx-talks`

[7] `developers.google.com/youtube`

[8] Accessed on 23/2/15.

Training	Test	Gender (%)	Extraverted	Stable	Agreeable	Conscientious	Open
en	*en*	80.5	0.029	0.050	0.030	0.021	0.021
es	*es*	**82.8**	**0.023**	**0.035**	0.024	0.024	0.025
es + *en→es*	*es*	75.1	0.031	0.042	0.024	**0.021**	**0.020**
en	*es→en*	62.6	0.032	0.048	**0.021**	0.027	0.030
it	*it*	**80.0**	**0.009**	0.028	0.020	**0.010**	0.019
it + *en→it*	*it*	59.1	0.013	0.028	**0.016**	0.014	**0.016**
en	*it→en*	61.7	0.031	0.063	0.020	0.024	0.025
nl	*nl*	67.6	**0.008**	**0.014**	**0.014**	**0.007**	**0.010**
nl + *en→nl*	*nl*	**74.0**	0.011	0.032	0.020	0.015	0.012
en	*nl→en*	53.2	0.028	0.076	0.018	0.017	0.023

Table 2: Cross-validation results on PAN15 for the settings as per Section 5.1. Gender is measured in accuracy; the remaining traits as mean squared error. Bold highlights the best result. English results are included for comparison.

Corpus	English	→French
Native	63.1	
Manual		66.6
MT		62.7

Table 3: Gender CV accuracy (%) on the English TED dataset, when translated manually and automatically.

Corpus	Accuracy (%)
TED61_{en}	58.3
TED61_{fr-en} (Manual)	60.0
TED61_{fr-en} (MT)	52.1

Table 4: Results when classifying gender on native, manually translated and machine translated English texts, from 61 TEDx and TED talks.

TED61_{en} dataset (leaving $n = 1295$ speakers), and this data is used to train a classification model. We then test our three smaller English datasets on this model: TED61_{en}, TED61_{fr-en} manual translated and TED61_{fr-en} machine translated. The results in this case are more comparable since we use the same model for all datasets and since their sizes are similar. The classification results are presented in Table 4. Again, signal is lost in automatic translation in comparison to manual translation. Interestingly, the manual translation scores higher than the native English, as if the translators are adding more gender indications to the text. Further analysis is required to clarify whether this is indeed a consequence of the manual translation or an artifact of the setting.

Author-aware translation may be viewed as a human-centric domain adaptation task: we can consider the two genders as two different domains, and apply domain adaptation techniques to train a better-suited model for each one. To assess this approach, we conducted a set of experiments with standard domain adaptation techniques for *en-fr*, including: separating the translation models and the language models by gender in various configurations, using only the target gender's training data from WIT3 (on top of the Europarl data), and separating tuning sets by gender. We split the IWSLT test sets by gender, and applied on each part the respective gender's model before concatenating the translations to compute a BLEU (Papineni et al., 2002) score. Unfortunately, none of these models showed a significant improvement, if at all, in comparison to our baseline that used both genders together. This suggests that alternative methods should be used for our task. We cannot say, however, that these results are conclusive; specifically, one difficulty in our experiments was obtaining enough female data, due to the relative small number of female speakers in WIT3.

6 Discussion

We are interested in understanding the impact which the consideration of author traits might have on automatic translation, in order to preserve projection of those traits in a target language. However, it is first necessary to understand the inverse: the effect of current translation approaches on the computational recognition of these traits. In the initial studies reported here we have explored two corpora: one of social media data; one of scripted speeches. Although linguistic signals of traits are weaker in the latter case, so far it appears that machine translation is detrimental to the automatic recognition of these traits. Though we have tried to account for as many confounding factors in this work as we could – particularly the availability of data – naturally there are still some open questions, and some obvious next steps. We fixed the learning parameters across languages and traits for comparative reasons, but would independent optimization provide better results? What is the impact of the translation quality on the subsequent classification performance? We would also like to understand the true relationship between linguistic features and traits across languages, along with how native speakers naturally observe these traits. Overall, however, we are encouraged to pursue our goal of personalized machine translation.

Acknowledgments

We would like to gratefully acknowledge the comments and feedback we received from the EMNLP reviewers.

References

Salah Ait-Mokhtar, Jean-Pierre Chanod, and Claude Roux. 2001. A multi-input dependency parser. In *Proceedings of IWPT*.

Shlomo Argamon, Moshe Koppel, Jonathan Fine, and Anat Rachel Shimoni. 2003. Gender, genre, and writing style in formal written texts. *Text*, 23(3):321–346.

Amittai Axelrod, Xiaodong He, and Jianfeng Gao. 2011. Domain adaptation via pseudo in-domain data selection. In *Proceedings of EMNLP*.

Alexandra Balahur and Marco Turchi. 2012. Multilingual sentiment analysis using machine translation? In *Proceedings of the 3rd Workshop in Computational Approaches to Subjectivity and Sentiment Analysis*, WASSA '12, pages 52–60, Stroudsburg, PA, USA. Association for Computational Linguistics.

Fabio Celli, Bruno Lepri, Joan-Isaac Biel, Daniel Gatica-Perez, Giuseppe Riccardi, and Fabio Pianesi. 2014. The workshop on computational personality recognition 2014. In *Proceedings of the ACM International Conference on Multimedia*, pages 1245–1246. ACM.

Mauro Cettolo, Christian Girardi, and Marcello Federico. 2012. WIT^3: Web inventory of transcribed and translated talks. In *Proceedings of EAMT*.

Mauro Cettolo, Jan Niehues, Sebastian Stüker, Luisa Bentivogli, and Marcello Federico. 2014. Report on the 11th iwslt evaluation campaign, iwslt 2014. In *Proceedings of the eleventh International Workshop on Spoken Language Translation (IWSLT), Lake Tahoe, CA*, pages 2–17.

Stanley F. Chen and Joshua Goodman. 1996. An empirical study of smoothing techniques for language modeling. In *Proceedings of the 34th annual meeting on Association for Computational Linguistics (ACL 1996)*, pages 310–318.

Boxing Chen, Roland Kuhn, and George Foster. 2013. Vector space model for adaptation in statistical machine translation. In *Proceedings of ACL*.

Morgane Ciot, Morgan Sonderegger, and Derek Ruths. 2013. Gender inference of Twitter users in non-English contexts. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1136–1145, Seattle, Washington, USA, October. Association for Computational Linguistics.

Thomas G. Dietterich. 2000. Ensemble methods in machine learning. In *Proceedings of the First International Workshop on Multiple Classifier Systems*, pages 1–15. Springer-Verlag.

George F. Foster, Cyril Goutte, and Roland Kuhn. 2010. Discriminative instance weighting for domain adaptation in statistical machine translation. In *Proceedings of EMNLP*.

Alastair J. Gill, Carsten Brockmann, and Jon Oberlander. 2012. Perceptions of alignment and personality in generated dialogue. In *Proceedings of the Seventh International Natural Language Generation Conference*, INLG '12, pages 40–48. Association for Computational Linguistics.

G. H. Golub and C. Reinsch. 1970. Singular value decomposition and least squares solutions. *Journal of Numerical Mathematics*, 14:403–420.

Li Gong, Aurélien Max, and François Yvon. 2012. Towards contextual adaptation for any-text translation. In *Proceedings of IWSLT*.

Bo Han, Paul Cook, and Timothy Baldwin. 2012. Automatically constructing a normalisation dictionary for microblogs. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL 2012)*, pages 421–432, Jeju Island, Korea.

Kenneth Heafield. 2011. KenLM: faster and smaller language model queries. In *Proceedings of the EMNLP 2011 Sixth Workshop on Statistical Machine Translation*, pages 187–197, Edinburgh, Scotland, United Kingdom, July.

Susan C. Herring and John C. Paolillo. 2006. Gender and genre variation in weblogs. *Journal of Sociolinguistics*, 10(4):439–459, September.

Dirk Hovy. 2015. Demographic factors improve classification performance. In *53rd Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Philipp Koehn and Hieu Hoang. 2007. Factored translation models. In *EMNLP-CoNLL*, pages 868–876.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, pages 48–54, Stroudsburg, PA, USA. Association for Computational Linguistics.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of ACL Demo and Poster Sessions*.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *Proceedings of MT Summit*.

Yajuan Lu, Jin Huang, and Qun Liu. 2007. Improving statistical machine translation performance by training data selection and optimization. In *Proceedings of EMNLP-CoNLL*.

Matthias R. Mehl, Samuel D. Gosling, and James W. Pennebaker. 2006. Personality in its natural habitat: manifestations and implicit folk theories of personality in daily life. *Journal of personality and social psychology*, 90(5):862–877, May.

Shachar Mirkin and Laurant Besacier. 2014. Data selection for compact adapted SMT models. In *Proceedings of the eleventh biennial conference of the Association for Machine Translation in the Americas (AMTA-2014)*, Vancouver, Canada, Oct.

Shachar Mirkin and Jean-Luc Meunier. 2015. Personalized machine translation: Predicting translational preferences. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Lisbon, Portugal. Association for Computational Linguistics.

Matthew L Newman, Carla J Groom, Lori D Handelman, and James W Pennebaker. 2008. Gender differences in language use: An analysis of 14,000 text samples. *Discourse Processes*, 45(3):211–236.

Scott Nowson and Alastair J. Gill. 2014. Look! Who's Talking? Projection of Extraversion Across Different Social Contexts. In *Proceedings of WCPR14, Workshop on Computational Personality Recognition at ACMM (22nd ACM International Conference on Multimedia)*.

Scott Nowson, Julien Perez, Caroline Brun, Shachar Mirkin, and Claude Roux. 2015. XRCE Personal Language Analytics Engine for Multilingual Author Profiling. In *Working Notes Papers of the CLEF 2015 Evaluation Labs*, CEUR Workshop Proceedings. CLEF and CEUR-WS.org, September.

Scott Nowson. 2006. *The Language of Weblogs: A study of genre and individual differences.* Ph.D. thesis, University of Edinburgh.

Jon Oberlander and Scott Nowson. 2006. Whose thumb is it anyway? Classifying author personality from weblog text. In *Proceedings of COLING/ACL-06: 44th Annual Meeting of the Association for Computational Linguistics and 21st International Conference on Computational Linguistics*, July.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1 (ACL 2003)*, ACL '03, pages 160–167.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of ACL*, Philadelphia, Pennsylvania, USA.

James W Pennebaker, Kate G Niederhoffer, and Matthias R Mehl. 2003. Psychological aspects of natural language use: Our words, our selves. *Annual Review of Psychology*, 54:547–577, January.

Beatrice Rammstedt and Oliver P. John. 2007. Measuring personality in one minute or less: A 10-item short version of the big five inventory in english and german. *Journal of Research in Personality*, 41(1):203–212, February.

Francisco Rangel, Fabio Celli, Paolo Rosso, Martin Potthast, Benno Stein, and Walter Daelemans. 2015. Overview of the 3rd Author Profiling Task at PAN 2015. In *Working Notes Papers of the CLEF 2015 Evaluation Labs*, CEUR Workshop Proceedings. CLEF and CEUR-WS.org, September.

Robert Schapire. 1990. The strength of weak learnability. *Journal of Machine Learning Research*, 5.

Andreas Stolcke. 2002. SRILM - an extensible language modeling toolkit. In *Proceedings Int. Conf. on Spoken Language Processing (INTERSPEECH 2002)*, pages 257–286.

Deborah Tannen. 1990. *You Just Don't Understand: Women and Men in Conversation*. Harper Collins, New York.

Marko Tkalčič, Berardina De Carolis, Marco de Gemmis, Ante Odić, and Andrej Košir. 2014. Preface: Empire 2014. In *Proceedings of the 2nd Workshop Emotions and Personality in Personalized Services (EMPIRE 2014)*. CEUR-WS.org, July.

Svitlana Volkova, Theresa Wilson, and David Yarowsky. 2013. Exploring demographic language variations to improve multilingual sentiment analysis in social media. In *EMNLP*, pages 1815–1827. ACL.

Trans-gram, Fast Cross-lingual Word-embeddings

$\text{rey}_{\text{es}} - \text{Mann}_{\text{de}} = \text{regina}_{\text{it}} - \text{femme}_{\text{fr}}$

Jocelyn Coulmance
105 rue La Fayette
75010 Paris
joc@proxem.com

Jean-Marc Marty *
105 rue La Fayette
75010 Paris
jmm@proxem.com

Guillaume Wenzek *
105 rue La Fayette
75010 Paris
guw@proxem.com

Amine Benhalloum
105 rue La Fayette
75010 Paris
aba@proxem.com

Abstract

We introduce *Trans-gram*, a simple and computationally-efficient method to simultaneously learn and align word-embeddings for a variety of languages, using only monolingual data and a smaller set of sentence-aligned data. We use our new method to compute aligned word-embeddings for twenty-one languages using English as a pivot language. We show that some linguistic features are aligned across languages for which we do not have aligned data, even though those properties do not exist in the pivot language. We also achieve state of the art results on standard cross-lingual text classification and word translation tasks.

1 Introduction

Word-embeddings are a representation of words with fixed-sized vectors. It is a distributed representation (Hinton, 1984) in the sense that there is not necessarily a one-to-one correspondence between vector dimensions and linguistic properties. The linguistic properties are distributed along the dimensions of the space.

A popular method to compute word-embeddings is the Skip-gram model (Mikolov et al., 2013a). This algorithm learns high-quality word vectors with a computation cost much lower than previous methods. This allows the processing of very important amounts of data. For instance, a 1.6 billion words dataset can be processed in less than one day.

Several authors came up with different methods to align word-embeddings across two languages (Klementiev et al., 2012; Mikolov et al., 2013b; Lauly et al., 2014; Gouws et al., 2015).

In this article, we introduce a new method called Trans-gram, which learns word embeddings aligned across many languages, in a simple and efficient fashion, using only sentence alignments rather than word alignments. We compare our method with previous approaches on a cross-lingual document classification task and on a word translation task and obtain state of the art results on these tasks. Additionally, word-embeddings for twenty-one languages are learned simultaneously - to our knowledge - for the first time, in less than two and a half hours. Furthermore, we illustrate some interesting properties that are captured such as cross-lingual analogies, e.g $\vec{\text{rey}}_{\text{es}} - \vec{\text{Mann}}_{\text{de}} + \vec{\text{femme}}_{\text{fr}} \approx \vec{\text{regina}}_{\text{it}}$ which can be used for disambiguation.

2 Review of Previous Work

A number of methods have been explored to train and align bilingual word-embeddings. These methods pursue two objectives: first, similar representations (i.e. spatially close) must be assigned to similar words (i.e. "semantically close") within each language - this is the **mono-lingual objective**; second, similar representations must be assigned to similar words across languages - this is the **cross-lingual objective**.

The simplest approach consists in separating the mono-lingual optimization task from the cross-lingual optimization task. This is for example the case in (Mikolov et al., 2013b). The idea is to separately train two sets of word-embeddings for each language and then to do a parametric estimation of the mapping between word-embeddings across languages. This method was further extended by (Faruqui and Dyer, 2014). Even though those algorithms proved to be viable and fast, it is not clear whether or not a simple mapping between whole languages exists. Moreover, they require word alignments which are a rare and expensive resource.

*These authors contributed equally.

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1109–1113,
Lisbon, Portugal, 17-21 September 2015.

Another approach consists in focusing entirely on the cross-lingual objective. This was explored in (Hermann and Blunsom, 2013; Lauly et al., 2014) where every couple of aligned sentences is transformed into two fixed-size vectors. Then, the model minimizes the Euclidean distance between both vectors. This idea allows processing corpus aligned at sentence-level rather than word-level. However, it does not leverage the abundance of existing mono-lingual corpora .

A popular approach is to jointly optimize the mono-lingual and cross-lingual objectives simultaneously. This is mostly done by minimizing the sum of mono-lingual loss functions for each language and the cross-lingual loss function. (Klementiev et al., 2012) proved this approach to be useful by obtaining state-of-the-art results on several tasks. (Gouws et al., 2015) extends their work with a more computationally-efficient implementation.

3 From Skip-Gram to Trans-Gram

3.1 Skip-gram

We briefly introduce the Skip-gram algorithm, as we will need it for further explanations. Skip-gram allows to train word embeddings for a language using mono-lingual data. This method uses a dual representation for words. Each word w has two embeddings: a target vector, $\vec{w}$ ($\in \mathbb{R}^D$), and a context vector, $\overleftarrow{w}$ ($\in \mathbb{R}^D$). The algorithm tries to estimate the probability of a word w to appear in the context of a word c. More precisely we are learning the embeddings $\vec{w}, \overleftarrow{c}$ so that: $\sigma(\vec{w} \cdot \overleftarrow{c}) = P(w|c)$ where σ is the sigmoid function.

A simplified version of the loss function minimized by Skip-gram is the following:

$$J = \sum_{s \in C} \sum_{w \in s} \sum_{c \in s[w-l:w+l]} -\log \sigma(\vec{w} \cdot \overleftarrow{c}) \quad (1)$$

where C is the set of sentences constituting the training corpus, and $s[w-l:w+l]$ is a word window on the sentence s centered around w. For the sake of simplicity this equation does not include the "negative-sampling" term, see (Mikolov et al., 2013a) for more details.

Skip-gram can be seen as a materialization of the distributional hypothesis (Harris, 1968): "Words used in similar contexts have similar meanings". We will now see how to extend this idea to cross-lingual contexts.

3.2 Trans-gram

In this section we introduce Trans-gram, a new method to compute aligned word-embeddings for a variety of languages.

Our method will minimize the summation of mono-lingual losses and cross-lingual losses. Like in BilBOWA (Gouws et al., 2015), we use Skip-gram as a mono-lingual loss. Assuming we are trying to learn aligned word vectors for languages e (e.g. English) and f (e.g. French), we note J_e and J_f the two mono-lingual losses.

In BilBOWA, the cross-lingual loss function is a distance between bag-of-words representations of two aligned sentences. But as (Levy and Goldberg, 2014) showed that the Skip-gram loss function extracts interesting linguistic features, we wanted to use a loss function for the cross-lingual objective that will be closer to Skip-gram than BilBOWA.

Therefore, we introduce a new task, Trans-gram, similar to Skip-gram. Each English sentence s_e in our aligned corpus $A_{e,f}$ is aligned with a French sentence s_f. In Skip-gram, the context picked for a target word w_e in a sentence s_e is the set of words c_e appearing in the window centered around w_e: $s_e[w_e - l : w_e + l]$. In Trans-gram, the context picked for a target word w_e in a sentence s_e will be all the words c_f appearing in s_f. The loss can thus be written as:

$$\Omega_{e,f} = \sum_{(s_e,s_f) \in A_{e,f}} \sum_{w_e \in s_e} \sum_{c_f \in s_f} -\log \sigma(\vec{w_e} \cdot \overleftarrow{c_f}) \quad (2)$$

This loss isn't symmetric with respect to the languages. We, therefore, use two cross-lingual objectives: $\Omega_{e,f}$ aligning e's target vectors and f's context vectors and $\Omega_{f,e}$ aligning f's target vectors and e's context vectors. By comparison BilBOWA only aligns e's target vectors and f's target vectors. The figure 1 illustrates the four objectives.

Notice that we make the assumption that the meaning of a word is uniformly distributed in the whole sentence. This assumption, although a naive one, gave us in practice excellent results. Also our method uses only sentence-aligned corpus and not word-aligned corpus which are rarer.

To add a third language i (e.g. Italian), we just have to add 3 new objectives (J_i, $\Omega_{e,i}$ and $\Omega_{i,e}$) to the global loss. If available we could also add $\Omega_{f,i}$ or $\Omega_{i,f}$ but in our case we only used corpora aligned with English.

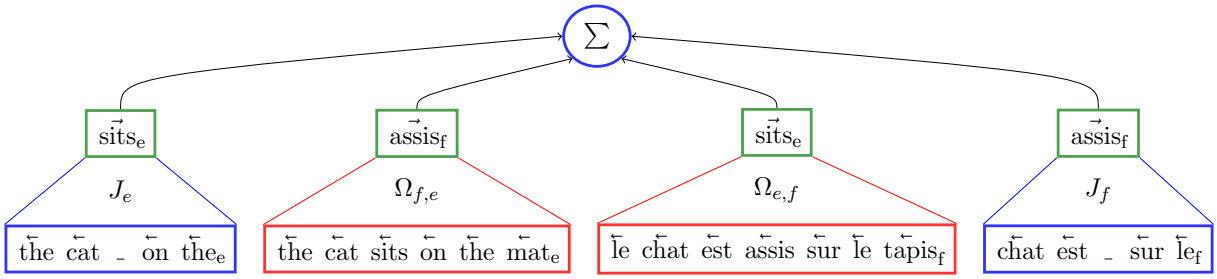


Figure 1: The four partial objectives contributing to the alignment of English and French: a Skip-gram objective per language (J_e and J_f) over a window surrounding a target word (blue) and two Trans-gram objectives ($\Omega_{e,f}$ and $\Omega_{f,e}$) over the whole sentence aligned with the sentence from which the target word is extracted (red).

4 Implementation

In our experiments, we used the Europarl (Koehn, 2005) aligned corpora. Europarl-v7 has two peculiarities: firstly, the corpora are aligned at sentence-level; secondly each pair of languages contains English as one of its members: for instance, there is no French/Italian pair. In other words, English is used as a pivot language. No bi-lingual lexicons nor other bi-lingual datasets aligned at the word level were used.

Using only the Europarl-v7 texts as both mono-lingual and bilingual data, it took 10 minutes to align 2 languages, and two and a half hours to align the 21 languages of the corpus, in a 40 dimensional space on a 6 core computer. We also computed 300 dimensions vectors using the Wikipedia extracts provided by (Al-Rfou et al., 2013) as monolingual data for each language. The training time was 21 hours.

5 Experiments

5.1 Reuters Cross-lingual Document Classification

We used a subset of the English and German sections of the Reuters RCV1/RCV2 corpora (Lewis and Li, 2004) (10000 documents each), as in (Klementiev et al., 2012), and we replicated the experimental setting. In the English dataset, there are four topics: CCAT (Corporate/Industrial), ECAT (Economics), GCAT (Government/Social), and MCAT (Markets). We used these topics as our labels and we only selected documents labeled with a single topic. We trained our classifier on the articles of one language, where each document was represented using an IDF weighted sum of the vectors of its words, we then tested it on the articles of the other language. The classifier used was an averaged perceptron, and we used the implementation from (Klementiev et al., 2012)[1]. The word vectors were computed on the Europarl-v7 parallel corpus with size 40 like other methods. For this task only the target vectors where used.

We report the percentage precision obtained with our method, in comparison with other methods, in Table 1. The table also include results obtained with 300 dimensions vectors trained by Trans-gram with the Europarl-v7 as parallel corpus and the Wikipedia as mono-lingual corpus. The previous state of the art results were detained (Gouws et al., 2015) with BilBOWA and (Lauly et al., 2014) with their Bilingual Auto-encoder model. This model learns word embeddings during a translation task that uses an encoder-decoder approach. We also report the scores from Klementiev et al. who introduced the task and the BiCVM model scores from (Hermann and Blunsom, 2013).

The results show an overall significant improvement over the other methods, with the added advantage of being computationally efficient.

5.2 P@k Word Translation

Next we evaluated our method on a word translation task, introduced in (Mikolov et al., 2013b) and used in (Gouws et al., 2015). The words were extracted from the publicly available WMT11[2] corpus. The experiments were done for two sets of translation: English to Spanish and Spanish to English. (Mikolov et al., 2013b) extracted the top $6K$ most frequent words and translated them with Google Translate. They used the top $5K$ pairs to train a translation matrix, and evaluated their method on the remaining $1K$. As our English and

[1]Thanks to S. Gouws for providing this implementation
[2]http://www.statmt.org/wmt11/

Method	En → De	De → En	Speed-up in training time
Klementiev et al.	77.6%	71.1%	×1
Bilingual Auto-encoder	**91.8%**	72.8%	×3
BiCVM	83.7%	71.4%	×320
BilBOWA	86,5%	75%	×800
Trans-gram	87,8%	**78,7%**	×600
Trans-gram (size 300 vectors EP+WIKI)	91,1%	**78,4%**	

Table 1: Comparison of Trans-gram with various methods for Reuters English/German classification

Method	En → Es P@1	En → Es P@5	Es → En P@1	Es → En P@5
Edit distance	13%	18%	24%	27%
Bing	55%		71%	
Translation Matrix	33%	35%	51%	52%
BilBOWA	39%	44%	**51%**	55%
Trans-gram	**45%**	**61%**	47%	**62%**

Table 2: Results on the translation task

Spanish vectors are already aligned we don't need the $5K$ training pairs and use only the $1K$ test pairs.

The reported score, the translation precision $P@k$, is the fraction of test-pairs where the target translation (Google Translate) is one of the k translations proposed by our model. For a given English word, w, our model takes its target vectors $\vec{w}$ and proposes the k closest Spanish word using the co-similarity of their vectors to $\vec{w}$. We compare ourselves to the "translation matrix" method and to the BilBowa aligned vectors. We also report the scores obtained by a trivial algorithm that uses edit-distance to determine the closest translation and by the Bing Translator service.

6 Interesting properties

6.1 Cross-lingual disambiguation

We now present the task of cross-lingual disambiguation as an example of possible uses of aligned multilingual vectors. The goal of this task is to find a suitable representation of each sense of a given polysemous word. The idea of our method is to look for a language in which the undesired senses are represented by unambiguous words and then to perform some arithmetic operation.

Let's illustrate the process with a concrete example: consider the French word "train", $\mathrm{train}_{\mathrm{fr}}$. The three closest Polish words to $\vec{\mathrm{train}}_{\mathrm{fr}}$ translate in English into "now", "a train" and "when". This seems a poor matching. In fact, $\mathrm{train}_{\mathrm{fr}}$ is polysemous. It can name a line of railroad cars, but it is also used to form progressive tenses. The French "Il est *en train de* manger" translates into "he is eat*ing*", or in Italian "*sta* mangiando".

As the Italian word "sta" is used to form progressive tenses, it's a good candidate to disambiguate $\mathrm{train}_{\mathrm{fr}}$. Let's introduce the vector $\vec{v} = \vec{\mathrm{train}}_{\mathrm{fr}} - \vec{\mathrm{sta}}_{\mathrm{it}}$. Now the three polish words closest to $\vec{v}$ translate in English into "a train", "a train" and "railroad". Therefore $\vec{v}$ is a better representation for the railroad sense of $\mathrm{train}_{\mathrm{fr}}$.

6.2 Transfer of linguistic features

Another interesting property of the vectors generated by Trans-gram is the transfer of linguistic features through a pivot language that does not possess these features.

Let's illustrate this by focusing on Latin languages, which possess some features that English does not, like rich conjugations. For example, in French and Italian the infinitives of "eat" are $\mathrm{manger}_{\mathrm{fr}}$ and $\mathrm{mangiare}_{\mathrm{it}}$, and the first plural persons are $\mathrm{mangeons}_{\mathrm{fr}}$ and $\mathrm{mangiamo}_{\mathrm{it}}$. Actually in our models we observe the following alignments: $\vec{\mathrm{manger}}_{\mathrm{fr}} \approx \vec{\mathrm{mangiare}}_{\mathrm{it}}$ and $\vec{\mathrm{mangeons}}_{\mathrm{fr}} \approx \vec{\mathrm{mangiamo}}_{\mathrm{it}}$. It is thus remarkable to see that features not present in English match in languages aligned through English as the only pivot language. We also found similar transfers for the genders of adjectives and are currently studying other similar properties captured by Trans-gram.

7 Conclusion

In this paper we provided the following contributions: Trans-gram, a new method to compute cross-lingual word-embeddings in a single word space; state of the art results on cross-lingual NLP tasks; a sketch of a cross-lingual calculus to help disambiguate polysemous words; the exhibition of linguistic features transfers through a pivot-language not possessing those features.

We are still exploring promising properties of the generated vectors and their applications in other NLP tasks (Sentiment Analysis, NER...).

References

Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. 2013. Polyglot: Distributed word representations for multilingual nlp. *arXiv preprint arXiv:1307.1662*.

Manaal Faruqui and Chris Dyer. 2014. Improving vector space word representations using multilingual correlation. Association for Computational Linguistics.

Stephan Gouws, Yoshua Bengio, and Greg Corrado. 2015. Bilbowa: Fast bilingual distributed representations without word alignments. In *Proceedings of the 25th international conference on Machine learning*, volume 15, pages 748–756.

Zellig Sabbettai Harris. 1968. Mathematical structures of language.

Karl Moritz Hermann and Phil Blunsom. 2013. Multilingual distributed representations without word alignment. *arXiv preprint arXiv:1312.6173*.

Geoffrey E Hinton. 1984. Distributed representations.

Alexandre Klementiev, Ivan Titov, and Binod Bhattarai. 2012. Inducing crosslingual distributed representations of words.

Philipp Koehn. 2005. A parallel corpus for statistical machine translation. In *MT Summit*.

Stanislas Lauly, Hugo Larochelle, Mitesh Khapra, Balaraman Ravindran, Vikas C Raykar, and Amrita Saha. 2014. An autoencoder approach to learning bilingual word representations. In *Advances in Neural Information Processing Systems*, pages 1853–1861.

Omer Levy and Yoav Goldberg. 2014. Neural word embedding as implicit matrix factorization. In *Advances in Neural Information Processing Systems*, pages 2177–2185.

Y.; Rose T.; Lewis, D. D.; Yang and F. Li. 2004. Rcv1: A new benchmark collection for text categorization research. In *Journal of Machine Learning Research*.

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

Tomas Mikolov, Quoc V Le, and Ilya Sutskever. 2013b. Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168*.

The Overall Markedness of Discourse Relations

Lifeng Jin and Marie-Catherine de Marneffe
Department of Linguistics
The Ohio State University
{jin, mcdm}@ling.osu.edu

Abstract

Discourse relations can be categorized as continuous or discontinuous in the hypothesis of continuity (Murray, 1997), with continuous relations expressing normal succession of events in discourse such as temporal, spatial or causal. Asr and Demberg (2013) propose a markedness measure to test the prediction that discontinuous relations may have more unambiguous connectives, but restrict the markedness calculation to relations with explicit connectives only. This paper extends their measure to explicit and implicit relations and shows that results from this extension better fit the continuity hypothesis predictions both for the English Penn Discourse (Prasad et al., 2008) and the Chinese Discourse (Zhou and Xue, 2015) Treebanks.

1 Introduction

Discourse relations between units of text are crucial for the production and understanding of discourse. Different taxonomies of discourse relations have been proposed (i.a. Hobbs (1985), Lascarides and Asher (1993) and Knott and Sanders (1998)). One taxonomy is based on deictic continuity (Segal et al., 1991; Murray, 1997): continuity in the sense of Segal et al. (1991) means that the same frame of reference is maintained, for example by subsequent sentences talking about the same event, without a shift in perspective (Asr and Demberg, 2012). For instance, a causal relation such as *I was tired, so I drank a cup of coffee.* is continuous, and adversatives show discontinuous relations: *I drank a cup of coffee but I was still tired.* Other continuous relations include temporal succession, topic succession and so on. The continuity hypothesis predicts that sentences connected by continuous relations are easier to understand than ones connected by discontinuous relations.

Previous work on continuity hypothesis (Maury and Teisserenc, 2005; Cain and Nash, 2011; Hoek and Zufferey, 2015) suggests that discourse connectives are indicators of the continuity of discourse and help the interlocutors predict the level of continuity of upcoming sentences. Segal et al. (1991) propose that connectives which signal discontinuous discourse relations, such as *but*, are the marked ones because they indicate harder-to-comprehend content. Asr and Demberg (2012, 2013) extend this idea to discourse relations, proposing that discourse relations which are discontinuous, or posing a conceptual difficulty (Haspelmath, 2006), may be less explicitly conveyed in text, or more explicitly marked by a connective which unambiguously conveys that specific relation than continuous ones. They propose a new measure called markedness to capture this, but when computed on the Penn Discourse Treebank, results do not fit the continuity theory well. This paper improves on Asr and Demberg (2013)'s measure and shows that the results on the Penn Discourse and the Chinese Discourse Treebanks fit the continuity hypothesis very well.

2 Discourse Treebanks

Penn Discourse Treebank The Penn Discourse Treebank (PDTB) is a corpus of Wall Street Journal articles annotated with discourse relations (Prasad et al., 2008). The discourse relations are organized in a hierarchical structure with three levels: a level 1 (e.g. TEMPORAL), level 1/level 2 (e.g. TEMPORAL.Asynchronous) or level 1/level 2/level 3 (e.g. TEMPORAL.Asynchronous.succession) relation can appear between two clauses within a sentence. Discourse relations with overt discourse connectives are annotated as "Explicit", whereas the relations with no discourse connective are annotated as "Implicit". The "AltLex" category is used when a non-connective expression conveys the relation. Table 1 gives the distribution of the

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1114–1119,
Lisbon, Portugal, 17-21 September 2015.

relation categories in the corpus.

Some connectives are labeled with multiple relations when it was difficult to pinpoint exactly one exact discourse relation for it. We follow Asr and Demberg (2013) and treat these cases as if there are multiple instances of the connective, each with one of the labels it received. This gives us a total of 35,870 relation instances.

	PDTB		CDTB	
Relation Category	**Count**	**%**	**Count**	**%**
Explicit	18459	45.5	1219	22.0
Implicit	16053	39.5	3935	71.1
AltLex	633	1.6	116	2.1
EntRel and NoRel	5464	13.4	264	4.8

Table 1: Relation category counts for the Penn Discourse and Chinese Discourse Treebanks.

Chinese Discourse Treebank The Chinese Discourse Treebank (CDTB, Zhou and Xue 2014) follows the PDTB annotation style and has annotations for 164 documents from Xinhua News. The main difference between CDTB and PDTB is that CDTB has a flat structure of only ten relations compared to the hierarchical relation structure in PDTB. Table 1 gives the distribution of the relation categories.

3 Rethinking the Markedness Measure

To quantify the conceptual difficulty of discourse relations, Asr and Demberg (2013) propose an information-theoretic measure "markedness", which tells us how tightly and uniquely a relation is associated with a connective. The measure uses normalized point-wise mutual information:

$$npmi(r;c) = \frac{\log p(r)p(c)}{\log p(r,c)} - 1 \qquad (1)$$

to get the markedness of a discourse relation:

$$markedness(r) = \sum_c p(c|r)\frac{npmi(r;c)+1}{2} \qquad (2)$$

where r is a relation and c is a discourse connective. Asr and Demberg (2013) propose this measure in the surprisal framework of Levy (2008), and restrict the scope of the data to only Explicit relations in PDTB. We will call this measure with only explicit relations "M-exp". Since surprisal is defined as the probability of a word given previous words and context (3), this restriction on the scope of relations does not seem reasonable. Surprisal is defined as

$$surprisal \propto -\log p(w_i|w_{1\ldots i-1}, CONTEXT) \qquad (3)$$

It can be argued that at the current word w_{i-1}, the distribution of upcoming discourse relations, available in CONTEXT, should play a role in determining the probability of the upcoming word w_i. In the surprisal model, the domain for w_i should be the same as w_{i-1}, which is all possible words. However, if we only calculate the distribution of explicit connectives as proposed by Asr and Demberg (2013), the candidates for w_i will change according to the prediction of whether an explicit relation is coming up or not. If the upcoming relation is an implicit relation, one then has access to a distribution of words without the connectives, whereas if one predicts an explicit relation, then one predicts the next word using a distribution of all the connectives as in M-exp. However surprisal should not be a model of deterministic decision making. It is more likely the case that given CONTEXT, one assigns probabilities to all words given the preceding context, which includes the case where no connective, in other words a zero or null connective, is predicted. The null connective may also be viewed as the probability mass for all the non-connective words predicted by CONTEXT where the connective is predicted to be omitted (Asr and Demberg, 2015).

The markedness measure can be analyzed in terms of point-wise mutual information (*pmi*), indicating the amount of information one relation has for the distribution of the words that follow it (Hume, 2011). Because *pmi* is proportionate to *npmi*, we can rewrite (2) as

$$markedness(r) \propto \sum_c p(c|r)pmi(r;c) \qquad (4)$$

We also have the mutual information measure:

$$I(X;Y) = \sum_y \sum_x p(x,y)pmi(x;y) \qquad (5)$$

For the mutual information of y_i in Y :

$$\begin{aligned} I(X;Y=y_i) &= \frac{\sum_x p(x,y_i)pmi(x;y_i)}{p(y_i)} \\ &= D_{KL}(p(x|y_i) \parallel p(x)) \\ &= \sum_x p(x|y_i)pmi(x;y_i) \\ &\propto markedness(y_i) \end{aligned} \qquad (6)$$

Level 1 Relation	Level 2 Relation	Level 3 Relation	Continuity
TEMPORAL	Asynchronous	precedence	Discontinuous
		succession	Discontinuous
	Synchronous		Ambiguous
CONTIGENCY	Cause		Continuous
	Condition		Unidentified
COMPARISON			Discontinuous
EXPANSION	Instantiation		Continuous
	Restatement		Continuous
	List		Continuous
	Alternative		Discontinuous
	Exception		Discontinuous
	Conjunction		Ambiguous

Table 2: Continuity of relations according to the continuity hypothesis (Asr and Demberg, 2012).

Therefore, the markedness measure can be understood as the Kullback-Leibler divergence of the univariate distribution of X from the conditional distribution of X given y_i (a discourse relation in our case). This shows the influence a relation has on the unigram distribution of words. Discourse relations strongly associated with certain connectives will have larger values of this measure than the ones with a weak association. In the previous discussion of surprisal, we have seen that one may treat the implicit cases as predicting a null connective, which will then expand the domain of X from explicit cases to all implicit and explicit cases. With this setup, we calculate "M-all" using all explicit and implicit relations, with a null connective accompanying all the implicit relations.

Continuity hypothesis has been linked with cognitive difficulties in discourse processing in previous studies (Segal et al., 1991; Murray, 1997). The markedness measure can also be linked to processing difficulties through surprisal theory. Surprisal theory proposes that processing difficulty during sentence processing can be seen as the work incurred by resource allocation during parallel disambiguation (Levy, 2008). If a relation has a high markedness value, it indicates that this relation has a strong influence on the distribution of upcoming candidate words. The stronger the influence is, the higher the resource allocation cost will be for the relation, thus more difficult to process.

4 Results on PDTB

Figure 1 compares the markedness of the PDTB level 1 relations as computed by the M-exp and M-all measures. According to the continuity hypothesis, the TEMPORAL relation is discontinuous (see Table 2 which gives the classification of discourse relations according to the continuity hypothesis). It should therefore have a high markedness value. However, M-exp assigns a low markedness to TEMPORAL, which Asr and Demberg (2013) note is unexpected. They ascribe this to the fact that temporal discourse connectives are often used to mark CONTINGENCY relations. However because of the high counts of Explicit connectives in TEMPORAL, whenever there is a connective that can indicate CONTINGENCY or TEMPORAL, one is more likely to predict TEMPORAL because of fewer null connective cases for TEMPORAL. In surprisal terms, whenever one predicts that there is a TEMPORAL relation next, one will more likely predict that there is an explicit discourse connective signaling the relation.

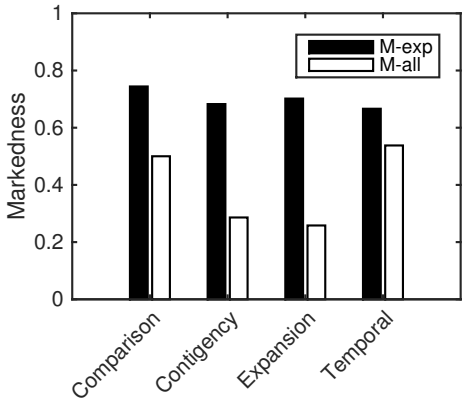


Figure 1: Comparison of markedness measures for PDTB level 1 relations.

EXPANSION is the least marked of all the relations in Figure 1 with M-all. An analysis of the level 2 relations explains this fact. Figure 2 com-

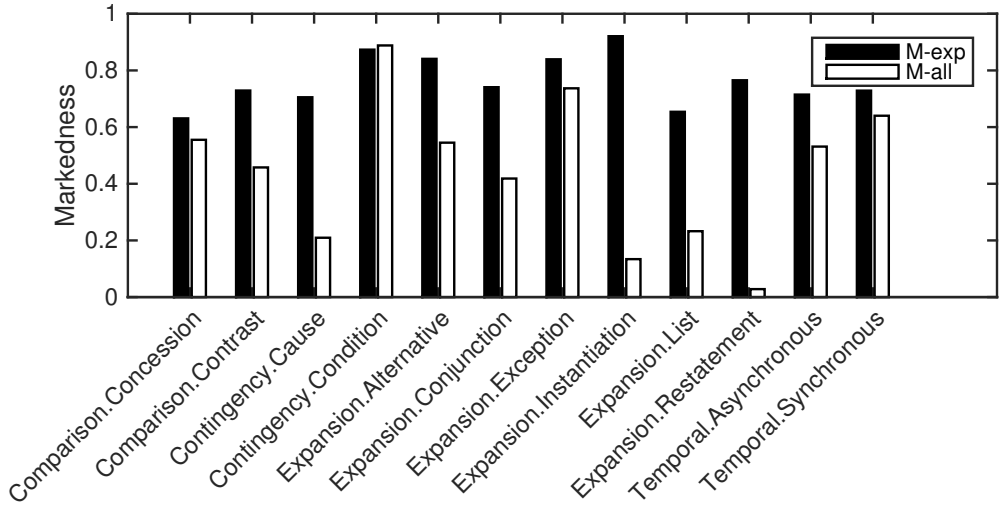


Figure 2: Comparison of markedness measures for PDTB level 2 relations.

pares both measures for the level 2 relations.[1] Using M-all, it is easy to see that discontinuous relations and ambiguous relations are more marked than the continuous ones. In the case of EXPANSION, the level 2 continuous relations are among the least marked ones, which are keeping the overall markedness low. Also, the ones which are discontinuous, especially *Exception*, are rare, so their influence to the overall score for EXPANSION is small. The most frequent relation, *Conjunction*, can be viewed as sometimes continuous and sometimes discontinuous, therefore the overall markedness rating for it is in the middle. All these factors contribute to the lowest markedness for EXPANSION. For CONTINGENCY, *Condition* is not classified as continuous or discontinuous, and it is highly marked, thus driving the overall score high. However if *Condition* is removed, then CONTINGENCY will be the least marked relation at the level 1.

At the level 3, there are two discontinuous relations of interest: *precedence* (e.g. *I had a cup of coffee before I took a bath*) and *succession* (e.g. *I took a bath after I had a cup of coffee*) under the TEMPORAL.Asynchronous relation. Table 3 compares the markedness measures for both relations. Asr and Demberg (2013) mention that there is no significant markedness distinction for them and we can see that *precedence* is slightly more marked than *succession* in M-exp, but the difference is small. M-all however shows that *succession* is more marked than *precedence*, reflecting the fact that *precedence* is easier to understand.

Metric	Precedence	Succession
M-exp	0.799	0.783
M-all	0.494	0.687

Table 3: Precedence and Succession Markedness.

	Arg1-Conn-Arg2		Conn-Arg2-Arg1	
	Implicit	Explicit	Implicit	Explicit
Precedence	567	931	0	55
Succession	171	867	0	234

Table 4: Counts for different argument orders of Precedence and Succession.

For *precedence*, the arguments can be in a normal temporal order, i.e. a forward temporal order (Arg1-Conn-Arg2, e.g. *I had some coffee before I went out.*) or in a backward temporal order (Conn-Arg2-Arg1, e.g. *Before I went out, I had some coffee.*). For *succession*, the temporal order and the argument order are reversed. Table 4 gives the counts in PTDB for both *precedence* and *succession* with different argument orders, showing that Arg1-Conn-Arg2 is the most frequent construction for both relations, which is forward temporal order for *precedence* and backward temporal order for *succession*, despite the fact that both of the relations can follow a forward temporal order. Therefore the results from M-all match the continuity hypothesis prediction that events in for-

[1]Pragmatic relations are not shown due to their small number of occurrences.

ward temporal order are easier to understand, thus less marked, than events in backward temporal order. Asr and Demberg (2012) explain that the relatively high count of Conn-Arg2-Arg1 constructions in *succession* is due to the fact that this construction actually places the events in the forward temporal order. We also notice that *precedence* has a lot more implicit occurrences than *succession*, meaning that inferring a normal temporal relation is much easier than inferring a reversed temporal relation.

5 Results on CDTB and Comparison

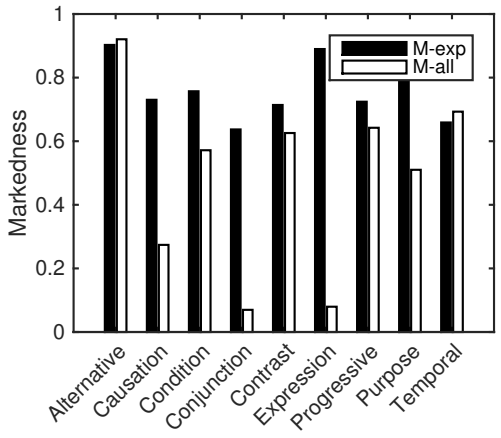


Figure 3: Markedness comparison for CDTB.

The results for CDTB, in Figure 3, shows the same trends as for English: overall, the markedness computed by M-all better fits the continuity hypothesis. EXPANSION is considered by M-exp as the second highest marked relation, whereas the continuity hypothesis predicts it to be one of the lowest marked relation, which is correctly captured by M-all. The reason for it to be low may be that discontinuous relations included by EXPANSION are rare so the frequent continuous relations dominate, just as for English. Using M-exp, CAUSATION is around the middle, but M-all correctly lowers it to the third least marked relation. TEMPORAL is now the second highest marked relation among all relations, as opposed to the second least marked one. Importantly, M-all correctly shows that discourses relations in English and Chinese behave similarly in terms of markedness, which indicates that the continuity hypothesis is valid across languages.

Sanders (2005) proposes the causality-by-default hypothesis, claiming that CAUSATION is the default discourse relation when processing discourse. However, looking at M-all scores for both languages, it is clear that CAUSATION is not the least marked relation in either language. In fact, EXPANSION can actually be seen as the least marked common relation in both languages, which may indicate that EXPANSION is the default discourse relation cross-linguistically, yet more investigations are needed to decide which one in EXPANSION is the default. CONJUNCTION is also among the least marked relations in Chinese, with 89% of its instances being implicit, but in English, CONJUNCTION has an average markedness score. This shows that there are also differences among languages on judgments of continuity of specific relations.

6 Conclusion

The continuity hypothesis predicts that discontinuous discourse relations are less expected than continuous relations and should be more marked. We expand Asr and Demberg (2013)'s measure from explicit relations only to explicit and implicit relations. We show that the results from that expansion fit the predictions of the theory very well, and such evidence demonstrates that discontinuous relations are indeed cognitively more difficult to process. Further we show that such difficulty is consistent across languages, indicating that discourse relations may not be influenced by idiosyncrasies of specific languages. Apart from the markedness measure, Asr and Demberg (2012) proposed an "implicitness" measure, modeling the continuity of a relation using the ratio of implicit cases to all cases. Incorporating explicit and implicit relations into the markedness measure has the advantage of not only providing a single measure but also one which better fits the continuity hypothesis and surprisal theory.

Acknowledgements

We thank William Schuler for productive discussions of the work presented here as well as our anonymous reviewers for their helpful comments.

References

Fatemeh Torabi Asr and Vera Demberg. 2012. Implicitness of discourse relations. In *Proceedings of COLING 2012*, pages 2669–2684.

Fatemeh Torabi Asr and Vera Demberg. 2013. On

the information conveyed by discourse markers. In *Proceedings of the Fourth Annual Workshop on Cognitive Modeling and Computational Linguistics (CMCL)*, pages 84–93. Association for Computational Linguistics.

Fatemeh Torabi Asr and Vera Demberg. 2015. Uniform surprisal at the level of discourse relations: negation markers and discourse connective omission. In *Proceedings of the 11th International Conference on Computational Semantics*, pages 118–128. Association for Computational Linguistics.

Kate Cain and Hannah M. Nash. 2011. The influence of connectives on young readers' processing and comprehension of text. *Journal of Educational Psychology*, 103(2):429–441.

Martin Haspelmath. 2006. Against markedness (and what to replace it with). *Journal of Linguistics*, 42:25–70, 3.

Jerry R Hobbs. 1985. *On the coherence and structure of discourse*. CSLI.

Jet Hoek and Sandrine Zufferey. 2015. Factors influencing the implicitation of discourse relations across languages. In *Proceedings 11th Joint ACL-ISO Workshop on Interoperable Semantic Annotation (isa-11)*, page 39.

Elizabeth Hume. 2011. Markedness. In Marc van Oostendrop, Colin J. Ewen, Elizabeth Hume, and Keren Rice, editors, *The Blackwell Companion to Phonology: Suprasegmental and prosodic phonology*, volume 2, pages 79–106. John Wiley & Sons.

Alistair Knott and Ted J.M. Sanders. 1998. The classification of coherence relations and their linguistic markers: An exploration of two languages. *Journal of Pragmatics*, 30:135–175.

Alex Lascarides and Nicholas Asher. 1993. Temporal interpretation, discourse relations, and common sense entailment. *Linguistics and Philosophy*, 16:437–493.

Roger Levy. 2008. Expectation-based syntactic comprehension. *Cognition*, 106(3):1126–1177.

Pascale Maury and Amélie Teisserenc. 2005. The role of connectives in science text comprehension and memory. *Language and Cognitive Processes*, 20(3):489–512.

John D. Murray. 1997. Connectives and narrative text: The role of continuity. *Memory & Cognition*, 25(2):227–236.

Rashmi Prasad, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Livio Robaldo, Aravind Joshi, and Bonnie Webber. 2008. The Penn Discourse TreeBank 2.0. In *Proceedings of LREC 2008*.

Ted Sanders. 2005. Coherence, causality and cognitive complexity in discourse. In *Proceedings of the Symposium on the Exploration and Modelling of Meaning*.

Erwin M. Segal, Judith F. Duchan, and Paula J. Scott. 1991. The role of interclausal connectives in narrative structuring: Evidence from adults' interpretations of simple stories. *Discourse Processes*, 14(1):27–54.

Yuping Zhou and Nianwen Xue. 2015. The Chinese Discourse TreeBank: a Chinese corpus annotated with discourse relations. *Language Resources and Evaluation*, 49(2):397–431.

Experiments in Open Domain Deception Detection

Verónica Pérez-Rosas and Rada Mihalcea
Computer Science and Engineering
University of Michigan
vrncapr@umich.edu, mihalcea@umich.edu

Abstract

The widespread use of deception in online sources has motivated the need for methods to automatically profile and identify deceivers. This work explores deception, gender and age detection in short texts using a machine learning approach. First, we collect a new open domain deception dataset also containing demographic data such as gender and age. Second, we extract feature sets including n-grams, shallow and deep syntactic features, semantic features, and syntactic complexity and readability metrics. Third, we build classifiers that aim to predict deception, gender, and age. Our findings show that while deception detection can be performed in short texts even in the absence of a predetermined domain, gender and age prediction in deceptive texts is a challenging task. We further explore the linguistic differences in deceptive content that relate to deceivers gender and age and find evidence that both age and gender play an important role in people's word choices when fabricating lies.

1 Introduction

Given the potential ethical and security risks associated with deceitful interactions, it is important to build computational tools able not only to detect deceivers but also to provide insights into the nature of deceptive behaviors. In particular, information related to the demographics of the deceivers could be potentially useful, as recent studies have shown that online users lie frequently about their appearance, gender, age or even education level.

There are multiple scenarios where it would be desirable to identify deceivers' demographics; for instance, identifying the age and gender of SMS senders or Twitter users might help improve parental controls, spam filtering, and user's security and privacy.

In this paper, we present a study on deception detection in an open domain, and also present an analysis of deceptive behavior in association with gender and age. Unlike previous studies, where domain-specific conversational transcripts and reviews have been used, this research targets the identification of deceit in short texts where domain and context are not available. We aim to build deception, age, and gender classifiers using short texts, and also explore the prediction of gender and age in deceptive content. Moreover, we present an analysis of the topics discussed by deceivers given their age and gender based on the assumption that, when lying in an open domain setting, deceivers will show natural bias towards specific topics related to gender and age.

2 Related work

To date, several studies have explored the identification of deceptive content in a variety of domains, including online dating, forums, social networks, and consumer reviews. (Toma and Hancock, 2010) conducted linguistic analyses in online dating profiles and identified correlations between deceptive profiles and self references, negations, and lower levels of words usage. A study for deception detection on essays and product reviews is presented in (Feng et al., 2012). (Ott et al., 2011) addressed the identification of spam in consumer reviews and also studied the human capability of detecting deceptive reviews, which was found not better than chance. In a following study, (Ott et al., 2013) presented an analysis of the sentiment associated to deceitful reviews focusing particularly in those containing negative

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1120–1125,
Lisbon, Portugal, 17-21 September 2015.

sentiment as it largely affects consumer purchase decisions. More recently (Yu et al., 2015) presented a study where authors analyze the role of deception in online networks by detecting deceptive groups in a social elimination-game.

This previous work has shown the effectiveness of features derived from text analysis, which frequently includes basic linguistic representations such as n-grams and sentence counts statistics (Mihalcea and Strapparava, 2009; Ott et al., 2011) and also more complex linguistic features derived from syntactic context free grammar trees and part of speech tags (Feng et al., 2012; Xu and Zhao, 2012). Other studies have focused on deception clues inspired from psychological studies. For instance, following the hypothesis that deceivers might create less complex sentences (DePaulo et al., 2003), researchers have incorporated syntactic complexity measures into the analysis. (Yancheva and Rudzicz, 2013) presented a study based on the analysis of syntactic units and found that syntactic complexity correlates with deceiver's age. Psycholinguistics lexicons, such as Linguistic Inquiry and Word Count (LIWC) (Pennebaker and Francis, 1999), have also been used to build deception models using machine learning approaches (Mihalcea and Strapparava, 2009; Almela et al., 2012) and showed that the use of semantic information is helpful for the automatic identification of deceit.

While there is a significant body of work on computational deception detection, except for (Yancheva and Rudzicz, 2013) who considered the relation between syntactic constructs and deceivers' age, to our knowledge there are no computational analyses of demographics in deceptive content. However, there have been a number of psychological studies on the role of gender and age in deceptive behavior. These studies have found interesting associations between deception and gender. For instance, (Toma et al., 2008) identified differences in self-presentation among genders. In this study men were found to lie more about their height and women lied more about their weight. (Kaina et al., 2011) found that females are more easily detectable when lying than their male counterparts. (Tilley et al., 2005) reported that females are more successful in deception detection than male receivers.

3 Open Domain Deception Dataset

We started our study by collecting a new open domain deception dataset consisting of freely contributed truths and lies. We used Amazon Mechanical Turk and asked each worker to contribute seven lies and seven truths, on topics of their own choice, each of them consisting of one single sentence. In an attempt to obtain truths and lies that represent everyday lying behavior, we asked our contributors to provide plausible lies and avoid non-commonsensical statements such as "I can fly." Since we did not enforce a particular topic, resulting truths and lies are open domain. Sample truths and lies are presented in Table 1. Note that the collected lies might include statements that are somehow unrealistic, even if plausible, e.g., "I own two Ferraris, one red and one black". We decided to also include these statements in order to aid the identification of differences in deceivers and true-tellers language, as we hypothesize that they might help reveal topics that naturally occur in truths and lies.

Additionally, we collect demographic data from the contributors, including their gender, age, country of origin, and education level. To avoid spam, contributions were manually verified by one of the paper authors. The final dataset consists of 7168 sentences from 512 unique contributors. Since each contributor provided seven lies and seven truths the dataset contains a total of 3584 truths and 3584 lies respectively. Participant's ages range from 18 to 72 years, with an average age of 34.14 and a standard deviation of 12.67.

4 Features

In this section, we describe the sets of features extracted, which will then be used to build our classifiers.

Unigrams We extract unigrams derived from the bag of words representation of truths and lies present in our dataset.

Shallow and deep syntax features These features consist of part of speech (POS) tags and lexicalized production rules derived from Probabilistic Context Free Grammar (PCFG) trees, obtained with the Berkeley Parser (Petrov et al., 2006).

Female	
Lie	Truth
I won 1 billion dollars in the Illinois state lottery last year and gave it all away to my mother.	My daughter is my best friend in the whole wide world, and i would give my life for hers.
On my last birthday i turned 119 years old and went sky diving as a gift to myself.	I graduated with a degree in information systems 10 years ago and still can't find a good job.
I'm allergic to alcohol	Giraffes are taller than zebras.
Male	
Lie	Truth
Barak obama was my guest last night; he offered me the administrative assistant job at white house in Washington.	Internet is one of the greatest invention of history of humankind with its ability to speed up the communication.
I own two Ferraris, one red and one black	I love to play soccer with my friends
I wake up at 11 o clock every day	I wake up at 6 am because I have to work at 7 am

Table 1: Sample open-domain lies and truths provided by male and a female participants

Semantic features These features include the 80 semantic classes present in the LIWC lexicon. Each feature represents the number of words in a sentence belonging to a specific semantic class.

Readability and Syntactic Complexity features This set includes the Flesch-Kincaid and Gunning Fog readability scores and 14 indexes of syntactic complexity derived from the syntactic analysis of each sentence; performed with the tool provided by Lu (Lu, 2010).

5 Classification of Deception, Gender, and Age in Short Texts

Our first experiment seeks to evaluate whether deception detection can be conducted using the open domain deception dataset described above. We performed the evaluations at user level, by collapsing all the lies from one user into one instance, and all the truths into another instance.

We build deception classifiers using the SVM algorithm[1] and the different sets of features. We performed a five-fold cross-validation, by training each time on 80% of the users and testing on the remaining 20%. During our evaluations truths and lies pertaining to a particular user were either on the training or testing set. Classification results on individual and combined sets of features are presented in Table 3. The best performing set of features are the POS tags, followed by features derived from production rules. The remaining sets of features achieved accuracy values ranging from 54% to 65%, which still represent a noticeable improvement over the random baseline. Note that we experimented with a few more feature sets combinations, including the use of all the features together, however we did not observe significant improvements.

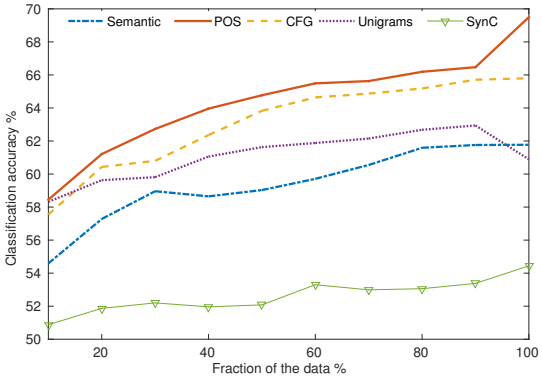


Figure 1: Learning curves for deception detection using five feature sets

To analyze the impact of the amount of data on the classifier learning process, we plot the learning curves on the different sets of features using incremental amounts of data as shown in Figure 1. Evaluations were conducted using five-fold cross validations on each incremental fraction of data. The learning trend suggests that most classifiers benefit from increasing amounts of training data.

[1] As implemented in the Weka toolkit, with default parameter settings.

Gender	Female	298
	Male	214
Age	Young ($\leq$ 35 years)	319
	Middle-aged/Elder (>35 years)	193

Table 2: Class distribution for gender and age

Feature set	Deception	Gender	Age
Baseline	50.00%	58.00%	62.00%
Unigrams	60.89%	54.25%	51.12%
Semantic	60.21%	57.28%	61.83%
POS	69.50%	49.95%	52.39%
CFG	65.39%	52.19%	54.74%
Readability	54.44%	58.16%	62.26%
Uni+Semantic	62.17%	63.04%	51.51%

Table 3: SVM classifiers trained for three prediction tasks: deception, gender, and age.

However, except for the POS features, the overall performance seems to stabilize when using 90% of the training data.

As a second experiment, we evaluate the ability of the classifier to predict gender and age in short open domain deceptive texts. Given the contributors' age distribution, which lies mainly in the range of 30-45 years, we opted to cluster the participants age into into two groups: young ($\leq$ 35 years) and middle-aged/elder (>35 years). Class distributions for age and gender are shown in Table 2. We performed the age prediction task on the two groups using the different sets of features and SVM classifiers. Classification accuracies are shown in Table 3. Reported baselines consist of a majority class baseline. Results show low to moderate improvement over the baseline for gender classification, with the combination of semantic features and unigrams being the best performing feature set. However, our classifiers performed poorly in the age prediction task, with accuracies below the majority class baseline.

Overall, the results suggest that age and gender prediction are challenging tasks when conducted in open domain deception data. One possible explanation for this is that the lack of context introduces noise into the analysis. For instance, the following sentence: "I'm 50 years old" can belong to either a male or a female, and it might be a lie for younger people or a truth for older people.

Lies			
Male		Female	
Other	2.22	Certain	1.87
Negate	2.08	Negate	1.63
Certain	2.06	You	1.59
Death	2.04	Motion	1.47
Anger	2.03	Down	1.45
You	1.77	Money	1.35
Friends	1.71	Anger	1.28
Othref	1.67	Future	1.20
Truths			
Male		Female	
Religion	1.67	Sleep	1.64
Family	1.65	Religion	1.61
Groom	1.60	See	1.50
Music	1.49	Discrepancy	1.39
Sports	1.45	Anxiety	1.36
School	1.42	Posfeel	1.33
Posfeel	1.35	Metaphor	1.33
Feel	1.32	TV	1.31

Table 4: Results from LIWC word class analysis for short open domain truths and lies.

6 Analyzing Language Used by Deceivers Given Age and Gender

In order to explore language differences among deceivers and true-tellers, we use the linguistic ethnography method (Mihalcea and Pulman, 2009) and obtain the most dominant semantic word classes in the LIWC lexicon associated to truth and lies provided by males and females. Results are shown in Table 4. From this table, we observe interesting patterns in word usage that are shared among genders. For instance, spontaneous lies often include *negation*, *certain*, and *you* words, which is in line with previous work on domain-specific deception (Mihalcea and Strapparava, 2009) that suggested that liars try to reinforce their lies through the use of stronger wording and detachment from the self. On the other hand, people appear to be less likely to lie when talking about their *family*, *religion*, and describing *positive* experiences. There are also LIWC classes associated to a specific gender. Male lies contain references to friends and others, while female lies contain references to money and future. Similarly, female true-tellers use metaphor words while male true-tellers use words related to sports and music.

Lies			
Age 18-34		Age 35-65	
Certain	2.04	Assent	2.13
Anger	1.98	Certain	1.83
Negate	1.82	Negate	1.68
Other	1.76	Anxiety	1.64
You	1.72	You	1.64
Down	1.64	Motion	1.54
Othref	1.53	Money	1.50
Death	1.49	Optim	1.38
Truths			
Age 18-34		Age 35-65	
Religion	1.83	Music	1.43
Tv	1.48	Sleep	1.42
Anxiety	1.45	Feel	1.34
Posfeel	1.37	Posfeel	1.33
See	1.30	See	1.31
Music	1.30	Sexual	1.28
School	1.29	Religion	1.27
Inhib	1.28	Family	1.25

Table 5: Results from LIWC word class analysis for short open domain truths and lies.

We also evaluate differences in word usage that might be attributed to deceiver's age. Resulting dominant classes and their scores are presented in Table 5. The analyses show interesting differences for deceiver's word usage across age. For instance, regardless of their gender, older deceivers use references to *anxiety*, *money*, and *motion*. On the other hand, younger deceivers language includes *anger*, *negate*, and *death* words. These differences suggest that indeed gender and age play a role on people words choices while fabricating lies.

7 Conclusions

In this paper, we presented our initial experiments in open domain deception detection. We targetted the deception detection on short text to address the cases where context is not available. In real settings, this can be useful when receiving a text message or when looking at anonymous posts in forums. We collected a new deception dataset consisting of one-sentence truths and lies, along with the demographics of the deceivers. Through several experiments, we showed that this data can be used to build deception classifiers for short open domain text. However, the classifiers do not perform very well while trying to predict gender and age. We further explored linguistic differences in deceptive content that relate to deceivers gender and age and found evidence that both age and gender play an important role on people's word choices when fabricating lies.

The dataset introduced in this paper is publicly available from http://lit.eecs.umich.edu.

Acknowledgments

This material is based in part upon work supported by National Science Foundation awards #1344257 and #1355633, by grant #48503 from the John Templeton Foundation, and by DARPA-BAA-12-47 DEFT grant #12475008. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation, the John Templeton Foundation, or the Defense Advanced Research Projects Agency.

References

Á. Almela, R. Valencia-García, and P. Cantos. 2012. Seeing through deception: A computational approach to deceit detection in written communication. In *Proceedings of the Workshop on Computational Approaches to Deception Detection*, pages 15–22, Avignon, France, April. Association for Computational Linguistics.

B. DePaulo, J. Lindsay, B. Malone, L. Muhlenbruck, K. Charlton, and H. Cooper. 2003. Cues to deception. *Psychological Bulletin*, 129(1).

S. Feng, R. Banerjee, and Y. Choi. 2012. Syntactic stylometry for deception detection. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers - Volume 2*, ACL '12, pages 171–175, Stroudsburg, PA, USA. Association for Computational Linguistics.

J. Kaina, M.G. Ceruti, K. Liu, S.C. McGirr, and J.B. Law. 2011. Deception detection in multicultural coalitions: Foundations for a cognitive model. Technical report, DTIC Document.

X. Lu. 2010. Automatic analysis of syntactic complexity in second language writing. *International Journal of Corpus Linguistics*, 15(4):474–496.

R. Mihalcea and S. Pulman. 2009. Linguistic ethnography: Identifying dominant word classes in text. In *Computational Linguistics and Intelligent Text Processing*, pages 594–602. Springer.

R. Mihalcea and C. Strapparava. 2009. The lie detector: Explorations in the automatic recognition of deceptive language. In *Proceedings of the Association for Computational Linguistics (ACL 2009)*, Singapore.

M. Ott, Y. Choi, C. Cardie, and J. Hancock. 2011. Finding deceptive opinion spam by any stretch of the imagination. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 309–319, Stroudsburg, PA, USA. Association for Computational Linguistics.

M. Ott, C. Cardie, and J. Hancock. 2013. Negative deceptive opinion spam. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Short Papers*, Atlanta, Georgia, USA, June. Association for Computational Linguistics.

J. Pennebaker and M. Francis. 1999. Linguistic inquiry and word count: LIWC. Erlbaum Publishers.

S. Petrov, L. Barrett, R. Thibaux, and D. Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, ACL-44, pages 433–440, Stroudsburg, PA, USA. Association for Computational Linguistics.

P. Tilley, J. F. George, and K. Marett. 2005. Gender differences in deception and its detection under varying electronic media conditions. In *Proceedings of the Proceedings of the 38th Annual Hawaii International Conference on System Sciences (HICSS'05) - Track 1 - Volume 01*, HICSS '05, pages 24.2–, Washington, DC, USA. IEEE Computer Society.

C. L. Toma and J. T. Hancock. 2010. Reading between the lines: Linguistic cues to deception in online dating profiles. In *Proceedings of the 2010 ACM Conference on Computer Supported Cooperative Work*, CSCW '10, pages 5–8, New York, NY, USA. ACM.

C. L Toma, J. T Hancock, and N. B Ellison. 2008. Separating fact from fiction: An examination of deceptive self-presentation in online dating profiles. *Personality and Social Psychology Bulletin*, 34(8):1023–1036.

Q. Xu and H. Zhao. 2012. Using deep linguistic features for finding deceptive opinion spam. In *Proceedings of COLING 2012: Posters*, pages 1341–1350, Mumbai, India, December. The COLING 2012 Organizing Committee.

M. Yancheva and F. Rudzicz. 2013. Automatic detection of deception in child-produced speech using syntactic complexity features. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 944–953, Sofia, Bulgaria, August. Association for Computational Linguistics.

D. Yu, Y. Tyshchuk, H. Ji, and W. A. Wallace. 2015. Detecting deceptive groups using conversations and network analysis. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 857–866.

A model of rapid phonotactic generalization

Tal Linzen
Department of Linguistics
New York University
linzen@nyu.edu

Timothy J. O'Donnell
Brain and Cognitive Sciences
Massachusetts Institute of Technology
timod@mit.edu

Abstract

The phonotactics of a language describes the ways in which the sounds of the language combine to form possible morphemes and words. Humans can learn phonotactic patterns at the level of abstract classes, generalizing across sounds (e.g., "words can end in a voiced stop"). Moreover, they rapidly acquire these generalizations, even before they acquire sound-specific patterns. We present a probabilistic model intended to capture this early-abstraction phenomenon. The model represents both abstract and concrete generalizations in its hypothesis space from the outset of learning. This—combined with a parsimony bias in favor of compact descriptions of the input data—leads the model to favor rapid abstraction in a way similar to human learners.

1 Introduction

Natural languages place restrictions on the ways in which sounds can combine to form words (the *phonotactics* of the language). The velar nasal [ŋ], for example, occurs at the end of English syllables, as in *ring* [ɹɪŋ] or *finger* [fɪŋgəɹ], but never at the beginning of a syllable: English does not have words like **ngir* [ŋɪɹ]. English speakers are aware of this constraint, and judge forms that start with a [ŋ] as impossible English words.

Sounds that share articulatory and/or perceptual properties often have similar phonotactic distributions. German, for example, allows voiced obstruents,[1] such as [b] and [g], to occur anywhere in the word except at its end: [bal] is a valid German word, but [lab] isn't.

Speakers use such features of sounds to form phonotactic generalizations, which can then apply to sounds that do not appear in their language. Although no English words start with either [sɹ] or [mb], English speakers judge *srip* to be a better potential word of English than *mbip* (Scholes, 1966); this is likely because [sɹ] shares properties with strident-liquid clusters that do exist in English, such as [sl] as in *slip* and [ʃɹ] as in *shrewd*, whereas [mb] does not benefit from any sonorant-stop onset sequences (*[nt])—none exist in English.

Recent studies have investigated how humans acquire generalizations over phonological classes in an artificial language paradigm (Linzen and Gallagher, 2014; Linzen and Gallagher, 2015). The central finding of these studies was that participants rapidly learned abstract phonotactic constraints and exhibited evidence of generalizations over classes of sounds before evidence of phoneme-specific knowledge.

This paper presents a probabilistic generative model, the Phonotactics As Infinite Mixture (PAIM) model, which exhibits similar behavior. This behavior arises from the combination of two factors: the early availability of abstract phonological classes in the learner's hypothesis space; and a parsimony bias implemented as a Dirichlet process mixture, which favors descriptions of the data using a single pattern over ones that make reference to multiple specific patterns.

2 Summary of behavioral data

The experiments are described in detail in Linzen and Gallagher (2014) and Linzen and Gallagher (2015); we summarize the main details here.

Design: Participants were exposed to varying numbers of auditorily-presented words in one of two artificial languages, VOICING and IDENTITY.

[1] See Hayes (2011) for an introduction to phonological features.

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1126–1131,
Lisbon, Portugal, 17-21 September 2015.

Exposure		Test		
ganu	gimi	CONF-	CONF-	NONCONF-
balu	bini	ATT	UNATT	UNATT
vimu	voni	zonu	dila	tumu
zalu	zili	zini	dimu	talu
ðano	ðamu			

Table 1: VOICING: Design for one of the lists (voiced exposure, [d] held out). The table shows the complete list of exposure words that a participant in the two exposure sets group might receive.

Exposure	Test	
	CONF-	CONF-
	ATT	UNATT
pipa	papi	keku
ʃuʃe	ʃeʃu	sasi
gapu	gugi	dʒidʒe
nuni	nanu	mamu
	NONCONF-	NONCONF-
	ATT	UNATT
kesa	kasi	pina
mudʒe	medʒa	nage
dʒuki	dʒuke	gaʃe
semi	sami	ʃipu

Table 2: IDENTITY: A complete list of exposure and test words that a participant in the one exposure set group might receive.

Following the exposure phase, they were asked to judge for a set of novel test words whether those words could be part of the language they had learned (the possible answers were "yes" or "no").

In the VOICING experiment, all exposure words began with consonants that had the same value for their voicing feature (all voiced or all voiceless, e.g., $A_1 = \{$g, v, z, ð, b$\}$ or $A_2 = \{$k, f, s, θ, p$\}$). Some of the sounds with the relevant voicing were held out to be used during testing (e.g., [d] for A_1 or [t] for A_2). All exposure and test words had the form CVMV (*tumi*), where C stands for the onset consonant, V for a randomized vowel, and M for [m], [n] or [l] (see Table 1).

Participants judged three types of novel test words: ones with the same onset as one or more of the words in exposure (CONF-ATT;[2] e.g., *zonu* for A_1); ones whose onset was not encountered in exposure but had the same voicing as the exposure onsets (CONF-UNATT; e.g., *dila*); and ones whose onset had different voicing from the exposure words (NONCONF-UNATT, e.g., *tomu*). The vowels and the second consonant were randomized across conditions such that only the onsets reliably discriminated the three conditions.

In the IDENTITY language, words had the form C_1VC_2V. The generalization in this language was $C_1 = C_2$ (e.g., *pipa*). Here it was probabilistic: only half of the words in the exposure stage conformed to the generalization. As such, there was a fourth test condition, NONCONF-ATT, of exposure words that did not conform to the generalization.

Participants were recruited on Amazon Mechanical Turk (280 participants in the VOICING experiment and 288 in the IDENTITY experiment). They were divided into four groups, which received 1, 2, 4 or 8 sets of words. In the VOICING experiment, each of the sets contained five words, one starting with each of the five CONF-ATT onsets; in the IDENTITY experiment, each of the sets contained eight words, one with each of the CONF-ATT and NONCONF-ATT consonant pairs (Tables 1 and 2).

Results: Human experimental results are plotted in Figure 1. Endorsement rates represent the proportion of trials in which participants judged the word to be well-formed. Participants learned generalizations involving abstract classes of sounds after a single exposure set: in the VOICING experiment, they judged voiced word onsets to be better than voiceless ones, and in the IDENTITY experiment they judged words with identical consonants as better than words with nonidentical ones.[3] Participants did not start distinguishing CONF-ATT from CONF-UNATT patterns until they received two or more sets of exposure to the language.

Participants continued to generalize to CONF-UNATT patterns even after significant exposure to the language. Endorsement rates were higher than 50% across the board, likely because even words with NONCONF-UNATT consonant patterns were similar to the exposure words in all other respects (e.g., length, syllable structure, number of vowels

[2] ATT (attested): the consonants in the word (though not the full word) were encountered in exposure; UNATT (unattested): consonants were not encountered in exposures; CONF (conforming): the consonants conform to the abstract pattern (voicing or identity); NONCONF (nonconforming): consonants don't conform to the abstract pattern.

[3] All differences discussed in this section are statistically significant.

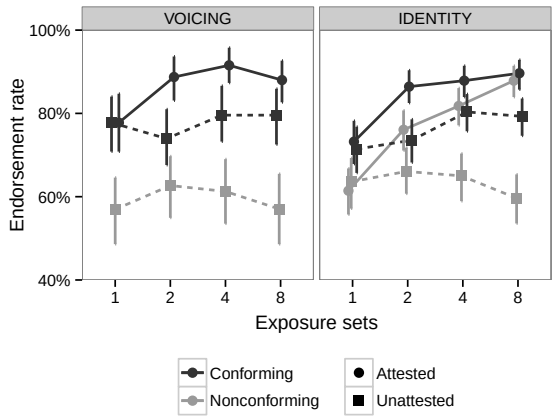


Figure 1: Human behavioral results. Error bars indicate bootstrapped 95% confidence intervals.

and consonants).

3 The model

PAIM is a generative model: it describes the probabilistic process by which the phonological forms of words are generated. Phonotactic knowledge is expressed as a set of word-form templates, represented as sequences of phonological classes. For example, the template ⟨[+voiced], V, C, V⟩ captures a generalization over words beginning with a voiced consonant.

Prior over phonological classes: Phonemes are represented as phonological feature-value matrices.[4] We generate a phonological class for each position in the template using this feature system and a parameter $p \in [0, 1]$, which controls the model's willingness to consider more or less abstract phonological classes: low values of p encourage underspecified classes, such as [] or [+voice], whereas high values of p favor highly specified classes, such as [+voice, labial, -continuant]. Given a particular value of p, we define the distribution $G(c)$ over classes as follows:

- For each phonological feature f whose set of possible values is V_f:
  1. Draw $P \sim \text{Bernoulli}(p)$.
  2. If $P = 1$, draw $v \sim \text{Uniform}(V_f)$ and include it in c.
  3. Otherwise, leave the feature unspecified, allowing the class to be abstract.

[4]The particular feature system is treated as a parameter of the model. In the simulation below we used a simplified version of the phonological feature inventory described in Hayes (2011), which only included features that are distinctive in English consonants.

Generating words from templates: Given a choice of phonological template t, we assume that each of the segments that instantiate t has the same probability of being sampled (cf. the "size principle" of Tenenbaum and Griffiths (2001)). Consider again the class c_0 = [+continuant, labial]. Under the assumption that the model's segment inventory is the English one, there are only two segments that are labial continuants: [v] and [f]. The probability of each one of them being generated from c_0 will be $P(s|c_0) = 1/2$.

Prior over template sets: The sounds of the language can be generated from a variety of templates at varying levels of abstraction. We therefore extend the model to be a mixture of template distributions of the type described above. The number of templates is inferred from the data using a Dirichlet process mixture model (Antoniak, 1974).

This prior can be constructed as a process. Suppose that s_i is an ordering of the input sounds, and that we know which templates generated the first $n-1$ sounds $s_1, \ldots, s_{n-1}$. If K is the number of templates that have been posited so far and $n_1, \ldots, n_K$ indicate the number of sounds that have been drawn from each template, then the probability distribution over the template z_n that the sound s_n will be drawn from is given by

$$P(z_n = k|z_{1:n-1}) \propto \begin{cases} n_k & \text{if } k \leq K \\ \alpha & \text{otherwise} \end{cases} \quad (1)$$

Since the probability that an existing template generated s_n is proportional to the number of segments currently assigned to that template, this prior encourages partitions in which a few templates explain most of the sounds (the "rich get richer" property), which amounts to a parsimony bias. Higher values of α can make this bias weaker.

Modeling phoneme spreading: To simulate the generalization made by participants in the IDENTITY experiment, templates must be able to state that two phonemes need to be identical. This is analogous to mechanisms of "spreading" widely assumed in phonology (Colavin et al., 2010; Goldsmith, 1976; McCarthy, 1986). For our simulations below, we simplify by only considering identity constraints between the initial and medial consonants in exposure and test forms. We sample a template over these positions as follows:

1. Draw a class $c_1 \sim G$, where G is the distribution over phonological classes defined above.
2. Draw $Q \sim \text{Bernoulli}(q)$.
3. If $Q = 1$, return an identity template, i.e., $\langle s, s \rangle$ such that $s \in c_1$.
4. Otherwise, draw $c_2 \sim G$ and return the template $\langle s_1, s_2 \rangle$ such that $s_1 \in c_1$ and $s_2 \in c_2$.

Inference: We perform inference to find the posterior over template sets given the exposure datasets used in the human experiments described above. We also infer the hyperparameters using the following prior distributions:

$$\begin{aligned} p &\sim \text{Beta}(1,1) \\ \alpha &\sim \text{Gamma}(2,4) \\ q &\sim \text{Beta}(1,1) \end{aligned} \tag{2}$$

Inference for the Dirichlet process mixture was performed using the Gibbs sampler described in Neal (2000). After each Gibbs sweep, slice sampling (Neal, 2003) was used to obtain a new value for p and q. A new value for α was sampled using the method described by Escobar and West (1995). We ran the sampler for 3000 iterations, discarded the first 100 samples and kept every fifth sample of the remaining samples, for a total of 580 samples from the posterior distribution.

Predicting human data: Participants in the behavioral experiments gave binary judgments ("could the word be part of the language?") rather than probability estimates. To link our model's predictions to participants' binary responses, we sample m template instantiations from the posterior predictive distribution.[5] If the relevant part of the test word appeared in these m samples, the model responds "yes"; this can be understood to be related to a sampling-based view of human inference (Vul et al., 2014). In the simulations below we fix m to be 10.

Human endorsement rates were consistently above 50%, while the model's ratings were often close to 0%. This is likely to be because human ratings were also informed by the unmodeled (fixed) parts of the templates, such as word length or number of vowels. We therefore linearly transform the model's ratings to the range exhibited by human participants: if the untransformed rate is r, the ultimate simulated rate will be $(1 + r)/2$.

[5]Template instantiations only include the modeled (specified) part of the template: an onset consonant in our model of the VOICING language or a consonant pair for the IDENTITY language.

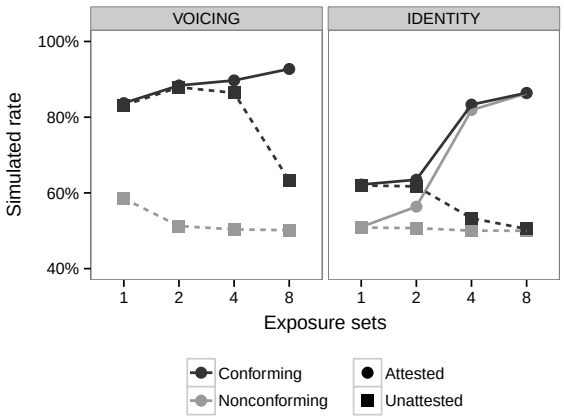


Figure 2: PAIM: Simulated endorsement rates.

4 Simulations

We only modeled those aspects of phonotactic templates that are relevant to the experimental results. For the VOICING experiment, we constrained the template to be ⟨_, V, C, V⟩ (inference is only performed on the first consonant); for the IDENTITY experiment, we constrained it to be ⟨_, V, _, V⟩.

Figure 2 shows the simulated endorsement rates. After a single exposure to each pattern (one exposure set), PAIM behaved in a qualitatively similar way to participants in both experiments: it distinguished CONF from NONCONF words, but did not distinguish ATT from UNATT words.

PAIM was less willing than humans to generalize to CONF-UNATT items after multiple exposure sets: in the IDENTITY experiment the generalization had no effect by the eighth exposure set; in the VOICING experiment its effect was weaker after eight than after four exposures sets. By contrast, human generalization in both languages showed no sign of weakening after multiple exposure sets.

5 Comparison to related models

Hayes and Wilson (2008) propose a Maximum Entropy model of phonotactics (MaxEnt; see also Goldwater and Johnson (2003)). Like PAIM, MaxEnt is based on phonological classes defined as feature matrices. Each class c is assigned a weight w_c. The predicted probability in MaxEnt of a sound s is

$$\hat{p}(s) = \frac{1}{Z} e^{\sum_c w_c I_c(s)} \tag{3}$$

where $Z = \sum_s \hat{p}(s)$ and $I_c(s) = 1$ if $s \in c$ and 0 otherwise.

We simulated endorsement rates from a MaxEnt model for the VOICING language. Following Hayes and Wilson (2008), we used l_2 regularization; that is, if the exposure words were $s_1, \ldots, s_n$, the objective function was

$$\sum_{i=1}^{n} \log \hat{p}(s_i) - \sum_{c} \frac{(w_c - \mu)^2}{2\sigma^2} \qquad (4)$$

Figure 3 shows the simulated endorsement rates for different values of σ (we set $\mu = 0$ in all simulations). For $\sigma = 0.05$, the model showed little learning after a single exposure set. When σ was set to higher values, MaxEnt rapidly preferred attested to unattested items, failing to reproduce the human early generalization pattern. Like PAIM, but unlike humans, generalization to CONF-UNATT items diminished after multiple exposure sets (in particular for $\sigma = 0.5$). A straightforward implementation of MaxEnt is therefore incapable of simulating the human results; better results could potentially be achieved with a regularization method that encouraged sparsity (Goodman, 2004; Johnson et al., 2015).

Another proposed model of phonotactics is the Minimal Generalization Learner, or MGL (Albright, 2009); Linzen and Gallagher (2014) showed that MGL can simulate relevant human behavioral data in some circumstances. In contrast with PAIM and MaxEnt, which converge to the empirical distribution given sufficient data, MGL reserves a fixed amount of probability mass to unseen events. It would therefore able to simulate a sustained generalization pattern.

Our prior over phonological classes bears some resemblance to the Rational Rules model of visual categorization (Goodman et al., 2008). In that model, classes are generated from a probabilistic context free grammar (PCFG); highly specified rules are therefore implicitly less probable, as in our model. Relatedly, Hayes and Wilson (2008) use a greedy feature selection procedure that starts from simpler phonological classes and gradually adds more complex ones; this procedure also encodes an implicit bias in favor of simple classes. Finally, our implementation of a parsimony bias using a Dirichlet process is related to similar biases incorporated into other models of language learning (Frank and Tenenbaum, 2011; Johnson et al., 2007; O'Donnell, 2015).

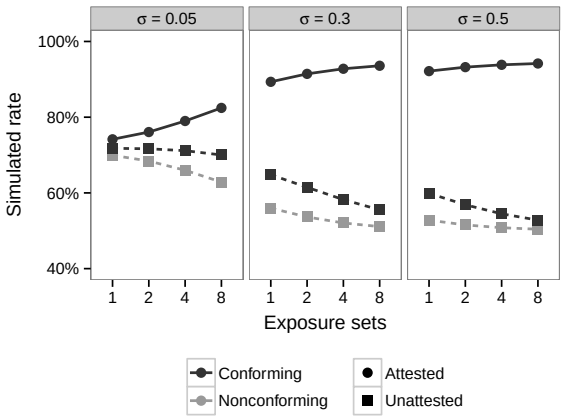


Figure 3: Maximum entropy model: Simulated endorsement rates for the VOICING language, with different values of the regularization parameter σ.

6 Discussion

We have presented a probabilistic model of phonotactic generalization that captures the pattern of rapid, abstract generalization characteristic of human learners. The model's performance is driven by two crucial assumptions. First, it allows hypotheses that make reference to abstract, broad classes of phones from the beginning of the learning process. Second, it prefers to learn compact or parsimonious explanations of the input corpus, using a small number of phonotactic templates. This second property is enforced by our use of the Dirichlet process as a prior over template sets.

These two properties interact. When the model has seen only a few data items, the availability of abstract generalizations allows it to explain all items using a single template, and the prior bias towards parsimony drives it to do so. As the number of data items increases, repeated instances of specific phonemes no longer seem like accidental observations from a more general template, but rather like significant templates in their own right; the model begins to capture such item-specificity.

The model stopped generalizing earlier than humans did; we intend to explore ways to explain this discrepancy. Additional human data would need to be collected to determine whether humans keep generalizing indefinitely, or eventually converge on the attested sounds. Finally, to facilitate inference, we only tested our model on the parts of the word that were relevant to the human data. In future work, we intend to extend the model to learn larger templates that include syllable structure and phonological tiers (Goldsmith, 1976).

References

Adam Albright. 2009. Feature-based generalisation as a source of gradient acceptability. *Phonology*, 26(1):9–41.

Charles E. Antoniak. 1974. Mixtures of Dirichlet processes with applications to Bayesian nonparametric problems. *The Annals of Statistics*, pages 1152–1174.

Rebecca S. Colavin, Roger Levy, and Sharon Rose. 2010. Modeling OCP-Place in Amharic with the Maximum Entropy phonotactic learner. In *Proceedings of the 46th meeting of the Chicago Linguistics Society*.

Michael D. Escobar and Mike West. 1995. Bayesian density estimation and inference using mixtures. *Journal of the American Statistical Association*, 90(430):577–588.

Michael C. Frank and Joshua B. Tenenbaum. 2011. Three ideal observer models for rule learning in simple languages. *Cognition*, 120(3):360–371.

John A. Goldsmith. 1976. *Autosegmental Phonology*. Ph.D. thesis, MIT.

Sharon Goldwater and Mark Johnson. 2003. Learning OT constraint rankings using a maximum entropy model. In *Proceedings of the Stockholm workshop on variation within Optimality Theory*, pages 111–120.

Noah D. Goodman, Joshua B. Tenenbaum, Jacob Feldman, and Thomas L. Griffiths. 2008. A rational analysis of rule-based concept learning. *Cognitive Science*, 32(1):108–154.

Joshua Goodman. 2004. Exponential priors for maximum entropy models. In *HLT-NAACL*, pages 305–312.

Bruce Hayes and Colin Wilson. 2008. A maximum entropy model of phonotactics and phonotactic learning. *Linguistic Inquiry*, 39(3):379–440.

Bruce Hayes. 2011. *Introductory phonology*. Wiley-Blackwell, Malden, MA and Oxford.

Mark Johnson, Thomas L. Griffiths, and Sharon Goldwater. 2007. Adaptor Grammars: A framework for specifying compositional nonparametric Bayesian models. In *Advances in Neural Information Processing Systems 19*, Cambridge, MA. MIT Press.

Mark Johnson, Joe Pater, Robert Staubs, and Emmanuel Dupoux. 2015. Sign constraints on feature weights improve a joint model of word segmentation and phonology. In *HLT-NAACL*, pages 303–313.

Tal Linzen and Gillian Gallagher. 2014. The timecourse of generalization in phonotactic learning. In *Proceedings of Phonology 2013*.

Tal Linzen and Gillian Gallagher. 2015. Rapid generalization in phonotactic learning. `http://tallinzen.net/media/papers/linzen_gallagher_2015.pdf`.

John J. McCarthy. 1986. OCP effects: Gemination and antigemination. *Linguistic Inquiry*, 17(2):207–263.

Radford M. Neal. 2000. Markov chain sampling methods for Dirichlet process mixture models. *Journal of Computational and Graphical Statistics*, 9(2):249–265.

Radford M. Neal. 2003. Slice sampling. *Annals of Statistics*, 31(3):705–767.

Timothy J. O'Donnell. 2015. *Productivity and reuse in language: A theory of linguistic computation and storage*. The MIT Press, Cambridge, Massachusetts.

Robert J. Scholes. 1966. *Phonotactic grammaticality*. Mouton, The Hague.

Joshua B. Tenenbaum and Thomas L. Griffiths. 2001. Generalization, similarity, and Bayesian inference. *Behavioral and Brain Sciences*, 24(4):629–640.

Edward Vul, Noah Goodman, Thomas L. Griffiths, and Joshua B. Tenenbaum. 2014. One and done? Optimal decisions from very few samples. *Cognitive Science*, 38(4):599–637.

Automatically Solving Number Word Problems by Semantic Parsing and Reasoning

Shuming Shi[1], Yuehui Wang[2*], Chin-Yew Lin[1], Xiaojiang Liu[1] and Yong Rui[1]
[1] Microsoft Research
{shumings, cyl, xiaojl, yongrui}@microsoft.com
[2] University of Science and Technology of China
wyh9346@mail.ustc.edu.cn

Abstract

This paper presents a semantic parsing and reasoning approach to automatically solving math word problems. A new meaning representation language is designed to bridge natural language text and math expressions. A CFG parser is implemented based on 9,600 semi-automatically created grammar rules. We conduct experiments on a test set of over 1,500 number word problems (i.e., verbally expressed number problems) and yield 95.4% precision and 60.2% recall.

1 Introduction

Computers, since their creation, have exceeded human beings in (speed and accuracy of) mathematical calculation. However, it is still a big challenge nowadays to design algorithms to automatically solve even primary-school-level math word problems (i.e., math problems described in natural language).

Efforts to automatically solve math word problems date back to the 1960s (Bobrow, 1964a, b). Previous work on this topic falls into two categories: symbolic approaches and statistical learning methods. In symbolic approaches (Bobrow, 1964a, b; Charniak, 1968; Bakman, 2007; Liguda & Pfeiffer, 2012), math problem sentences are transformed to certain structures by pattern matching or verb categorization. Equations are then derived from the structures. Statistical learning methods are employed in two recent papers (Kushman et al., 2014; Hosseini et al., 2014).

Most (if not all) previous symbolic approaches suffer from two major shortcomings. First, natural language (NL) sentences are processed by simply applying pattern matching and/or transformation rules in an ad-hoc manner (refer to the related work section for more details). Second, surprisingly, they seldom report evaluation results about the effectiveness of the methods (except for some examples for demonstration purposes). For the small percentage of work with evaluation results available, it is unclear whether the patterns and rules are specially designed for specific sentences in a test set.

1). One number is 16 more than another. If the smaller number is subtracted from 2/3 of the larger, the result is 1/4 of the sum of the two numbers. Find the numbers.

2). Nine plus the sum of an even integer and its square is 3 raised to the power of 4. What is the number?

3). The tens digit of a two-digit number is 3 more than the units digit. If the number is 8 more than 6 times the sum of the digits, find the number.

4). If the first and third of three consecutive even integers are added, the result is 12 less than three times the second integer. Find the integers.

Figure 1: Number word problem examples

In this paper, we present a computer system called SigmaDolphin which automatically solves math word problems by semantic parsing and reasoning. We design a meaning representation language called DOL (abbreviation of dolphin language) as the structured semantic representation of NL text. A semantic parser is implemented to transform math problem text into DOL trees. A reasoning module is included to derive math expressions from DOL trees and to calculate final answers. Our approach falls into the symbolic category, but makes improvements over previous symbolic methods in the following ways,

* Work done while this author was an intern at Microsoft Research

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1132–1142,
Lisbon, Portugal, 17-21 September 2015.

1) We introduce a systematic way of parsing NL text, based on context-free grammar (CFG).

2) Evaluation is enhanced in terms of both data set construction and evaluation mechanisms. We split the problem set into a development set (called dev set) and a test set. Only the dev set is accessible during our algorithm design (especially in designing CFG rules and in implementing the parsing algorithm), which avoids over-tuning towards the test set. Three metrics (precision, recall, and F1) are employed to measure system performance from multiple perspectives, in contrast to all previous work (including the statistical ones) which only measures accuracy.

We target, in experiments, a subtype of word problems: number word problems (i.e., verbally expressed number problems, as shown in Figure 1). We hope to extend our techniques to handle general math word problems in the future.

We build a test set of over 1,500 problems and make a quantitative comparison with state-of-the-art statistical methods. Evaluation results show that our approach significantly outperforms baseline methods on our test set. Our system yields an extremely high precision of 95.4% and a reasonable recall of 60.2%, which shows promising application of our system in precision-critical situations.

2 Related Work

2.1 Math word problem solving

Most previous work on automatic word problem solving is symbolic. STUDENT (Bobrow, 1964a, b) handles algebraic problems by first transforming NL sentences into *kernel sentences* using a small set of transformation patterns. The kernel sentences are then transformed to math expressions by recursive use of pattern matching. CARPS (Charniak, 1968, 1969) uses a similar approach to solve English rate problems. The major difference is the introduction of a tree structure as the internal representation of the information gathered for one object. Liguda & Pfeiffer (2012) propose modeling math word problems with augmented semantic networks. Addition/subtraction problems are studied most in early research (Briars & Larkin, 1984; Fletcher, 1985; Dellarosa, 1986; Bakman, 2007; Ma et al., 2010). Please refer to Mukherjee & Garain (2008) for a review of symbolic approaches before 2008.

No empirical evaluation results are reported in most of the above work. Almost all of these approaches parse NL text by simply applying pattern matching rules in an ad-hoc manner. For example, as mentioned in Bobrow (1964b), due to the pattern "($, AND $)", the system would incorrectly divide "Tom has 2 apples, 3 bananas, and 4 pears." into two "sentences": "Tom has 2 apples, 3 bananas." and "4 pears."

WolframAlpha[1] shows some examples[2] of automatically solving elementary math word problems, with technique details unknown to the general public. Other examples on the web site demonstrate a large coverage of short phrase queries on math and other domains. By randomly selecting problems from our dataset and manually testing on their web site, we find that it fails to handle most problems in our problem collection.

Statistical learning methods have been proposed recently in two papers: Hosseini et al. (2014) solve single step or multi-step homogenous addition and subtraction problems by learning verb categories from the training data. Kushman et al. (2014) can solve a wide range of word problems, given that the equation systems and solutions are attached to problems in the training set. The method of the latter paper (referred to as KAZB henceforth) is used as one of our baselines.

2.2 Semantic parsing

There has been much work on analyzing the semantic structure of NL strings. In semantic role labeling and frame-semantic parsing (Gildea & Jurafsky, 2002; Carreras & Marquez, 2004; Marquez et al., 2008; Baker et al., 2007; Das et al., 2014), predicate-argument structures are discovered from text as their shallow semantic representation. In math problem solving, we need a deeper and richer semantic representation from which to facilitate the deriving of math expressions.

Another type of semantic parsing work (Zelle & Mooney, 1996; Zettlemoyer & Collins, 2005; Zettlemoyer & Collins, 2007; Wong & Mooney, 2007; Cai & Yates, 2013; Berant et al., 2013; Kwiatkowski et al., 2013; Berant & Liang, 2014) maps NL text into logical forms by supervised or semi-supervised learning. Some of them are based on or related to combinatory categorial grammar (CCG) (Steedman, 2000). Abstract Meaning Representation (AMR) (Banarescu et al., 2013) keeps richer semantic information than CCG and logical

[1] http://www.wolframalpha.com

[2] https://www.wolframalpha.com/examples/Elementary-Math.html (bottom-right part)

forms. In Section 3.1.4, we discuss the differences between DOL, AMR, and CCG, and explain why we choose DOL as the meaning representation language for math problem solving.

3 Approach

Consider the first problem in Figure 1 (written below for convenience),

One number is 16 more than another. If the smaller number is subtracted from 2/3 of the larger, the result is 1/4 of the sum of the two numbers. Find the numbers.

To automatically solve this problem, the computer system needs to figure out, somehow, that 1) two numbers x, y are demanded, and 2) they satisfy the equations below,

$$x = 16 + y \quad (1)$$

$$(2/3)x - y = (x + y) / 4 \quad (2)$$

To achieve this, *reasoning* must be performed based on common sense knowledge and the information provided by the source problem. Given the difficulty of performing reasoning directly on unstructured and ambiguous natural language text, it is reasonable to transform the source text into a structured, less ambiguous representation.

Our approach contains three modules:

1) A *meaning representation language* called DOL newly designed by us as the semantic representation of natural language text.
2) A *semantic parser* which transforms natural language sentences of a math problem into DOL representation.
3) A *reasoning* module to derive math expressions from DOL representation.

English: Nine plus the sum of an even integer and its square is 3 raised to the power of 4. What is the number?

DOL trees:

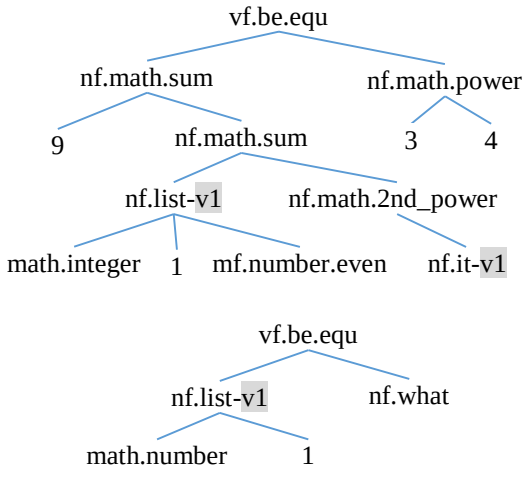


Figure 2: DOL example

3.1 DOL: Meaning representation language

Every meaningful piece of NL text is represented in DOL as a semantic tree of various node types. Figure 2 shows the DOL representation of the second problem of Figure 1. It contains two semantic trees, corresponding to the two sentences.

3.1.1 Node types

Node types of a DOL tree include *constants*, *classes*, and *functions*. Each interim node of a tree is always a function; and each leaf node can be a constant, a class, or a zero-argument function.

Constants in DOL refer to specific objects in the world. A constant can be a number (e.g., 3.57), a lexical string (like "New York"), or an entity.

Classes: An entity class refers to a category of entities sharing common semantic properties. For example, all cities are represented by the class *location.city*; and *math.number* is a class for all numbers. It is clear that,

3.14159 ∈ math.number
city.new_york ∈ location.city

A class C_1 is a *sub-class* (denoted by ⊆) of another class C_2 if and only if every instance of C_1 are in C_2. The following holds according to common sense knowledge,

math.number ⊆ math.expression
person.pianist ⊆ person.performer

Template classes are classes with one or more parameters, just like template classes in C++. The most important template class in DOL is

t.list<*c*,*m*,*n*>

where *c* is a class; *m* and *n* are integers. Each instance of this class is a list containing at least *m* and at most *n* elements of type *c*. For example, each instance of t.list<math.number,2,+∞> is a list containing at least 2 numbers.

Functions are used in DOL as the major way to form larger language units from smaller ones. A function is comprised of a *name*, a list of *core arguments*, and a *return type*. DOL enables function overloading (again borrowing ideas from programming languages). That is, one function name can have multiple core-argument specifications. Below are two specifications for fn.math.sum (which appears in the example of Figure 2).

nf.math.sum!1: $1: math.expression; $2: math.expression return type: math.expression return value: The sum of its arguments
nf.math.sum!2: $1: t.list<math.expression,2,+∞> return type: math.expression return value: The sum of the elements in $1

Here "$1: math.expression" means the first argument has type math.expression.

DOL supports three kinds of functions: noun functions, verb functions, and modifier functions.

Noun functions map entities to their properties or to other entities having specific relations with the argument(s). For example, *nf.math.sum* maps math expressions to their sum. Noun functions are used to represent noun phrases in natural language text. More noun functions are shown in Table 1.

Among all noun functions, *nf.list* has a special important position due to its high frequency in DOL trees. The function is specified below,

nf.list
$1: class; $2: math.number
return type: t.list<$1>
return value: An entity list with cardinality $2 and element type $1

For example nf.list(math.number,5) returns a list containing 5 elements of type math.number. It is the semantic representation of "five numbers".

Pronoun functions are special zero-argument noun functions. Examples are nf.it (representing an already-mentioned entity or event) and nf.what (denoting an unknown entity or entity list).

Verb functions act as sentences or sub-sentences in DOL. As an example, vf.be.equ (in Figure 2) is a verb function that has two arguments of the quantity type.

vf.be.equ
$1: quantity.generic; $2: quantity.generic
return type: t.vf
Meaning: Two quantities $1 and $2 have the same value

In addition to core arguments ($1, $2, etc.), many functions can take additional *extended arguments* as their modifiers. Our last function type called *modifier functions* often take the role of extended arguments, to modify noun functions, verb functions, or other modifier functions. Modifier functions are used in DOL as the semantic representation of adjectives, adverb phrases (including conjunctive adverb phrases), and prepositional phrases in natural languages. In the example of Figure 2, the function mf.number.even modifies the noun function nf.list as its extended argument.

3.1.2 Entity variables

Variables are assigned to DOL sub-trees for indicating the co-reference of sub-trees to entities and for facilitating the construction of logical forms and math expressions from DOL. In Figure 2, the same variable v1 (meaning a variable with ID 1) is assigned to two sub-trees in the first sentence and one sub-tree in the second sentence. Thus the three sub-trees refer to the same entity.

Function	Remarks
nf.math.numerator $1: math.fraction ret: math.number	Get the numerator of fraction $1
nf.math.gcd $1: t.list<math.integer,2,+∞> ret: math.integer	Get the greatest common divisor of the elements of $1
nf.e.height $1: e.concrete ret: quantity.length	Get the height of $1 which is a concrete entity
vf.believe $1: e.agent; $2: t.vf.std ret: t.vf	Agent $1 believes that $2 is true as a predicate
mf.number.even ret: t.mf.adj	Indicating the property of being an even number

Table 1: Example DOL functions

3.1.3 Key features of DOL

DOL has some nice characteristics that are critical to building a high-precision math problem solving system. That is why we invent DOL as our meaning representation language instead of employing an existing one.

First, DOL is a strongly typed language. Every function has clearly defined argument types and a return type. A valid DOL tree must satisfy the *type-compatibility* property:

Type-compatibility: The type of each child of a function node should match the corresponding argument type of the function.

For example, in Figure 2, the return type of nf.math.power is math.expression, which matches the second argument of vf.be.equ. However, the following two trees (yielded from the corresponding pieces of text) are invalid because they do not satisfy type-compatibility.

sum of 100 [unreasonable text]
~~nf.math.sum!2(100)~~ [invalid DOL tree]

sum of 3 and Jordan [unreasonable text]
~~nf.math.sum!2({3, "Jordan"})~~ [invalid tree]

Second, we maintain in DOL an open-domain type system. The type system contains over 1000 manually verified classes and more automatically generated ones (refer to Section 3.2.1 for more details). Such a comprehensive type system makes it possible to define various kinds of functions and to perform type-compatibility checking. In contrast, most previous semantic languages have at most 100+ types at the grammar level. In addition, by introducing template classes, we avoid maintaining a lot of potentially duplicate types and reduce the type system management efforts. To the best of our knowledge, template classes are not

available in other semantic representation languages.

Third, DOL has built-in data structures like t.list and nf.list which greatly facilitate both function declaration and text representation (especially math text representation). For example, the two variants of nf.math.sum (refer to Section 3.1.1 for their specifications) are enough to represent the following English phrases:

3 plus 5 → nf.math.sum!1(3, 5)
sum of 3, 5, 7, and 9 → nf.math.sum!2(nf.list(3, 5, 7, 9))
sum of ten thousand numbers → nf.math.sum!2(nf.list(math.number,10000))

Without t.list or nf.list, we would have to define a lot of overloaded functions for nf.math.sum to deal with different numbers of addends.

3.1.4 Comparing with other languages

Among all meaning representation languages, AMR (Banarescu et al., 2013) is most similar to DOL. Their major differences are: First, they use very different mechanisms to represent noun phrases. In AMR, a sentence (e.g., "the boy destroyed the room") and a noun phrase (e.g., "the boy's destruction of the room") can have the same representation. While in DOL, a sentence is always represented by a verb function; and a noun phrase is always a noun function or a constant. Second, DOL has a larger type system and is stricter in type compatibility checking. Third, DOL has template classes and built-in data structures like t.list and nf.list to facilitate the representation of math concepts.

CCG (Steedman, 2000) provides a transparent interface between syntax and semantics. In CCG, semantic information is defined on words (e.g., "λx.odd(x)" for "odd" and "λx.number(x)" for "number"). In contrast, DOL explicitly connects NL text patterns to semantic elements. For example, as shown in Table 2 (Section 3.2.1), one CFG grammar rule connects pattern "{$1} raised to the power of {$2}" to function nf.math.power.

Logical forms are another way of meaning representation. We choose not to transform NL text directly to logical forms for two reasons: On one hand, state-of-the-art methods for mapping NL text into logical forms typically target short, one-sentence queries in restricted domains. However, many math word problems are long and contain multiple sentences. On the other hand, variable-id assignment is a big issue in direct logical form construction for many math problems. Let's use the following problem (i.e., the first problem of Figure 1) to illustrate,

One number is 16 more than another. If the smaller number is subtracted from 2/3 of the larger, the result is 1/4 of the sum of the two numbers. Find the numbers.

For this problem, it is difficult to determine whether "the smaller number" refers to "one number" or "another" in directly constructing logical forms. It is therefore a challenge to construct a correct logical form for such kinds of problems.

Our solution to the above challenge is assigning a new variable ID (which is different from the IDs of "one number" and "another") and to delay the final variable-ID assignment to the reasoning stage. To enable this mechanism, the meaning representation language should support a lazy variable ID assignment and keep as much information (e.g., determiners, plurals, modifiers) from the noun phrases as possible. DOL is a language that always keeps the structure information of phrases, whether or not it has been assigned a variable ID.

In summary, compared with other languages, DOL has some unique features which make it more suitable for our math problem solving scenario.

3.2 Semantic Parsing

Our parsing algorithm is based on context-free grammar (CFG) (Chomsky, 1956; Backus, 1959; Jurafsky & Martin, 2000), a commonly used mathematical system for modeling constituent structure in natural languages.

3.2.1 CFG for connecting DOL and NL

The core part of a CFG is the set of grammar rules. Example English grammar rules for building syntactic parsers include "S → NP VP", "NP → CD | DT NN | NP PP", etc. Table 2 shows some example CFG rules in our system for mapping DOL nodes to natural language word sequences. The left side of each rule is a DOL element (a function, class, or constant); and the right side is a sequence of words and arguments. The grammar rules are consumed by our parser for building DOL trees from NL text.

So far there are 9,600 grammar rules in our system. For every DOL node type, the lexicon and grammar rules are constructed together in a semi-automatic way. Math-related classes, functions, and constants and their grammar rules are manually built by referring to text books and online tu-

torials. About 35 classes and 200 functions are obtained in this way. Additional instances of each element type are constructed in the ways below.

Classes: Additional classes and grammar rules are obtained from two data sources: Freebase[3] types, and automatically extracted lexical semantic data. By treating Freebase types as DOL classes and the mapping from types to lexical names as grammar rules, we get the first version of grammar for classes. To improve coverage, we run a term peer similarity and hypernym extraction algorithm (Hearst, 1992; Shi et al., 2010; Zhang et al., 2011) on a web snapshot of 3 billion pages, and get a peer-similarity graph and a collection of is-a pairs. An is-a pair example is (Megan Fox, actress), where "Megan Fox" and "actress" are instance and type names respectively. In our peer similarity graph, "Megan Fox" and "Britney Spears" have a high similarity score. The peer similarity graph is used to clean the is-a data collection (with the idea that peer terms often share some common type names). Given the cleaned is-a data, we sort the type names by weight and manually create classes for top-1000 type names. For example, create a class person.actress and add a grammar rule "person.actress → actress". For the other 2000 type names in the top 3000, we create classes and rules automatically, in the form of "class.TN → TN", where TN is a type name. For example, create rule "class.succulent → succulent" for name "succulent".

vf.be.equ($1,$2) → {$1} be equal to {$2} | {$1} equal {$2} | {$1} be {$2}
vf.give($1,$2,$3) → {$1} give {$2} to {$3} | {$1} give {$3} {$2}
nf.math.sum!1($1,$2) → {$1} plus {$2} | {$2} added to {$1}
nf.math.sum!2($1) → sum of {$1} | addition of {$1}
nf.math.power($1,$2) → {$1} raised to the {power|exponent} of {$2}
nf.list($1,$2) → {$2} {$1}
mf.number.even → even
mf.condition.if($1) → if {$1}
mf.approximately → approximately | roughly
education.university → university
math.number → number
math.integer → integer

Table 2: Example grammar for connecting DOL and NL

Functions: Additional noun functions are automatically created from Freebase properties and attribute extraction results (Pasca et al., 2006; Durme et al., 2008), using a similar procedure with creating classes from Freebase types and is-a extraction results. We have over 50 manually defined math-related verb functions. Our future plan is automatically generating verb functions from databases like PropBank (Kingsbury & Palmer, 2002), FrameNet (Fillmore et al., 2003), and VerbNet[4] (Schuler, 2005). Additional modifier functions are automatically created from an English adjective and adverb list, in the form of "mf.adj.TN → TN" and "mf.adv.TN → TN" where TN is the name of an adjective or adverb.

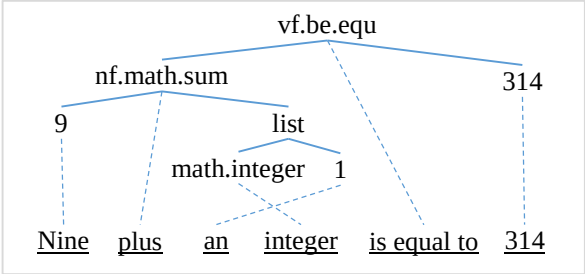


Figure 3: The DOL semantic parse tree for "Nine plus an integer is equal to 314"

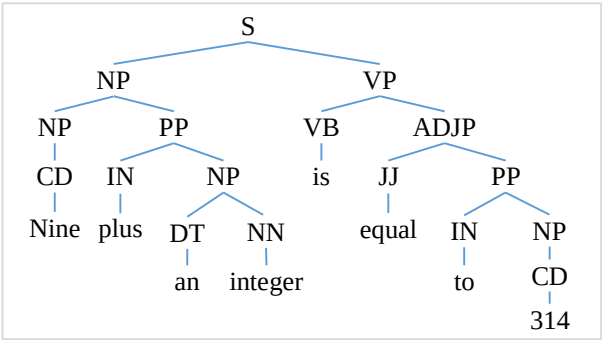


Figure 4: A syntactic parse tree

3.2.2 Parsing

Parsing for CFG is a well-studied topic with lots of algorithms invented (Kasami, 1965; Earley, 1970). The core idea behind almost all the algorithms is exploiting dynamic programming to achieve efficient search through the space of possible parse trees. For syntactic parsing, a well-known serious problem is ambiguity: the appearance of many syntactically correct but semantically unreasonable parse trees. Modern syntactic parsers reply on statistical information to reduce

[3] Freebase: http://www.freebase.com/

[4] VerbNet: http://verbs.colorado.edu/~mpalmer/projects/verbnet.html

ambiguity. They are often based on probabilistic CFGs (PCFGs) or probabilistic lexicalized CFGs trained on hand-labeled TreeBanks.

With the new set of DOL-NL grammar rules (examples in Table 2) and the type-compatibility property (Section 3.1.3), ambiguity can hopefully be greatly reduced, because semantically unreasonable parsing often results in invalid DOL trees.

We implement a top-down parser for our new CFG of Section 3.2.1, following the Earley algorithm (Earley, 1970). No probabilistic information is attached in the grammar rules because no Treebanks are available for learning statistical probabilities for the new CFG. Figure 3 shows the parse tree returned by our parser when processing a simple sentence. The DOL tree can be obtained by removing the dotted lines (corresponding to the non-argument part in the right side of the grammar rules). A traditional syntactic parse tree is shown in Figure 4 for reference.

During parsing, a score is calculated for each DOL node. The score of a tree *T* is the weighted average of the scores of its sub-trees,

$$\boldsymbol{S}(\boldsymbol{T}) = \frac{\sum_{i=1}^{k} \boldsymbol{L}(\boldsymbol{T_i}) \cdot \boldsymbol{S}(\boldsymbol{T_i})}{\sum_{i=1}^{k} \boldsymbol{L}(\boldsymbol{T_i})} \cdot \boldsymbol{p}(\boldsymbol{T}) \qquad (3)$$

where T_i is a sub-tree, and $L(T_i)$ is the number of words to which the sub-tree corresponds in the original text. If the type-compatibility property for *T* is satisfied, $p(T)$=1; otherwise $p(T)$=0.

All leaf nodes are assigned a score of 1.0, except for pure lexical string nodes (which are used as named entity names). The score of a lexical string node is set to $1/(1+\mu n)$, where *n* is the number of words in the node, and μ (=0.2 in experiments) is a parameter whose value does not have much impact on parsing results. Such a score function encourages interpreting a word sequence with our grammar than treating it as an entity name.

Among all candidate DOL trees yielded during parsing, we return the one with the highest score as the final parsing result. A null tree is returned if the highest score is zero.

3.3 Reasoning

The reasoning module is responsible for deriving *math expressions* from DOL trees and calculating problem answers by solving equation systems. Math expressions have different definitions in different contexts. In some definitions, equations and inequations are excluded from math expressions. In this paper, equations and inequations (like "*a*=*b*" and "*ax*+*b*>0") are called s-expressions because they represent mathematical sentences, while other math expressions (like "*x*+5") are named n-expressions since they are essentially noun phrases. Our definition of "*math expressions*" therefore includes both n-expressions and s-expressions.

Different types of nodes may generate different types of math expressions. In most cases, s-expressions are derived from verb function nodes and modifier function nodes, while n-expressions are generated from constants and noun function nodes. For example, the s-expression "9+*x*=314" can be derived from the DOL tree of Figure 3, if variable *x* represents the integer. In the same Figure, The n-expression "9+x" is derived from the left sub-tree.

The pseudo-codes of our math expression derivation algorithm are shown in Figure 5. The algorithm generates the math expression for a DOL tree T by first calling the expression derivation procedure of sub-trees, and then applying the semantic interpretation of T. All the s-expressions derived so far are stored in an expression list named XL.

```
Algorithm MathExpDerivation
   Input: DOL tree T
   Output: Math expression X(T)
   Global data structure: Expression list XL
1: For each child Ci of T
2:    X(Ci) = MathExpDerivation(Ci)
3:    If X(Ci) is an s-expression
4:       Add X(Ci) to XL
5: X(T) ← Applying the semantic interpretation
of T
6: Return X(T)
```

Figure 5: Math expression derivation algorithm

vf.be.equ($1,$2) → X($1) = X($2)	(1)
nf.math.sum!1($1,$2) → X($1) + X($2)	(2)
nf.math.sum!2($1) → $\sum_{\mathbf{e}\in\$\mathbf{1}} \mathbf{X}(\mathbf{e})$	(3)
nf.math.gcd($1) → gcd({X(e) | **e** ∈ $**1**})	(4)
nf.list($1,$2) → V = (v_1, v_2…, v_n), *n*=X($2)	(5)
mf.number.even → X($↑) % 2 = 0	(6)

Table 3: Example semantic interpretations

The semantic interpretation of DOL nodes plays a critical role in the algorithm. Table 3 shows some example interpretations of some representative DOL functions. In the table, $1, $2 etc. are function arguments, and $↑ for a modifier node denotes the node which the modifier modifies. So far the semantic interpretations are built manually. Please note that it is not necessary to make semantic interpretations for every DOL

node in solving number word problems. For example, most class nodes and many adverb nodes can have null interpretations at the moment.

4 Experiments

4.1 Experimental setup

Datasets: Our problem collection[5] contains 1,878 math number word problems, collected from two web sites: algebra.com[6] (a web site for users to post math problems and get help from tutors) and answers.yahoo.com[7]. Problems on both sites are organized into categories. For algebra.com, problems are randomly sampled from the number word problems category; for answers.yahoo.com, we first randomly sample an initial set of problems from the math category and then ask human annotators to manually choose number word problems from them. Math equations[8] and answers to the problems are manually added by human annotators.

We randomly split the dataset into a dev set (for algorithm design and debugging) and a test set. More subsets are extracted to meet the requirements of the baseline methods (see below). Table 4 shows the statistics of the datasets.

Baseline methods: We compare our approach with two baselines: KAZB (Kushman et al., 2014) and BasicSim.

KAZB is a learning-based statistical method which solves a problem by mapping it to one of the *equation templates* determined by the annotated equations in the training data. We run the ALLEQ version of their algorithm since it performs much better than the other two (i.e., 5EQ and 5EQ+ANS). Their codes support only linear equations and require that there are at least two problems for each equation template (otherwise an exception will be thrown). By choosing problems from the collection that meet these requirements, we build a sub-dataset called LinearT2. In the dataset of KAZB, each equation template corresponds to at least 6 problems. So we form another sub-dataset called LinearT6 by removing from the test set the problems for which the associated equation template appears less than 6 times.

BasicSim is a simple statistical method which works by computing the similarities between a testing problem and those in the training set, and then applying the equations of the most similar problem. This method has similar performance with KAZB on their dataset, but does not have the two limitations mentioned above. Therefore we adopt it as the second baseline.

For both baselines, experiments are conducted using 5-fold cross-validation with the dev set always included in the training data. In other words, we always use the dev set and 4/5 of the test set as training data for each fold.

Evaluation metrics: Evaluation is performed in the setting that a system can choose NOT to answer all problems in the test set. In other words, one has the flexibility of generating answers only when she knows how to solve it or she is confident about her answer. In this setting, the following three metrics are adopted in reporting evaluation results (assuming, in a test set of size n, a system generates answers for m problems, where k of them are correct):

Precision: k/m

Recall (or coverage): k/n

F1: $2PR/(P+R) = 2k/(m+n)$

Dataset		#problems	#sentences (average)	#words (average)
All	dev	374	1.79	20.3
	test	1,504	1.75	22.5
Linear	dev	247	1.78	19.6
	test	986	1.72	19.0
LinearT2	dev	172	1.85	18.8
	test	669	1.71	17.4
LinearT6	dev	71	1.96	16.8
	test	348	1.80	16.1

Table 4: Dataset statistics (Linear: problems with linear equations; T2: problems corresponding to template size ≥ 2)

4.2 Experimental results

The Overall evaluation results are summarized in Table 5, where "Dolphin" represents our approach. The results show that our approach significantly outperforms (with $p<<0.01$ according to two-tailed t-test) the two baselines on every test set, in terms of precision, recall, and F-measure. Our approach achieves a particularly high precision of 95%. That means once an answer is provided by our approach, it has a very high probability of being correct.

Please note that our grammar rules and parsing algorithm are NOT tuned for the evaluation data. Only the dev set is referred to in system building.

[5] Available from http://research.microsoft.com/en-us/projects/dolphin/

[6] http://www.algebra.com

[7] https://answers.yahoo.com/

[8] Math equations are used in the baseline approaches as part of training data.

Since the baselines generate results for *all* problems, the precision, recall, and F1 are all the same for each dataset.

Dataset	Method	Precision (%)	Recall (%)	F1 (%)
LinearT6	KAZB	49.1	49.1	49.1
	BasicSim	59.7	59.7	59.7
	Dolphin	**98.1**	**72.9**	**83.6**
LinearT2	KAZB	37.5	37.5	37.5
	BasicSim	46.3	46.3	46.3
	Dolphin	**97.3**	**68.0**	**80.0**
Linear	BasicSim	32.3	32.3	32.3
	Dolphin	**95.7**	**63.6**	**76.4**
Test set all	BasicSim	29.0	29.0	29.0
	Dolphin	**95.4**	**60.2**	**73.8**

Table 5: Evaluation results

The reason for such a high precision is that, by transforming NL text to DOL trees, the system "*understands*" the problem (or has structured and accurate information about quantity relations). Therefore it is more likely to generate correct results than statistical methods who simply "guess" according to features. By examining the problems in the dev set that we cannot generate answers, we find that most of them are due to empty parsing results.

On the other hand, statistical approaches have the advantage of generating answers without understanding the semantic meaning of problems (as long as there are similar problems in the training data). So they are able to handle (with probably low precision) problems that are complex in terms of language and logic.

Please pay attention that our experimental results reported here are on *number word problems*. General math word problems are much harder to our approach because the entity types, properties, relations, and actions contained in general word problems are much larger in quantity and more complex in quality. We are working on extending our approach to general math word problems. Now our DOL language and CFG grammar already have a good coverage on common entity types, but the coverage on properties, relations, and actions is quite limited. As a result, our parser fails to parse many sentences in general math word problems because they contain properties, relations or actions that are unknown to our system. We also observe that sometimes we are able to parse a problem successfully, but cannot derive math expressions in the reasoning stage. This is often because some relations or actions in the problem are not modeled appropriately. As future work, we plan to extend our DOL lexicon and grammar to improve the coverage of properties, relations, and actions. We also plan to study the mechanism of modeling relations and actions.

5 Conclusion

We proposed a semantic parsing and reasoning approach to automatically solve math number word problems. We have designed a new meaning representation language DOL to bridge NL text and math expressions. A CFG parser is implemented to parse NL text to DOL trees. A reasoning module is implemented to derive math expressions from DOL trees, by applying the semantic interpretation of DOL nodes. We achieve a high precision and a reasonable recall on our test set of over 1,500 problems. We hope to extend our techniques to handling general math word problems and to other domains (like physics and chemistry) in the future.

Acknowledgments

We would like to thank the annotators for their efforts in assigning math equations and answers to the problems in our dataset. Thanks to the anonymous reviewers for their helpful comments and suggestions.

Reference

J.W. Backus. 1959. The syntax and semantics of the proposed international algebraic language of the Zurich ACM-GAMM conference. Proceedings of the International Conference on Information Processing, 1959.

Y. Bakman. 2007. Robust understanding of word problems with extraneous information. http://arxiv.org/abs/math/0701393. Accessed Feb. 2nd, 2015.

C. Baker, M. Ellsworth, and K. Erk. 2007. SemEval-2007 Task 19: Frame semantic structure extraction. In Proceedings of SemEval.

L. Banarescu, C. Bonial, S. Cai, M. Georgescu, K. Griffitt, U. Hermjakob, K. Knight, P. Koehn, M. Palmer, and N. Schneider. 2013. Abstract meaning representation for sembanking. In Proc. of the Linguistic Annotation Workshop and Interoperability with Discourse.

J. Berant, A. Chou, R. Frostig, and P. Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In Empirical Methods in Natural Language Processing (EMNLP).

J. Berant and P. Liang. 2014. Semantic Parsing via Paraphrasing. In ACL'2014.

D.G. Bobrow. 1964a. Natural language input for a computer problem solving system. Report MAC-TR-1, Project MAC, MIT, Cambridge, June

D.G. Bobrow. 1964b. Natural language input for a computer problem solving system. Ph.D. Thesis, Department of Mathematics, MIT, Cambridge

D.L. Briars, J.H. Larkin. 1984. An integrated model of skill in solving elementary word problems. Cognition and Instruction, 1984, 1 (3) 245-296.

Q. Cai and A. Yates. 2013. Large-scale semantic parsing via schema matching and lexicon extension. In Association for Computational Linguistics (ACL).

X. Carreras. and L. Marquez. 2004. Introduction to the CoNLL-2004 shared task: Semantic role labeling. In Proceedings of CoNLL.

E. Charniak. 1968. CARPS: a program which solves calculus word problems. Report MAC-TR-51, Project MAC, MIT, Cambridge, July

E. Charniak. 1969. Computer solution of calculus word problems. In Proceedings of international joint conference on artificial intelligence. Washington, DC, pp 303–316

N. Chomsky. 1956. Three models for the description of language. Information Theory, IRE Transactions on, 2(3), 113-124.

S. Clark, and J. Curran. 2007. Wide-coverage efficient statistical parsing with CCG and log-linear models. Computational Linguistics, 33(4):493-552.

D. Das, D. Chen, A.F.T. Martins, N. Schneider and N.A. Smith. 2014. Frame-Semantic Parsing. Computational Linguistics 40:1, pages 9-56

D. Dellarosa. 1986. A computer simulation of children's arithmetic word problem solving. Behavior Research Methods, Instruments, & Computers, 18:147–154

V. Durme, T. Qian, and L. Schubert. 2008. Class-driven attribute extraction. In Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1, pp. 921-928. Association for Computational Linguistics, 2008.

J. Earley. 1970. An efficient context-free parsing algorithm. Communications of the ACM, 13(2), 94-102.

C.J. Fillmore, C.R. Johnson, and M.R. Petruck. 2003. Background to FrameNet. International Journal of Lexicography, 16(3).

C.R. Fletcher. 1985. Understanding and solving arithmetic word problems: a computer simulation. Behavior Research Methods, Instruments, & Computers, 17:565–571

D. Gildea, and D. Jurafsky. 2002. Automatic labeling of semantic roles. Computational Linguistics, 28(3).

M. Hearst. 1992. Automatic Acquisition of Hyponyms from Large Text Corpora. In Fourteenth International Conference on Computational Linguistics, Nantes, France.

M.J. Hosseini, H. Hajishirzi, O. Etzioni, and N. Kushman. 2014. Learning to Solve Arithmetic Word Problems with Verb Categorization. In EMNLP'2014.

D. Jurafsky, and J.H. Martin. 2000. Speech & language processing. Pearson Education India.

T. Kasami. 1965. An efficient recognition and syntax-analysis algorithm for context-free languages (Technical report). AFCRL. 65-758.

P. Kingsbury, and M. Palmer. 2002. From TreeBank to PropBank. In Proceedings of LREC.

N. Kushman, Y. Artzi, L. Zettlemoyer, and R. Barzilay. 2014. Learning to automatically solve algebra word problems. In Proc. of the Annual Meeting of the Association for Computational Linguistics (ACL).

T. Kwiatkowski, E. Choi, Y. Artzi, and L. Zettlemoyer. 2013. Scaling semantic parsers with on-the-fly ontology matching. In Empirical Methods in Natural Language Processing (EMNLP).

I. Lev, B. MacCartney, C. Manning, and R. Levy. 2004. Solving logic puzzles: From robust processing to precise semantics. In Proceedings of the Workshop on Text Meaning and Interpretation. Association for Computational Linguistics.

C. Liguda, T. Pfeiffer. 2012. Modeling Math Word Problems with Augmented Semantic Networks. NLDB'2012, pp. 247-252.

Y. Ma, Y. Zhou, G. Cui, R. Yun, R. Huang. 2010. Frame-based calculus of solving arithmetic multi-step addition and subtraction word problems. In International Workshop on Education Technology and Computer Science, vol. 2, pp. 476–479.

L. Marquez, X. Carreras, K.C. Litkowski, and S. Stevenson. 2008. Semantic role labeling: an introduction to the special issue. Computational Linguistics, 34(2).

A. Mukherjee and U. Garain. 2008. A review of methods for automatic understanding of natural language mathematical problems. Artificial Intelligence Review, 29(2).

M. Pasca, D. Lin, J. Bigham, A. Lifchits, and A. Jain. 2006. Organizing and searching the world wide web of facts-step one: the one-million fact extraction challenge. In AAAI (Vol. 6, pp. 1400-1405).

K.K. Schuler. 2005. VerbNet: A broad-coverage, comprehensive verb lexicon. Dissertation. http://repository.upenn.edu/dissertations/AAI3179808

S. Shi, H. Zhang, X. Yuan, and J.-R. Wen. 2010. Corpus-based semantic class mining: distributional vs. pattern-based approaches. In Proceedings of the 23rd International Conference on Computational Linguistics, pages 993–1001. Association for Computational Linguistics.

M. Steedman. 2000. The Syntactic Process. The MIT Press.

Y. W. Wong and R. J. Mooney. 2007. Learning synchronous grammars for semantic parsing with lambda calculus. In Association for Computational Linguistics (ACL), pages 960–967.

M. Zelle and R.J. Mooney. 1996. Learning to parse database queries using inductive logic proramming. In Association for the Advancement of Artificial Intelligence (AAAI), pages 1050–1055.

L.S. Zettlemoyer and M. Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In Uncertainty in Artificial Intelligence (UAI), pages 658–666.

L.S. Zettlemoyer and M. Collins. 2007. Online Learning of Relaxed CCG Grammars for Parsing to Logical Form. In Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL).

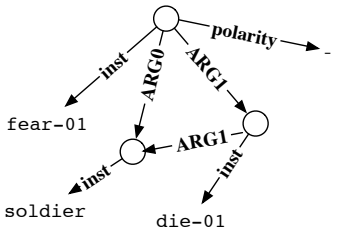
F. Zhang, S. Shi, J. Liu, S. Sun, and C.-Y. Lin. 2011. Nonlinear evidence fusion and propagation for hyponymy relation mining. In ACL, volume 11, pages 1159–1168.

Parsing English into Abstract Meaning Representation Using Syntax-Based Machine Translation

Michael Pust, Ulf Hermjakob, Kevin Knight, Daniel Marcu, Jonathan May
Information Sciences Institute
Computer Science Department
University of Southern California
{pust, ulf, knight, marcu, jonmay}@isi.edu

Abstract

We present a parser for Abstract Meaning Representation (AMR). We treat English-to-AMR conversion within the framework of string-to-tree, syntax-based machine translation (SBMT). To make this work, we transform the AMR structure into a form suitable for the mechanics of SBMT and useful for modeling. We introduce an AMR-specific language model and add data and features drawn from semantic resources. Our resulting AMR parser significantly improves upon state-of-the-art results.



The soldier was not afraid of dying.
The soldier was not afraid to die.
The soldier did not fear death.

Figure 1: An Abstract Meaning Representation (AMR) with several English renderings.

1 Introduction

Abstract Meaning Representation (AMR) is a compact, readable, whole-sentence semantic annotation (Banarescu et al., 2013). It includes entity identification and typing, PropBank semantic roles (Kingsbury and Palmer, 2002), individual entities playing multiple roles, as well as treatments of modality, negation, etc. AMR abstracts in numerous ways, e.g., by assigning the same conceptual structure to *fear* (v), *fear* (n), and *afraid* (adj). Figure 1 gives an example.

AMR parsing is a new research problem, with only a few papers published to date (Flanigan et al., 2014; Wang et al., 2015) and a publicly available corpus of more than 10,000 English/AMR pairs.[1] New research problems can be tackled either by developing new algorithms/techniques (Flanigan et al., 2014; Wang et al., 2015) or by adapting existing algorithms/techniques to the problem at hand. In this paper, we investigate the second approach.

The AMR parsing problem bears a strong formal resemblance to syntax-based machine translation (SBMT) of the string-to-tree variety, in that a string is transformed into a nested structure in both cases. Because of this, it is appealing to apply the substantial body of techniques already invented for SBMT[2] to AMR parsing. By re-using an SBMT inference engine instead of creating custom inference procedures, we lose the ability to embed some task-specific decisions into a custom transformation process, as is done by Flanigan et al. (2014) and Wang et al. (2015). However, we reap the efficiency gains that come from working within a tested, established framework. Furthermore, since production-level SBMT systems are widely available, anyone wishing to generate AMR from text need only follow our recipe and retrain an existing framework with relevant data to quickly obtain state-of-the-art results.

Since SBMT and AMR parsing are, in fact, distinct tasks, as outlined in Figure 2, to adapt the SBMT parsing framework to AMR parsing, we develop novel representations and techniques. Some of our key ideas include:

1. Introducing an AMR-equivalent representation that is suitable for string-to-tree SBMT rule extraction and decoding (Section 4.1).

[1] LDC Catalog number 2014T12

[2] See e.g. the related work section of Huck et al. (2014).

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1143–1154,
Lisbon, Portugal, 17-21 September 2015.

	SBMT	AMR parsing
Target	tree	graph
Nodes	labeled	unlabeled
Edges	unlabeled	labeled
Alignments	words to leaves	words to leaves + words to edges
Children	ordered	unordered
Accuracy Metric	BLEU (Papineni et al., 2002)	Smatch (Cai and Knight, 2013)

Figure 2: Differences between AMR parsing and syntax-based machine translation (SBMT).

2. Proposing a target-side reordering technique that leverages the fact that child nodes in AMR are unordered (Section 4.4).
3. Introducing a hierarchical AMR-specific language model to ensure generation of likely parent-child relationships (Section 5).
4. Integrating several semantic knowledge sources into the task (Section 6).
5. Developing tuning methods that maximize Smatch (Cai and Knight, 2013) (Section 7).

By applying these key ideas, which constitute lightweight changes on a baseline SBMT system, we achieve state-of-the-art AMR parsing results. We next describe our baseline, and then describe how we adapt it to AMR parsing.

2 Syntax-Based Machine Translation

Our baseline SBMT system proceeds as follows. Given a corpus of (source string, target tree, source-target word alignment) sentence translation training tuples and a corpus of (source, target, score) sentence translation tuning tuples:

1. **Rule extraction:** A grammar of string-to-tree rules is induced from training tuples using the GHKM algorithm (Galley et al., 2004; Galley et al., 2006).
2. **Local feature calculation:** Statistical and indicator features, as described by Chiang et al. (2009), are calculated over the rule grammar.
3. **Language model calculation:** A Kneser-Ney-interpolated 5-gram language model (Chen and Goodman, 1996) is learned from the yield of the target training trees.
4. **Decoding:** A beamed bottom-up chart decoder (Pust and Knight, 2009; Hopkins and Langmead, 2010) calculates the optimal derivations given a source string and feature parameter set.
5. **Tuning:** Feature parameters are optimized using the MIRA learning approach (Chiang et al., 2009) to maximize the objective, typically BLEU (Papineni et al., 2002), associated with a tuning corpus.

We initially use this system with no modifications and pretend that English–AMR is a language pair indistinct from any other.

Corpus	Sentences	Tokens
Training	10,313	218,021
Development	1,368	29,484
Test	1,371	30,263

Table 1: Data splits of AMR 1.0, used in this work. Tokens are English, after tokenization.

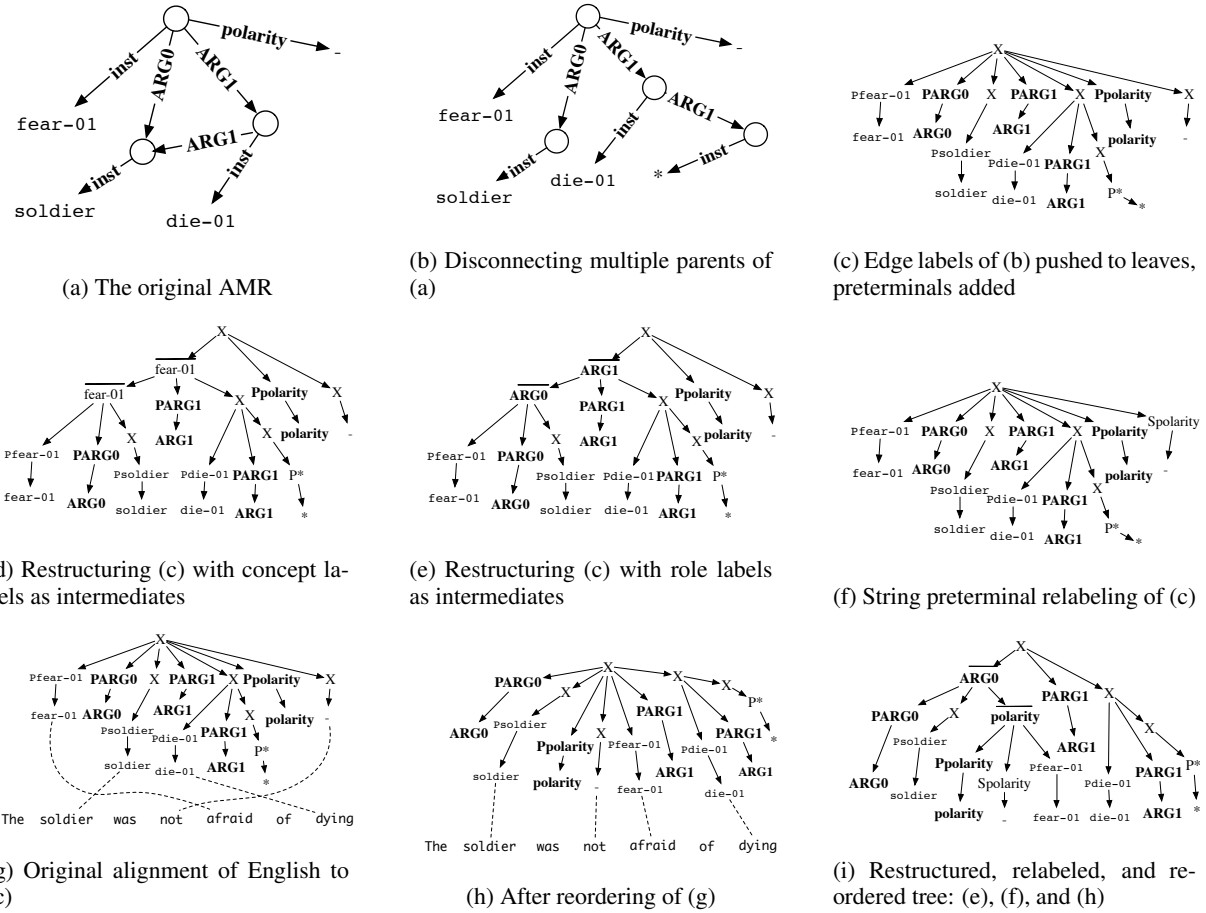
3 Data and Comparisons

We use English–AMR data from the AMR 1.0 corpus, LDC Catalog number 2014T12. In contrast to narrow-domain data sources that are often used in work related to semantic parsing (Price, 1990; Zelle, 1995; Kuhlmann et al., 2004), the AMR corpus covers a broad range of news and web forum data. We use the training, development, and test splits specified in the AMR corpus (Table 1). The training set is used for rule extraction, language modeling, and statistical rule feature calculation. The development set is used both for parameter optimization and qualitatively for hill-climbing. The test set is held out blind for evaluation. We preprocess the English with a simple rule-based tokenizer and, except where noted, lowercase all data. We obtain English–AMR alignments by using the unsupervised alignment approach of Pourdamghani et al. (2014), which linearizes the AMR and then applies the models of Brown et al. (1993) with an additional symmetrization constraint.

All parsing results reported in this work are obtained with the Smatch 1.0 software (Cai and Knight, 2013). We compare our results to those of Flanigan et al. (2014) on the AMR 1.0 data splits; we run that work's JAMR software according to the provided instructions.[3] We also compare our results to published scores in the recent work of Wang et al. (2015). Their work uses slightly different data than that used here [4] but in practice we have not seen significant variation in results.

[3] https://github.com/jflanigan/jamr

[4] LDC2013E117, a pre-released version of LDC2014T12 that is not generally available.



(a) The original AMR

(b) Disconnecting multiple parents of (a)

(c) Edge labels of (b) pushed to leaves, preterminals added

(d) Restructuring (c) with concept labels as intermediates

(e) Restructuring (c) with role labels as intermediates

(f) String preterminal relabeling of (c)

(g) Original alignment of English to (c)

(h) After reordering of (g)

(i) Restructured, relabeled, and reordered tree: (e), (f), and (h)

Figure 3: Transformation of AMR into tree structure that is acceptable to GHKM rule extraction (Galley et al., 2004; Galley et al., 2006) and yields good performance.

4 AMR Transformations

In this section we discuss various transformations to our AMR data. Initially, we concern ourselves with converting AMR into a form that is amenable to GHKM rule extraction and string to tree decoding. We then turn to structural transformations designed to improve system performance. Figure 3 progressively shows all the transformations described in this section; the example we follow is shown in its original form in Figure 3a. We note that all transformations are done internally; the input to the final system is a sentence and the output is an AMR. We further observe that all transformations are data-driven and language agnostic.

4.1 Massaging AMRs into Syntax-Style Trees

The relationships in AMR form a directed acyclic graph (DAG), but GHKM requires a tree, so we must begin our transformations by discarding some information. We arbitrarily disconnect all but a single parent from each node (see Figure 3b). This is the only lossy modification we make to our AMR data. As multi-parent relationships occur 1.05 times per training sentence and at least once in 48% of training sentences, this is indeed a regrettable loss. We nevertheless make this modification, since it allows us to use the rest of our string-to-tree tools.

AMR also contains labeled edges, unlike the constituent parse trees we are used to working with in SBMT. These labeled edges have informative content and we would like to use the alignment procedure of Pourdamghani et al. (2014), which aligns words to edges as well as to terminal nodes. So that our AMR trees are compatible with both our desired alignment approach and our desired rule extraction approach, we propagate edge labels to terminals via the following procedure:

1. For each node n in the AMR tree we create a corresponding node m with the all-purpose symbol 'X' in the SBMT-like tree. Outgoing edges from n come in two flavors: *concept*

edges, labeled 'inst,' which connect n to a terminal *concept* such as `fear-01`, and *role edges*, which have a variety of labels such as **ARG0** and **name**, and connect n to another instance or to a *string*.[5] A node has one instance edge and zero or more role edges. We consider each type of edge separately.

2. For each outgoing role edge we insert two unlabeled edges into the corresponding transformation; the first is an edge from m to a terminal bearing the original edge's role label (a so-called *role label edge*), and the second (a *role filler edge*) connects m to the transformation of the original edge's target node, which we process recursively. String targets of a role receive an 'X' preterminal to be consistent with the form of role filler edges.
3. For the outgoing concept edge we insert an unlabeled edge connecting m and the concept. It is unambiguous to determine which of m's edges is the concept edge and which edges constitute role label edges and their corresponding role filler edges, as long as paired label and filler edges are adjacent.
4. Since SBMT expects trees with preterminals, we simply replicate the label identities of concepts and role labels, adding a marker ('P' in Figure 3) to distinguish preterminals.

The complete transformation can be seen in Figure 3c. Apart from multiple parent ancestry, the original AMR can be reconstructed deterministically from this SBMT-compliant rewrite.

4.2 Tree Restructuring

While the transformation in Figure 3c is acceptable to GHKM, and hence an entire end-to-end AMR parser may now be built with SBMT tools, the resulting parser does not exhibit very good performance (Table 3, first line). The trees we are learning on are exceedingly flat, and thus yield rules that do not generalize sufficiently. Rules produced from the top of the tree in Figure 3c, such as that in Figure 4a, are only appropriate for cases where `fear-01` has exactly three roles: **ARG0** (agent), **ARG1** (patient), and **polarity**.

We follow the lead of Wang et al. (2010), who in turn were influenced by similar approaches in monolingual parsing (Collins, 1997; Charniak, 2000), and re-structure trees at nodes with more than three children (i.e. instances with more than one role), to allow generalization of flat structures.

However, our trees are unlike syntactic constituent trees in that they do not have labeled nonterminal nodes, so we have no natural choice of an intermediate ("bar") label. We must choose a meaningful label to characterize an instance and its roles. We initially choose the concept label, resulting in trees like that in Figure 3d, in which a chain of fear-01 nodes is used to unflatten the root, which has instance `fear-01`.

This attempt at re-structuring yields rules like that in Figure 4b, which are general in form but are tied to the concept context in which they were extracted. This leads to many redundant rules and blows up the nonterminal vocabulary size to approximately 8,000, the size of the concept vocabulary. Furthermore, the rules elicited by this procedure encourage undesirable behavior such as the immediate juxtaposition of two rules generating **ARG1**.

We next consider restructuring with the immediately dominant role labels, resulting in trees like that in Figure 3e and rules like that in Figure 4c. The shape of the structure added is the same as in Figure 3d but the bar nodes now take their labels from their second children. This approach leads to more useful rules with fewer undesirable properties.

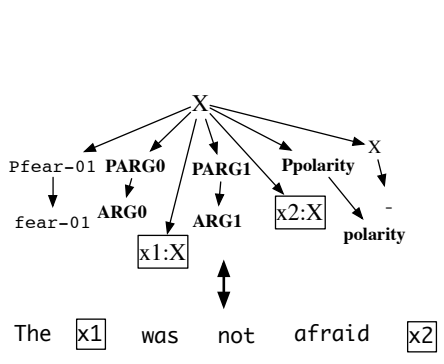
4.3 Tree Relabeling

AMR strings have an effective preterminal label of 'X,' which allows them to compete with full AMR instances at decode time. However, whether or not a role is filled by a string or an instance is highly dependent on the kind of role being filled. The **polarity** and **mode** roles, for instance, are nearly always filled by strings, but **ARG0** and **ARG1** are always filled by instances. The **quant** role, which is used for representation of numerical quantities, can be filled by an instance (e.g. for approximate quantities such as 'about 3') or a string. To capture this behavior we relabel string preterminals of the tree with labels indicating role identity and string subsumption. This relabeling, replaces, for example, one 'X' preterminal in Figure 3c with "Spolarity," as shown in Figure 3f.

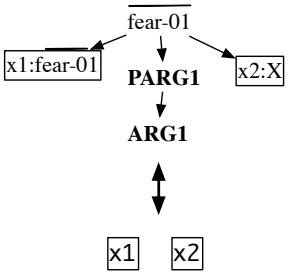
4.4 Tree Reordering

Finally, let us consider the alignments between English and AMR. As is known in SBMT, nonmonotone alignments can lead to large, unwieldy

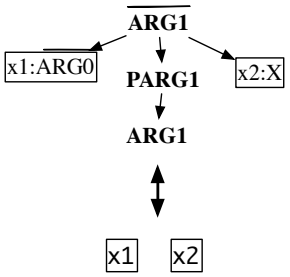
[5] In Figure 3 the negative polarity marker '-' is a string. Disconnected referents labeled '*' are treated as AMR instances with no roles.



(a) A rule extracted from the AMR tree of Figure 3c. All roles seen in training must be used.



(b) A rule from the AMR tree of Figure 3d. Many nearly identical rules of this type are extracted, and this rule can be used multiple times in a single derivation.



(c) A rule from the AMR tree of Figure 3e. This rule can be used independent of the concept context it was extracted from and multiple reuse is discouraged.

Figure 4: Impact of restructuring on rule extraction.

rules and in general make decoding more difficult (May and Knight, 2007). While this is often an unavoidable fact of life when trying to translate between two natural languages with different syntactic behavior, it is an entirely artificial phenomenon in this case. AMR is an *unordered* representation, yet in order to use an SBMT infrastructure we must declare an order of the AMR tree. This means we are free to choose whatever order is most convenient to us, as long as we keep role label edges immediately adjacent to their corresponding role filler edges to preserve conversion back to the edge-labeled AMR form. We thus choose the order that is as close as possible to that of the source yet still preserves these constraints. We use a simple greedy bottom-up approach that permutes the children of each internal node of the unrestructured tree so as to minimize crossings. This leads to a 79% overall reduction in crossings and is exemplified in Figure 3g (before) and Figure 3h (after). We may then restructure our trees, as described above, in an instance-outward manner. The final restructured, relabeled, and reordered tree is shown in Figure 3i.

5 AMR Language Models

We now turn to language models of AMRs, which help us prefer reasonable target structures over unreasonable ones.

Our first language model is unintuitively simple—we pretend there is a language called *AMRese* that consists of yields of our restructured AMRs. An example AMRese string from Figure 3i is '**ARG0** `soldier polarity -` `fear-01` **ARG1** `die-01` **ARG1** *.' We then build a standard n-gram model for AMRese.

It also seems sensible to judge the correctness of an AMR by calculating the empirical probability of the concepts and their relations to each other. This is the motivation behind the following model of an AMR:[6]

We define an AMR instance $i = (c, R)$, where c is a *concept* and R is a set of *roles*. We define an AMR role $r = (l, i)$, where l is a role label, and i is an AMR instance labeled l. For an AMR instance i let $\hat{c}_i$ be the concept of i's parent instance, and $\hat{l}_i$ be the label of the role that i fills with respect to its parent. We also define the special instance and role labels ROOT and STOP. Then, we define $P_{\text{AMR}}(i|\hat{l}_i, \hat{c}_i)$, the conditional probability of AMR instance i given its ancestry as $P_{\text{AMR}}(i = (c, R)|\hat{l}_i, \hat{c}_i) = P(c|\hat{l}_i, \hat{c}_i) \prod_{r \in R} P_{\text{Role}}(r|c) \times P(\text{STOP}|c)$, where $P_{\text{Role}}(r = (l, i)|c) = P(l|c)P_{\text{AMR}}(i|l, c)$.

We define $P(c|\hat{l}_i, \hat{c}_i)$, $P(l|c)$, and $P(\text{STOP}|c)$ as empirical conditional probabilities, Witten-Bell interpolated (Witten and Bell, 1991) to lower-order models by progressively discarding context from the right.[7] We model exactly one STOP event per instance. We define the probability of a full-sentence AMR i as $P_{\text{AMR}}(i|\text{ROOT})$ where ROOT in this case serves as both parent concept and role label.

[6] This model is only defined over AMRs that can be represented as trees, and not over all AMRs. Since tree AMRs are a prerequisite of our system we did not yet investigate whether this model could be sufficiently generalized.

[7] That is, $P(c|\hat{l}_i, \hat{c}_i)$ is interpolated with $P(c|\hat{l}_i)$ and then $P(c)$.

System	Tune	Test
AMRese n-gram LM	61.7	59.7
AMR LM	59.1	57.1
both LMs	62.3	60.6

Table 2: The effect of AMRese n-gram and AMR LMs on Smatch quality.

As an example, the instance associated with concept `die-01` in Figure 3b has $\hat{l}_i$ = **ARG1** and $\hat{c}_i$ = `fear-01`, so we may score it as $P(\texttt{die-01}|\textbf{ARG1}, \texttt{fear-01}) \times P(\textbf{ARG1}|\texttt{die-01}) \times P(\text{STOP}|\texttt{die-01}) \times P(*|\textbf{ARG1}, \texttt{die-01})$

In Table 2 we compare the effect of varying LMs on Smatch quality. The AMR LM by itself is inferior to the AMRese n-gram LM, but combining the two yields superior quality.

6 Adding External Semantic Resources

Although we are engaged in the task of semantic parsing, we have not yet discussed the use of any semantic resources. In this section we rectify that omission.

6.1 Rules from Numerical Quantities and Named Entities

While the majority of string-to-tree rules in SBMT systems are extracted from aligned parallel data, it is common practice to dynamically generate additional rules to handle the translation of dates and numerical quantities, as these follow common patterns and are easily detected at decode-time. We follow this practice here, and additionally detect person names at decode-time using the Stanford Named Entity Recognizer (Finkel et al., 2005). We use cased, tokenized source data to build the decode-time rules. We add indicator features to these rules so that our tuning methods can decide how favorable the resources are. We leave as future work the incorporation of named-entity rules for other classes, since most available named-entity recognition beyond person names is at a granularity level that is incompatible with AMR (e.g. we can recognize 'Location' but not distinguish between 'City' and 'Country').

6.2 Hierarchical Semantic Categories

In order to further generalize our rules, we modify our training data AMRs once more, this time replacing the identity preterminals over concepts

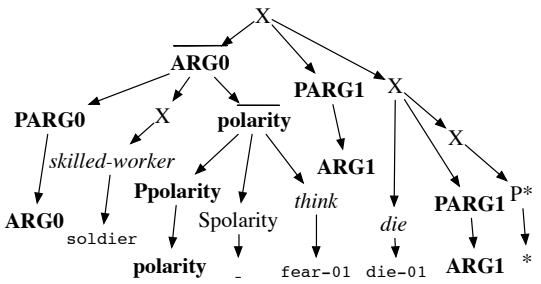


Figure 5: Final modification of the AMR data; semantically clustered preterminal labels are added to concepts.

with preterminals designed to enhance the applicability of our rules in semantically similar contexts. For each concept c expressed in AMR, we consult WordNet (Fellbaum, 1998) and a curated set of gazetteers and vocabulary lists to identify a hierarchy of increasingly general semantic categories that describe the concept. So as not to be overwhelmed by the many fine-grained distinctions present in WordNet, we pre-select around 100 salient semantic categories from the WordNet ontology. When traversing the WordNet hierarchy, we propagate a smoothed count[8] of the number of examples seen per concept sense,[9] combining counts when paths meet. For each selected semantic category s encountered in the traversal, we calculate a weight by dividing the propagated example count for c at s by the frequency s was proposed over all AMR concepts. We then assign c to the highest scoring semantic category s. An example calculation for the concept `computer` is shown in Figure 7.

We apply semantic categories to our data as replacements for identity preterminals of concepts. This leads to more general, more widely-applicable rules. For example, with this transformation, we can parse correctly not only contexts in which "soldiers die," but also contexts in which other kinds of "skilled workers die." Figure 5 shows the addition of semantic preterminals to the tree from Figure 3i. We also incorporate semantic categories into the AMR LM. For concept c, let s_c be the semantic category of c. Then we reformu-

[8] We use very simple smoothing, and add 0.1 to the provided example counts.

[9] Since WordNet senses do not correspond directly to PropBank or AMR senses, we simply use a lexical match and must consider all observed senses for that match.

System	Sec.	Tune	Test
flat trees	4.1	51.6	49.9
concept restructuring	4.2	57.2	55.3
role restructuring (rr)	4.2	**60.8**	**58.6**
rr + string preterminal relabeling (rl)	4.3	**61.3**	**59.7**
rr + rl + reordering (ro)	4.4	**61.7**	**59.7**
rr + rl + ro + AMR LM	5	**62.3**	**60.6**
rr + rl + ro + AMR LM + date/number/name rules (dn)	6.1	**63.3**	**61.3**
rr + rl + ro + AMR LM + dn + semantic categories (sc)	6.2	**66.2**	**64.3**
rr + rl + ro + AMR LM + dn + sc + morphological normalization (mn)	6.3	**67.3**	**65.4**
rr + rl + ro + AMR LM + dn + sc + mn, rule-based alignments	6.4	**68.3**	**66.3**
rr + rl + ro + AMR LM + dn + sc + mn, rule-based + unsupervised alignments	6.4	**69.0**	**67.1**
JAMR (Flanigan et al., 2014)	9	58.8	58.2
dependency parse-based (Wang et al., 2015)	9	N/A	63

Table 3: AMR parsing Smatch scores for the experiments in this work. We provide a cross-reference to the section of this paper that describes each of the evaluated systems. Entries in **bold** are improvements over JAMR (Flanigan et al., 2014). Test entries underlined are improvements over the dependency-based work of Wang et al. (2015). Human inter-annotator Smatch performance is in the 79-83 range (Cai and Knight, 2013).

English	AMRese
tigers	`tiger`
asbestos	`asbestos`
quietly	`quiet`
nonexecutive	`executive` **polarity** '-'
broke up	`break-up-08`

Table 4: Lexical conversions to AMRese form due to the morphological normalization rules described in Section 6.3.

late $P_{\text{AMR}}(i|\hat{l}_i, \hat{c}_i)$ as $P_{\text{AMR}}(i = (c, R)|\hat{l}_i, \hat{c}_i) = P(s_c|\hat{l}_i, s_{\hat{c}_i}, \hat{c}_i)P(c|s_c, \hat{l}_i, s_{\hat{c}_i}, \hat{c}_i) \prod_{r \in R} P_{\text{Role}}(r|c) \times P(\text{STOP}|s_c, c)$, where $P_{\text{Role}}(r = (l, i)|c) = P(l|s_c, c) \times P_{\text{AMR}}(i|l, c)$.

6.3 Morphological Normalization

While we rely heavily on the relationships between words in-text and concept nodes expressed in parallel training data, we find this is not sufficient for complete coverage. Thus we also include a run-time module that generates AMRese base forms at the lexical level, expressing relationships such as those depicted in Table 4. We build these dictionary rules using three resources:

1. An inflectional morphological normalizing table, comprising a lexicon with 84,558 entries, hand-written rules for regular inflectional morphology, and hand-written lists of irregular verbs, nouns, and adjectives.
2. Lists of derivational mappings (e.g. 'quietly' → 'quiet', 'naval' → 'navy').
3. PropBank framesets, which we use, e.g., to map the morphologically normalized 'break up' (from 'broke up') into a sense match, such as `break-up-08`.

6.4 Semantically informed Rule-based Alignments

For our final incorporation of semantic resources we revisit the English-to-AMR alignments used to extract rules. As an alternative to the unsupervised approach of Pourdamghani et al. (2014), we build alignments by taking a linguistically-aware, supervised heuristic approach to alignment:

First, we generate a large number of potential links between English and AMR. We attempt to link English and AMR tokens after conversion through resources such as a morphological analyzer, a list of 3,235 pertainym pairs (e.g. adj-'gubernatorial' → noun-'governor'), a list of 2,444 adverb/adjective pairs (e.g. 'humbly' → 'humble'), a list of 2,076 negative polarity pairs (e.g. 'illegal' → 'legal'), and a list of 2,794 known English-AMR transformational relationships (e.g. 'asleep' → `sleep-01`, 'advertiser' → `person` **ARG0-of** `advertise-01`, 'Greenwich Mean Time' → `GMT`). These links are then culled based on context and AMR structure. For example, in the sentence "The big fish ate the little fish," ini-

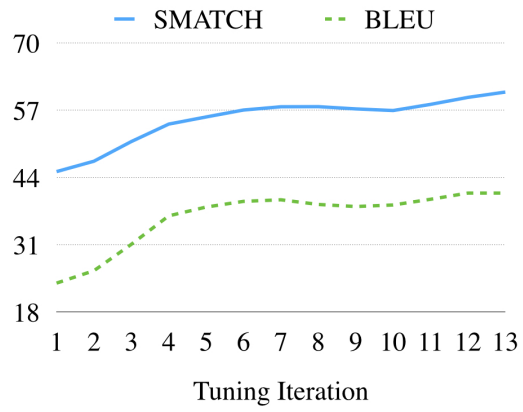


Figure 6: BLEU of AMRese and Smatch correlate closely when tuning.

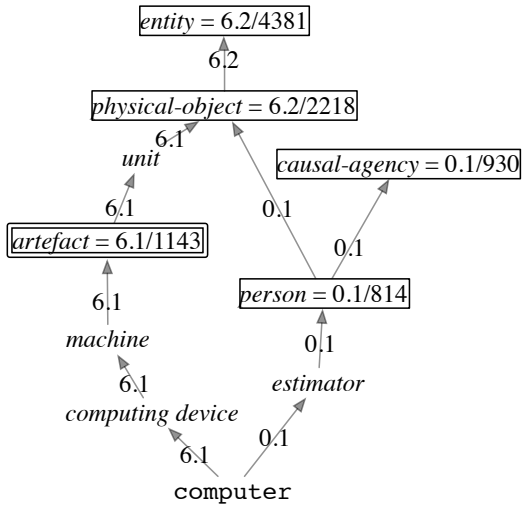


Figure 7: WordNet hierarchy for `computer`. Pre-selected salient WordNet categories are boxed. Smoothed sense counts are propagated up the hierarchy and re-combined at join points. Scores are calculated by dividing propagated sense count by count of the category's prevalence over the set of AMR concepts. The double box indicates the selection of *artefact* as the category label for `computer`.

tially both English 'fish' are aligned to both AMR `fish`. However, based on the context of 'big' and 'little' the spurious links are removed.

In our experiments we explore both replacing the unsupervised alignments of Pourdamghani et al. (2014) with these alignments and concatenating the two alignment sets together, essentially doubling the size of the training corpus. Because the different alignments yield different target-side tree reorderings, it is necessary to build separate 5-gram AMRese language models.[10] When using both alignment sets together, we also use both AMRese language models simultaneously.

7 Tuning

We would like to tune our feature weights to maximize Smatch directly. However, a very convenient alternative is to compare the AMRese yields of candidate AMR parses to those of reference AMRese strings, using a BLEU objective and forest-based MIRA (Chiang et al., 2009). Figure 6 shows that MIRA tuning with BLEU over AMRese tracks closely with Smatch. Note that, for experiments using reordered AMR trees, this requires obtaining similarly permuted reference tuning AMRese and hence requires alignments on the development corpus. When using unsupervised alignments we may simply run inference on the trained alignment model to obtain development alignments. The rule-based aligner runs one sentence at a time and can be employed on the development corpus. When using both sets of alignments, each approach's AMRese is used as a development reference (i.e. each development sentence has two possible reference translations).

[10]The AMR LM is insensitive to reordering so we do not need to vary it when varying alignments.

8 Results and Discussion

Our AMR parser's performance is shown in Table 3. We progressively show the incremental improvements and compare to the systems of Flanigan et al. (2014) and Wang et al. (2015). Purely transforming AMR data into a form that is compatible with the SBMT pipeline yields suboptimal results, but by adding role-based restructuring, relabeling, and reordering, as described in Section 4, we are able to surpass Flanigan et al. (2014). Adding an AMR LM and semantic resources increases scores further, outperforming Wang et al. (2015). Rule-based alignments are an improvement upon unsupervised alignments, but concatenating the two alignments is even better. We compare rule set sizes of the various systems in Table 5; initially we improve the rule set by removing numerous overly brittle rules but then successive changes progressively add useful rules. The parser is available for public download and use at `http://amr.isi.edu`.

System	Rules
flat trees	1,430,124
concept restructuring	678,265
role restructuring (rr)	660,582
rr + preterminal relabeling (rl)	661,127
rr + rl + semantic categories (sc)	765,720
rr + rl + sc + reordering (ro)	790,624
rr + rl + sc + ro + rule-based alignments	908,318
rr + rl + sc + ro + both alignments	1,306,624

Table 5: Comparison of extracted rule set size on the systems evaluated in this work. Note that, as compared to Table 3, only systems that affect the rule size are listed.

9 Related Work

The first work that addressed AMR parsing was that of Flanigan et al. (2014). In that work, multiple discriminatively trained models are used to identify individual concept instances and then a minimum spanning tree algorithm connects the concepts. That work was extended and improved upon by Werling et al. (2015). Recent work by Wang et al. (2015) also uses a two-pass approach; dependency parses are modified by a tree-walking algorithm that adds edge labels and restructures to resolve discrepancies between dependency standards and AMR's specification. In contrast to these works, we use a single-pass approach and re-use existing machine translation architecture, adapting to the AMR parsing task by modifying training data and adding lightweight AMR-specific features.

Several other recent works have used a machine translation approach to semantic parsing, but all have been applied to domain data that is much narrower and an order of magnitude smaller than that of AMR, primarily the Geoquery corpus (Zelle and Mooney, 1996). The WASP system of Wong and Mooney (2006) uses hierarchical SMT techniques and does not apply semantic-specific improvements; its extension (Wong and Mooney, 2007) incorporates a target-side reordering component much like the one presented in Section 4.4. Jones et al. (2012a) cast semantic parsing as an instance of hyperedge replacement grammar transduction; like this work they use an IBM model-influenced alignment algorithm and a GHKM-based extraction algorithm. Andreas et al. (2013) use phrase-based and hierarchical SMT techniques on Geoquery. Like this work, they perform a transformation of the input semantic representation so that it is amenable to use in an existing machine translation system. However, they are unable to reach the state of the art in performance. Li et al. (2013) directly address GHKM's word-to-terminal alignment requirement by extending that algorithm to handle word-to-node alignment.

Our SBMT system is grounded in the theory of tree transducers, which were applied to the task of semantic parsing by Jones et al. (2011; 2012b).

Semantic parsing in general and AMR parsing specifically can be considered a subsumption of many semantic resolution sub-tasks, e.g. named entity recognition (Nadeau and Sekine, 2007), semantic role labeling (Gildea and Jurafsky, 2002), word sense disambiguation (Navigli, 2009) and relation finding (Bach and Badaskar, 2007).

10 Conclusion

By restructuring our AMRs we are able to convert a sophisticated SBMT engine into a baseline semantic parser with little additional effort. By further restructuring our data to appropriately model the behavior we want to capture, we are able to rapidly achieve state-of-the-art results. Finally, by incorporating novel language models and external semantic resources, we are able to increase quality even more. This is not the last word on AMR parsing, as fortunately, machine translation technology provides more low-hanging fruit to pursue.

Acknowledgments

Thanks to Julian Schamper and Allen Schmaltz for early attempts at this problem. This work was sponsored by DARPA DEFT (FA8750-13-2-0045), DARPA BOLT (HR0011-12-C-0014), and DARPA Big Mechanism (W911NF-14-1-0364).

References

Jacob Andreas, Andreas Vlachos, and Stephen Clark. 2013. Semantic parsing as machine translation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 47–52, Sofia, Bulgaria, August. Association for Computational Linguistics.

Nguyen Bach and Sameer Badaskar. 2007. A Review of Relation Extraction. Unpublished. `http://www.cs.cmu.edu/~nbach/papers/A-survey-on-Relation-Extraction.pdf`.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186, Sofia, Bulgaria, August. Association for Computational Linguistics.

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

Shu Cai and Kevin Knight. 2013. Smatch: an evaluation metric for semantic feature structures. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 748–752, Sofia, Bulgaria, August. Association for Computational Linguistics.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of the 1st North American Chapter of the Association for Computational Linguistics Conference*, NAACL 2000, pages 132–139, Stroudsburg, PA, USA. Association for Computational Linguistics.

Stanley F. Chen and Joshua Goodman. 1996. An empirical study of smoothing techniques for language modeling. In *Proceedings of the 34th Annual Meeting on Association for Computational Linguistics*, ACL '96, pages 310–318, Stroudsburg, PA, USA. Association for Computational Linguistics.

David Chiang, Kevin Knight, and Wei Wang. 2009. 11,001 new features for statistical machine translation. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 218–226, Boulder, Colorado, June. Association for Computational Linguistics.

Michael Collins. 1997. Three generative, lexicalised models for statistical parsing. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, pages 16–23, Madrid, Spain, July. Association for Computational Linguistics.

Christiane Fellbaum. 1998. *WordNet: An Electronic Lexical Database*. Bradford Books.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 363–370, Ann Arbor, Michigan, June. Association for Computational Linguistics.

Jeffrey Flanigan, Sam Thomson, Jaime Carbonell, Chris Dyer, and Noah A. Smith. 2014. A discriminative graph-based parser for the abstract meaning representation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1426–1436, Baltimore, Maryland, June. Association for Computational Linguistics.

Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. 2004. What's in a translation rule? In Daniel Marcu Susan Dumais and Salim Roukos, editors, *HLT-NAACL 2004: Main Proceedings*, pages 273–280, Boston, Massachusetts, USA, May 2 - May 7. Association for Computational Linguistics.

Michel Galley, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang, and Ignacio Thayer. 2006. Scalable inference and training of context-rich syntactic translation models. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 961–968, Sydney, Australia, July. Association for Computational Linguistics.

Daniel Gildea and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. *Computational Linguistics*, 28(3):245–288, September.

Mark Hopkins and Greg Langmead. 2010. SCFG decoding without binarization. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 646–655, Cambridge, MA, October. Association for Computational Linguistics.

Matthias Huck, Hieu Hoang, and Philipp Koehn. 2014. Augmenting string-to-tree and tree-to-string translation with non-syntactic phrases. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 486–498, Baltimore, Maryland, USA, June. Association for Computational Linguistics.

Bevan Jones, Mark Johnson, and Sharon Goldwater. 2011. Formalizing semantic parsing with tree transducers. In *Proceedings of the Australasian Language Technology Association Workshop 2011*, pages 19–28, Canberra, Australia, December.

Bevan Jones, Jacob Andreas, Daniel Bauer, Karl Moritz Hermann, and Kevin Knight. 2012a.

Semantics-based machine translation with hyperedge replacement grammars. In *Proceedings of COLING 2012*, pages 1359–1376, Mumbai, India, December. The COLING 2012 Organizing Committee.

Bevan Jones, Mark Johnson, and Sharon Goldwater. 2012b. Semantic parsing with bayesian tree transducers. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 488–496, Jeju Island, Korea, July. Association for Computational Linguistics.

Paul Kingsbury and Martha Palmer. 2002. From treebank to propbank. In *In Language Resources and Evaluation*.

Gregory Kuhlmann, Peter Stone, Raymond J. Mooney, and Jude W. Shavlik. 2004. Guiding a reinforcement learner with natural language advice: Initial results in robocup soccer. In *The AAAI-2004 Workshop on Supervisory Control of Learning and Adaptive Systems*, July.

Peng Li, Yang Liu, and Maosong Sun. 2013. An extended ghkm algorithm for inducing lambda-scfg. In Marie desJardins and Michael L. Littman, editors, *AAAI*. AAAI Press.

Jonathan May and Kevin Knight. 2007. Syntactic re-alignment models for machine translation. In Jason Eisner and Taku Kudo, editors, *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 360–368, Prague, Czech Republic, June 28 – June 30. Association for Computational Linguistics.

David Nadeau and Satoshi Sekine. 2007. A survey of named entity recognition and classification. *Linguisticae Investigationes*, 30(1):3–26, January. Publisher: John Benjamins Publishing Company.

Roberto Navigli. 2009. Word sense disambiguation: A survey. *ACM Comput. Surv.*, 41(2):10:1–10:69, February.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 311–318, Stroudsburg, PA, USA. Association for Computational Linguistics.

Nima Pourdamghani, Yang Gao, Ulf Hermjakob, and Kevin Knight. 2014. Aligning English strings with Abstract Meaning Representation graphs. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 425–429, Doha, Qatar, October. Association for Computational Linguistics.

P. J. Price. 1990. Evaluation of spoken language systems: The ATIS domain. In *Proceedings of the Workshop on Speech and Natural Language*, HLT '90, pages 91–95, Stroudsburg, PA, USA. Association for Computational Linguistics.

Michael Pust and Kevin Knight. 2009. Faster mt decoding through pervasive laziness. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, pages 141–144, Boulder, Colorado, June. Association for Computational Linguistics.

Wei Wang, Jonathan May, Kevin Knight, and Daniel Marcu. 2010. Re-structuring, re-labeling, and re-aligning for syntax-based machine translation. *Computational Linguistics*, 36(2):247–277, June.

Chuan Wang, Nianwen Xue, and Sameer Pradhan. 2015. A transition-based algorithm for amr parsing. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 366–375, Denver, Colorado, May–June. Association for Computational Linguistics.

Keenon Werling, Gabor Angeli, and Christopher D. Manning. 2015. Robust subgraph generation improves abstract meaning representation parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 982–991, Beijing, China, July. Association for Computational Linguistics.

I.H. Witten and T.C. Bell. 1991. The zero-frequency problem: Estimating the probabilities of novel events in adaptive text compression. *IEEE Transactions on Information Theory*, 37(4).

Yuk Wah Wong and Raymond Mooney. 2006. Learning for semantic parsing with statistical machine translation. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 439–446, New York City, USA, June. Association for Computational Linguistics.

Yuk Wah Wong and Raymond Mooney. 2007. Learning synchronous grammars for semantic parsing with lambda calculus. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 960–967, Prague, Czech Republic, June. Association for Computational Linguistics.

John M. Zelle and Raymond J. Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence - Volume 2*, AAAI'96, pages 1050–1055. AAAI Press.

John Marvin Zelle. 1995. *Using Inductive Logic Programming to Automate the Construction of Natural Language Parsers*. Ph.D. thesis, University of Texas at Austin, Austin, TX, USA.

The Forest Convolutional Network: Compositional Distributional Semantics with a Neural Chart and without Binarization

Phong Le and **Willem Zuidema**
Institute for Logic, Language and Computation
University of Amsterdam, the Netherlands
{p.le,zuidema}@uva.nl

Abstract

According to the principle of compositionality, the meaning of a sentence is computed from the meaning of its parts and the way they are syntactically combined. In practice, however, the syntactic structure is computed by automatic parsers which are far-from-perfect and not tuned to the specifics of the task. Current recursive neural network (RNN) approaches for computing sentence meaning therefore run into a number of practical difficulties, including the need to carefully select a parser appropriate for the task, deciding how and to what extent syntactic context modifies the semantic composition function, as well as on how to transform parse trees to conform to the branching settings (typically, binary branching) of the RNN. This paper introduces a new model, the Forest Convolutional Network, that avoids all of these challenges, by taking a parse forest as input, rather than a single tree, and by allowing arbitrary branching factors. We report improvements over the state-of-the-art in sentiment analysis and question classification.

1 Introduction

For many natural language processing tasks we need to compute meaning representations for sentences from meaning representations of words. In a recent line of research on 'recursive neural networks' (e.g., Socher et al. (2010)), both the word and sentence representations are vectors, and the word vectors ("embeddings") are borrowed from work in distributional semantics or neural language modelling. Sentence representations, in this approach, are computed by recursively applying a neural network that combines two vectors into one (typically according to the syntactic structure provided by an external parser). The network, which thus implements a 'composition function', is optimized for delivering sentence representations that support a given semantic task: sentiment analysis (Irsoy and Cardie, 2014; Le and Zuidema, 2015), paraphrase detection (Socher et al., 2011), semantic relatedness (Tai et al., 2015) etc. Studies with recursive neural networks have yielded promising results on a variety of such tasks.

In this paper, we represent a new recursive neural network architecture that fits squarely in this tradition, but aims to solve a number of difficulties that have arisen in existing work. In particular, the model we propose addresses three issues:

1. how to make the composition functions adaptive, in the sense that they operate adequately for the many different types of combinations (e.g., adjective-noun combinations are of a very different type than VP-PP combinations);

2. how to deal with different branching factors of nodes in the relevant syntactic trees (i.e., we want to avoid having to binarize syntactic trees,[1] but also do not want ternary productions to be completely independent from binary productions);

3. how to deal with uncertainty about the correct parse inside the neural architecture (i.e., we don't want to work with just the best or k-best parses for a sentence according to an external model, but receive an entire distribution over possible parsers).

[1] Eisner (2001, Chapter 2) shows that using flat rules is linguistically beneficial, "most crucially, a flat lexical entry corresponds to the local domain of a headword-the word together with all its semantic arguments and modifiers". From the computational perspective, flat rules make trees less deep, thus avoiding the vanishing gradient problem and capturing long range dependencies.

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1155–1164,
Lisbon, Portugal, 17-21 September 2015.

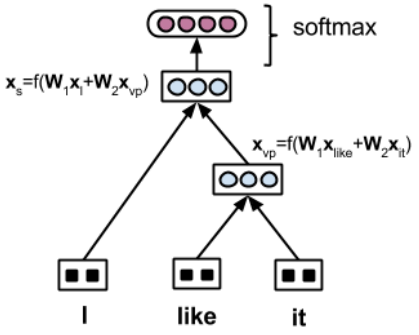


Figure 1: Recursive Neural Network. For simplicity, bias vectors are removed.

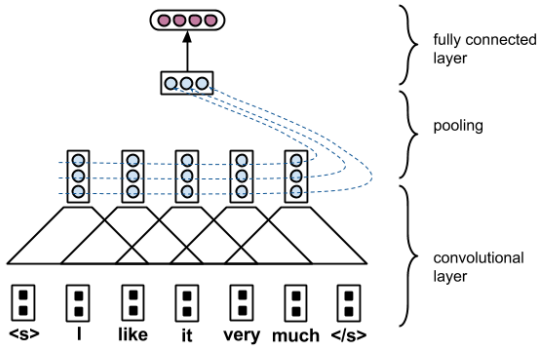


Figure 2: Convolutional Neural Network (one convolutional layer and one fully connected layer) with a window-size-3 kernel.

To solve these challenges we take inspiration from two other traditions: the convolutional neural networks and classic parsing algorithms based on dynamic programming. Including convolution in our network provides a direct solution for issue (2), and turns out, somewhat unexpectedly, to also provide a solution for issue (1). Introducing the chart representation from classic parsing into our architecture then allows us to tackle issue (3). The resulting model, the Forest Convolutional Network, outperforms all other models on a sentiment analysis and question classification task.

2 Background

This section is to introduce the recursive neural network (RNN) and convolutional neural network (CNN) models, on which our work is based.

2.1 Recursive Neural Network

A recursive neural network (RNN) (Goller and Küchler, 1996) is a feed-forward neural network where, given a tree structure, we recursively apply the same weight matrices at each inner node in a bottom-up manner. In order to see how an RNN works, consider the following example. Assume that there is a constituent with parse tree $(S\ \ I\ \ (VP\ \ like\ \ it))$ (Figure 1), and that $\mathbf{x}_I, \mathbf{x}_{like}, \mathbf{x}_{it} \in \mathbb{R}^d$ are the vectorial representations of the three words I, $like$ and it, respectively. We use a neural network which consists of a weight matrix $\mathbf{W}_1 \in \mathbb{R}^{d\times d}$ for left children and a weight matrix $\mathbf{W}_2 \in \mathbb{R}^{d\times d}$ for right children to compute the vector for a parent node in a bottom up manner. Thus, we compute $\mathbf{x}_{VP}$

$$\mathbf{x}_{VP} = f(\mathbf{W}_1\mathbf{x}_{like} + \mathbf{W}_2\mathbf{x}_{it} + \mathbf{b}) \qquad (1)$$

where $\mathbf{b}$ is a bias vector and f is an (non-linear) activation function. Having computed $\mathbf{x}_{VP}$, we can then move one level up in the hierarchy and compute $\mathbf{x}_S$

$$\mathbf{x}_S = f(\mathbf{W}_1\mathbf{x}_I + \mathbf{W}_2\mathbf{x}_{VP} + \mathbf{b})$$

This process is continued until we reach the root node.

For classification tasks, we put a *softmax* layer on the top of the root node, and compute the probability of assigning a class c to an input $\mathbf{x}$ by

$$Pr(c|\mathbf{x}) = \text{softmax}(c) = \frac{e^{u(c,\mathbf{y}_{\text{top}})}}{\sum_{c'\in C} e^{u(c',\mathbf{y}_{\text{top}})}} \qquad (2)$$

where $\left[u(c_1, \mathbf{y}_{top}), ..., u(c_{|C|}, \mathbf{y}_{top})\right]^T = \mathbf{W}_u\mathbf{y}_{top} + \mathbf{b}_u$; C is the set of all possible classes; $\mathbf{W}_u \in \mathbb{R}^{|C|\times d}, \mathbf{b}_u \in \mathbb{R}^{|C|}$ are a weight matrix and a bias vector.

Training an RNN uses the gradient descent method to minimize an objective function $J(\theta)$. The gradient $\partial J/\partial\theta$ is efficiently computed thanks to the back-propagation through structure algorithm (Goller and Küchler, 1996).

Departing from the original RNN model, many extensions have been proposed to enhance its compositionality (Socher et al., 2013; Irsoy and Cardie, 2014; Le and Zuidema, 2015) and applicability (Le and Zuidema, 2014b). The model we are going to propose can be considered as an extension of RNN with an ability to solve the three issues introduced in Section 1.

2.2 Convolutional Neural Network

A convolutional neural network (CNN) (LeCun et al., 1998) is also a feed-forward neural network; it consists of one or more convolutional layers (often with a pooling operation) followed by one or more fully connected layers. This architecture was

invented for computer vision. It then has been widely applied to solve natural language processing tasks (Collobert et al., 2011; Kalchbrenner et al., 2014; Kim, 2014).

To illustrate how a CNN works, the following example uses a simplified model proposed by Collobert et al. (2011) which consists of one convolutional layer with the max pooling operation, followed by one fully connected layer (Figure 2). This CNN uses a *kernel* with window size 3; when we slide this kernel along the sentence "⟨s⟩ I like it very much ⟨/s⟩", we get five vectors:

$$\mathbf{u}^{(1)} = \mathbf{W}_1\mathbf{x}_{<s>} + \mathbf{W}_2\mathbf{x}_I + \mathbf{W}_3\mathbf{x}_{like} + \mathbf{b}_c$$
$$\mathbf{u}^{(2)} = \mathbf{W}_1\mathbf{x}_I + \mathbf{W}_2\mathbf{x}_{like} + \mathbf{W}_3\mathbf{x}_{it} + \mathbf{b}_c$$
$$...$$
$$\mathbf{u}^{(5)} = \mathbf{W}_1\mathbf{x}_{very} + \mathbf{W}_2\mathbf{x}_{much} + \mathbf{W}_3\mathbf{x}_{</s>} + \mathbf{b}_c$$

where $\mathbf{W}_1, \mathbf{W}_2, \mathbf{W}_3 \in \mathbb{R}^{d\times m}$ are weight matrices, $\mathbf{b}_c \in \mathbb{R}^m$ is a bias vector. The max pooling operation is then applied to those resulted vectors in an element-wise manner:

$$\mathbf{x} = \big[\max_{1\le i\le 5} \mathbf{u}_1^{(i)}, ..., \max_{1\le i\le 5} \mathbf{u}_j^{(i)}, ...\big]^T$$

Finally, a fully connected layer is employed

$$\mathbf{y} = f(\mathbf{W}\mathbf{x} + \mathbf{b})$$

where $\mathbf{W}$, $\mathbf{b}$ are a real weight matrix and bias vector, respectively; f is an activation function.

Intuitively, a window-size-k kernel extracts (local) features from k-grams, and is thus able to capture k-gram composition. The max pooling operation is for reducing dimension, forcing the network to discriminate important features from others by assigning high values to them. For instance, if the network is used for sentiment analysis, local features corresponding to k-grams containing the word "like" should receive high values in order to be propagated to the top layer.

3 Forest Convolutional Network

We now first propose a solution to the issues (1) and (2) (i.e., making the composition functions adaptive and dealing with different branching factors), called Recursive convolutional neural network (RCNN), and then a solution to the third issue (i.e., dealing with uncertainty about the correct parse), called Chart Neural Network (ChNN). A combination of them, Forest Convolutional Network (FCN), will be introduced lastly.

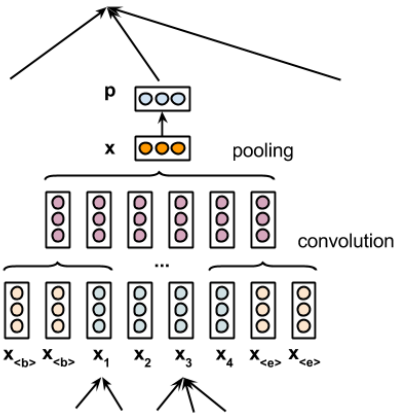


Figure 3: Recursive Convolutional Neural Network with a nonlinear window-size-3 kernel.

3.1 Recursive Convolutional Neural Network[2]

Given a subtree $p \rightarrow x_1 \ ... \ x_l$, an RCNN (Figure 3), like a CNN, slides a window-size-k kernel along the sequence of children $(x_1, ..., x_l)$ to compute a pool of vectors. The max pooling operation followed by a fully connected layer is then applied to this pool to compute a vector for the parent p.

This RCNN differs from the CNN introduced in Section 2.2 at two points. First, we use a non-linear kernel: after linearly transforming input vectors, an activation function is applied. Second, we put $k-1$ padding tokens <b> at the beginning of the children sequence and $k-1$ padding tokens <e> at the end. This thus guarantees that all the children contribute equally to the resulted vector pool, which now has $l+k-1$ vectors.

It is obvious that this RCNN can solve the second issue (i.e., dealing with different branching factors), we now show how it can make the composition functions adaptive. We first see what happens if the window size k is larger than the number of children l, for instance $k=3$ and $l=2$. There are four vectors in the pool

$$\mathbf{u}^{(1)} = f(\mathbf{W}_1\mathbf{x}_{<b>} + \mathbf{W}_2\mathbf{x}_{<b>} + \mathbf{W}_3\mathbf{x}_1 + \mathbf{b}_c)$$
$$\mathbf{u}^{(2)} = f(\mathbf{W}_1\mathbf{x}_{<b>} + \mathbf{W}_2\mathbf{x}_1 + \mathbf{W}_3\mathbf{x}_2 + \mathbf{b}_c)$$
$$\mathbf{u}^{(3)} = f(\mathbf{W}_1\mathbf{x}_1 + \mathbf{W}_2\mathbf{x}_2 + \mathbf{W}_3\mathbf{x}_{<e>} + \mathbf{b}_c)$$
$$\mathbf{u}^{(4)} = f(\mathbf{W}_1\mathbf{x}_2 + \mathbf{W}_2\mathbf{x}_{<e>} + \mathbf{W}_3\mathbf{x}_{<e>} + \mathbf{b}_c)$$

where $\mathbf{W}_1, \mathbf{W}_2, \mathbf{W}_3$ are weight matrices, $\mathbf{b}_c$ is a

[2]While finalizing the current paper we discovered a paper by Zhu et al. (2015) proposing a similar model which is evaluated on syntactic parsing. Our work goes substantially beyond theirs, however, as it takes a parse forest rather than a single tree as input.

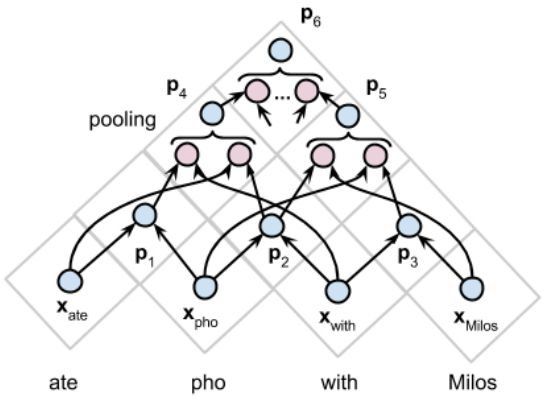


Figure 4: Chart Neural Network.

bias vector, f is an activation function. These four resulted vectors correspond to four ways of composing the two children:

(1) the first child stands alone (e.g., when the information of the second child is not important, it is better to ignore it),

(2,3) the two children are composed with two different weight matrix sets,

(4) the second child stands alone.

Now, imagine that we must handle binary syntactic rules with different head positions such as $S \rightarrow NP\ VP$ (e.g. "Jones runs") where the second child is the head and $VP \rightarrow VBD\ NP$ (e.g., "ate spaghetti") where the first child is the head. We can set those weight matrices such that when multiplying $\mathbf{W}_2$ by the vector of a head, we have a vector with high-value entries. And when multiplying $\mathbf{W}_2$ by the vector of a non-head, or when multiplying $\mathbf{W}_1$ or $\mathbf{W}_3$ by a vector, the resulted vector has low-value entries. This is possible thanks to the max pooling operation and that heads are often more informative than non-heads.

If the window size k is smaller than the number of children l, the argument above is still valid in some cases such as head position. However, there is no longer a direct interaction between any two children whose distance is larger than k.[3] In practice, this problem is not serious because rules with a large number of children are very rare.

3.2 Chart Neural Network

Unseen sentences are always parsed by an automatic parser, which is far from perfect and task-independent. Therefore, a good solution is to give the system a set of parses and let it decide which parse is the best or to combine some of them. The RNN model handles one extreme where this set contains only one parse. We now consider the other extreme where the set contains all possible parses. Because the number of all possible binary parse trees of a length-n sentence is the n-th Catalan number, processing individual parses is not practical. We thus propose a new model working on charts in the CKY style (Younger, 1967), called Chart Neural Network (ChNN).

We describe this model by the following example. Given a phrase "ate pho with Milos", a ChNN will process its parse chart as in Figure 4. Because any 2-word constituent has only one parse, the computation for p_1, p_2, p_3 is identical to Equation 1. For 3-word constituent p_4, because there are two possible productions $p_4 \rightarrow ate\ p_2$ and $p_4 \rightarrow p_1\ with$, we compute one vector for each production

$$\begin{aligned} \mathbf{u}^{(1)} &= f(\mathbf{W}_1\mathbf{x}_{ate} + \mathbf{W}_2\mathbf{p}_2 + \mathbf{b}) \\ \mathbf{u}^{(2)} &= f(\mathbf{W}_1\mathbf{p}_1 + \mathbf{W}_2\mathbf{x}_{with} + \mathbf{b}) \end{aligned} \quad (3)$$

and then apply the max pooling operation to these two vectors to compute $\mathbf{p}_4$. We do the same to compute $\mathbf{p}_5$. Finally, at the top, there are three productions $p_6 \rightarrow ate\ p_5$, $p_6 \rightarrow p_1\ p_3$ and $p_6 \rightarrow p_4\ Milos$. Similarly, we compute one vector for each production and employ the max pooling operation to compute $\mathbf{p}_6$.

Because this ChNN processes a chart like the CKY algorithm, its time complexity is $O(n^2d^2 + n^3d)$ where n and d are the sentence length and the dimension of vectors, respectively.[4] A ChNN is thus notably more complex than an RNN, whose complexity is $O(nd^2)$. Like chart parsing, the complexity can be reduced significantly by pruning the chart before applying the ChNN. This will be discussed right below.

3.3 Forest Convolutional Network

We now introduce the Forest Convolutional Network (FCN) model, which is a combination of the RCNN and the ChNN. The idea is to use an automatic parser to prune a chart[5], debinarize productions (if applicable), and then apply a ChNN

[3] An indirect interaction can be set up through pooling.

[4] In each cell, we apply the matrix-vector multiplication two times and (if the cell is not a leaf) apply the max pooling to a pool of maximally n d-D vectors.

[5] Pruning a chart by an automatic parser is also not perfect. However, the quality of a pruned chart can get very close to human annotation. For instance, the chart pruner proposed by Huang (2008) has a forest oracle of 97.8% F-

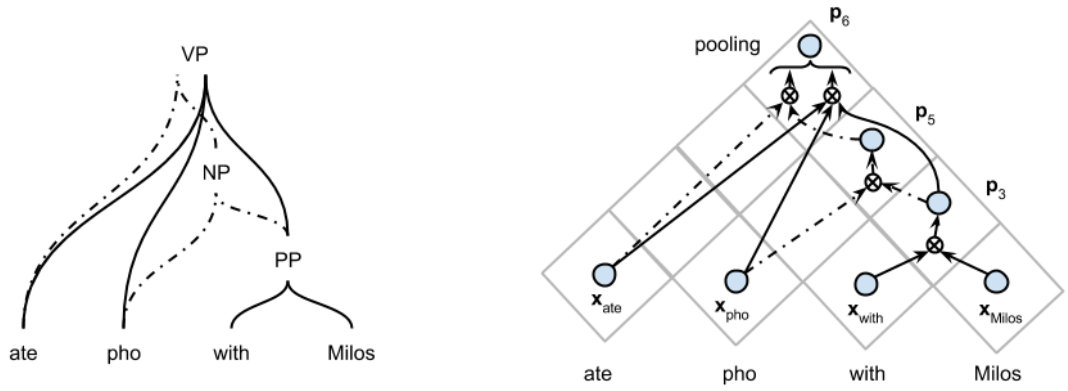


Figure 5: Forest of parses (left) and Forest Convolutional Network (right). $\otimes$ denotes a convolutional layer followed by the max pooling operation and a fully connected layer as in Figure 3.

where the computation in Equation 3 is replaced by a convolutional layer followed by the max pooling operation and a fully connected layer as in the RCNN.

Figure 5 shows an illustration how the FCN works on the phrase "ate pho with Milos". A forest of parses, given by an external parser, comprises two parses $(VP\ ate\ pho\ (PP\ with\ Milos))$ (solid lines) and $(VP\ ate\ (NP\ pho\ (PP\ with\ Milos)))$ (dash-dotted lines). The first parse is the preferred reading if Milos is a person, but the second one is a possible reading (for instance, if Milos is the name of a sauce). Instead of forcing the external parser to decide which one is correct, we let the FCN network do that because it has more information about the context and domain, which are embedded in training data. What the network should do is depicted in Figure 5-right.

Training Training an FCN is similar to training an RNN. We use the mini-batch gradient descent method to minimize an objective function J, which depends on which task this network is applied to. For instance, if the task is sentiment analysis, J is the cross-entropy over the training sentence set $\mathcal{D}$ plus an L2-norm regularization term

$$J(\theta) = -\frac{1}{|\mathcal{D}|}\sum_{s\in\mathcal{D}}\sum_{p\in s}\log Pr(c_p|\mathbf{p}) + \frac{\lambda}{2}||\theta||^2$$

where θ is the parameter set, c_p is the sentiment class of phrase p, $\mathbf{p}$ is the vector representation at the node covering p, $Pr(c_p|\mathbf{p})$ is computed by the softmax function, and λ is the regularization parameter.

The gradient $\partial J/\partial\theta$ is computed efficiently thanks to the back-propagation through structure (Goller and Küchler, 1996). We use the AdaGrad method (Duchi et al., 2011) to automatically update the learning rate for each parameter.

4 Experiments

We evaluate the FCN model with two tasks: question classification and sentiment analysis. The evaluation metric is the classification accuracy.

Our networks were initialized with the 300-D GloVe word embeddings trained on a corpus of 840B words[6] (Pennington et al., 2014). The initial values for a weight matrix were uniformly sampled from the symmetric interval $\left[-\frac{1}{\sqrt{n}}, \frac{1}{\sqrt{n}}\right]$ where n is the number of total input units. In each experiment, a development set was used to tune the model. We run the model ten times and chose the run with the highest performance on the development set. We employed early stopping: training is halted if performance on the development set does not improve after three consecutive epochs.

4.1 Sentiment Analysis

The Stanford Sentiment Treebank (SST)[7] (Socher et al., 2013) which consists of 5-way fine-grained sentiment labels (very negative, negative, neutral, positive, very positive) for 215,154 phrases of 11,855 sentences. We used the standard splitting: 8544 sentences for training, 1101 for development, and 2210 for testing. The average sentence length is 19.1. In addition, the treebank

score on section 23 of the Penn Treebank whereas resulted forests are very compact: the average number of hyperedges per forest is 123.1.

[6] http://nlp.stanford.edu/projects/GloVe/

[7] http://nlp.stanford.edu/sentiment/treebank.html

Model	Fine-grained	Binary
RNTN	45.7	85.4
CNN	48.0	88.1
DCNN	48.5	86.8
PV	48.7	87.8
DRNN	49.8	86.6
LSTM-RNN	49.9	88.0
CT-LSTM	**51.0**	88.0
FCN (dep.)	50.4	88.2
FCN (const.)	**51.0**	**89.1**

Table 1: Accuracies at sentence level on the SST dataset. FCN (dep.) and FCN (const.) denote the FCN with dependency forests and with constituent forests, respectively. The accuracies of RNTN, CNN, DCNN, PV, DRNN, LSTM-RNN and CT-LSTM are copied from the corresponding papers (see text).

also supports binary sentiment (positive, negative) classification by removing neutral labels, leading to: 6920 sentences for training, 872 for development, and 1821 for testing.

All sentences were parsed by Liang Huang's dependency parser[8] (Huang and Sagae, 2010). We used this parser because it generates parse forests and that dependency trees are less deep than constituent trees. In addition, because the SST was annotated in a constituency manner, we also employed the Charniak's constituent parser (Charniak and Johnson, 2005) with Huang (2008)'s forest pruner. We found that the beam width 16 for the dependency parser and the log probability beam 10 for the other worked best. Lower values harmed the system's performance and higher values were not beneficial.

Our FCN has the dimension of vectors at inner nodes 200, a window size for the convolutional kernel of 7, and the activation function $tanh$. It was trained with the learning rate 0.01, the regularization parameter 10^{-4}, and the mini batch size 5. To reduce the average depth of the network, the fully connected layer following the convolutional layer was removed (i.e., $\mathbf{p} = \mathbf{x}$, see Figure 3).

We compare the FCN against other models: the Recursive neural tensor network (RNTN) (Socher et al., 2013), the Convolutional neural network (CNN) (Kim, 2014), the Dynamic convolutional neural network (DCNN) (Kalchbrenner et al., 2014), the Paragraph vectors (PV) (Le and Mikolov, 2014), the Deep recursive neural network (DRNN) (Irsoy and Cardie, 2014), the Recursive neural network with Long short term memory (LSTM-RNN) (Le and Zuidema, 2015) and the Constituent Tree LSTM (CT-LSTM) (Tai et al., 2015).[9]

Table 1 shows the results. Our FCN using constituent forests achieved the highest accuracies in both fine-grained task and binary task, 51% and 89.1%. Comparing to CT-LSTM, although there is no difference in the fine-grained task, the difference in the binary task is significant (1.1%). Comparing to LSTM-RNN, the differences in both tasks are all remarkable (1.1% and 1.1%).

Constituent parsing is clearly more helpful than dependency parsing: the improvements that the FCN got are 0.6% in the fine-grained task and 0.9% in the binary task. We conjecture that, because sentences in the treebank were parsed by a constituent parser (here is the Stanford parser), training with constituent forests is easier.

4.2 Question Classification

In this task we used the TREC question dataset[10] (Li and Roth, 2002) which contains 5952 questions (5452 questions for training and 500 questions for testing). The task is to assign a question to one in six types: ABBREVIATION, ENTITY, DESCRIPTION, HUMAN, LOCATION, NUMERIC. The average length of the questions in the training set is 10.2 whereas in the test set is 7.5. This difference is due to the fact that those questions are from different sources. All questions were parsed by Liang Huang's dependency parser with the beam width 16.

We randomly picked 5% of the training set (272 questions) for validation. Our FCN has the dimension of vectors at inner nodes 200, a window size for the convolutional kernel of 5, and the activation function $tanh$. It was trained with the learning rate 0.01, the regularization parameter 10^{-4}, and the mini batch size 1. The vectors representing the two padding tokens <b>, <e> were fixed to $\mathbf{0}$.

We compare the FCN against the Convolutional neural network (CNN) (Kim, 2014), the Dynamic convolutional neural network (DCNN) (Kalch-

[8] http://acl.cs.qc.edu/~lhuang/software

[9] LSTM-RNN and CT-LSTM are very similar: they are RNNs using LSTMs for composition. Their difference is that LSTM-RNN uses one input gate for each child where as CT-LSTM uses only one input gate for all children.

[10] http://cogcomp.cs.illinois.edu/Data/QA/QC

Model	Acc. (%)
DCNN	93.0
MaxEnt$_H$	93.6
CNN-non-static	93.6
SVM$_S$	**95.0**
LSTM-RNN	93.4
FCN	<u>94.8</u>

Table 2: Accuracies on the TREC question type classification dataset. The accuracies of DCNN, MaxEnt$_H$, CNN-non-static, and SVM$_S$ are copied from the corresponding papers (see text).

brenner et al., 2014), MaxEnt$_H$ (Huang et al., 2008) (which uses MaxEnt with uni-bi-trigrams, POS, wh-word, head word, word shape, parser, hypernyms, WordNet) and the SVM$_S$ (Silva et al., 2011) (which uses SVM with, in addition to features used by MaxEnt$_H$, 60 hand-coded rules). We also include the LSTM-RNN (Le and Zuidema, 2015) whose accuracy was computed by running their published source code[11] on binary trees from the Stanford Parser[12] (Klein and Manning, 2003). This network was also initialized by the 300-D GloVe word embeddings.

Table 2 shows the results.[13] The FCN achieved the second best accuracy, only lightly lower than SVM$_S$ (0.2%). This is a promising result because our network used only parse forests, unsupervisedly pre-trained word embeddings whereas SVM$_S$ used heavily engineered resources. The difference between FCN and the third best is remarkable (1.2%). Interestingly, LSTM-RNN did not perform well on this dataset. This is likely because the questions are short and the parse trees quite shallow, such that the two problems that the LSTM was invented for (long range dependency and vanishing gradient) do not play much of a role.

4.3 Visualization

We visualize the charts we obtained in the sentiment analysis task as in Figure 6. To identify how important each cell is for determining the final vector at the root, we compute the number of features of each that are actually propagated all the way to the root in the successive max pooling operations. The circles in a graph are proportional to this number. Here, to make the contribution of each individual cell clearer, we have set the window size to 1 to avoid direct interactions between cells.

At the lexical level, we can see that the FCN can discriminate important words from the others. Two words "most" and "incoherent" are the key of the sentiment of this sentence: if one of them is replaced by another word (e.g. replacing "most" by "few" or "incoherent" by "coherent"), the sentiment will flip. The punctuation "." however also has a high contribution to the root. This happens to other charts as well. We conjecture that the network uses the vector of "." to store neutral features and propagate them to the root whenever it can not find more useful features in other vectors. Our future work is to examine this.

At the phrasal level, the network tends to group words in grammatical constituents, such as "most of the action setups", "are incoherent". Ill-formed constituents such as "of the action" and "incoherent ." receive little attention from the network.

Interestingly, we can see that the circle of "incoherent" is larger than the circles of any inner cells, suggesting that the network is able to make use of parses containing direct links from that word to the root. This is evidence that the network has an ability of selecting (or combining) parses that are beneficial to this sentiment analysis task.

5 Related Work

The idea that a composition function must be able to change its behaviour on the fly according to input vectors is explored by Socher et al. (2013), Le and Zuidema (2015), among others. The tensor in the former is multiplied with the vector representations of the phrases it is going to combine to define a composition function (a matrix) on the fly, and then multiplies again with these vectors to yield a compound representation. In the LSTM architecture of the latter, there is one input gate for each child in order to control how the vector of the child affects the composition at the parent node. Because the input gate is a function of the vector of the child, the composition function has an *infinite* number of behaviours. In this paper, we instead slide a kernel function along the sequence of children to generate different ways of composition. Although the number of behaviours is limited (and depends on the window size), it simultane-

[11] https://github.com/lephong/lstm-rnn

[12] http://nlp.stanford.edu/software/lex-parser.shtml

[13] While finalizing the current paper we discovered a paper by Ma et al. (2015) proposing a convolutional network model for dependency trees. They report a new state-of-the-art accuracy of 95.6%.

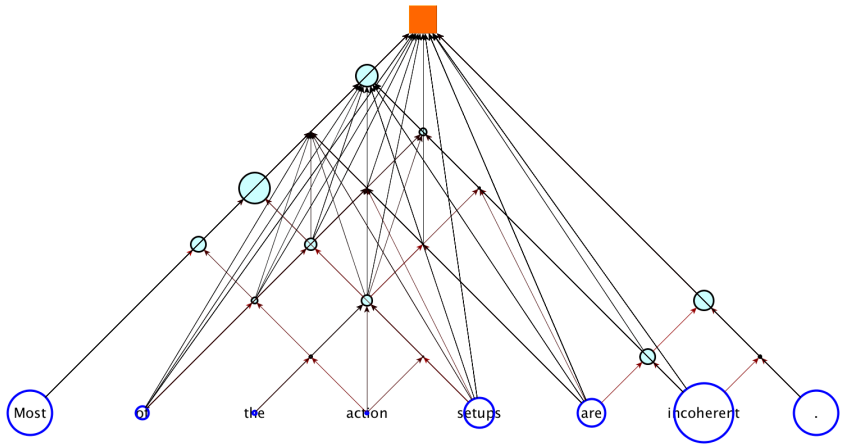


Figure 6: Chart of sentence "Most of the action setups are incoherent ." The size of a circle is propositional to the number of the cell's features that are propagated to the root.

ously provides us with a solution to handle rules with different branching sizes.

Some approaches try to overcome the problem of varying branching sizes. Le and Zuidema (2014b) use different sets of weight matrices for different branching sizes, thus requiring a large number of parameters. Because large branching-size rules are rare, many parameters are infrequently updated during training. Socher et al. (2014), for dependency trees, use a weight matrix for each relative position to the head word (e.g., first-left, second-right). Le and Zuidema (2014a) replace relative positions by dependency relations (e.g., OBJ, SUBJ). These approaches strongly depend on input parse trees and are very sensitive to parsing errors. The approach presented in this paper, on the other hand, does not need the information about the head word position and is less sensitive to parsing errors. Moreover, its number of parameters is independent from the maximal branching size.

Convolutional networks have been widely applied to solve natural language processing tasks. Collobert et al. (2011), Kalchbrenner et al. (2014), and Kim (2014) use convolutional networks to deal with varying length sequences. Recently, Zhu et al. (2015) and Ma et al. (2015) try to intergrate syntactic information by employing parse trees. Ma et al. (2015) extend the work of Kim (2014) by taking into acount dependency relations so that long range dependencies could be captured. The model proposed by Zhu et al. (2015), which is very similar to our Recursive convolutional neural network model, is to use a convolutional network for the composition purpose. Our work, although also employing a convolutional network and syntactic information, goes beyond them: we address the issue of how to deal with uncertainty about the correct parse inside the neural architecture. Therefore, instead of using a single parse, our proposed FCN model takes as input a forest of parses.

Related to our FCN is the Gated recursive convolutional neural network model proposed by Cho et al. (2014) which is stacking $n-1$ convolutional neural layers using a window-size-2 gated kernel (where n is the sentence length). Mapping their network into a chart, each cell is only connected to the two cells right below it. What makes this network special is the gated kernel which is a 3-gate switcher for choosing one of three options: directly transmit the left/right child's vector to the parent node, or compose the vectors of the two children. Thanks to this, the network can capture any binary parse trees by setting those gates properly. However, because only one gate is allowed to open in a cell, the network is not able to capture an arbitrary forest. Our FCN is thus more expressive and flexible than their model.

6 Conclusions

We proposed the Forest Convolutional Network (FCN) model that addresses the three issues: (1) how to make the composition functions adaptive, (2) how to deal with different branching factors of nodes in the relevant syntactic trees, (3) how to deal with uncertainty about the correct parse inside the neural architecture. The key principle is to carry out many different ways of computation

and then choose or combine some of them. For more details, the two first issues are solved by employing a convolutional net for composition. To the third issue, the network takes input as a forest of parses instead of a single parse as in traditional approaches.

Our future work is to focus on how to choose/combine different ways of computation. For instance, we might replace the max pooling by different pooling operations such as mean pooling, k-max pooling (Kalchbrenner et al., 2014), and stochastic pooling (Zeiler and Fergus, 2013). We can even bias the selection/combination toward grammatical constituents by weighing cells by their inside probabilities.

Acknowledgments

We thank our three anonymous reviewers for their comments. This work was funded by the Faculty of Humanities of the University of Amsterdam, through a Digital Humanities fellowship to PL and a position in the New Generation Initiative (NGO) for WZ.

References

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 173–180. Association for Computational Linguistics.

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder–decoder approaches. *Syntax, Semantics and Structure in Statistical Translation*, page 103.

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

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, pages 2121–2159.

Jason Eisner. 2001. *Smoothing a Probabilistic Lexicon via Syntactic Transformations*. Ph.D. thesis, University of Pennsylvania, July. 318 pages.

Christoph Goller and Andreas Küchler. 1996. Learning task-dependent distributed representations by backpropagation through structure. In *International Conference on Neural Networks*, pages 347–352. IEEE.

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

Zhiheng Huang, Marcus Thint, and Zengchang Qin. 2008. Question classification using head words and their hypernyms. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 927–936. Association for Computational Linguistics.

Liang Huang. 2008. Forest reranking: Discriminative parsing with non-local features. In *ACL*, pages 586–594.

Ozan Irsoy and Claire Cardie. 2014. Deep recursive neural networks for compositionality in language. In *Advances in Neural Information Processing Systems*, pages 2096–2104.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 655–665, Baltimore, Maryland, June. Association for Computational Linguistics.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar, October. Association for Computational Linguistics.

Dan Klein and Christopher D Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 423–430. Association for Computational Linguistics.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1188–1196.

Phong Le and Willem Zuidema. 2014a. The inside-outside recursive neural network model for dependency parsing. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Phong Le and Willem Zuidema. 2014b. Inside-outside semantics: A framework for neural models of semantic composition. In *NIPS 2014 Workshop on Deep Learning and Representation Learning*.

Phong Le and Willem Zuidema. 2015. Compositional distributional semantics with long short term memory. In *Proceedings of the Joint Conference on Lexical and Computational Semantics (*SEM)*. Association for Computational Linguistics.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Xin Li and Dan Roth. 2002. Learning question classifiers. In *Proceedings of the 19th international conference on Computational linguistics-Volume 1*, pages 1–7. Association for Computational Linguistics.

Mingbo Ma, Liang Huang, Bowen Zhou, and Bing Xiang. 2015. Dependency-based convolutional neural networks for sentence embedding. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 174–179, Beijing, China, July. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. *Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014)*, 12.

Joao Silva, Luísa Coheur, Ana Cristina Mendes, and Andreas Wichert. 2011. From symbolic to subsymbolic information in question classification. *Artificial Intelligence Review*, 35(2):137–154.

Richard Socher, Christopher D. Manning, and Andrew Y. Ng. 2010. Learning continuous phrase representations and syntactic parsing with recursive neural networks. In *Proceedings of the NIPS-2010 Deep Learning and Unsupervised Feature Learning Workshop*.

Richard Socher, Eric H. Huang, Jeffrey Pennington, Andrew Y. Ng, and Christopher D. Manning. 2011. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. *Advances in Neural Information Processing Systems*, 24:801–809.

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

Richard Socher, Andrej Karpathy, Quoc V Le, Christopher D Manning, and Andrew Y Ng. 2014. Grounded compositional semantics for finding and describing images with sentences. *Transactions of the Association for Computational Linguistics*, 2:207–218.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1556–1566, Beijing, China, July. Association for Computational Linguistics.

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

Matthew D Zeiler and Rob Fergus. 2013. Stochastic pooling for regularization of deep convolutional neural networks. *arXiv preprint arXiv:1301.3557*.

Chenxi Zhu, Xipeng Qiu, Xinchi Chen, and Xuanjing Huang. 2015. A re-ranking model for dependency parser with recursive convolutional neural network. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1159–1168, Beijing, China, July. Association for Computational Linguistics.

Alignment-Based Compositional Semantics for Instruction Following

Jacob Andreas and **Dan Klein**
Computer Science Division
University of California, Berkeley
{jda,klein}@cs.berkeley.edu

Abstract

This paper describes an alignment-based model for interpreting natural language instructions in context. We approach instruction following as a search over plans, scoring sequences of actions conditioned on structured observations of text and the environment. By explicitly modeling both the low-level compositional structure of individual actions and the high-level structure of full plans, we are able to learn both grounded representations of sentence meaning and pragmatic constraints on interpretation. To demonstrate the model's flexibility, we apply it to a diverse set of benchmark tasks. On every task, we outperform strong task-specific baselines, and achieve several new state-of-the-art results.

1 Introduction

In instruction-following tasks, an agent executes a sequence of actions in a real or simulated environment, in response to a sequence of natural language commands. Examples include giving navigational directions to robots and providing hints to automated game-playing agents. Plans specified with natural language exhibit compositionality both at the level of individual actions and at the overall sequence level. This paper describes a framework for learning to follow instructions by leveraging structure at both levels.

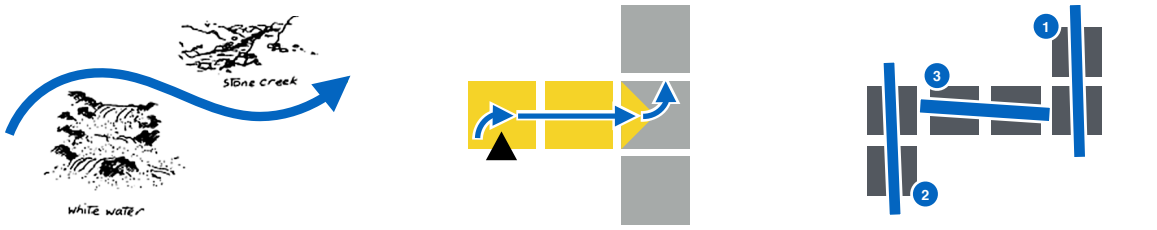
Our primary contribution is a new, alignment-based approach to grounded compositional semantics. Building on related logical approaches (Reddy et al., 2014; Pourdamghani et al., 2014), we recast instruction following as a pair of nested, structured alignment problems. Given instructions and a candidate plan, the model infers a *sequence-to-sequence* alignment between sentences and atomic actions. Within each sentence–action pair, the model infers a *structure-to-structure* alignment between the syntax of the sentence and a graph-based representation of the action.

At a high level, our agent is a block-structured, graph-valued conditional random field, with alignment potentials to relate instructions to actions and transition potentials to encode the environment model (Figure 3). Explicitly modeling sequence-to-sequence alignments between text and actions allows flexible reasoning about action sequences, enabling the agent to determine which actions are specified (perhaps redundantly) by text, and which actions must be performed automatically (in order to satisfy pragmatic constraints on interpretation). Treating instruction following as a sequence prediction problem, rather than a series of independent decisions (Branavan et al., 2009; Artzi and Zettlemoyer, 2013), makes it possible to use general-purpose planning machinery, greatly increasing inferential power.

The fragment of semantics necessary to complete most instruction-following tasks is essentially predicate–argument structure, with limited influence from quantification and scoping. Thus the problem of sentence interpretation can reasonably be modeled as one of finding an alignment between language and the environment it describes. We allow this structure-to-structure alignment—an "overlay" of language onto the world—to be mediated by linguistic structure (in the form of dependency parses) and structured perception (in what we term *grounding graphs*). Our model thereby reasons directly about the relationship between language and observations of the environment, without the need for an intermediate logical representation of sentence meaning. This, in turn, makes it possible to incorporate flexible feature representations that have been difficult to integrate with previous work in semantic parsing.

We apply our approach to three established

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1165–1174,
Lisbon, Portugal, 17-21 September 2015.



. . . right round the white water but stay quite close 'cause you don't otherwise you're going to be in that stone creek . . .

(a) Map reading

Go down the yellow hall. Turn left at the intersection of the yellow and the gray.

(b) Maze navigation

Clear the right column. Then the other column. Then the row.

(c) Puzzle solving

Figure 1: Example tasks handled by our framework. The tasks feature noisy text, over- and under-specification of plans, and challenging search problems.

instruction-following benchmarks: the map reading task of Vogel and Jurafsky (2010), the maze navigation task of MacMahon et al. (2006), and the puzzle solving task of Branavan et al. (2009). An example from each is shown in Figure 1. These benchmarks exhibit a range of qualitative properties—both in the length and complexity of their plans, and in the quantity and quality of accompanying language. Each task has been studied in isolation, but we are unaware of any published approaches capable of robustly handling all three. Our general model outperforms strong, task-specific baselines in each case, achieving relative error reductions of 15–20% over several state-of-the-art results. Experiments demonstrate the importance of our contributions in both compositional semantics and search over plans. We have released all code for this project at `github.com/jacobandreas/instructions`.

2 Related work

Existing work on instruction following can be roughly divided into two families: semantic parsers and linear policy estimators.

Semantic parsers Parser-based approaches (Chen and Mooney, 2011; Artzi and Zettlemoyer, 2013; Kim and Mooney, 2013) map from text into a formal language representing commands. These take familiar structured prediction models for semantic parsing (Zettlemoyer and Collins, 2005; Wong and Mooney, 2006), and train them with task-provided supervision. Instead of attempting to match the structure of a manually-annotated semantic parse, semantic parsers for instruction following are trained to maximize a reward signal provided by black-box execution of the predicted command in the environment. (It is possible to think of response-based learning for question answering (Liang et al., 2013) as a special case.)

This approach uses a well-studied mechanism for compositional interpretation of language, but is subject to certain limitations. Because the environment is manipulated only through black-box execution of the completed semantic parse, there is no way to incorporate current or future environment state into the scoring function. It is also in general necessary to hand-engineer a task-specific formal language for describing agent behavior. Thus it is extremely difficult to work with environments that cannot be modeled with a fixed inventory of predicates (e.g. those involving novel strings or arbitrary real quantities).

Much of contemporary work in this family is evaluated on the maze navigation task introduced by MacMahon et al. (2006). Dukes (2013) also introduced a "blocks world" task for situated parsing of spatial robot commands.

Linear policy estimators An alternative family of approaches is based on learning a policy over primitive actions directly (Branavan et al., 2009; Vogel and Jurafsky, 2010).[1] Policy-based approaches instantiate a Markov decision process representing the action domain, and apply standard supervised or reinforcement-learning approaches to learn a function for greedily selecting among actions. In linear policy approximators, natural language instructions are incorporated directly into state observations, and reading order

[1]This is distinct from semantic parsers in which greedy inference happens to have an interpretation as a policy (Vlachos and Clark, 2014).

becomes part of the action selection process.

Almost all existing policy-learning approaches make use of an unstructured parameterization, with a single (flat) feature vector representing all text and observations. Such approaches are thus restricted to problems that are simple enough (and have small enough action spaces) to be effectively characterized in this fashion. While there is a great deal of flexibility in the choice of feature function (which is free to inspect the current and future state of the environment, the whole instruction sequence, etc.), standard linear policy estimators have no way to model compositionality in language or actions.

Agents in this family have been evaluated on a variety of tasks, including map reading (Anderson et al., 1991) and gameplay (Branavan et al., 2009).

Though both families address the same class of instruction-following problems, they have been applied to a totally disjoint set of tasks. It should be emphasized that there is nothing inherent to policy learning that prevents the use of compositional structure, and nothing inherent to general compositional models that prevents more complicated dependence on environment state. Indeed, previous work (Branavan et al., 2011; Narasimhan et al., 2015) uses aspects of both to solve a different class of gameplay problems. In some sense, our goal in this paper is simply to combine the strengths of semantic parsers and linear policy estimators for fully general instruction following. As we shall see, however, this requires changes to many aspects of representation, learning and inference.

3 Representations

We wish to train a model capable of following commands in a simulated environment. We do so by presenting the model with a sequence of training pairs $(\mathbf{x}, \mathbf{y})$, where each $\mathbf{x}$ is a sequence of *natural language instructions* $(x_1, x_2, \ldots, x_m)$, e.g.:

(*Go down the yellow hall.*, *Turn left.*, ...)

and each $\mathbf{y}$ is a demonstrated *action sequence* $(y_1, y_2, \ldots, y_n)$, e.g.:

(`rotate(90)`, `move(2)`, ...)

Given a start state, $\mathbf{y}$ can equivalently be characterized by a sequence of (state, action, state) triples resulting from execution of the environment model. An example instruction is shown in Figure 2a. An example action, situated in the environment where it occurs, is shown in Figure 2e.

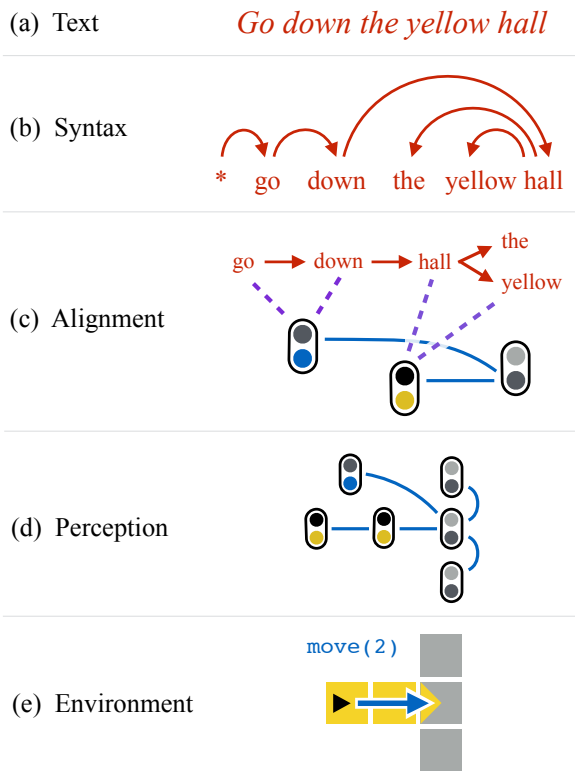


Figure 2: Structure-to-structure alignment connecting a single sentence (via its syntactic analysis) to the environment state (via its grounding graph). The connecting alignments take the place of a traditional semantic parse and allow flexible, feature-driven linking between lexical primitives and perceptual factors.

Our model performs compositional interpretation of instructions by leveraging existing structure inherent in both text and actions. Thus we interpret x_i and y_j not as raw strings and primitive actions, but rather as structured objects.

Linguistic structure We assume access to a pretrained parser, and in particular that each of the instructions x_i is represented by a tree-structured dependency parse. An example is shown in Figure 2b.

Action structure By analogy to the representation of instructions as parse trees, we assume that each (state, action, state) triple (provided by the environment model) can be characterized by a *grounding graph*. The structure and content of this representation is task-specific. An example grounding graph for the maze navigation task is

shown in Figure 2d. The example contains a node corresponding to the primitive action `move(2)` (in the upper left), and several nodes corresponding to locations in the environment that are visible after the action is performed.

Each node in the graph (and, though not depicted, each edge) is decorated with a list of features. These features might be simple indicators (e.g. whether the primitive action performed was `move` or `rotate`), real values (the distance traveled) or even string-valued (English-language names of visible landmarks, if available in the environment description). Formally, a grounding graph consists of a tuple $(V, E, \mathcal{L}, f_V, f_E)$, with

- V a set of vertices
- $E \in V \times V$ a set of (directed) edges
- $\mathcal{L}$ a space of labels (numbers, strings, etc.)
- $f_V : V \to 2^{\mathcal{L}}$ a vertex feature function
- $f_E : E \to 2^{\mathcal{L}}$ an edge feature function

In this paper we have tried to remain agnostic to details of graph construction. Our goal with the grounding graph framework is simply to accommodate a wider range of modeling decisions than allowed by existing formalisms. Graphs might be constructed directly, given access to a structured virtual environment (as in all experiments in this paper), or alternatively from outputs of a perceptual system. For our experiments, we have remained as close as possible to task representations described in the existing literature. Details for each task can be found in the accompanying software package.

Graph-based representations are extremely common in formal semantics (Jones et al., 2012; Reddy et al., 2014), and the version presented here corresponds to a simple generalization of familiar formal methods. Indeed, if $\mathcal{L}$ is the set of all atomic entities and relations, f_V returns a unique label for every $v \in V$, and f_E always returns a vector with one active feature, we recover the existentially-quantified portion of first order logic exactly, and in this form can implement large parts of classical neo-Davidsonian semantics (Parsons, 1990) using grounding graphs.

Crucially, with an appropriate choice of $\mathcal{L}$ this formalism also makes it possible to go beyond set-theoretic relations, and incorporate string-valued features (like names of entities and landmarks) and real-valued features (like colors and positions) as well.

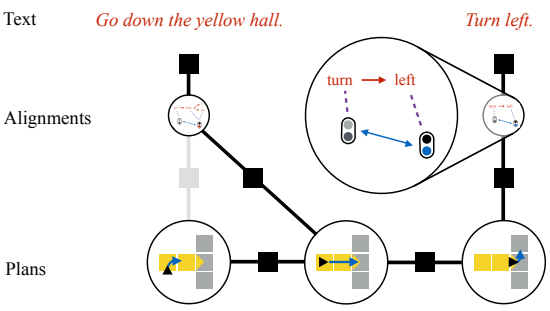


Figure 3: Our model is a conditional random field that describes distributions over state-action sequences conditioned on input text. Each variable's domain is a structured value. Sentences align to a subset of the state–action sequences, with the rest of the states filled in by pragmatic (planning) implication. State-to-state structure represents planning constraints (environment model) while state-to-text structure represents compositional alignment. All potentials are log-linear and feature-driven.

Lexical semantics We must eventually combine features provided by parse trees with features provided by the environment. Examples here might include simple conjunctions (`word=yellow` $\wedge$ `rgb=(0.5, 0.5, 0.0)`) or more complicated computations like edit distance between landmark names and lexical items. Features of the latter kind make it possible to behave correctly in environments containing novel strings or other features unseen during training.

This aspect of the syntax–semantics interface has been troublesome for some logic-based approaches: while past work has used related machinery for selecting lexicon entries (Berant and Liang, 2014) or for rewriting logical forms (Kwiatkowski et al., 2013), the relationship between text and the environment has ultimately been mediated by a discrete (and indeed finite) inventory of predicates. Several recent papers have investigated simple grounded models with real-valued output spaces (Andreas and Klein, 2014; McMahan and Stone, 2015), but we are unaware of any fully compositional system in recent literature that can incorporate observations of these kinds.

Formally, we assume access to a joining feature function $\phi : (2^{\mathcal{L}} \times 2^{\mathcal{L}}) \to \mathbb{R}^d$. As with grounding graphs, our goal is to make the general framework as flexible as possible, and for individual experiments have chosen ϕ to emulate modeling decisions from previous work.

4 Model

As noted in the introduction, we approach instruction following as a sequence prediction problem. Thus we must place a distribution over sequences of actions conditioned on instructions. We decompose the problem into two components, describing interlocking models of "path structure" and "action structure". Path structure captures how sequences of instructions give rise to sequences of actions, while action structure captures the compositional relationship between individual utterances and the actions they specify.

Path structure: aligning utterances to actions

The high-level path structure in the model is depicted in Figure 3. Our goal here is to permit both under- and over-specification of plans, and to expose a planning framework which allows plans to be computed with lookahead (i.e. non-greedily).

These goals are achieved by introducing a sequence of latent alignments between instructions and actions. Consider the multi-step example in Figure 1b. If the first instruction *go down the yellow hall* were interpreted immediately, we would have a presupposition failure—the agent is facing a wall, and cannot move forward at all. Thus an implicit `rotate` action, unspecified by text, must be performed before any explicit instructions can be followed.

To model this, we take the probability of a (text, plan, alignment) triple to be log-proportional to the sum of two quantities:

1. a path-only score $\psi(n;\theta) + \sum_j \psi(y_j;\theta)$
2. a path-and-text score, itself the sum of all pair scores $\psi(x_i, y_j;\theta)$ licensed by the alignment

(1) captures our desire for pragmatic constraints on interpretation, and provides a means of encoding the inherent plausibility of paths. We take $\psi(n;\theta)$ and $\psi(y;\theta)$ to be linear functions of θ. (2) provides context-dependent interpretation of text by means of the structured scoring function $\psi(x, y;\theta)$, described in the next section.

Formally, we associate with each instruction x_i a sequence-to-sequence alignment variable $a_i \in 1 \dots n$ (recalling that n is the number of actions). Then we have[2]

$$p(\mathbf{y},\mathbf{a}|\mathbf{x};\theta) \propto \exp\left\{\psi(n) + \sum_{j=1}^{n}\psi(y_j) + \sum_{i=1}^{m}\sum_{j=1}^{n}\mathbf{1}[a_j = i]\,\psi(x_i, y_j)\right\} \quad (1)$$

We additionally place a monotonicity constraint on the alignment variables. This model is globally normalized, and for a fixed alignment is equivalent to a linear-chain CRF. In this sense it is analogous to IBM Model I (Brown et al., 1993), with the structured potentials $\psi(x_i, y_j)$ taking the place of lexical translation probabilities. While alignment models from machine translation have previously been used to align words to fragments of semantic parses (Wong and Mooney, 2006; Pourdamghani et al., 2014), we are unaware of such models being used to align entire instruction sequences to demonstrations.

Action structure: aligning words to percepts

Intuitively, this scoring function $\psi(x, y)$ should capture how well a given utterance describes an action. If neither the utterances nor the actions had structure (i.e. both could be represented with simple bags of features), we would recover something analogous to the conventional policy-learning approach. As structure is essential for some of our tasks, $\psi(x, y)$ must instead fill the role of a semantic parser in a conventional compositional model.

Our choice of $\psi(x, y)$ is driven by the following fundamental assumptions: *Syntactic relations approximately represent semantic relations. Syntactic proximity implies relational proximity.* In this view, there is an additional hidden structure-to-structure alignment between the grounding graph and the parsed text describing it. [3] Words line up with nodes, and dependencies line up with relations. Visualizations are shown in Figure 2c and the zoomed-in portion of Figure 3.

As with the top-level alignment variables, this approach can viewed as a simple relaxation of a familiar model. CCG-based parsers assume that syntactic type strictly determines semantic type,

[2] Here and the remainder of this paper, we suppress the dependence of the various potentials on θ in the interest of readability.

[3] It is formally possible to regard the sequence-to-sequence and structure-to-structure alignments as a single (structured) random variable. However, the two kinds of alignments are treated differently for purposes of inference, so it is useful to maintain a notational distinction.

and that each lexical item is associated with a small set of functional forms. Here we simply allow all words to license all predicates, multiple words to specify the same predicate, and some edges to be skipped. We instead rely on a scoring function to impose soft versions of the hard constraints typically provided by a grammar. Related models have previously been used for question answering (Reddy et al., 2014; Pasupat and Liang, 2015).

For the moment let us introduce variables b to denote these structure-to-structure alignments. (As will be seen in the following section, it is straightforward to marginalize over all choices of b. Thus the structure-to-structure alignments are never explicitly instantiated during inference, and do not appear in the final form of $\psi(x, y)$.) For a fixed alignment, we define $\psi(x, y, b)$ according to a recurrence relation. Let x^i be the ith word of the sentence, and let y^j be the jth node in the action graph (under some topological ordering). Let $c(i)$ and $c(j)$ give the indices of the dependents of x^i and children of y^j respectively. Finally, let x^{ik} and y^{jl} denote the associated dependency type or relation. Define a "descendant" function:

$$d(i, j) = \big\{(k, l) : \; k \in c(i), \; l \in c(j), \; (k, l) \in b\big\}$$

Then,

$$\psi(x^i, y^j, b) = \exp\Bigg\{\theta^\top \phi(x^i, y^j) + \sum_{(k,l)\in d(x,y)} \Big[\theta^\top \phi\big(x^{ik}, y^{jl}\big) \cdot \; \psi(x^k, y^l, b)\Big]\Bigg\}$$

This is just an unnormalized synchronous derivation between x and y—at any aligned (node, word) pair, the score for the entire derivation is the score produced by combining that word and node, times the scores at all the aligned descendants. Observe that as long as there are no cycles in the dependency parse, it is perfectly acceptable for the relation graph to contain cycles and even self-loops—the recurrence still bottoms out appropriately.

5 Learning and inference

Given a sequence of training pairs $(\mathbf{x}, \mathbf{y})$, we wish to find a parameter setting that maximizes $p(\mathbf{y}|\mathbf{x}; \theta)$. If there were no latent alignments a or b, this would simply involve minimization of a convex objective. The presence of latent variables complicates things. Ideally, we would like to sum over the latent variables, but that sum is intractable. Instead we make a series of variational approximations: first we replace the sum with a maximization, then perform iterated conditional modes, alternating between maximization of the conditional probability of $\mathbf{a}$ and θ. We begin by initializing θ randomly.

Algorithm 1 Computing structure-to-structure alignments

x^i are words in reverse topological order
y^j are grounding graph nodes (root last)
chart is an $m \times n$ array
for $i = 1$ to $|x|$ **do**
  for $j = 1$ to $|y|$ **do**
    score $\leftarrow \exp\{\theta^\top \phi(x^i, y^j)\}$
    for $(k, l) \in d(i, j)$ **do**
      $s \leftarrow \sum_{l \in c(j)} \Big[\exp\{\theta^\top \phi(x^{ik}, y^{jl})\} \cdot$ *chart*$[k, l]\Big]$
      score $\leftarrow$ *score* $\cdot\, s$
    end for
    chart$[i, j] \leftarrow score$
  end for
end for
return *chart*$[n, m]$

As noted in the preceding section, the variable b does not appear in these equations. Conditioned on $\mathbf{a}$, the sum over structure-to-structure $\psi(x, y) = \sum_b \psi(x, y, b)$ can be performed exactly using a simple dynamic program which runs in time $\mathcal{O}(|x||y|)$ (assuming out-degree bounded by a constant, and with $|x|$ and $|y|$ the number of words and graph nodes respectively). This is Algorithm 1.

In our experiments, θ is optimized using L-BFGS (Liu and Nocedal, 1989). Calculation of the gradient with respect to θ requires computation of a normalizing constant involving the sum over $p(\mathbf{x}, \mathbf{y}', \mathbf{a})$ for *all* $\mathbf{y}'$. While in principle the normalizing constant can be computed using the forward algorithm, in practice the state spaces under consideration are so large that even this is intractable. Thus we make an additional approximation, constructing a set $\tilde{Y}$ of alternative actions and taking

$$p(\mathbf{y}, \mathbf{a}|\mathbf{x}) \approx \sum_{j=1}^{n} \frac{\exp\left\{\psi(y_j) + \sum_{i=1}^{m} \mathbf{1}[a_i = j]\psi(x_i, y_i)\right\}}{\sum_{\tilde{y} \in \tilde{Y}} \exp\left\{\psi(\tilde{y}) + \sum_{i=1}^{m} \mathbf{1}[a_i = j]\psi(x_i, \tilde{y})\right\}}$$

$\tilde{Y}$ is constructed by sampling alternative actions from the environment model. Meanwhile, maximization of $\mathbf{a}$ can be performed exactly using the Viterbi algorithm, without computation of normalizers.

Inference at test time involves a slightly different pair of optimization problems. We again perform iterated conditional modes, here on the alignments $\mathbf{a}$ and the unknown output path $\mathbf{y}$. Maximization of $\mathbf{a}$ is accomplished with the Viterbi algorithm, exactly as before; maximization of $\mathbf{y}$ also uses the Viterbi algorithm, or a beam search when this is computationally infeasible. If bounds on path length are known, it is straightforward to adapt these dynamic programs to efficiently consider paths of all lengths.

6 Evaluation

As one of the main advantages of this approach is its generality, we evaluate on several different benchmark tasks for instruction following. These exhibit great diversity in both environment structure and language use. We compare our full system to recent state-of-the-art approaches to each task. In the introduction, we highlighted two core aspects of our approach to semantics: compositionality (by way of grounding graphs and structure-to-structure alignments) and planning (by way of inference with lookahead and sequence-to-sequence alignments). To evaluate these, we additionally present a pair of ablation experiments: *no grounding graphs* (an agent with an unstructured representation of environment state), and *no planning* (a reflex agent with no lookahead).

Map reading Our first application is the map navigation task established by Vogel and Jurafsky (2010), based on data collected for a psychological experiment by Anderson et al. (1991) (Figure 1a). Each training datum consists of a map with a designated starting position, and a collection of landmarks, each labeled with a spatial coordinate and a string name. Names are not always unique, and landmarks in the test set are never observed during training. This map is accompanied by a set of instructions specifying a path from the starting position to some (unlabeled) destination point. These instruction sets are informal and redundant, involving as many as a hundred utterances. They are transcribed from spoken text, so grammatical errors, disfluencies, etc. are common. This is a prime example of a domain that does not lend itself to logical representation—grammars may be too rigid, and previously-unseen landmarks and real-valued positions are handled more easily with feature machinery than predicate logic.

	P	R	F_1
Vogel and Jurafsky (2010)	0.46	0.51	**0.48**
Andreas and Klein (2014)	0.43	0.51	0.45
Model [no planning]	0.44	0.46	0.45
Model [no grounding graphs]	0.52	0.52	0.52
Model [full]	0.51	0.60	**0.55**

Table 1: Evaluation results for the map-reading task. P is precision, R is recall and F_1 is F-measure. Scores are calculated with respect to transitions between landmarks appearing in the reference path (for details see Vogel and Jurafsky (2010)). We use the same train / test split. Some variant of our model achieves the best published results on all three metrics.

Feature	Weight
`word=top ∧ side=North`	1.31
`word=top ∧ side=South`	0.61
`word=top ∧ side=East`	−0.93
`dist=0`	4.51
`dist=1`	2.78
`dist=4`	1.54

Table 2: Learned feature values. The model learns that the word *top* often instructs the navigator to position itself above a landmark, occasionally to position itself below a landmark, but rarely to the side. The bottom portion of the table shows learned *text-independent* constraints: given a choice, near destinations are preferred to far ones (so shorter paths are preferred overall).

The map task was previously studied by Vogel and Jurafsky (2010), who implemented SARSA with a simple set of features. By combining these features with our alignment model and search procedure, we achieve state-of-the-art results on this task by a substantial margin (Table 1).

Some learned feature values are shown in Table 2. The model correctly infers cardinal directions (the example shows the preferred side of a destination landmark modified by the word *top*). Like Vogel et al., we see support for both allocentric references (*you are on top of the hill*) and egocentric references (*the hill is on top of you*). We can also see pragmatics at work: the model learns useful text-independent constraints—in this case, that near destinations should be preferred to far ones.

Maze navigation The next application we consider is the maze navigation task of MacMahon et al. (2006) (Figure 1b). Here, a virtual agent is sit-

	Success (%)
Kim and Mooney (2012)	57.2
Chen (2012)	**57.3**
Model [no planning]	58.9
Model [no grounding graphs]	51.7
Model [full]	**59.6**
Kim and Mooney (2013) [reranked]	62.8
Artzi et al. (2014) [semi-supervised]	**65.3**

Table 3: Evaluation results for the maze navigation task. "Success" shows the percentage of actions resulting in a correct position and orientation after observing a single instruction. We use the leave-one-map-out evaluation employed by previous work.[4] All systems are trained on full action sequences. Our model outperforms several task-specific baselines, as well as a baseline with path structure but no action structure.

uated in a maze (whose hallways are distinguished with various wallpapers, carpets, and the presence of a small set of standard objects), and again given instructions for getting from one point to another. This task has been the subject of focused attention in semantic parsing for several years, resulting in a variety of sophisticated approaches.

Despite superficial similarity to the previous navigation task, the language and plans required for this task are quite different. The proportion of instructions to actions is much higher (so redundancy much lower), and the interpretation of language is highly compositional.

As can be seen in Table 3, we outperform a number of systems purpose-built for this navigation task. We also outperform both variants of our system, most conspicuously the variant without grounding graphs. This highlights the importance of compositional structure. Recent work by Kim and Mooney (2013) and Artzi et al. (2014) has achieved better results; these systems make use of techniques and resources (respectively, discriminative reranking and a seed lexicon of hand-annotated logical forms) that are largely orthogonal to the ones used here, and might be applied to improve our own results as well.

Puzzle solving The last task we consider is the Crossblock task studied by Branavan et al. (2009) (Figure 1c). Here, again, natural language is used to specify a sequence of actions, in this case the solution to a simple game. The environment is simple enough to be captured with a flat feature representation, so there is no distinction between the full model and the variant without grounding graphs.

[4]We specifically targeted the single-sentence version of this evaluation, as an alternative full-sequence evaluation does not align precisely with our data condition.

	Match (%)	Success (%)
No text	54	78
Branavan '09	**63**	–
Model [no planning]	64	66
Model [full]	**70**	**86**

Table 4: Results for the puzzle solving task. "Match" shows the percentage of predicted action sequences that exactly match the annotation. "Success" shows the percentage of predicted action sequences that result in a winning game configuration, regardless of the action sequence performed. Following Branavan et al. (2009), we average across five random train / test folds. Our model achieves state-of-the-art results on this task.

Unlike the other tasks we consider, Crossblock is distinguished by a challenging associated search problem. Here it is nontrivial to find *any* sequence that eliminates all the blocks (the goal of the puzzle). Thus this example allows us measure the effectiveness of our search procedure.

Results are shown in Table 4. As can be seen, our model achieves state-of-the-art performance on this task when attempting to match the human-specified plan exactly. If we are purely concerned with task completion (i.e. solving the puzzle, perhaps not with the exact set of moves specified in the instructions) we can measure this directly. Here, too, we substantially outperform a no-text baseline. Thus it can be seen that text induces a useful heuristic, allowing the model to solve a considerable fraction of problem instances not solved by naïve beam search.

The problem of inducing planning heuristics from side information like text is an important one in its own right, and future work might focus specifically on coupling our system with a more sophisticated planner. Even at present, the results in this section demonstrate the importance of lookahead and high-level reasoning in instruction following.

7 Conclusion

We have described a new alignment-based compositional model for following sequences of natural language instructions, and demonstrated the effectiveness of this model on a variety of tasks. A fully general solution to the problem of contextual interpretation must address a wide range of well-studied problems, but the work we have described

here provides modular interfaces for the study of a number of fundamental linguistic issues from a machine learning perspective. These include:

Pragmatics How do we respond to presupposition failures, and choose among possible interpretations of an instruction disambiguated only by context? The mechanism provided by the sequence-prediction architecture we have described provides a simple answer to this question, and our experimental results demonstrate that the learned pragmatics aid interpretation of instructions in a number of concrete ways: ambiguous references are resolved by proximity in the map reading task, missing steps are inferred from an environment model in the maze navigation task, and vague hints are turned into real plans by knowledge of the rules in Crossblock. A more comprehensive solution might explicitly describe the process by which instruction-givers' own beliefs (expressed as distributions over sequences) give rise to instructions.

Compositional semantics The graph alignment model of semantics presented here is an expressive and computationally efficient generalization of classical logical techniques to accommodate environments like the map task, or those explored in our previous work (Andreas and Klein, 2014). More broadly, our model provides a compositional approach to semantics that does not require an explicit formal language for encoding sentence meaning. Future work might extend this approach to tasks like question answering, where logic-based approaches have been successful.

Our primary goal in this paper has been to explore methods for integrating compositional semantics and the pragmatic context provided by sequential structures. While there is a great deal of work left to do, we find it encouraging that this general approach results in substantial gains across multiple tasks and contexts.

Acknowledgments

The authors would like to thank S.R.K. Branavan for assistance with the Crossblock evaluation. The first author is supported by a National Science Foundation Graduate Fellowship.

References

Anne H. Anderson, Miles Bader, Ellen Gurman Bard, Elizabeth Boyle, Gwyneth Doherty, Simon Garrod, Stephen Isard, Jacqueline Kowtko, Jan McAllister, Jim Miller, et al. 1991. The HCRC map task corpus. *Language and speech*, 34(4):351–366.

Jacob Andreas and Dan Klein. 2014. Grounding language with points and paths in continuous spaces. In *Proceedings of the Conference on Natural Language Learning*.

Yoav Artzi and Luke Zettlemoyer. 2013. Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association for Computational Linguistics*, 1(1):49–62.

Yoav Artzi, Dipanjan Das, and Slav Petrov. 2014. Learning compact lexicons for CCG semantic parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1273–1283, Doha, Qatar, October. Association for Computational Linguistics.

Jonathan Berant and Percy Liang. 2014. Semantic parsing via paraphrasing. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, page 92.

S.R.K. Branavan, Harr Chen, Luke S. Zettlemoyer, and Regina Barzilay. 2009. Reinforcement learning for mapping instructions to actions. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 82–90. Association for Computational Linguistics.

S.R.K. Branavan, David Silver, and Regina Barzilay. 2011. Learning to win by reading manuals in a Monte-Carlo framework. In *Proceedings of the Human Language Technology Conference of the Association for Computational Linguistics*, pages 268–277.

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

David L. Chen and Raymond J. Mooney. 2011. Learning to interpret natural language navigation instructions from observations. In *Proceedings of the Meeting of the Association for the Advancement of Artificial Intelligence*, volume 2, pages 1–2.

David L Chen. 2012. Fast online lexicon learning for grounded language acquisition. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 430–439.

Kais Dukes. 2013. Semantic annotation of robotic spatial commands. In *Language and Technology Conference (LTC)*.

Bevan Jones, Jacob Andreas, Daniel Bauer, Karl Moritz Hermann, and Kevin Knight. 2012. Semantics-based machine translation with hyperedge replacement grammars. In *Proceedings of the International Conference on Computational Linguistics*, pages 1359–1376.

Joohyun Kim and Raymond J. Mooney. 2012. Unsupervised PCFG induction for grounded language learning with highly ambiguous supervision. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 433–444.

Joohyun Kim and Raymond J. Mooney. 2013. Adapting discriminative reranking to grounded language learning. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Tom Kwiatkowski, Eunsol Choi, Yoav Artzi, and Luke Zettlemoyer. 2013. Scaling semantic parsers with on-the-fly ontology matching. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Percy Liang, Michael I. Jordan, and Dan Klein. 2013. Learning dependency-based compositional semantics. *Computational Linguistics*, 39(2):389–446.

Dong Liu and Jorge Nocedal. 1989. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45(1-3):503–528.

Matt MacMahon, Brian Stankiewicz, and Benjamin Kuipers. 2006. Walk the talk: Connecting language, knowledge, and action in route instructions. *Proceedings of the Meeting of the Association for the Advancement of Artificial Intelligence*, 2(6):4.

Brian McMahan and Matthew Stone. 2015. A Bayesian model of grounded color semantics. *Transactions of the Association for Computational Linguistics*, 3:103–115.

Karthik Narasimhan, Tejas Kulkarni, and Regina Barzilay. 2015. Language understanding for text-based games using deep reinforcement learning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Terence Parsons. 1990. *Events in the semantics of English*. MIT Press.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Nima Pourdamghani, Yang Gao, Ulf Hermjakob, and Kevin Knight. 2014. Aligning english strings with abstract meaning representation graphs. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Siva Reddy, Mirella Lapata, and Mark Steedman. 2014. Large-scale semantic parsing without question-answer pairs. *Transactions of the Association for Computational Linguistics*, 2:377–392.

Andreas Vlachos and Stephen Clark. 2014. A new corpus and imitation learning framework for context-dependent semantic parsing. *Transactions of the Association for Computational Linguistics*, 2:547–559.

Adam Vogel and Dan Jurafsky. 2010. Learning to follow navigational directions. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 806–814. Association for Computational Linguistics.

Yuk Wah Wong and Raymond Mooney. 2006. Learning for semantic parsing with statistical machine translation. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 439–446, New York, New York.

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, pages 658–666.

Do we need bigram alignment models? On the effect of alignment quality on transduction accuracy in G2P

Steffen Eger
Text Technology Lab
Goethe University Frankfurt am Main
steeger@em.uni-frankfurt.de

Abstract

We investigate the need for *bigram* alignment models and the benefit of *supervised* alignment techniques in grapheme-to-phoneme (G2P) conversion. Moreover, we quantitatively estimate the relationship between alignment quality and overall G2P system performance. We find that, in English, bigram alignment models do perform better than unigram alignment models on the G2P task. Moreover, we find that supervised alignment techniques may perform considerably better than their unsupervised brethren and that few manually aligned training pairs suffice for them to do so. Finally, we estimate a highly significant impact of alignment quality on overall G2P transcription performance and that this relationship is linear in nature.

1 Introduction

Grapheme-to-phoneme (G2P) conversion is the problem of converting a string of letters into a string of phonetic symbols. Closely related to G2P are other string transduction problems in natural language processing (NLP) such as transliteration (Sherif and Kondrak, 2007), lemmatization (Dreyer et al., 2008), and spelling error correction (Brill and Moore, 2000). The classical learning paradigm in each of these settings is to train a model on pairs of strings $\{(\mathbf{x}, \mathbf{y})\}$ and then to evaluate model performance on test data. While there are exceptions (e.g., (Rao et al., 2015)), most state-of-the-art modelings (e.g., (Jiampojamarn et al., 2007; Bisani and Ney, 2008; Jiampojamarn et al., 2008; Jiampojamarn et al., 2010; Novak et al., 2012)) view string transduction as a two-stage process in which string pairs $(\mathbf{x}, \mathbf{y})$ in the training data are first *aligned*, and then a subsequent (e.g., sequence labeling) module is learned on the aligned data.

ph	oe	n	i	x
f	i	n	ɪ	ks

Table 1: Sample monotone many-to-many alignment between $\mathbf{x} =$ phoenix and $\mathbf{y} =$ fɪnɪks.

State-of-the-art alignments in G2P are characterized by the following properties:

(i) Alignments are *monotone* in that the ordering of characters in input and output sequences is preserved by the alignments. Furthermore, they are *many-to-many* in the sense that several $\mathbf{x}$ sequence characters may be matched up with several $\mathbf{y}$ sequence characters as illustrated in Table 1.

(ii) The alignment is a *latent variable* and learnt in an *unsupervised* manner from pairs of strings in the training data.

(iii) The unsupervised alignment models are *unigram alignment* models insofar as the overall score that the alignment model assigns an alignment is the same for all orderings of the matched-up subsequences (context independence).

To illustrate point (iii), consider, in the field of lemmatization, the case of aligning an inflected word form with the extended infinitive in German, such as *absagt* ('rejects') with *abzusagen* ('to reject'). Critically, the insertion *-zu-* appears in infixal position and a plausible alignment might be as in Table 2. Then, correctly aligning certain

a	b	ϵ	s	a	g	t
a	b	zu	s	a	g	en

Table 2: Alignment between *absagen* and *abzusagen*. Empty string denoted by ϵ.

analogous forms such as *zusagt* ('accepts') with

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1175–1185,
Lisbon, Portugal, 17-21 September 2015.

their corresponding extended infinitive *zuzusagen* ('to accept') is beyond the scope of a unigram alignment model since this cannot distinguish the linguistically correct alignment from the following linguistically incorrect alignment

ϵ	z	u	s	a	g	t
zu	z	u	s	a	g	en

precisely because it has no notion of context.

In this work, we **firstly** address bigram alignment models in G2P. We investigate whether there are phenomena in G2P that require bigram alignment models and, more generally, whether bigram alignment models produce better alignments — with respect to a human gold standard — than unigram alignment models within the G2P setting. We do so, **secondly**, in a *supervised* setting where the model learns from gold-standard alignments. While this may seem an odd scenario at first sight, modern alignment toolkits in the related field of machine translation typically include the possibility to learn both in a supervised and unsupervised manner (Liu et al., 2010; Liu and Sun, 2015). The rationale behind supervised learning models may be that they perform better than unsupervised models, and if alignment quality has a large impact upon subsequent string translation performance, then a supervised model may be a suitable alternative. **Thirdly**, we investigate how alignment quality affects overall G2P performance. This allows us to address whether it is worthwhile to work on better alignment models, which bigram and supervised alignment models promise to be. To our knowledge, all three outlined aspects of alignments — bigram models, supervised learning, and *systematically* estimating the relationship between alignment quality and overall string transduction performance — are novel in the G2P setting and its related fields as outlined; however, see also the related work section.

This work is structured as follows. Section 2 presents definitions and algorithms for uni- and bigram alignment models. Section 3 surveys related work. Section 4 presents our data and Section 5 our experiments. We conclude in Section 6.

2 Uni- and bigram alignment models

We first formally define the problem of aligning two strings $\mathbf{x}$ and $\mathbf{y}$ over arbitrary alphabets in a *monotone and many-to-many* manner. Let $\ell_x = |\mathbf{x}|$ and $\ell_y = |\mathbf{y}|$ denote the lengths of $\mathbf{x}$ and $\mathbf{y}$, respectively. Let $\mathbb{N} = \{0, 1, 2, \ldots\}$, and let $S \subseteq \mathbb{N}^2 \backslash \{(0,0)\}$ be a set defining the valid match-up operations between $\mathbf{x}$ characters and $\mathbf{y}$ characters. In other words, when $(s,t) \in S$, then this means we allow matches of subsequences of $\mathbf{x}$ of length s and subsequences of $\mathbf{y}$ of length t.[1]

It is convenient to define a monotone many-to-many alignment of $\mathbf{x}$ and $\mathbf{y}$ as a $2 \times k$ (for $k \geq 1$ arbitrary) nonnegative integer matrix $\mathbf{A}_{\mathbf{x},\mathbf{y}} \in \mathbb{N}^{2 \times k}$ satisfying $\mathbf{A}_{\mathbf{x},\mathbf{y}} \mathbb{1}_k = \begin{pmatrix} \ell_x \\ \ell_y \end{pmatrix}$, i.e., the two rows of $\mathbf{A}_{\mathbf{x},\mathbf{y}}$ sum up to the lengths of the respective strings,[2] and where each column of $\mathbf{A}_{\mathbf{x},\mathbf{y}}$ lies in S. For any such alignment, we let $(\mathbf{x}_1, \ldots, \mathbf{x}_k)$ be the corresponding induced segmentation of $\mathbf{x}$ and $(\mathbf{y}_1, \ldots, \mathbf{y}_k)$ be the corresponding induced segmentation of $\mathbf{y}$.

Example. For any $S \supseteq \{(1,1), (1,2), (2,1)\}$, the alignment of $\mathbf{x}$ = phoenix and $\mathbf{y}$ = finɪks shown in Table 1 may be represented by the matrix $\mathbf{A}_{\mathbf{x},\mathbf{y}} = \begin{pmatrix} 2 & 2 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 2 \end{pmatrix}$. The corresponding induced segmentations are (ph,oe,n,i,x) and (f,i,n,ɪ,ks).

Let $\mathcal{A}_S(\mathbf{x}, \mathbf{y})$ denote the class of all alignments of $\mathbf{x}$ and $\mathbf{y}$. We call a function $f : \mathcal{A}_S(\mathbf{x}, \mathbf{y}) \to \mathbb{R}$ an *alignment model*. We call an alignment model f a *unigram alignment model* if f takes the form, for any $\mathbf{A}_{\mathbf{x},\mathbf{y}} \in \mathcal{A}_S(\mathbf{x}, \mathbf{y})$,

$$f(\mathbf{A}_{\mathbf{x},\mathbf{y}}) = \sum_{i=1}^{k} \mathsf{sim}_1(\mathbf{x}_i, \mathbf{y}_i) \tag{1}$$

where sim_1 is an arbitrary (real-valued) similarity function measuring similarity of two subsequences. We call an alignment model f a *bigram alignment model* if f takes the form

$$f(\mathbf{A}_{\mathbf{x},\mathbf{y}}) = \sum_{i=1}^{k} \mathsf{sim}_2\Big((\mathbf{x}_i, \mathbf{y}_i), (\mathbf{x}_{i-1}, \mathbf{y}_{i-1})\Big) \tag{2}$$

where sim_2 is an arbitrary (real-valued) similarity function measuring similarity of successive pairs of subsequences.

Example. Let $\mathsf{sim}_1(\mathbf{u}, \mathbf{v})$ be equal to $|\mathbf{u}| \cdot |\mathbf{v}|$ and let $f_{\text{uni}}(\mathbf{A}_{\mathbf{x},\mathbf{y}})$ be as in Eq. (1). Then, f_{uni} is a unigram alignment model that assigns the score

[1] This is sometimes denoted in the manner M-N (e.g., 3-2, 1-0), indicating that M characters of one string may be matched up with N characters of the other string. Analogously, we could write here s-t rather than (s,t).

[2] Here, $\mathbb{1}_k$ denotes the unit vector of dimension k.

$1+1+0+1+1+1+2=7$ to the alignment given in Table 2.

Example. Let $\mathsf{sim}_2\big((\mathbf{u},\mathbf{v}),(\mathbf{u}',\mathbf{v}')\big) = (|\mathbf{u}| \cdot |\mathbf{v}|)^{|\mathbf{v}'|}$ if $|\mathbf{u}| = |\mathbf{u}'|-1$ or $\mathbf{u}=\mathbf{v}$ and -2 otherwise. Let $f_{\text{bi}}(\mathbf{A}_{\mathbf{x},\mathbf{y}})$ be as in Eq. (2). Then, f_{bi} is a bigram alignment model assigning the score $(1\cdot 1)^0+(1\cdot 1)^1+(0\cdot 2)^1+(1\cdot 1)^2+(1\cdot 1)^1+(1\cdot 1)^1-2=3$ to the alignment in Table 2.

In statistical alignment modeling, the task is to find an *optimal* alignment (i.e., one with maximal score) given strings $\mathbf{x}$ and $\mathbf{y}$ and given the alignment model f. When f is a unigram model, this can be solved efficiently via dynamic programming (DP). When f is a bigram alignment model, then finding the optimal alignment can still be solved via DP, by introducing a variable M_{ijqw} denoting the score of the best alignment of $\mathbf{x}(1:i)$ and $\mathbf{y}(1:j)$ that ends in the match-up of $\mathbf{x}(q:i)$ with $\mathbf{y}(w:j)$.[3] The variable M_{ijqw} satisfies a recurrence leading to a DP algorithm, shown in Algorithm 1. The actual alignment can be found by storing pointers to the maximizing steps taken. Running time of the algorithm is $\mathcal{O}(\ell_{\mathbf{x}}^2\ell_{\mathbf{y}}^2|S|)$. Note also that the sketched algorithm is supervised insofar as it assumes that the similarity values $\mathsf{sim}_2(\cdot,\cdot)$ are known. Typically, such alignment algorithms can be converted into unsupervised algorithms in which similarity measures sim are learnt iteratively, e.g., in an EM-like fashion (cf., e.g., Eger (2012), Eger (2013)); however, in this paper, we only investigate the supervised base version as indicated.

3 Related work

Monotone alignments have a long tradition in NLP. The classical Needleman-Wunsch algorithm (Needleman and Wunsch, 1970) computes the optimal alignment between two sequences when only single character matches, mismatches, and skips are allowed. It is a special case of the unigram model (1) for which $S=\{(1,0),(0,1),(1,1)\}$ and sim_1 takes on values from $\{0,-1\}$, depending on whether compared subsequences match or not. As is well-known, this alignment specification is equivalent to the edit distance problem (Levenshtein, 1966) in which the minimal number of insertions, deletions and substitutions is sought that transforms one string into another. *Substring-to-substring edit operations* — or equivalently, (monotone) many-to-many alignments — have appeared in the NLP context, e.g., in Deligne et al. (1995), Brill and Moore (2000), Jiampojamarn et al. (2007), Bisani and Ney (2008), Jiampojamarn et al. (2010), or, significantly earlier, in Ukkonen (1985), Véronis (1988). *Learning* edit distance/monotone alignments in an *unsupervised* manner has been the topic of, e.g., Ristad and Yianilos (1998), Cotterell et al. (2014), besides the works already mentioned. All of these approaches are special cases of our unigram model — i.e., they consider particular S (most prominently, $S=\{(1,0),(0,1),(1,1)\}$) and sim_1.[4] Eger (2015b), Yao and Kondrak (2015), and Eger (2015a) generalize to alignments of multiple strings, but likewise only consider unigram alignment models in their experiments.

Probably the most closely related work to ours is Jiampojamarn and Kondrak (2010). There, older and specialized alignment techniques such as ALINE (Kondrak, 2000) (as well as partly heuristic/semi-automatic alignment methods) are compared with variants of the M2M alignment algorithm, which we also survey. This work does not consider supervised alignments or bigram alignments, as we do. Moreover, Jiampojamarn and Kondrak (2010) also evaluate the impact of alignment quality on overall G2P system accuracy by running a few experiments, finding that better alignment quality does not always translate into better G2P accuracy, but that there is a "strong correlation" between the two. We more thorougly investigate this question, using, arguably, more heterogeneous aligners, and many more experiments. We also quantitatively estimate how alignment quality influences G2P system accuracy on two different languages via linear regression.

Goldwater et al. (2006) study the effect of context in (unsupervised) word/sequence segmentation, which may be considered the one-dimensional specialization of sequence alignment, using a Bayesian method. They find that bigram models greatly outperform unigram models for their task.

Of course, our study is also related to the field of machine translation and its studies on the rela-

[3] We denote by $\mathbf{x}(a:b)$ the substring $x_a x_{a+1}\cdots x_b$ of the string $x_1x_2\cdots x_t$.

[4] In Cotterell et al. (2014), context influences alignments, so that the approach goes beyond the unigram model sketched in (1) (but does not allow for many-to-many match-ups). The contextual dependencies in this model are set up differently from the bigram dependencies in our paper.

Algorithm 1

procedure BIGRAM-ALIGN($\mathbf{x} = x_1 \dots x_n, \mathbf{y} = y_1 \dots y_m$; S, sim_2)
  $M_{ijqw} \leftarrow -\infty$ for all $(i, j, q, w) \in \mathbb{Z}^4$
  $M_{0000} \leftarrow 0$
  for $i = 0 \dots n$ **do**
    for $j = 0 \dots m$ **do**
      for $q = 0 \dots i + 1$ **do**
        for $w = 0 \dots j + 1$ **do**
          if $(i, j, q, w) \neq (0, 0, 0, 0)$ **then**
            if $(i - q + 1, j - w + 1) \in S$ **then**
              $M_{ijqw} = \max_{(a,b)\in S} M_{q-1,w-1,q-a,w-b} + \mathsf{sim}_2\Big(\big(\mathbf{x}(q{:}i), \mathbf{y}(w{:}j)\big), \big(\mathbf{x}(q-a{:}q-1), \mathbf{y}(w-b{:}w-1)\big)\Big)$

tionship between alignment quality and translation performance (Ganchev et al., 2008). In machine translation, the monotonicity assumption of string transduction does typically not hold, however, rendering alignment and translation techniques different and more heuristic in nature.

4 Data and systems

4.1 Data

For English, we conduct experiments on the General American (GA) variant of the Combilex data set (Richmond et al., 2009). This contains about 128 000 grapheme-phoneme pairs as exemplified in Table 3. Importantly, Combilex provides gold-standard alignments, which we will make use of for the supervised alignment models as well as for measuring alignment quality. For German, we randomly extract 3 000 G2P string pairs from CELEX (Baayen et al., 1995). We had a native speaker manually align them so that gold standard alignments are available here, too. Both data sets contain quite complex match-ups of character subsequences such as (2,3) as in English *s-oi-r-ee-s*/s-wOA-r-P-z or (4,1) as in *w-eigh-t*/w-P-t but the majority of match-ups are of type (1,1), (2,1), and, to a lesser degree, (1,2) and (3,1).

Grapheme string	Phoneme string
g-e-n-e-r-a-l	dZ-E-n-@-r-@-l
p-r-o-b-a-t-ion-a-r-y	p-r-@U-b-eI-S-n=-E-r-i
w-oo-d-e-d	w-U-d-@-d
M-u-r-m-a-n-s-k	m-U@-r-m-A-n-s-k

Table 3: Sample grapheme-phoneme string pairs in Combilex, using Combilex notation for the phoneme strings. Gold-standard alignments indicated in an intuitive manner.

4.2 Alignment toolkits/models

The **M2M aligner** (Jiampojamarn et al., 2007), which is based on EM maximum likelihood estimation of alignment parameters, is the classical unsupervised unigram many-to-many aligner in G2P. As has been pointed out (Kubo et al., 2011), M2M greatly overfits the data.[5] This means that when the M2M aligner is given the freedom to align two sequences without restrictions, it matches them up as a whole. The reason is that a (probabilistic) unigram alignment model adds log-probabilities of matched-up subsequences, which, if not appropriately corrected for, makes alignments with few match-ups a priori more likely than alignments with many match-ups, when probabilities of individual match-ups are uniformly or randomly initialized (as is typically the case for EM maximum likelihood estimation in unsupervised models). To address this, M2M must artifically restrain, in our language, the set S to be $\{(1,1),(1,2),(2,1)\}$. In contrast, the **Mpaligner** (Kubo et al., 2011) introduces a prior (or penalty) in the alignment model which favors 'short' matches (s, t) over 'long' ones. Finally, the **Phonetisaurus** aligner (Novak et al., 2012) modifies the M2M aligner by adding additional soft constraints.

Our own alignment model is, as indicated, supervised. We implement a **unigram** alignment model where we specify $\mathsf{sim}_1(\mathbf{u}, \mathbf{v})$ as

$$\alpha \cdot \text{logp}((\mathbf{u}, \mathbf{v})) + \beta \cdot \text{logp}((|\mathbf{u}|, |\mathbf{v}|)) + \gamma \cdot \text{logp}(\mathbf{u}) + \delta \cdot \text{logp}(\mathbf{v}).$$

Here, $\text{logp}(\mathbf{z})$ denotes the log-probability — estimated from the training data — of observing the

[5] See also the discussion in (Goldwater et al., 2006) for the related word segmentation problem.

object $\mathbf{z}$, and α, β, γ and δ are parameters. This specification says that the subsequences $\mathbf{u}$ and $\mathbf{v}$ are similar insofar as (i) $\mathbf{u}$ and $\mathbf{v}$ have been paired frequently in the training data, (ii) the length of $\mathbf{u}$ and the length of $\mathbf{v}$ have been paired frequently, (iii)/(iv) $\mathbf{u}$/$\mathbf{v}$ by itself is likely. We refer to this unigram alignment model as $\mathsf{uni}_{\alpha,\beta,\gamma,\delta}$. We also implement a **bigram** alignment model where we specify $\mathsf{sim}_2\big((\mathbf{u},\mathbf{v}),(\mathbf{u}',\mathbf{v}')\big)$ as

$$\begin{aligned}&\alpha\cdot\mathrm{logp}\big((\mathbf{u},\mathbf{v})\,|\,(\mathbf{u}',\mathbf{v}')\big)\\+&\beta\cdot\mathrm{logp}\big((|\mathbf{u}|,|\mathbf{v}|)\,|\,(|\mathbf{u}'|,|\mathbf{v}'|)\big)\\+&\gamma\cdot\mathrm{logp}\big(\mathbf{u}|\mathbf{u}'\big)+\delta\cdot\mathrm{logp}\big(\mathbf{v}|\mathbf{v}'\big).\end{aligned}$$

Here, $\mathrm{logp}(\mathbf{z}\,|\,\mathbf{z}')$ denotes the logarithm of the conditional probability of observing the object $\mathbf{z}$ following the object $\mathbf{z}'$. We refer to this bigram alignment model as $\mathsf{bi}_{\alpha,\beta,\gamma,\delta}$.

4.3 Transduction systems

We use two string transduction systems for our experiments. The first one is **DirecTL+** (Jiampojamarn et al., 2010), a discriminative string-to-string translation system incorporating joint n-gram features. DirecTL+ is an extension of the model presented in Jiampojamarn et al. (2008) which treats string transduction as a source sequence segmentation and subsequent sequence labeling task. In addition, we use **Phonetisaurus** (Novak et al., 2012), a weighted finite state-based joint n-gram model employing recurrent neural network language model N-best rescoring in decoding. Both systems take aligned pairs of strings as input and from this construct a monotone translation model.[6]

4.4 Measuring alignment quality

We employ two measures of alignment quality. First, we use **word accuracy**, defined as the fraction of correctly aligned sequence pairs in a test sample. This is a very strict measure that penalizes even tiny deviations from the gold standard. Additionally, we measure the **edit distance** between the true alignment $\mathbf{A}_{\mathbf{x},\mathbf{y}}$ and the predicted alignment $\hat{\mathbf{A}}_{\mathbf{x},\mathbf{y}}$. To implement this, we view the two induced segmentations that constitute an alignment — e.g., (ph,oe,n,i,x) and (f,i,n,ɪ,ks) — as strings including splitting signs. Thus, we can compute the edit distance between the gold-standard segmented $\mathbf{x}$ string and the predicted segmentation, and analogously for the $\mathbf{y}$ sequence. Then, we define the edit distance between $\mathbf{A}_{\mathbf{x},\mathbf{y}}$ and $\hat{\mathbf{A}}_{\mathbf{x},\mathbf{y}}$ as the *sum* of these two string edit distances. For a test sample, we indicate so-defined *average* edit distance, averaged over all pairs in the sample.

5 Experiments

5.1 Alignment quality

To measure alignment quality for the different systems, for English, we run experiments on sets of size x+5 000, where x = 1 000, 2 000, 5 000, 10 000, and 20 000. For the supervised models, we consider x as the training data and the 5 000 additional string pairs as test data.[7] To quantify effects when training data is very little, we let x also range over 100 and 500 string pairs for the supervised models. For the unsupervised models, we simply take all x+5 000 string pairs as data to learn from (but evaluate performance only on the 5 000 string pairs, for comparability).

Results are shown in Tables 4, 5, and 6. We first note (Table 4) that the *unsupervised* models perform decently, obtaining accuracy rates of 80% and beyond under appropriate parametrizations. We also observe the M2M aligner's deterioration in performance as we increase its degrees of freedom (allowing it to match subsequences of larger length), confirming our previous remarks. The Mpaligner does not suffer from this problem as it penalizes large matches. Phonetisaurus suffers from the same problems as M2M, but to a lesser degree. Overall, we find that, under optimal parametrizations, Phonetisaurus produces best alignments, followed by Mpalign and M2M. However, peak performances of all three unsupervised aligners are close. Unsurprisingly, the *supervised* alignment models perform better than the unsupervised ones (Tables 5 and 6). Surprisingly, however, they do so with very little training data; fewer than 100 aligned string pairs suffice to outperform the unsupervised models under good calibrations. When there is sufficient training data, the supervised models perform splendidly, with a peak accuracy of 99.43% for the bigram alignment model that includes appropriate features (scoring lengths of aligned subsequences,

[6] We run both systems with parameters determined by some manual tuning, without trying to systematically optimize their individual performances, however.

[7] For all our below experiments involving the supervised aligners, we set S to a ('pessimistically' large) value of $\{(a,b)\,|\,1\leq a\leq 6, 1\leq b\leq 6\}$. Also, for the bigram models, we add special sequence boundary markers.

etc.). We also note that the bigram alignment model is almost consistently better than the unigram alignment model, with a surplus of about 1% point, depending on specific parametrizations.

We performed an analogous analysis for the German data. Results are quite similar except that unigram and bigram alignment model have indistinguishable performance on the German data, indicating (the known fact) that G2P is a more complex task in English, apparently not requiring bigram alignment models.

x	$\text{uni}_{0,0,1,1}$	$\text{uni}_{1,0,0,0}$	$\text{uni}_{1,1,1,1}$
100	70.34	58.13	**87.22**
500	81.94	84.64	**95.60**
1000	84.56	90.38	**96.17**
2000	85.41	93.47	**97.13**
5000	86.56	96.11	**97.72**
10000	86.13	97.07	**98.14**
20000	86.60	97.90	**98.34**

Table 5: Unigram model and its alignment accuracies in % for various training sizes.

x	$\text{bi}_{0,0,1,1}$	$\text{bi}_{1,0,0,0}$	$\text{bi}_{1,1,1,1}$
100	73.96	58.02	**87.28**
500	87.62	85.31	**95.26**
1000	91.87	90.73	**97.32**
2000	93.29	94.11	**97.96**
5000	95.58	97.01	**99.03**
10000	96.07	98.12	**99.17**
20000	97.21	98.73	**99.43**

Table 6: Bigram model and its alignment accuracies in % for various training sizes.

Error analysis Concerning errors that the unigram model commits and the bigram model does not, the majority of errors (roughly 80%) involve match-ups of *ed/d* and d. For example, the unigram model aligns as in

t	w	i	n	k	le	d
t	w	I	N	k	@l	d

while the gold-standard alignment is

t	w	i	n	k	l	ed
t	w	I	N	k	@l	d

While all match-ups in both alignments are plausible, the bigram model assigns here higher probability to the correct *ed/d* match-up in terminal position (consistently favored in the data set), which has a particular meaning there, namely, that of a suffix marker for past tense.[8,9] In the German data, there is a single instance where the unigram and bigram alignment model disagree, namely, in the alignment of *s-t-o-ff-f-l-a-sch-e*/S-t-O-f-f-l-&-S-@, which the unigram model falsely aligns as *s-t-o-f-ff-l-a-sch-e*/S-t-O-f-f-l-&-S-@; note that in the correct alignment *f* must follow *ff*, not vice versa, which depends on context information, e.g., that o/O signifies a short vowel which is followed by a double consonant, not a single consonant.

All remaining errors that the bigram alignment models commits are, for the best considered parametrization and training set size, typically due to match-up types not seen in the training data, and thus mostly concern foreign names or writings (e.g., *Bh-u-tt-o*/b-u-t-F, falsely aligned as *B-hu-tt-o*/b-u-t-F). A few other errors might be corrected when the feature coefficients $\alpha, \beta, \gamma, \delta$ were optimized on a development set rather than set manually. We find no indication that our G2P data, either for English or German, would further benefit from n-gram alignment models of order $n > 2$.

5.2 Alignment quality vs. overall G2P performance

Next, we estimate the relationship between alignment quality and overall G2P performance (transcription accuracy). To this end, for the English data, we use the 5 000 aligned string pairs from the previous experiment on alignment quality and feed them in — as training data — to either DirecTL+ or Phonetisaurus as outlined in Section 4. We then evaluate G2P performance — in terms of **word accuracy** (fraction of correctly transcribed strings) — on a distinct test set of size 10 000. Figure 1 shows a plot of overall G2P accuracy vs. *training set size for the aligner* (ranging over the x values in the last section); and a second plot that sketches G2P accuracy as a function of corresponding alignment accuracy. We first note that, as the supervised aligner receives more training

[8] Similar cases are, e.g., alignments of the type *f-ee-d-b-a-ck*/f-i-d-b-a-k, which the unigram model falsely aligns as *f-e-ed-b-a-ck*/f-i-d-b-a-k. Here, too, the unigram is unable to account for the almost exclusive terminal position of the *ed/d* match-up in the data.

[9] Other errors involve 'unusual/foreign' spelling/pronunciation pairs such as *Ph-oe-n-i-c-ia*/f-@-n-i-S-@ (wrongly aligned as *Ph-o-en-i-c-ia*/f-@-n-i-S-@ by the unigram model) or *m-a-d-e-m-oi-s-e-ll-e-'s*/m-a-d-@-m-w@-z-E-l-0-z (*m-a-d-e-m-o-i-s-e-ll-e-'s*/m-a-d-@-m-w-@-z-E-l-0-z), where the bigram alignment model has apparently gathered the more appropriate statistics.

x	Mpalign	$M2M_{2,2}$	$M2M_{3,3}$	$M2M_{6,6}$	$Phon_{2,2}$	$Phon_{3,3}$	$Phon_{6,6}$
1000	76.48	77.87	34.59	18.96	**78.27**	78.15	11.70
2000	78.05	78.03	34.45	18.87	**79.24**	77.07	12.43
5000	76.68	77.93	35.09	19.72	79.77	**80.47**	17.63
10000	78.86	77.97	35.03	21.35	79.60	**81.30**	23.57
20000	79.87	78.60	37.09	22.90	80.09	**83.37**	34.61

Table 4: Unsupervised aligners and their alignment accuracies in % for various data sizes as described in the text. Subscripts a, b denote restrictions on maximal lengths of subsequences allowed in match-ups (a/b corresponds to **x**/**y** subsequences).

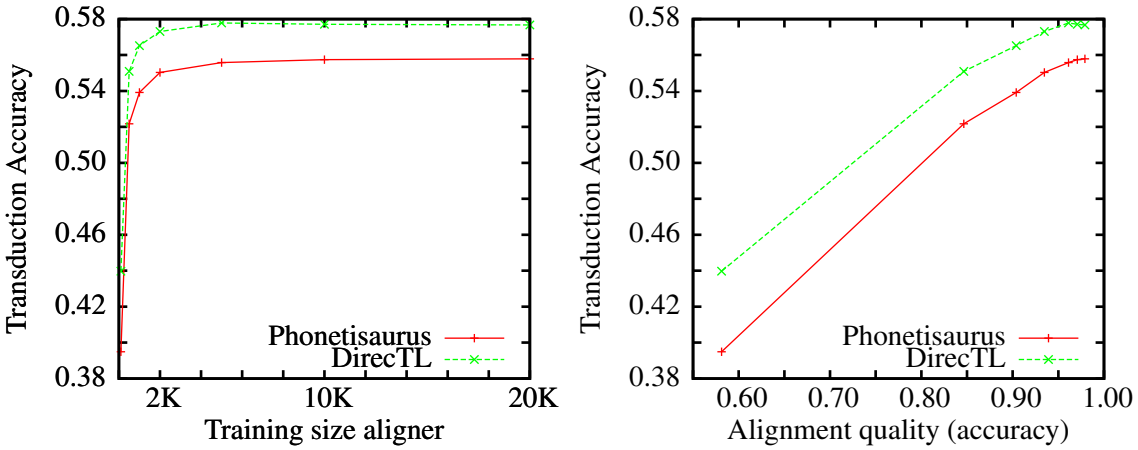


Figure 1: Left: Overall G2P accuracy as a function of training set size of supervised aligner $uni_{1,0,0,0}$. Right: G2P accuracy as a function of alignment quality (measured in accuracy).

data from which to align the 5 000 string pairs, the overall G2P accuracy of both DirecTL+ and Phonetisaurus increase substantially (and as a convex function of training set size). Apparently, the better alignments produced by more training data for the particular supervised aligner considered directly translate into better overall G2P accuracy. The other plot in the figure shows that, indeed, there seems to be a *linear* trend coupling alignment quality with overall G2P performance. Table 7 pairs G2P accuracy with alignment accuracy of selected systems, all run in the $x = 20\,000$ setting. While, in the table, better alignments do not necessarily imply better overall G2P performance, the two best alignments also lead to the two best overall G2P performances (although, in this case, the second best alignment is paired with the best overall G2P performance); conversely, the worst alignment quality is coupled with the worst overall G2P performance.

Overall, we ran 249 experiments (including the German data) in which we trained DirecTL+ or Phonetisaurus with alignments of specific qualities obtained from particularly parametrized aligners. In each of these cases, we obtained an alignment quality score and a subsequent overall G2P system performance. The English part of this data is sketched in Figure 2. This figure seems to corroborate the linear relationship (apparently present in Figure 1) between alignment quality and overall G2P system accuracy, particularly, when alignment quality is measured in the more fine-grained metric of edit distance. To formally test

	Alignment acc.	Phon.	DirecTL+
Mpalign	79.87	55.48	57.54
$M2M_{3,3}$	37.09	49.25	53.71
$Phon_{3,3}$	83.37	54.05	56.11
$uni_{0,0,1,1}$	86.60	53.19	55.49
$uni_{1,1,1,1}$	98.34	**55.72**	**57.78**
$bi_{1,1,1,1}$	**99.43**	55.71	57.71

Table 7: Systems, alignment accuracies of corresponding produced alignments and transcription accuracy of Phonetisaurus and DirecTL+ when trained with the respective alignments.

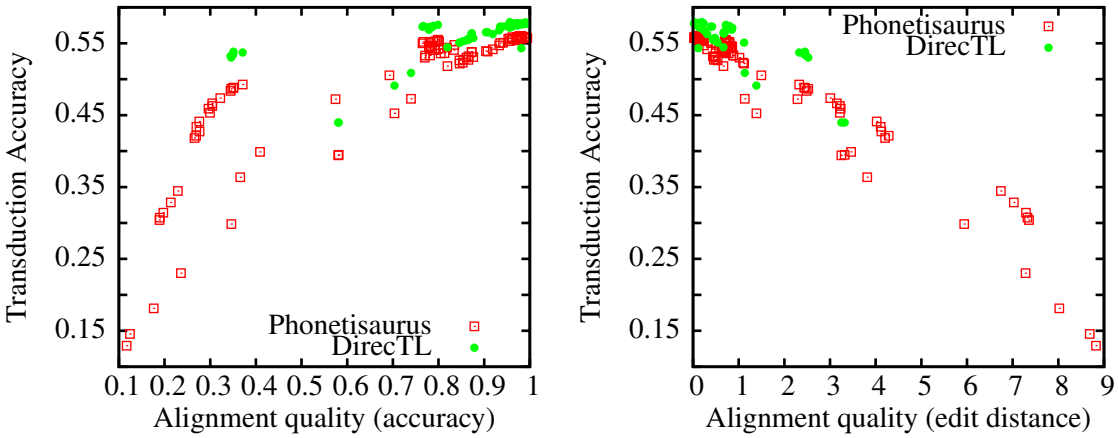


Figure 2: Overall G2P accuracy vs. alignment quality. Left: Alignment quality measured in accuracy. Right: Alignment quality measured in edit distance. English data only.

this, we regress overall G2P system performance (measured in word accuracy) on edit distance and other variables.[10] This yielded the coefficients as given in Table 8; in each case, the goodness-of-fit of the linear model was quite large, with R^2 values above 90% for the English data and about 84% for the German data. Also, the coefficients on alignment quality were highly significantly different from zero. The table shows that the coefficients are on the order of about -3.80% to -4.70%, meaning that, all else being equal, increasing alignment quality by 1 edit distance to the gold-standard alignment increases overall G2P by about 3.80 to 4.70%.

	DirecTL+	Phonetisaurus
English	-3.80^{***}	-4.14^{***}
German	-	-4.68^{***}

Table 8: Coefficients on edit distance in the regression of G2P accuracy on edit distance and further variables. For German, DirecTL+ is omitted due to its long run times.

So far, we have estimated the effects of alignment quality on overall G2P system performance for a **fixed** size of training data, namely, 5 000 aligned string pairs. To see whether this relationship changes when we vary the amount of training data, we run several more experiments. In these, we align training sets of sizes 100, 500, 1 000, 2 000, 10 000, 20 000, 40 000 and 60 000 via our several alignment systems. Then we feed the aligned data to the Phonetisaurus system (we omit DirecTL+ here because of its long run times) and compute overall G2P accuracy on a disjoint test set of size 28 000 approximately. This time, we only use the unsupervised aligners and the gold-standard alignments directly, omitting results for our various supervised aligners. Note, however, that these aligners could, in principle, imitate the gold-standard alignments with a very high degree of precision, as previously seen. Table 9

	$M2M_{3,3}$	Mpalign	$Phon_{3,3}$	Gold
100	5.38	6.43	0.19	**9.60**
500	16.80	22.43	5.08	**23.93**
1K	25.79	31.46	18.70	**33.37**
2K	35.31	42.01	37.74	**43.64**
10K	58.44	64.05	63.06	**64.60**
20K	67.70	71.70	71.51	**72.21**
40K	74.69	78.45	78.13	**78.65**
60K	78.00	81.07	80.92	**81.17**

Table 9: Overall G2P accuracy in % as a function training size of aligned data and alignment system.

shows that training G2P systems from the human gold standard alignments in each case yields better overall G2P transcriptions than training them from either of the three unsupervised alignments considered here. However, we note that the surplus over the unsupervised alignments decreases as training set size increases. This may be due to the fact that the unsupervised aligners themselves create better alignments once they are boot-

[10] These include binary dummy variables for the specific systems as well as alignment consistency and its square — measured in conditional entropy $H(Y|X)$ (Pervouchine et al., 2009) — in the regression.

strapped from larger data sets (cf. Table 4). Additionally, the effect of alignment quality on overall G2P system performance may simply vanish as training set sizes become large enough because the translation modules can better accomodate 'noisy' data as long as its size is sufficiently large. Figure

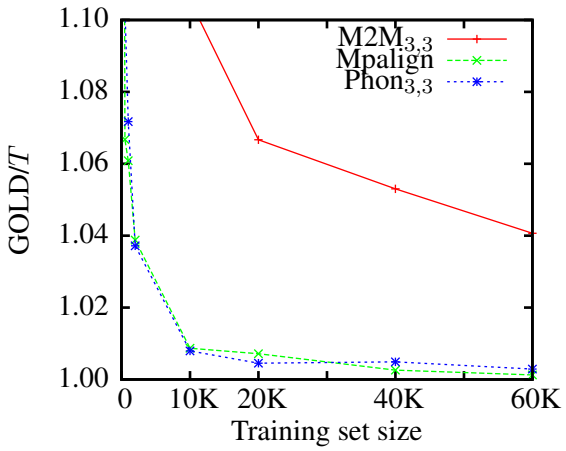


Figure 3: Ratio of transcription accuracy when using gold standard alignments (GOLD) and when using alignments generated by T = M2M$_{3,3}$, Mpalign, and Phon$_{3,3}$, respectively, as a function of size of aligned training set.

3 sketches the decreasing influence of alignment system on overall G2P system performance as size of the aligned data increases.

6 Conclusion

We have investigated the need for *bigram* alignment models and the benefit of *supervised* alignment techniques in G2P. We have also quantitatively estimated the relationship between alignment quality and overall G2P system performance. We have found that, in English, bigram alignment models do perform better than unigram alignment models on the G2P task (we find almost no differences between unigram and bigram models for the German sample of G2P data we considered). Moreover, we have found that supervised alignment techniques may perform considerably better than their unsupervised brethren and that few manually aligned training pairs suffice for them to do so. Finally, we have estimated a highly significant impact of alignment quality on overall G2P transcription performance and that this relationship is linear in nature. At a particular training size, a linear regression model has estimated that improving alignment quality by 1 edit distance toward the gold standard alignments leads to an 3.80-4.70% increase in G2P transcription accuracy. However, we have also found that the importance of good alignments on G2P accuracy appears to dimish as data set size increases, possibly because the translation modules can accomodate more 'noisy' data in this scenario.

As a 'policy' implication, we recommend the use of supervised alignment techniques particularly when the size of the G2P corpus is small or when high quality alignments, as an end in themselves, are required. In this case, constructing a few dozen or few hundred alignments in an unsupervised manner and correcting them by hand (to serve as an input for a supervised technique) may be highly beneficial.

In future work, it may be worthwhile to study the impact of alignment techniques on overall system performance in other string transduction problems such as transliteration, lemmatization, and spelling error correction.

Our supervised uni- and bigram aligners are available via `https://github.com/SteffenEger/`.

Acknowledgments

I thank three anonymous reviewers and Tim vor der Brück for valuable suggestions.

References

H. Baayen, R. Piepenbrock, and L. Gulikers. 1995. The CELEX2 lexical database. ldc96l14.

Maximilian Bisani and Hermann Ney. 2008. Joint-sequence models for grapheme-to-phoneme conversion. *Speech Communication*, 50(5):434–451.

Eric Brill and Robert C. Moore. 2000. An improved error model for noisy channel spelling correction. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, ACL '00, pages 286–293, Stroudsburg, PA, USA. Association for Computational Linguistics.

Ryan Cotterell, Nanyun Peng, and Jason Eisner. 2014. Stochastic contextual edit distance and probabilistic FSTs. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 625–630, Baltimore, June.

Sabine Deligne, Franois Yvon, and Frédéric Bimbot. 1995. Variable-length sequence matching for phonetic transcription using joint multigrams. In *EUROSPEECH*. ISCA.

Markus Dreyer, Jason Smith, and Jason Eisner. 2008. Latent-variable modeling of string transductions with finite-state methods. In *EMNLP*, pages 1080–1089. ACL.

Steffen Eger. 2012. S-restricted monotone alignments: Algorithm, search space, and applications. In *COLING'12*, pages 781–798.

Steffen Eger. 2013. Sequence alignment with arbitrary steps and further generalizations, with applications to alignments in linguistics. *Inf. Sci.*, 237:287–304.

Steffen Eger. 2015a. Improving g2p from wiktionary and other (web) resources. In *Proceedings of Interspeech*. accepted.

Steffen Eger. 2015b. Multiple many-to-many sequence alignment for combining string-valued variables: A G2P experiment. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 909–919.

Kuzman Ganchev, Joo Graa, and Ben Taskar. 2008. Better alignments = better translations? In Kathleen McKeown, Johanna D. Moore, Simone Teufel, James Allan, and Sadaoki Furui, editors, *ACL*, pages 986–993. The Association for Computational Linguistics.

Sharon Goldwater, Thomas L. Griffiths, and Mark Johnson. 2006. Contextual dependencies in unsupervised word segmentation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 673–680, Sydney, Australia, July. Association for Computational Linguistics.

Sittichai Jiampojamarn and Grzegorz Kondrak. 2010. Letter-phoneme alignment: An exploration. In Jan Hajic, Sandra Carberry, and Stephen Clark, editors, *ACL*, pages 780–788. The Association for Computational Linguistics.

Sittichai Jiampojamarn, Grzegorz Kondrak, and Tarek Sherif. 2007. Applying many-to-many alignments and hidden markov models to letter-to-phoneme conversion. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 372–379, Rochester, New York, April. Association for Computational Linguistics.

Sittichai Jiampojamarn, Colin Cherry, and Grzegorz Kondrak. 2008. Joint processing and discriminative training for letter-to-phoneme conversion. In *Proceedings of ACL-08: HLT*, pages 905–913, Columbus, Ohio, June. Association for Computational Linguistics.

Sittichai Jiampojamarn, Colin Cherry, and Grzegorz Kondrak. 2010. Integrating joint n-gram features into a discriminative training framework. In *HLT-NAACL*, pages 697–700. The Association for Computational Linguistics.

Grzegorz Kondrak. 2000. A new algorithm for the alignment of phonetic sequences. In *Proceedings of the 1st North American Chapter of the Association for Computational Linguistics Conference*, NAACL 2000, pages 288–295, Stroudsburg, PA, USA. Association for Computational Linguistics.

Keigo Kubo, Hiromichi Kawanami, Hiroshi Saruwatari, and Kiyohiro Shikano. 2011. Unconstrained many-to-many alignment for automatic pronunciation annotation. In *Proceedings of Asia-Pacific Signal and Information Processing Association Annual Summit and Conference 2011 (APSIPA2011)*, October.

VI Levenshtein. 1966. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady*, 10:707.

Yang Liu and Maosong Sun. 2015. Contrastive unsupervised word alignment with non-local features. In *Proceedings of AAAI 2015*.

Yang Liu, Qun Liu, and Shouxun Lin. 2010. Discriminative word alignment by linear modeling. *Computational Linguistics*, pages 303–339.

Saul B. Needleman and Christian D. Wunsch. 1970. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48(3):443–453, March.

Josef R. Novak, Nobuaki Minematsu, and Keikichi Hirose. 2012. WFST-based grapheme-to-phoneme conversion: Open source tools for alignment, model-building and decoding. In *Proceedings of the 10th International Workshop on Finite State Methods and Natural Language Processing*, pages 45–49, Donostia–San Sebastin, July. Association for Computational Linguistics.

Vladimir Pervouchine, Haizhou Li, and Bo Lin. 2009. Transliteration alignment. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1 - Volume 1*, ACL '09, pages 136–144, Stroudsburg, PA, USA. Association for Computational Linguistics.

Kanishka Rao, Fuchun Peng, Hasim Sak, and Franoise Beaufays. 2015. Grapheme-to-phoneme conversion using long short-term memory recurrent neural networks. In *ICASSP 2015*.

Korin Richmond, Robert A. J. Clark, and Susan Fitt. 2009. Robust LTS rules with the Combilex speech technology lexicon. In *INTERSPEECH*, pages 1295–1298. ISCA.

Eric Sven Ristad and Peter N. Yianilos. 1998. Learning string-edit distance. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20(5):522–532.

Tarek Sherif and Grzegorz Kondrak. 2007. Substring-based transliteration. In John A. Carroll, Antal van den Bosch, and Annie Zaenen, editors, *ACL*. The Association for Computational Linguistics.

Esko Ukkonen. 1985. Algorithms for approximate string matching. *Information and Control*, 64:100–118.

Jean Véronis. 1988. Computerized correction of phonographic errors. *Computers and the Humanities*, 22(1):43–56.

Lei Yao and Grzegorz Kondrak. 2015. Joint generation of transliterations from multiple representations. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 943–952, Denver, Colorado, May–June. Association for Computational Linguistics.

Keyboard Logs as Natural Annotations for Word Segmentation

Fumihiko Takahashi*
Yahoo Japan Corporation
Midtown Tower, 9-7-1 Akasaka, Minato-ku,
Tokyo, Japan
ftakahas@yahoo-corp.jp

Shinsuke Mori
ACCMS, Kyoto University
Yoshida Honmachi, Sakyo-ku,
Kyoto, Japan
forest@i.kyoto-u.ac.jp

Abstract

In this paper we propose a framework to improve word segmentation accuracy using input method logs. An input method is software used to type sentences in languages which have far more characters than the number of keys on a keyboard. The main contributions of this paper are: 1) an input method server that proposes word candidates which are not included in the vocabulary, 2) a publicly usable input method that logs user behavior (like typing and selection of word candidates), and 3) a method for improving word segmentation by using these logs. We conducted word segmentation experiments on tweets from Twitter, and showed that our method improves accuracy in this domain. Our method itself is domain-independent and only needs logs from the target domain.

1 Introduction

The first step of almost all natural language processing (NLP) for languages with ambiguous word boundaries (such as Japanese and Chinese) is solving the problem of word identification ambiguity. This task is called word segmentation (WS) and the accuracy of state-of-the-art methods based on machine learning techniques is more than 98% for Japanese and 95% for Chinese (Neubig et al., 2011; Yang and Vozila, 2014). Compared to languages like English with clear word boundaries, this ambiguity poses an additional problem for NLP tasks in these languages. To make matters worse, the domains of the available training data often differ from domains where there is a high demand for NLP, which causes a severe degradation in WS performance. Examples include machine translation of patents, text mining of medical texts, and marketing on the micro-blog site Twitter[1]. Some papers have reported low accuracy on WS or the joint task of WS and part-of-speech (POS) tagging of Japanese or Chinese in these domains (Mori and Neubig, 2014; Kaji and Kitsuregawa, 2014; Liu et al., 2014)

To cope with this problem, we propose a way to collect information from people as they type Japanese or Chinese on computers. These languages use far more characters than the number of keys on a keyboard, so users use software called an input method (IM) to type text in these languages. Unlike written texts in these languages, which lack word boundary information, text entered with an IM can provide word boundary information that can used by NLP systems. As we show in this paper, logs collected from IMs are a valuable source of word boundary information.

An IM consists of a client (front-end) and a server (back-end). The client receives a key sequence typed by the user and sends a phoneme sequence (*kana* in Japanese or *pinyin* in Chinese) or some predefined commands to the server. The server converts the phoneme sequence into normal written text as a word sequence or proposes word candidates for the phoneme sequence in the region specified by the user. We noticed that the actions performed by people using the IM (such as typing and selecting word candidates) provide information about word boundaries, including context information.

In this paper, we first describe an IM for Japanese which allows us to collect this information. We then propose an automatic word segmenter that uses IM logs as a language resource to improve its performance. Finally, we report experimental results showing that our method increases the accuracy of a word segmenter on Twitter texts by using logs collected from a browser add-on ver-

*This work was done when the first author was at Kyoto University.

[1]https://twitter.com/ (accessed in 2015 May).

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1186–1196,
Lisbon, Portugal, 17-21 September 2015.

sion of our IM.

The three main contributions of this paper are:

- an IM server that proposes word candidates which are not included in the vocabulary (Section 3),
- a publicly usable IM that logs user behavior (such as typing and selection of word candidates) (Section 4),
- a method for improving word segmentation by using these logs (Section 5).

To the best of our knowledge, this is the first paper proposing a method for using IM logs to successfully improve WS.

2 Related Work

The main focus of this paper is WS. Corpus-based, or empirical, methods were proposed in the early 90's (Nagata, 1994). Then (Mori and Kurata, 2005) extended it by lexicalizing the states like many researches in that era, grouping the word-POS pairs into clusters inspired by the class-based n-gram model (Brown et al., 1992), and making the history length variable like a POS tagger in English (Ron et al., 1996). In parallel, there were attempts at solving Chinese WS in a similar way (Sproat and Chang, 1996). WS or the joint task of WS and POS tagging can be seen as a sequence labeling problem. So conditional random fields (CRFs) (Peng et al., 2004; Lafferty et al., 2001) have been applied to this task and showed better performance than POS-based Markov models (Kudo et al., 2004). The training time of sequence-based methods tends to be long, especially when we use partially annotated data. Thus a simple method based on pointwise classification has been shown to be as accurate as sequence-based methods and fast enough to make active learning practically possible (Neubig et al., 2011). Since the pointwise method decides whether there is a word boundary or not between two characters without referring to other word boundary decisions in the same sentence, it is straightforward to train the model from partially annotated sentences. We adopt this WS system for our experiments.

Along with the evolution of models, the NLP community has become increasingly aware of the importance of language resources (Neubig and Mori, 2010; Mori and Neubig, 2014). So Mori and Oda (2009) proposed to incorpolate dictionaries for human into a WS system with a different word definition. CRFs were also extended to enable training from partially annotated sentences (Tsuboi et al., 2008). When using partially annotated sentences for WS training data, word boundary information exists only between some character pairs and is absent for others. This extension was adopted in Chinese WS to make use of so-called natural annotations (Yang and Vozila, 2014; Jiang et al., 2013). In that work, tags in hyper-texts were regarded as annotations and used to improve WS performance. The IM logs used in this paper are also classified as natural annotations, but contain much more noise. In addition, we need an IM that is specifically designed to collect logs as natural annotations.

Server design is the most important factor in capturing information from IM logs. The most popular IM servers are based on statistical language modeling (Mori et al., 1999; Chen and Lee, 2000; Maeta and Mori, 2012). Their parameters are trained from manually segmented sentences whose words are annotated with phoneme sequences, and from sentences automatically annotated with NLP tools which are also based on machine learning models trained on the annotated sentences. Thus normal IM servers are not capable of presenting out-of-vocabulary (OOV) words (which provide large amounts of information on word boundaries) as conversion candidates. To make our IM server capable of presenting OOV words, we extend a statistical IM server based on (Mori et al., 2006), and ensure that it is computationally efficient enough for practical use by the public.

The target domain in our experiments is Twitter, a site where users post short messages called tweets. Since tweets are an immediate and powerful reflection of public attitudes and social trends, there have been numerous attempts at extracting information from them. Examples include information analysis of disasters (Sakai et al., 2010), estimation of depressive tendencies (Tsugawa et al., 2013), speech diarization (Higashinaka et al., 2011), and many others. These works require preprocessing of tweets with NLP tools, and WS is the first step. So it is clear that there is strong demand for improving WS accuracy. Another reason why we have chosen Twitter for the test domain is that the tweets typed using our server are open and

we can avoid privacy problems. Our method does not utilize any other characteristics of tweets. So it also works in other domains such as blogs.

3 Input Method Suggesting OOV Words

In this section we propose a practical statistical IM server that suggests OOV word candidates in addition to words in its vocabulary.

3.1 Statistical Input Method

An input method (IM) is software which converts a phoneme sequence into a word sequence. This is useful for languages which contain far more characters than keys on a keyboard. Since there are some ambiguities in conversion, a conversion engine based on a word n-gram model has been proposed (Chen and Lee, 2000). Today, almost all IM engines are based on statistical methods.

For the LM unit, instead of words we propose to adopt word-pronunciation pairs $u = \langle \boldsymbol{y}, w \rangle$. Thus given a phoneme sequence $\boldsymbol{y}_1^l = y_1 y_2 \cdots y_l$ as the input, the goal of our IM engine is to output a word sequence $\hat{\boldsymbol{w}}_1^m$ that maximizes the probability $P(\boldsymbol{w}, \boldsymbol{y}_1^l)$ as follows:

$$\begin{aligned} \hat{\boldsymbol{w}}_1^m &= \operatorname*{\mathbf{argmax}}_{\boldsymbol{w}} P(\boldsymbol{w}, \boldsymbol{y}_1^l), \\ P(\boldsymbol{w}, \boldsymbol{y}_1^l) &= \prod_{i=1}^{m+1} P(u_i | \boldsymbol{u}_{i-n+1}^{i-1}), \end{aligned}$$

where the concatenation of $\boldsymbol{y}_i$ in each u_i is equal to the input: $\boldsymbol{y}_1^l = \boldsymbol{y}_1 \boldsymbol{y}_2 \cdots \boldsymbol{y}_m$. In addition u_j ($j \leq 0$) are special symbols introduced to simplify the notation and u_{m+1} is a special symbol indicating a sentence boundary.

As in existing statistical IM engines, parameters are estimated from a corpus whose sentences are segmented into words annotated with their pronunciations as follows:

$$P(u_i | \boldsymbol{u}_{i-n+1}^{i-1}) = \frac{F(\boldsymbol{u}_{i-n+1}^{i})}{F(\boldsymbol{u}_{i-n+1}^{i-1})}, \qquad (1)$$

where $F(\cdot)$ denotes the frequency of a pair sequence in the corpus. In contrast to IM engines based on a word n-gram model, ours does not require an additional model describing relationships between words and pronunciations, and thus it is much simpler and more practical.

Existing statistical IM engines only need an accurate automatic word segmenter to estimate the parameters of the word n-gram model. As the equation above shows, our pair-based engine also needs an accurate way of automatically estimating pronunciation (phoneme sequences). However, recently an automatic pronunciation estimator (Mori and Neubig, 2011) that delivers as accurate as state-of-the-art word segmenters has been proposed. As we explain in Section 6, in our experiments both our IM engine and existing ones delivered accuracy of 91%.

3.2 Enumerating Substrings as Candidate Words

Essentially, the IM engine which we have explained above does not have the ability to enumerate words which are unknown to the word segmenter and the pronunciation estimator used to build the training data. The aim of our research is to gather language information from user behavior as they use an IM. So we extend the basic IM engine to enumerate all the substrings in a corpus with all possible pronunciations. For that purpose, we adopt the notion of a stochastically segmented corpus (SSC) (Mori and Takuma, 2004) and extend it to the pronunciation annotation to words.

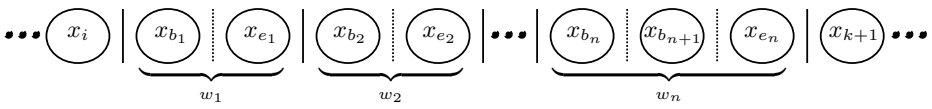
3.2.1 Stochastically Segmented Corpora

An SSC is defined as a combination of a raw corpus C_r (hereafter referred to as the character sequence $\boldsymbol{x}_1^{n_r}$) and word boundary probabilities of the form P_i, which is the probability that a word boundary exists between two characters x_i and x_{i+1}. These probabilities are estimated by a model based on logistic regression (LR)(Fan et al., 2008) trained on a manually segmented corpus referring to the same features as those used in (Neubig et al., 2011). Since there are word boundaries before the first character and after the last character of the corpus, $P_0 = P_{n_r} = 1$. Then word n-gram frequencies on an SSC are calculated as follows:

Word 0-gram frequency: This is defined as the expected number of words in the SSC:

$$f(\cdot) = 1 + \sum_{i=1}^{n_r - 1} P_i.$$

Word n-gram frequency ($n \geq 1$): Consider the situation in which a word sequence $\boldsymbol{w}_1^n$ occurs in the SSC as a subsequence beginning at the $(i+1)$-th character and ending at the k-th character and each word w_m in the word sequence is equal to the character sequence beginning at the



$$f_r(w_1^n) = P_i(1-P_{b_1})P_{e_1}(1-P_{b_2})P_{e_2}\cdots(1-P_{b_n})(1-P_{b_{n+1}})P_{e_n}$$

Figure 1: Word n-gram frequency in a stochastically segmented corpus.

b_m-th character and ending at the e_m-th character ($\boldsymbol{x}_{b_m}^{e_m} = w_m,\ 1 \leq \forall m \leq n;\ e_m + 1 = b_{m+1},\ 1 \leq \forall m \leq n-1;\ b_1 = i+1;\ e_n = k$) (See Figure 1 for an example). The word n-gram frequency of a word sequence $f_r(\boldsymbol{w}_1^n)$ in the SSC is defined by the summation of the stochastic frequency at each occurrence of the character sequence of the word sequence $\boldsymbol{w}_1^n$ over all of the occurrences in the SSC:

$$f_r(\boldsymbol{w}_1^n) = \sum_{(i,\boldsymbol{e}_1^n)\in O_n} P_i \left[\prod_{m=1}^{n} \left\{ \prod_{j=b_m}^{e_m-1} (1-P_j) \right\} P_{e_m} \right],$$

where $\boldsymbol{e}_1^n = (e_1, e_2, \cdots, e_n)$ and $O_n = \{(i, e_1^n) | \boldsymbol{x}_{b_m}^{e_m} = w_m, 1 \leq m \leq n\}$.

We calculate word n-gram probabilities by dividing word n-gram frequencies by word $(n-1)$-gram frequencies. For a detailed explanation and a mathematical proof of this method, please refer to (Mori and Takuma, 2004).

3.2.2 Pseudo-Stochastically Segmented Corpora

The computational costs (in terms of both time and space) for calculating an n-gram model from an SSC are very high[2], so it is not a practical technique for implementing an IM engine. In order to reduce the computational costs we approximate an SSC using a deterministically tagged corpus, which is called a pseudo-stochastically segmented corpus (pSSC) (Kameko et al., 2015). The following is the method for producing a pSSC from an SSC.

- For i = 1 to $n_r - 1$
  1. output a character x_i,
  2. generate a random number $0 \leq p < 1$,
  3. output a word boundary if $p < P_i$ or output nothing otherwise.

Now we have a corpus in the same format as a standard segmented corpus with variable (non-constant) segmentation.

[2]This is because an SSC has many words and word fragments. Additionally, word n-gram frequencies must be calculated using floating point numbers instead of integers.

3.2.3 Pseudo-Stochastically Tagged Corpora

We can annotate a word with its all possible pronunciations and their probabilities, as is done in an SSC. We call a corpus containing sequences of words ($w_1 w_2 \cdots w_i \cdots$) annotated with a sequence of pairs of a pronunciation and its probability ($\langle \boldsymbol{y}_{i,1}, p_{i,1} \rangle, \langle \boldsymbol{y}_{i,2}, p_{i,2} \rangle, \cdots$, where $\sum_j p_{i,j} = 1$, for $\forall i$) a stochastically tagged corpus (STC)[3]. We can estimate these probabilities using an LR model built from sentences annotated with pronunciations (Mori and Neubig, 2011).

Similar to pSSC we then produce a pseudo-stochastically tagged sentence (pSTC) from an STC as follows:

- For each w_i in the sentence
  1. generate a random number $0 \leq p < 1$,
  2. annotate w_i with its j-th phoneme sequence $\boldsymbol{y}_{i,j}$, where $\sum_1^{j-1} p_{i,j} \leq p < \sum_1^{j} p_{i,j}$

Now we have a corpus in the same format as a standard corpus annotated with variable pronunciation.

By estimating the parameters in Equation (1) from a pSTC derived from a pSSC, our IM engine can also suggest OOV word candidates with various possible segmentation and pronunciations without incurring high computational costs.

3.2.4 Suggestion of OOV Words

Here we give an intuitive explanation why our IM engine can suggest OOV words for a certain phoneme sequence. Let us take an OOV word example: "横アリ/yo-ko-a-ri," an abbreviation of "横浜アリーナ" (Yokohama city arena). A WS system tends to segment it into "横" (side) and "アリ" (ant) because they are frequent nouns. In a pSSC, however, some occurrences of the string "横アリ" are remain concatenated as the correct word. For pronunciation, the first character has two possible pronunciations "yo-ko" and "o-u."

[3]Because the existence or non-existence of a word boundary information can also be expressed as a tag, a stochastically tagged corpus includes stochastic segmentation.

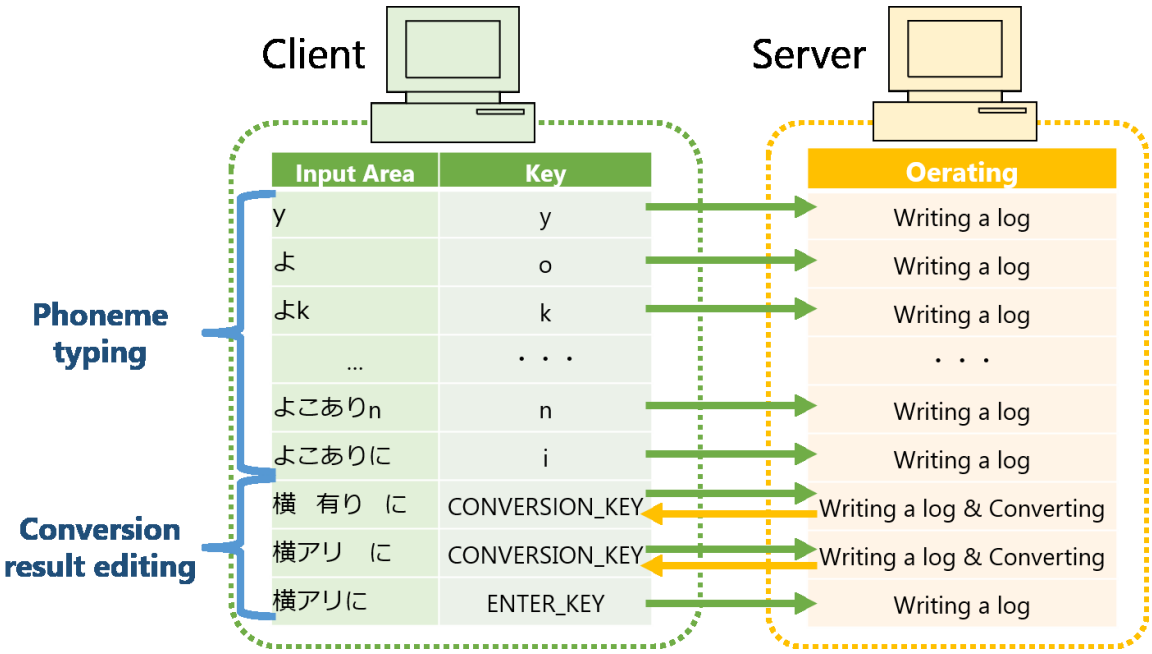


Figure 2: Input method for collecting logs.

So deterministic pronunciation estimation of this new word has the risk of outputting the erroneous result "o-u-a-ri." This prevents our engine from presenting "横アリ" as a conversion candidate for the input "yo-ko-a-ri." The pSTC, however, contains two possible pronunciations for this word and allows our engine to present the OOV word "横アリ" for the input "yo-ko-a-ri."

Thus when the user of our IM engine types "yo-ko-a-ri-ni-i-ku" and selects "横アリ に (to) 行く (go)," the engine can learn an OOV word "横アリ/yo-ko-a-ri" with context "に/ni 行く/i-ku".

4 Input Method Logs

In this section we first propose an IM which allows us to collect user logs. We then examine the characteristics of these logs and some difficulties in using them as language resources.

4.1 Collecting Logs from an Input Method

As Figure 2 shows, the client of our IM, running on the user's PC, is used to input characters and to modify conversion results. The server logs both input from the client and the results of conversions performed in response to requests from the client.

Our IM has two phases: phoneme typing and conversion result editing. In each phase, the client sends the typed keys to the server with a timestamp and its IP address.

Phoneme typing: First the user inputs ASCII characters for a phoneme sequence. If the phoneme sequence itself is what the user wants to write, the user may not go to the next phase. The server records the keys typed to enter the phoneme sequence, cursor movements, and the phoneme sequence if the user selects it as-is.

Conversion result editing: Then the user presses a space key to make the IM engine convert the phoneme sequence to the most likely word sequence based on Equation (1). Sometimes the user changes some word boundaries, makes the IM engine enumerate candidate words covering the region, and selects the intended one from the list of candidates. The server records a space key and the final word sequence.

4.2 Characteristics of Input Method Logs

Table 1 shows an example of interesting log messages from the same IP address[4]. In many cases, users type sentence fragments but not a complete sentence. So in the example there are six fragments within a short period indicated by the timestamps. If the user selects the phoneme sequence as-is without going to the conversion result editing phase, we can expect that there are word boundaries on both sides of the phoneme sequence. In-

[4]In reality, logs from different IPs are stored in the order that they were received.

Table 1: Input method logs of a tweet '横アリに比べると安めかと' (It is cheap compared with Yokohama arena).

Timestamp	Phoneme sequence	Edit result	Note
18:37:11.21	よこありに/yo-ko-a-ri-ni	横アリ/yo-ko-a-ri に/ni	(with Yokohama arena)
18:37:12.60	くらっべる/ku-ra-b-be-ru	くらっ/ku-ra-b ベル/be-ru	Mistyping
18:37:14.94	くらべる/ku-ra-be-ru	比べ/ku-ra-be る/ru	Revised input (compare)
18:37:15.32	と/to	N/A	(*inflectional ending*)
18:37:19.82	ものの/mo-no-no	N/A	Discarded in the twitter
18:37:22.42	やすめかと/ya-su-me-ka-to	安め/ya-su-me か/ka と/to	(cheap)

side the phoneme sequence, however, there is no information. If the user goes to the conversion result editing phase, we can expect that the final word sequence has correct word boundary information.

There are two main problems that make it difficult to directly use IM logs as a training corpus for word segmentation. The first problem is fragmentation. IM users send the phoneme sequences for sentence fragments to the engine to avoid editing long conversion results that require many cursor movements. Thus the phoneme sequence and the final word sequence tend to be sentence fragments (as we noted above) and as a result they lose context information. The second problem is noise. Word boundary information is unreliable even when it is present because of mistakenly selected conversions or words entered separately. From these observations, the IM logs are treated as partially segmented sentence fragments that include noise.

5 Word Segmentation Using Input Method Logs

In this section we first explain various ways to generate language resources for a word segmenter from IM logs. We then describe an automatic word segmenter which utilizes these resources. In the examples below we use the three-valued notation (Mori and Oda, 2009) to denote partial segmentation as follows:

- | : there is a word boundary,
- – : there is not a word boundary,
- ␣ : there is no information.

5.1 Input Method Logs as Language Resources

The phoneme sequences and edit results in the final selection themselves are considered to be partially segmented sentences. We call the corpus generated directly from the logs "**Log-as-is**." Examples in Table 1 are converted as following.

Example of Log-as-is (12 annotations)

横–ア–リ | に
く–ら–っ | べ–ル
比–べ | る
と
も␣の␣の
安–め | か | と

Here the number of annotations is the sum of "–" and "|". In this example, one entry corresponds to one entry of the training data for the word segmenter. As you can easily imagine, Log-as-is may contain mistaken results (noise) and short entries (fragmentation). Both are harmful for a word segmenter.

To cope with the fragmentation problem, we propose to connect some logs based on their timestamps. If the difference between the timestamps of two sequential logs is short, both logs are probably from the same sentence. So we connect two sequential logs if the time difference between the last key of the first log and the first key of the second log is smaller than a certain threshold s. In the experiment we set $s = 500[\mathtt{ms}]$ based on observations of our behavior[5]. This method is referred to as "**Log-chunk**." Using this method, we obtain the following from the examples in Table 1.

Example of Log-chunk (15 annotations)

横–ア–リ | に | く–ら–っ | べ–ル
比–べ | る | と | も␣の␣の
安–め | か | と

We see that Log-chunk contains more context information than Log-as-is.

For preventing the noise problem, we propose to filter out logs with a small number of conversions. We expect that an edited sentence will have many OOV words and not much noise. Therefore we use logs which were converted more than n_c times. In the experiment we set $n_c = 2$ based on

[5]The results were stable for s in preliminary experiments.

Table 2: Corpus specifications.

	#sent.	#words	#char.
Training			
BCCWJ	56,753	1,324,951	1,911,660
Newspaper	8,164	240,097	361,843
Conversation	11,700	147,809	197,941
Test			
BCCWJ-test	6,025	148,929	212,261
TWI-test	2,976	37,010	58,316

observations of our behavior[6]. This method is referred to as "**Log-mconv**." Using this method, the examples in Table 1 becomes the following.

Example of Log-mconv (3 annotations)
横–ア–リ | に

As this example shows, Log-mconv contains short entries (fragmentation) like Log-as-is. However, we expect that the annotated tweets do not include mistaken boundaries or conversions that were discarded.

Obviously we can combine Log-chunk and Log-mconv to avoid both the fragmentation and noise problems. This combination is referred to as "**Log-chunk-mconv**."

5.2 Training a Word Segmenter on Logs

The IM logs give us partially segmented sentence fragments, so we need a word segmenter capable of learning from them. We can use a word segmenter based on a sequence classifier (Tsuboi et al., 2008; Yang and Vozila, 2014; Jiang et al., 2013) or one based on a pointwise classifier (Neubig et al., 2011). Although both types are viable, we adopt the latter in the experiments because it requires much less training time while delivering comparable accuracy.

Here is a brief explanation of the word segmenter based on the pointwise method. For more detail the reader may refer to (Neubig et al., 2011). The input is an unsegmented character sequence $\boldsymbol{x} = x_1 x_2 \cdots x_k$. The word segmenter decides if there is a word boundary $t_i = 1$ or not $t_i = 0$ by using support vector machines (SVMs) (Fan et al., 2008)[7]. The features are character n-grams and character type n-grams ($n = 1, 2, 3$) around the decision points in a window with a width of 6 characters. Additional features are triggered if character n-grams in the window match with character sequences in the dictionary. This approach is called pointwise because the word boundary decision is made without referring to the other decisions on the points $j \neq i$. As you can see from the explanation given above, we can also use partially segmented sentences from IM logs for training in the standard way.

Table 3: Language resources derived from logs.

	#sentence fragments	#annotations
Log-as-is	32,119	39,708
Log-chunk	8,685	63,144
Log-mconv	4,610	10,852
Log-chunk-mconv	1,218	14,242

6 Evaluation

As an evaluation of our methods, we measured the accuracy of WS without using logs (the baseline) and using logs converted by several methods. There are two test corpora: one is the general domain corpus from which we built the baseline WS, and the other is the same domain that the IM logs were collected from, Twitter.

6.1 Corpora

The annotated corpus we used to build the baseline word segmenter is the manually annotated part (core data) of the Balanced Corpus of Contemporary Written Japanese (BCCWJ) (Maekawa, 2008), plus newspaper articles and daily conversation sentences. We also used a 234,652-word dictionary (UniDic) provided with the BCCWJ. A small portion of the BCCWJ core data is reserved for testing. In addition, we manually segmented sentences randomly obtained from Twitter[8] during the same period as the log collection for the test corpus. Table 2 shows the details of these corpora.

[6]The results were stable for n_c in the preliminary experiments.

[7]The reason why we use SVM for word segmentation is that the accuracy is generally higher than that based on LR. It was so in the experiments of this paper. The F-measure of LR on TWI-test was 91.30 (Recall = 89.50, Precision = 93.17), which is lower than that of SVM (see Table 4). To make an SSC, however, we use an LR model because we need word boundary probabilities.

[8]We extracted body text from 1,592 tweets excluding mentions, hash tags, URLs, and ticker symbols. Then we divided the body text into sentences by separating on newline characters, resulting in 2,976 sentences.

Table 4: WS accuracy on the tweets.

	Recall [%]	Precision [%]	F-measure
Baseline	90.31	94.05	92.14
+ Log-as-is	90.33	93.77	92.02
+ Log-chunk	91.04	94.29	92.64
+ Log-mconv	90.62	94.09	92.32
+ Log-chunk-mconv	91.40	94.45	92.90

Table 5: WS accuracy on BCCWJ.

	Recall [%]	Precision [%]	F-measure
Baseline	99.01	98.97	98.99
+ Log-as-is	99.02	98.87	98.94
+ Log-chunk	99.05	98.88	98.96
+ Log-mconv	98.98	98.91	98.95
+ Log-chunk-mconv	98.98	98.92	98.95

6.2 Models using Input Method Logs

To make the training data for our IM server, we first chose randomly selected tweets (786,331 sentences) in addition to the unannotated part of the BCCWJ (358,078 sentences). We then trained LR models which estimate word boundary probabilities and pronunciation probabilities for words (and word candidates) from the training data shown in Table 2 and UniDic. We made a pSTC for our IM engine from 1,207,182 sentences randomly obtained from Twitter by following the procedure which we explained in Subsection 3.2.3[9].

We launched our IM as a browser add-on for Twitter and collected 19,770 IM logs from 7 users between April 24 and December 31, 2014. Following the procedures in Section 5.1, we obtained the language resources shown in Table 3. We combined them with the training corpus and dictionaries to build four WSs, which we compared with the baseline.

6.3 Results and Discussion

Following the standard in WS experiments, the evaluation criteria are recall, precision, and F-measure (their harmonic mean). Recall is the number of correctly segmented words divided by the number of words in the test corpus. Precision is the number of correctly segmented words divided by the number of words in the system output.

Table 4 and 5 show WS accuracy on TWI-test and BCCWJ-test, respectively. The difference in accuracy of the baseline method on BCCWJ-test and TWI-test shows that WS of tweets is very difficult. The fact that the precision on TWI-test is much higher than the recall indicates that the baseline model suffers from over-segmentation. This over-segmentation problem is mainly caused by OOV words being divided into known words. For example, “横アリ” (Yokohama arena) is divided into the two know words “横” (side) and “アリ” (ant).

When we compare the F-measures on TWI-test, all the models referring to the IM logs outperform the baseline model trained only from the BCCWJ. The highest is the Log-chunk-mconv model and the improvement over the baseline is statistically significant (significance level: 1%). In addition the accuracies of the five methods on the BCCWJ (Table 5) are almost the same and there is no statistical significance (significance level: 1%) between any two of them.

We analyzed the words misrecognized by the WSs, which we call error words. Table 6 shows the number of error words, the number of OOV words, and the ratio of OOV words to error words. Here the vocabulary is the set of the words appearing in the training data or in UniDic (see Table 2). Although the result of the WS trained on Log-as-is contains more error words than the baseline, the OOV ratio is less than the baseline. This means that the IM logs have a potential to reduce errors caused by OOV words.

Table 6 also indicates that the best method Log-chunk-mconv had the greatest success in reducing

[9]There is no overlap with the test data.

Table 6: Ratio of OOV words in error words.

	#Error words	#OOV words	(ratio[%])
Baseline	446	103	(23.09)
+ Log-as-is	467	89	(19.06)
+ Log-chunk	428	81	(18.93)
+ Log-mconv	443	88	(19.86)
+ Log-chunk-mconv	413	74	(17.79)

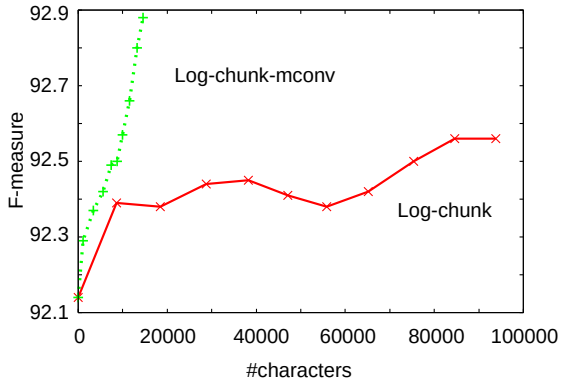


Figure 3: Relationship between WS accuracy on the tweets and log size.

errors caused by OOV words. However, the majority of error words are in-vocabulary words. It can be said that our log chunking method (Log-chunk or Log-chunk-mconv) enabled the WSs to eliminate many known word errors by using context information.

To investigate the impact of the log size, we measured WS accuracy on TWI-test when varying the log size during training. Figure 3 shows the results. Table 4 says that Log-chunk-mconv and Log-chunk increase the accuracy nicely. The graph, however, clarifies that Log-chunk-mconv achieves high accuracy with fewer training data converted from logs. In other words, the method Log-chunk-mconv is good at distilling the informative parts and filtering out the noisy parts. These characteristics are very important properties to have as we consider deploying our IM to a wider audience. An IM is needed to type Japanese and the number of Japanese speakers is more than 100 million. If we can use input logs of even 1% of them for the same or longer period[10], the idea we propose in this paper can improve WS accuracy on various domains efficiently and automatically.

As a final remark, this paper describes a successful example of how to build a useful tool for the NLP community. This process has three steps: 1) design a useful NLP application that can collect user logs, 2) deploy it for public use, and 3) devise a method for mining data from the logs.

[10]The number of users using our system in this paper is 7 for 8 months.

7 Conclusion

This paper described the design of a publicly usable IM which collects natural annotations for use as training data for another system. Specifically, we (1) described how to construct an IM server that suggests OOV word candidates, (2) designed a publicly usable IM that collects logs of user behavior, and (3) proposed a method for using this data to improve word segmenters. Tweets from Twitter are a promising source of data with great potential for NLP, which is one reason why we used them as the target domain for our experiments. The experimental results showed that our methods improve accuracy in this domain. Our method itself is domain-independent and only needs logs from the target domain, so it is worth testing on other domains and with much longer periods of data collection.

Acknowledgments

This work was supported by JSPS Grants-in-Aid for Scientific Research Grant Number 26280084 and Microsoft CORE project. We thank Dr. Hisami Suzuki, Dr. Koichiro Yoshino, and Mr. Daniel Flannery for their valuable comments and suggestions on the manuscript. We are also grateful to the anonymous users of our input method.

References

Peter F. Brown, Vincent J. Della Pietra, Peter V. deSouza, Jennifer C. Lai, and Robert L. Mercer. 1992. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479.

Zheng Chen and Kai-Fu Lee. 2000. A new statistical approach to Chinese pinyin input. In *Proceedings*

of the 38th Annual Meeting of the Association for Computational Linguistics, pages 241–247.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874.

Ryuichiro Higashinaka, Noriaki Kawamae, Kugatsu Sadamitsu, Yasuhiro Minami, Toyomi Meguro, Kohji Dohsaka, and Hirohito Inagaki. 2011. Building a conversational model from two-tweets. *IEEE Transactions on ASRU*, pages 330–335.

Wenbin Jiang, Meng Sun, Yajuan Lu, Yating Yang, and Qun Liu. 2013. Discriminative learning with natural annotations: Word segmentation as a case study. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 761–769.

Nobuhiro Kaji and Masaru Kitsuregawa. 2014. Accurate word segmentation and POS tagging for Japanese microblogs: Corpus annotation and joint modeling with lexical normalization. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 99–109.

Hirotaka Kameko, Shinsuke Mori, and Yoshimasa Tsuruoka. 2015. Can symbol grounding improve low-level NLP? Word segmentation as a case study. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Taku Kudo, Kaoru Yamamoto, and Yuji Matsumoto. 2004. Applying conditional random fields to Japanese morphological analysis. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 230–237.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth ICML*, pages 282–289.

Yijia Liu, Yue Zhang, Wangxiang Che, Ting Liu, and Fan Wu. 2014. Domain adaptation for CRF-based Chinese word segmentation using free annotations. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 864–874.

Kikuo Maekawa. 2008. Balanced corpus of contemporary written Japanese. In *Proceedings of the 6th Workshop on Asian Language Resources*, pages 101–102.

Hirokuni Maeta and Shinsuke Mori. 2012. Statistical input method based on a phrase class n-gram model. In *Workshop on Advances in Text Input Methods*.

Shinsuke Mori and Gakuto Kurata. 2005. Class-based variable memory length markov model. In *Proceedings of the InterSpeech2005*, pages 13–16.

Shinsuke Mori and Graham Neubig. 2011. A pointwise approach to pronunciation estimation for a TTS front-end. In *Proceedings of the InterSpeech2011*, pages 2181–2184.

Shinsuke Mori and Graham Neubig. 2014. Language resource addition: Dictionary or corpus? In *Proceedings of the Nineth International Conference on Language Resources and Evaluation*, pages 1631–1636.

Shinsuke Mori and Hiroki Oda. 2009. Automatic word segmentation using three types of dictionaries. In *Proceedings of the Eighth International Conference Pacific Association for Computational Linguistics*, pages 1–6.

Shinsuke Mori and Daisuke Takuma. 2004. Word n-gram probability estimation from a Japanese raw corpus. In *Proceedings of the Eighth International Conference on Speech and Language Processing*, pages 1037–1040.

Shinsuke Mori, Tsuchiya Masatoshi, Osamu Yamaji, and Makoto Nagao. 1999. Kana-kanji conversion by a stochastic model. *Transactions of Information Processing Society of Japan*, 40(7):2946–2953. (in Japanese).

Shinsuke Mori, Daisuke Takuma, and Gakuto Kurata. 2006. Phoneme-to-text transcription system with an infinite vocabulary. In *Proceedings of the 21st International Conference on Computational Linguistics*, pages 729–736.

Masaaki Nagata. 1994. A stochastic Japanese morphological analyzer using a forward-DP backward-A* n-best search algorithm. In *Proceedings of the 15th International Conference on Computational Linguistics*, pages 201–207.

Graham Neubig and Shinsuke Mori. 2010. Word-based partial annotation for efficient corpus construction. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation*, pages 2723–2727.

Graham Neubig, Yosuke Nakata, and Shinsuke Mori. 2011. Pointwise prediction for robust, adaptable Japanese morphological analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, pages 529–533.

Fuchun Peng, Fangfang Feng, and Andrew McCallum. 2004. Chinese segmentation and new word detection using conditional random fields. In *Proceedings of the 20th International Conference on Computational Linguistics*, pages 562–568.

Dana Ron, Yoram Singer, and Naftali Tishby. 1996. The power of amnesia: Learning probabilistic automata with variable memory length. *Machine Learning*, 25:117–149.

Takeshi Sakai, Makoto Okazaki, and Yutaka Matsuo. 2010. Earthquake shakes Twitter users: Real-time event detection by social sensors. In *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, pages 851–860.

Richard Sproat and Chilin Shih William Gale Nancy Chang. 1996. A stochastic finite-state word-segmentation algorithm for Chinese. *Computational Linguistics*, 22(3):377–404.

Yuta Tsuboi, Hisashi Kashima, Shinsuke Mori, Hiroki Oda, and Yuji Matsumoto. 2008. Training conditional random fields using incomplete annotations. In *Proceedings of the 22nd International Conference on Computational Linguistics*, pages 897–904.

Sho Tsugawa, Yukiko Mogi, Yusuke Kikuchi, Fumio Kishino, Kazuyuki Fujita, Yuichi Itoh, and Hiroyuki Ohsaki. 2013. On estimating depressive tendencies of Twitter users utilizing their tweet data. In *VR'13*, pages 1–4.

Fan Yang and Paul Vozila. 2014. Semi-supervised Chinese word segmentation using partial-label learning with conditional random fields. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 90–98.

Long Short-Term Memory Neural Networks for Chinese Word Segmentation

Xinchi Chen, Xipeng Qiu*, Chenxi Zhu, Pengfei Liu, Xuanjing Huang
Shanghai Key Laboratory of Intelligent Information Processing, Fudan University
School of Computer Science, Fudan University
825 Zhangheng Road, Shanghai, China
{xinchichen13,xpqiu,czhu13,pfliu14,xjhuang}@fudan.edu.cn

Abstract

Currently most of state-of-the-art methods for Chinese word segmentation are based on supervised learning, whose features are mostly extracted from a local context. These methods cannot utilize the long distance information which is also crucial for word segmentation. In this paper, we propose a novel neural network model for Chinese word segmentation, which adopts the long short-term memory (LSTM) neural network to keep the previous important information in memory cell and avoids the limit of window size of local context. Experiments on PKU, MSRA and CTB6 benchmark datasets show that our model outperforms the previous neural network models and state-of-the-art methods.

1 Introduction

Word segmentation is a fundamental task for Chinese language processing. In recent years, Chinese word segmentation (CWS) has undergone great development. The popular method is to regard word segmentation task as a sequence labeling problem (Xue, 2003; Peng et al., 2004). The goal of sequence labeling is to assign labels to all elements in a sequence, which can be handled with supervised learning algorithms such as Maximum Entropy (ME) (Berger et al., 1996) and Conditional Random Fields (CRF) (Lafferty et al., 2001). However, the ability of these models is restricted by the design of features, and the number of features could be so large that the result models are too large for practical use and prone to overfit on training corpus.

Recently, neural network models have increasingly used for NLP tasks for their ability to minimize the effort in feature engineering (Collobert et al., 2011; Socher et al., 2013; Turian et al., 2010; Mikolov et al., 2013b; Bengio et al., 2003). Collobert et al. (2011) developed the SENNA system that approaches or surpasses the state-of-the-art systems on a variety of sequence labeling tasks for English. Zheng et al. (2013) applied the architecture of Collobert et al. (2011) to Chinese word segmentation and POS tagging, also he proposed a perceptron style algorithm to speed up the training process with negligible loss in performance. Pei et al. (2014) models tag-tag interactions, tag-character interactions and character-character interactions based on Zheng et al. (2013). Chen et al. (2015) proposed a gated recursive neural network (GRNN) to explicitly model the combinations of the characters for Chinese word segmentation task. Each neuron in GRNN can be regarded as a different combination of the input characters. Thus, the whole GRNN has an ability to simulate the design of the sophisticated features in traditional methods.

Despite of their success, a limitation of them is that their performances are easily affected by the size of the context window. Intuitively, many words are difficult to segment based on the local information only. For example, the segmentation of the following sentence needs the information of the long distance collocation.

冬天 (winter)，能 (can) 穿 (wear) 多少 (amount) 穿 (wear) 多少 (amount)；夏天 (summer)，能 (can) 穿 (wear) 多 (more) 少 (little) 穿 (wear) 多 (more) 少 (little)。

Without the word “夏天 (summer)” or “冬天 (winter)”, it is difficult to segment the phrase “能穿多少穿多少”. Therefore, we usually need utilize the non-local information for more accurate word segmentation. However, it does not work by simply increasing the context window size. As reported in (Zheng et al., 2013), the performance drops smoothly when the window size is larger than 3. The reason is that the number of its parameters is so large that the trained network has

*Corresponding author.

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1197–1206,
Lisbon, Portugal, 17-21 September 2015.

overfitted on training data. Therefore, it is necessary to capture the potential long-distance dependencies without increasing the size of the context window.

In order to address this problem, we propose a neural model based on Long Short-Term Memory Neural Network (LSTM) (Hochreiter and Schmidhuber, 1997) that explicitly model the previous information by exploiting input, output and forget gates to decide how to utilize and update the memory of pervious information. Intuitively, if the LSTM unit detects an important feature from an input sequence at early stage, it easily carries this information (the existence of the feature) over a long distance, hence, capturing the potential useful long-distance information. We evaluate our model on three popular benchmark datasets (PKU, MSRA and CTB6), and the experimental results show that our model achieves the state-of-the-art performance with the smaller context window size (0,2).

The contributions of this paper can be summarized as follows.

- We first introduce the LSTM neural network for Chinese word segmentation. The LSTM can capture potential long-distance dependencies and keep the previous useful information in memory, which avoids the limit of the size of context window.
- Although there are relatively few researches of applying dropout method to the LSTM, we investigate several dropout strategies and find that dropout is also effective to avoid the overfitting of the LSTM.
- Despite Chinese word segmentation being a specific case, our model can be easily generalized and applied to the other sequence labeling tasks.

2 Neural Model for Chinese Word Segmentation

Chinese word segmentation is usually regarded as character-based sequence labeling. Each character is labeled as one of {B, M, E, S} to indicate the segmentation. {B, M, E} represent *Begin*, *Middle*, *End* of a multi-character segmentation respectively, and S represents a *Single* character segmentation.

The neural model is usually characterized by three specialized layers: (1) a character embedding layer; (2) a series of classical neural network layers and (3) tag inference layer. An illustration is shown in Figure 1.

The most common tagging approach is based on a local window. The window approach assumes that the tag of a character largely depends on its neighboring characters. Given an input sentence $c^{(1:n)}$, a window of size k slides over the sentence from character $c^{(1)}$ to $c^{(n)}$, where n is the length of the sentence. As shown in Figure 1, for each character $c^{(t)}(1 \leq t \leq n)$, the context characters ($c^{(t-2)}$,$c^{(t-1)}$,$c^{(t)}$,$c^{(t+1)}$,$c^{(t+2)}$) are fed into the lookup table layer when the window size k is 5. The characters exceeding the sentence boundaries are mapped to one of two special symbols, namely "start" and "end" symbols. The character embeddings extracted by the lookup table layer are then concatenated into a single vector $\mathbf{x}^{(t)} \in \mathbb{R}^{H_1}$, where $H_1 = k \times d$ is the size of layer 1. Then $\mathbf{x}^{(t)}$ is fed into the next layer which performs linear transformation followed by an element-wise activation function g such as sigmoid function $\sigma(x) = (1+e^{-x})^{-1}$ and hyperbolic tangent function $\phi(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$here.

$$\mathbf{h}^{(t)} = g(\mathbf{W}_1\mathbf{x}^{(t)} + \mathbf{b}_1), \quad (1)$$

where $\mathbf{W}_1 \in \mathbb{R}^{H_2 \times H_1}$, $\mathbf{b}_1 \in \mathbb{R}^{H_2}$, $\mathbf{h}^{(t)} \in \mathbb{R}^{H_2}$. H_2 is a hyper-parameter which indicates the number of hidden units in layer 2. Given a set of tags $\mathcal{T}$ of size $|\mathcal{T}|$, a similar linear transformation is performed except that no non-linear function is followed:

$$\mathbf{y}^{(t)} = \mathbf{W}_2\mathbf{h}^{(t)} + \mathbf{b}_2, \quad (2)$$

where $\mathbf{W}_2 \in \mathbb{R}^{|\mathcal{T}| \times H_2}$, $\mathbf{b}_2 \in \mathbb{R}^{|\mathcal{T}|}$. $\mathbf{y}^{(t)} \in \mathbb{R}^{|\mathcal{T}|}$ is the score vector for each possible tag. In Chinese word segmentation, the most prevalent tag set $T \subseteqq \mathcal{T}$ is {B, M, E, S} as mentioned above.

To model the tag dependency, a transition score $\mathbf{A}_{ij}$ is introduced to measure the probability of jumping from tag $i \in T$ to tag $j \in T$ (Collobert et al., 2011). Although this model works well for Chinese word segmentation and other sequence labeling tasks, it just utilizes the information of context of a limited-length window. Some useful long distance information is neglected.

3 Long Short-Term Memory Neural Network for Chinese Word Segmentation

In this section, we introduce the LSTM neural network for Chinese word segmentation.

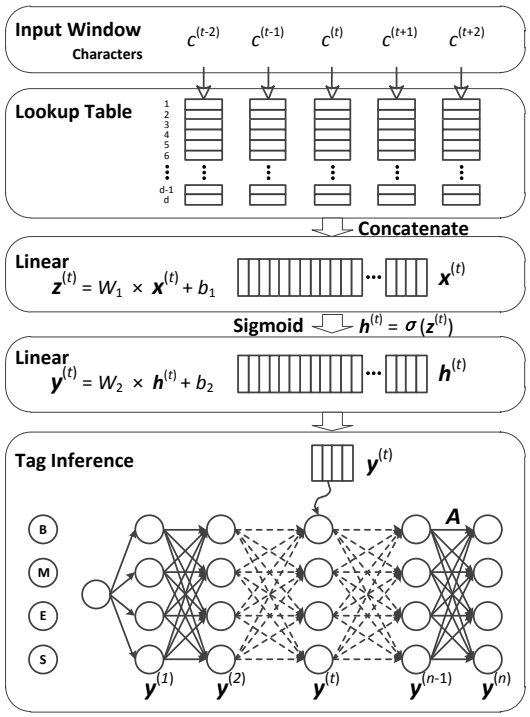


Figure 1: General architecture of neural model for Chinese word segmentation.

3.1 Character Embeddings

The first step of using neural network to process symbolic data is to represent them into distributed vectors, also called embeddings (Bengio et al., 2003; Collobert and Weston, 2008).

Formally, in Chinese word segmentation task, we have a character dictionary $\mathcal{C}$ of size $|\mathcal{C}|$. Unless otherwise specified, the character dictionary is extracted from the training set and unknown characters are mapped to a special symbol that is not used elsewhere. Each character $c \in \mathcal{C}$ is represented as a real-valued vector (character embedding) $\mathbf{v}_c \in \mathbb{R}^d$ where d is the dimensionality of the vector space. The character embeddings are then stacked into an embedding matrix $\mathbf{M} \in \mathbb{R}^{d \times |\mathcal{C}|}$. For a character $c \in \mathcal{C}$, the corresponding character embedding $\mathbf{v}_c \in \mathbb{R}^d$ is retrieved by the lookup table layer. And the lookup table layer can be regarded as a simple projection layer where the character embedding for each context character is achieved by table lookup operation according to its index.

3.2 LSTM

The long short term memory neural network (LSTM) (Hochreiter and Schmidhuber, 1997) is an extension of the recurrent neural network (RNN).

The RNN has recurrent hidden states whose output at each time is dependent on that of the previous time. More formally, given a sequence $\mathbf{x}^{(1:n)} = (\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(t)}, \ldots, \mathbf{x}^{(n)})$, the RNN updates its recurrent hidden state $\mathbf{h}^{(t)}$ by

$$\mathbf{h}^{(t)} = g(\mathbf{U}\mathbf{h}^{(t-1)} + \mathbf{W}\mathbf{x}^{(t)} + \mathbf{b}), \quad (3)$$

where g is a nonlinear function as mentioned above.

Though RNN has been proven successful on many tasks such as speech recognition (Vinyals et al., 2012), language modeling (Mikolov et al., 2010) and text generation (Sutskever et al., 2011), it can be difficult to train them to learn long-term dynamics, likely due in part to the vanishing and exploding gradient problem (Hochreiter and Schmidhuber, 1997).

The LSTM provides a solution by incorporating memory units that allow the network to learn when to forget previous information and when to update the memory cells given new information. Thus, it is a natural choice to apply LSTM neural network to word segmentation task since the LSTM neural network can learn from data with long range temporal dependencies (memory) due to the considerable time lag between the inputs and their corresponding outputs. In addition, the LSTM has been applied successfully in many NLP tasks, such as text classification (Liu et al., 2015) and machine translation (Sutskever et al., 2014).

The core of the LSTM model is a memory cell $\mathbf{c}$ encoding memory at every time step of what inputs have been observed up to this step (see Figure 2) . The behavior of the cell is controlled by three "gates", namely input gate $\mathbf{i}$, forget gate $\mathbf{f}$ and output gate $\mathbf{o}$. The operations on gates are defined as element-wise multiplications, thus gate can either scale the input value if the gate is non-zero vector or omit input if the gate is zero vector. The output of output gate will be fed into the next time step $t+1$ as previous hidden state and input of upper layer of neural network at current time step t. The definitions of the gates, cell update and output are as follows:

$$\mathbf{i}^{(t)} = \sigma(\mathbf{W}_{ix}\mathbf{x}^{(t)} + \mathbf{W}_{ih}\mathbf{h}^{(t-1)} + \mathbf{W}_{ic}\mathbf{c}^{(t-1)}), \quad (4)$$

$$\mathbf{f}^{(t)} = \sigma(\mathbf{W}_{fx}\mathbf{x}^{(t)} + \mathbf{W}_{fh}\mathbf{h}^{(t-1)} + \mathbf{W}_{fc}\mathbf{c}^{(t-1)}), \quad (5)$$

$$\mathbf{c}^{(t)} = \mathbf{f}^{(t)} \odot \mathbf{c}^{(t-1)} + \mathbf{i}^{(t)} \odot \phi(\mathbf{W}_{cx}\mathbf{x}^{(t)} + \mathbf{W}_{ch}\mathbf{h}^{(t-1)}), \quad (6)$$

$$\mathbf{o}^{(t)} = \sigma(\mathbf{W}_{ox}\mathbf{x}^{(t)} + \mathbf{W}_{oh}\mathbf{h}^{(t-1)} + \mathbf{W}_{oc}\mathbf{c}^{(t)}), \quad (7)$$

$$\mathbf{h}^{(t)} = \mathbf{o}^{(t)} \odot \phi(\mathbf{c}^{(t)}), \quad (8)$$

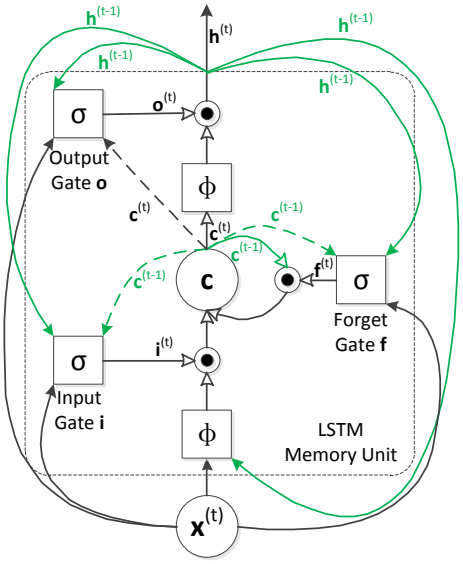


Figure 2: LSTM Memory Unit. The memory unit contains a cell **c** which is controlled by three gates. The green links show the signals at time $t-1$, while the black links show the current signals. The dashed links represent the weight matrices from beginning to end are diagonal. Moreover, the solid pointers mean there are weight matrices on the connections, and hollow pointers mean none. The current output signal, $\mathbf{h}^{(t)}$, will fed back to the next time $t+1$ via three gates, and is the input of the higher layer of the neural network as well.

where σ and ϕ are the logistic sigmoid function and hyperbolic tangent function respectively; $\mathbf{i}^{(t)}$, $\mathbf{f}^{(t)}$, $\mathbf{o}^{(t)}$ and $\mathbf{c}^{(t)}$ are respectively the input gate, forget gate, output gate, and memory cell activation vector at time step t, all of which have the same size as the hidden vector $\mathbf{h}^{(t)} \in \mathbb{R}^{H_2}$; the parameter matrices Ws with different subscripts are all square matrices; $\odot$ denotes the element-wise product of the vectors. Note that $\mathbf{W}_{ic}$, $\mathbf{W}_{fc}$ and $\mathbf{W}_{oc}$ are diagonal matrices.

3.3 LSTM Architectures for Chinese Word Segmentation

To fully utilize the LSTM, we propose four different structures of neural network to select the effective features via memory units. Figure 3 illustrates our proposed architectures.

LSTM-1 The LSTM-1 simply replace the hidden neurons in Eq. (1) with LSTM units (See Figure 3a).

The input of the LSTM unit is from a window of context characters. For each character, $c^{(t)}, (1 \leq t \leq n)$, the input of the LSTM unit $\mathbf{x}^{(t)}$,

$$\mathbf{x}^{(t)} = \mathbf{v}_c^{(t-k_1)} \oplus \cdots \oplus \mathbf{v}_c^{(t+k_2)}, \quad (9)$$

is concatenated from character embeddings of $c^{(t-k_1):(t+k_2)}$, where k_1 and k_2 represent the numbers of characters from left and right contexts respectively. The output of the LSTM unit is used in final inference function (Eq. (11)) after a linear transformation.

LSTM-2 The LSTM-2 can be created by stacking multiple LSTM hidden layers on top of each other, with the output sequence of one layer forming the input sequence for the next (See Figure 3b). Here we use two LSTM layers. Specifically, input of the upper LSTM layer takes $\mathbf{h}^{(t)}$ from the lower LSTM layer without any transformation. The input of the first layer is same to LSTM-1, and the output of the second layer is as same operation as LSTM-1.

LSTM-3 The LSTM-3 is a extension of LSTM-1, which adopts a local context of LSTM layer as input of the last layer (See Figure 3c). For each time step t, we concatenate the outputs of a window of the LSTM layer into a vector $\hat{\mathbf{h}}^{(t)}$,

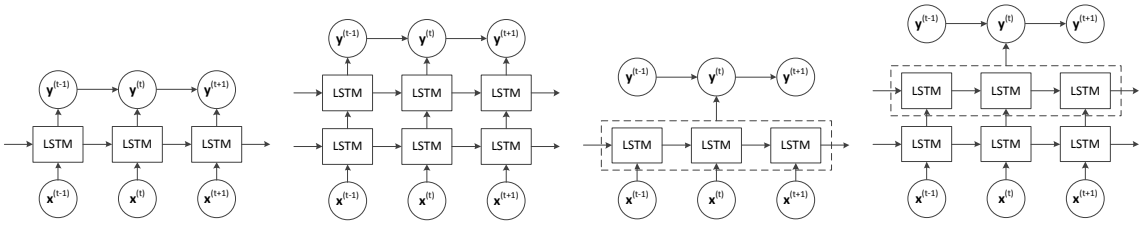
$$\hat{\mathbf{h}}^{(t)} = \mathbf{h}^{(t-m_1)} \oplus \cdots \oplus \mathbf{h}^{(t+m_2)}, \quad (10)$$

where m_1 and m_2 represent the lengths of time lags before and after current time step.Finally, $\hat{\mathbf{h}}^{(t)}$ is used in final inference function (Eq. (11)) after a linear transformation.

LSTM-4 The LSTM-4 (see Figure 3d) is a mixture of the LSTM-2 and LSTM-3, which consists of two LSTM layers. The output sequence of the lower LSTM layer forms the input sequence of the upper LSTM layer. The final layer adopts a local context of upper LSTM layer as input.

3.4 Inference at Sentence Level

To model the tag dependency, previous neural network models (Collobert et al., 2011; Zheng et al., 2013; Pei et al., 2014) introduced the transition score $\mathbf{A}_{ij}$ for measuring the probability of jumping from tag $i \in T$ to tag $j \in T$. For a input sentence $c^{(1:n)}$ with a tag sequence $y^{(1:n)}$, a sentence-level score is then given by the sum of tag transition scores and network tagging scores:



(a) LSTM-1 (b) LSTM-2 (c) LSTM-3 (d) LSTM-4

Figure 3: Our proposed LSTM architectures for Chinese word segmentation.

$$s(c^{(1:n)}, y^{(1:n)}, \theta) = \sum_{t=1}^{n} \left(\mathbf{A}_{y^{(t-1)}y^{(t)}} + \mathbf{y}^{(t)}_{y^{(t)}} \right), \quad (11)$$

where $\mathbf{y}^{(t)}_{y^{(t)}}$ indicates the score of tag $y^{(t)}$, and $\mathbf{y}^{(t)}$ is computed by the network as in Eq. (2). The parameter set of our model $\theta = \{\mathbf{M}, \mathbf{A}, \mathbf{W}_{ic}, \mathbf{W}_{fc}, \mathbf{W}_{oc}, \mathbf{W}_{ix}, \mathbf{W}_{fx}, \mathbf{W}_{ox}, \mathbf{W}_{ih}, \mathbf{W}_{fh}, \mathbf{W}_{oh}, \mathbf{W}_{cx}, \mathbf{W}_{ch}\}$.

4 Training

4.1 Max-Margin criterion

We use the Max-Margin criterion to train our model. Intuitively, the Max-Margin criterion provides an alternative to probabilistic, likelihood based estimation methods by concentrating directly on the robustness of the decision boundary of a model (Taskar et al., 2005). We use $Y(x_i)$ to denote the set of all possible tag sequences for a given sentence x_i and the correct tag sequence for x_i is y_i. The parameter set of our model is θ. We first define a structured margin loss $\Delta(y_i, \hat{y})$ for predicted tag sequence $\hat{y}$:

$$\Delta(y_i, \hat{y}) = \sum_{t}^{n} \eta \mathbf{1}\{y_i^{(t)} \neq \hat{y}^{(t)}\}, \quad (12)$$

where n is the length of sentence x_i and η is a discount parameter. The loss is proportional to the number of characters with incorrect tags in the proposed tag sequence. For a given training instance (x_i, y_i),the predicted tag sequence $\hat{y}_i \in Y(x_i)$ is the one with the highest score:

$$\hat{y}_i = \underset{y \in Y(x_i)}{\arg\max}\, s(x_i, y, \theta), \quad (13)$$

where the function $s(\cdot)$ is sentence-level score and defined in equation (11).

Given a set of training set $\mathcal{D}$, the regularized objective function is the loss function $J(\theta)$ including a l_2-norm term:

$$J(\theta) = \frac{1}{|\mathcal{D}|} \sum_{(x_i, y_i) \in \mathcal{D}} l_i(\theta) + \frac{\lambda}{2} \|\theta\|_2^2, \quad (14)$$

where $l_i(\theta) = \max(0, s(x_i, \hat{y}_i, \theta) + \Delta(y_i, \hat{y}_i) - s(x_i, y_i, \theta))$.

To minimize $J(\theta)$, we use a generalization of gradient descent called subgradient method (Ratliff et al., 2007) which computes a gradient-like direction.

Following (Socher et al., 2013), we also use the diagonal variant of AdaGrad (Duchi et al., 2011) with minibatchs to minimize the objective. The parameter update for the i-th parameter $\theta_{t,i}$ at time step t is as follows:

$$\theta_{t,i} = \theta_{t-1,i} - \frac{\alpha}{\sqrt{\sum_{\tau=1}^{t} g_{\tau,i}^2}} g_{t,i}, \quad (15)$$

where α is the initial learning rate and $g_\tau \in \mathbb{R}^{|\theta_i|}$ is the subgradient at time step τ for parameter θ_i. In addition, the process of back propagation is followd Hochreiter and Schmidhuber (1997).

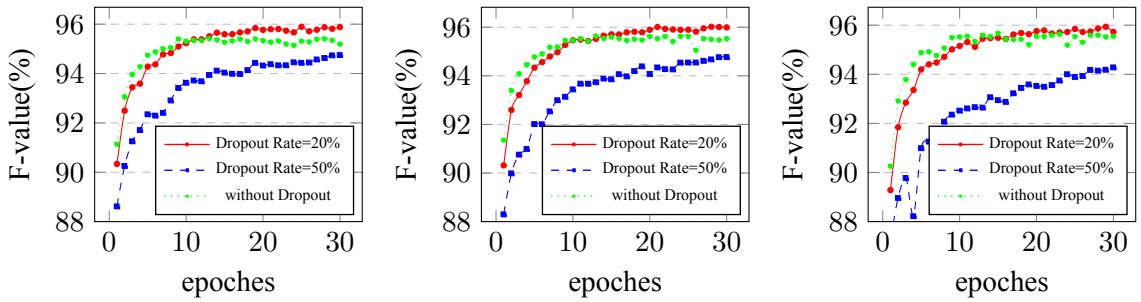
4.2 Dropout

Dropout is one of prevalent methods to avoid overfitting in neural networks (Srivastava et al., 2014). When dropping a unit out, we temporarily remove it from the network, along with all its incoming and outgoing connections. In the simplest case, each unit is omitted with a fixed probability p independent of other units, namely dropout rate, where p is also chosen on development set.

5 Experiments

5.1 Datasets

We use three popular datasets, PKU, MSRA and CTB6, to evaluate our model. The PKU



(a) LSTM-1(2,2) (b) LSTM-1 (1,2) (c) LSTM-1(0,2)

Figure 4: Performances of LSTM-1 with the different context lengths and dropout rates on PKU development set.

Context length	$(k_1, k_2) = (0, 2)$
Character embedding size	$d = 100$
Hidden unit number	$H_2 = 150$
Initial learning rate	$\alpha = 0.2$
Margin loss discount	$\eta = 0.2$
Regularization	$\lambda = 10^{-4}$
Dropout rate on input layer	$p = 0.2$

Table 1: Settings of the hyper-parameters.

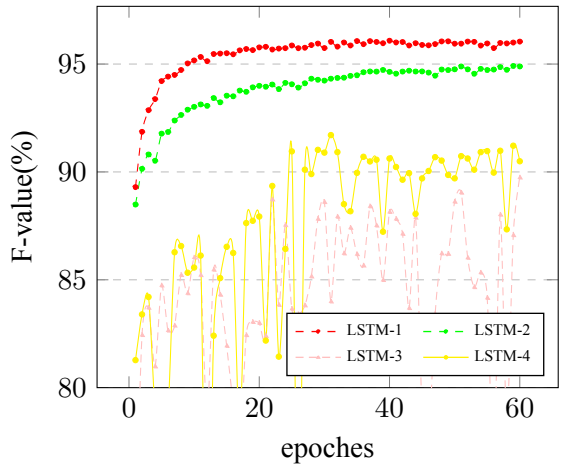


Figure 5: Performances of LSTM-1 (0,2) with 20% dropout on PKU development set.

and MSRA data are provided by the second International Chinese Word Segmentation Bakeoff (Emerson, 2005), and CTB6 is from Chinese TreeBank 6.0 (LDC2007T36) (Xue et al., 2005), which is a segmented, part-of-speech tagged and fully bracketed corpus in the constituency formalism. These datasets are commonly used by previous state-of-the-art models and neural network models. In addition, we use the first 90% sentences of the training data as training set and the rest 10% sentences as development set for PKU and MSRA datasets. For CTB6 dataset, we divide the training, development and test sets according to (Yang and Xue, 2012)

All datasets are preprocessed by replacing the Chinese idioms and the continuous English characters and digits with a unique flag.

For evaluation, we use the standard bake-off scoring program to calculate precision, recall, F1-score and out-of-vocabulary (OOV) word recall.

5.2 Hyper-parameters

Hyper-parameters of neural model impact the performance of the algorithm significantly. According to experiment results, we choose the hyper-parameters of our model as showing in Figure 1. The minibatch size is set to 20. Generally, the number of hidden units has a limited impact on the performance as long as it is large enough. We found that 150 is a good trade-off between speed and model performance. The dimensionality of character embedding is set to 100 which achieved the best performance. All these hyper-parameters are chosen according to their average performances on three development sets.

For the context lengths (k_1, k_2) and dropout strategy, we give detailed analysis in next section.

5.3 Dropout and Context Length

We first investigate the different dropout strategies, including dropout at different layers and with different dropout rate p. As a result, we found that it is a good trade-off between speed and model performance to drop the input layer only with dropout rate $p_{input} = 0.2$. However, it does not show any significant improvement to dropout on hidden LSTM layers.

Context Length	Dropout rate=20%			Dropout rate=50%			without Dropout		
	P	R	F	P	R	F	P	R	F
LSTM-1 (2,2)	95.8	95.3	95.6	94.8	94.4	94.6	95.2	94.9	95.1
LSTM-1 (1,2)	95.7	95.3	95.5	94.8	94.4	94.6	95.4	94.9	95.2
LSTM-1 (0,2)	**95.8**	**95.5**	**95.7**	94.6	94.2	94.4	95.4	95.0	95.2

Table 2: Performances of LSTM-1 with the different context lengths and dropout rates on PKU test set.

models	Contextr Length = (0,2)		
	P	R	F
LSTM-1	**95.8**	**95.5**	**95.7**
LSTM-2	95.1	94.5	94.8
LSTM-3	89.1	90.4	89.8
LSTM-4	92.1	91.7	91.9

Table 3: Performance on our four proposed models on PKU test set.

Due to space constraints, we just give the performances of LSTM-1 model on PKU dataset with different context lengths (k_1, k_2) and dropout rates in Figure 4 and Table 2. From Figure 4, we can see that 20% dropout converges slightly slower than the one without dropout, but avoids overfitting. 50% or higher dropout rate seems to be underfitting since its training error is also high.

Table 2 shows that the LSTM-1 model performs consistently well with the different context length, but the LSTM-1 model with short context length saves computational resource, and gets more efficiency. At the meanwhile, the LSTM-1 model with context length (0,2) can receive the same or better performance than that with context length (2,2), which shows that the LSTM model can well model the pervious information, and it is more robust for its insensitivity of window size variation.

We employ context length (0,2) with the 20% dropout rate in the following experiments to balance the tradeoff between accuracy and efficiency.

5.4 Model Selection

We also evaluate the our four proposed models with the hyper-parameter settings in Table 1. For LSTM-3 and LSTM-4 models, the context window length of top LSTM layer is set to (2,0). For LSTM-2 and LSTM-4,the number of upper hidden LSTM layer is set to 100. We use PKU dataset to select the best model. Figure 5 shows the results of the four models on PKU development set from first epoch to 60-th epoch. We see that the LSTM-1 is the fastest one to converge and achieves the best performance. The LSTM-2 (two LSTM layers) get worse, which shows the performance seems not to benefit from deep model. The LSTM-3 and LSTM-4 models do not converge, which could be caused by the complexity of models.

The results on PKU test set are also shown in Table 3, which again show that the LSTM-1 achieves the best performance. Therefore, in the rest of the paper we will give more analysis based on the LSTM-1 with hyper-parameter settings as showing in Table 1.

5.5 Experiment Results

In this section, we give comparisons of the LSTM-1 with pervious neural models and state-of-the-art methods on the PKU, MSRA and CTB6 datasets.

We first compare our model with two neural models (Zheng et al., 2013; Pei et al., 2014) on Chinese word segmentation task with random initialized character embeddings. As showing in Table 4, the performance is boosted significantly by utilizing LSTM unit. And more notably, our window size of the context characters is set to (0,2), while the size of the other models is (2,2).

Previous works found that the performance can be improved by pre-training the character embeddings on large unlabeled data. We use word2vec [1] (Mikolov et al., 2013a) toolkit to pre-train the character embeddings on the Chinese Wikipedia corpus. The obtained embeddings are used to initialize the character lookup table instead of random initialization. Inspired by (Pei et al., 2014), we also utilize bigram character embeddings which is simply initialized as the average of embeddings of two consecutive characters.

Table 5 shows the performances with additional pre-trained and bigram character embeddings. Again, the performances boost significantly as a result. Moreover, when we use bigram embeddings only, which means we do close test without pre-training the embeddings on other extra corpus, our model still perform competitively compared

[1] http://code.google.com/p/word2vec/

models	PKU			MSRA			CTB6		
	P	R	F	P	R	F	P	R	F
(Zheng et al., 2013)	92.8	92.0	92.4	92.9	93.6	93.3	94.0*	93.1*	93.6*
(Pei et al., 2014)	93.7	93.4	93.5	94.6	94.2	94.4	94.4*	93.4*	93.9*
LSTM	**95.8**	**95.5**	**95.7**	**96.7**	**96.2**	**96.4**	**95.0**	**94.8**	**94.9**

Table 4: Performances on three test sets with random initialized character embeddings. The results with * symbol are from our implementations of their methods.

models	PKU			MSRA			CTB6		
	P	R	F	P	R	F	P	R	F
+Pre-train									
(Zheng et al., 2013)	93.5	92.2	92.8	94.2	93.7	93.9	93.9*	93.4*	93.7*
(Pei et al., 2014)	94.4	93.6	94.0	95.2	94.6	94.9	94.2*	93.7*	94.0*
LSTM	96.3	95.6	96.0	96.7	96.5	96.6	95.9	95.5	95.7
+bigram									
LSTM	96.3	95.9	96.1	97.1	97.1	97.1	95.6	95.3	95.5
+Pre-train+bigram									
(Pei et al., 2014)	-	-	95.2	-	-	97.2	-	-	-
LSTM	**96.6**	**96.4**	**96.5**	**97.5**	**97.3**	**97.4**	**96.2**	**95.8**	**96.0**

Table 5: Performances on three test sets with pre-trained and bigram character embeddings. The results with * symbol are from our implementations of their methods.

Models	PKU	MSRA	CTB6
(Tseng et al., 2005)	95.0	96.4	-
(Zhang and Clark, 2007)	95.1	97.2	-
(Sun and Xu, 2011)	-	-	95.7
(Zhang et al., 2013)	96.1	97.4	-
This work	**96.5**	**97.4**	**96.0**

Table 6: Comparison of our model with state-of-the-art methods on three test sets.

with previous neural models with pre-trained embedding and bigram embeddings.

Table 6 lists the performances of our model as well as previous state-of-the-art systems. (Zhang and Clark, 2007) is a word-based segmentation algorithm, which exploit features of complete words, while the rest of the list are character-based word segmenters, whose features are mostly extracted from a window of characters. Moreover, some systems (such as Sun and Xu (2011) and Zhang et al. (2013)) also exploit kinds of extra information such as unlabeled data or other knowledge. Despite our model only uses simple bigram features, it outperforms previous state-of-the-art models which use more complex features.

Since that we do not focus on the speed of the algorithm in this paper, we do not optimize the speed a lot. On PKU dataset, it takes about 3 days to train the model (last row of Table 5) using CPU (Intel(R) Xeon(R) CPU E5-2665 @ 2.40GHz) only. All implementation is based on Python.

6 Related Work

Chinese word segmentation has been studied with considerable efforts in the NLP community. The most popular word segmentation methods is based on sequence labeling (Xue, 2003). Recently, researchers have tended to explore neural network based approaches (Collobert et al., 2011) to reduce efforts of feature engineering (Zheng et al., 2013; Pei et al., 2014; Qi et al., 2014; Chen et al., 2015). The features of all these methods are extracted from a local context and neglect the long distance information. However, previous information is also crucial for word segmentation. Our model adopts the LSTM to keep the previous important information in memory and avoids the limitation of ambiguity caused by limit of the size of context window.

7 Conclusion

In this paper, we use LSTM to explicitly model the previous information for Chinese word segmentation, which can well model the potential long-

distance features. Though our model use smaller context window size (0,2), it still outperforms the previous neural models with context window size (2,2). Besides, our model can also be easily generalized and applied to other sequence labeling tasks.

Although our model achieves state-of-the-art performance, it only makes use of previous context. The future context is also useful for Chinese word segmentation. In future work, we would like to adopt the bidirectional recurrent neural network (Schuster and Paliwal, 1997) to process the sequence in both directions.

Acknowledgments

We would like to thank the anonymous reviewers for their valuable comments. This work was partially funded by the National Natural Science Foundation of China (61472088, 61473092), National High Technology Research and Development Program of China (2015AA015408), Shanghai Science and Technology Development Funds (14ZR1403200).

References

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *The Journal of Machine Learning Research*, 3:1137–1155.

A.L. Berger, V.J. Della Pietra, and S.A. Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71.

Xinchi Chen, Xipeng Qiu, Chenxi Zhu, and Xuanjing Huang. 2015. Gated recursive neural network for Chinese word segmentation. In *Proceedings of Annual Meeting of the Association for Computational Linguistics*.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of ICML*.

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

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159.

T. Emerson. 2005. The second international Chinese word segmentation bakeoff. In *Proceedings of the Fourth SIGHAN Workshop on Chinese Language Processing*, pages 123–133. Jeju Island, Korea.

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

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*.

PengFei Liu, Xipeng Qiu, Xinchi Chen, Shiyu Wu, and Xuanjing Huang. 2015. Multi-timescale long short-term memory neural network for modelling sentences and documents. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH*.

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

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

Wenzhe Pei, Tao Ge, and Chang Baobao. 2014. Max-margin tensor neural network for chinese word segmentation. In *Proceedings of ACL*.

F. Peng, F. Feng, and A. McCallum. 2004. Chinese segmentation and new word detection using conditional random fields. *Proceedings of the 20th international conference on Computational Linguistics*.

Yanjun Qi, Sujatha G Das, Ronan Collobert, and Jason Weston. 2014. Deep learning for character-based information extraction. In *Advances in Information Retrieval*, pages 668–674. Springer.

Nathan D Ratliff, J Andrew Bagnell, and Martin A Zinkevich. 2007. (online) subgradient methods for structured prediction. In *Eleventh International Conference on Artificial Intelligence and Statistics (AIStats)*.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *Signal Processing, IEEE Transactions on*, 45(11):2673–2681.

Richard Socher, John Bauer, Christopher D Manning, and Andrew Y Ng. 2013. Parsing with compositional vector grammars. In *In Proceedings of the ACL conference*. Citeseer.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Weiwei Sun and Jia Xu. 2011. Enhancing Chinese word segmentation using unlabeled data. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 970–979.

Ilya Sutskever, James Martens, and Geoffrey E Hinton. 2011. Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 1017–1024.

Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112.

Huihsin Tseng, Pichuan Chang, Galen Andrew, Daniel Jurafsky, and Christopher Manning. 2005. A conditional random field word segmenter for sighan bakeoff 2005. In *Proceedings of the fourth SIGHAN workshop on Chinese language Processing*, volume 171.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 384–394. Association for Computational Linguistics.

Oriol Vinyals, Suman V Ravuri, and Daniel Povey. 2012. Revisiting recurrent neural networks for robust asr. In *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, pages 4085–4088. IEEE.

Naiwen Xue, Fei Xia, Fu-Dong Chiou, and Martha Palmer. 2005. The Penn Chinese TreeBank: Phrase structure annotation of a large corpus. *Natural language engineering*, 11(2):207–238.

N. Xue. 2003. Chinese word segmentation as character tagging. *Computational Linguistics and Chinese Language Processing*, 8(1):29–48.

Yaqin Yang and Nianwen Xue. 2012. Chinese comma disambiguation for discourse analysis. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 786–794. Association for Computational Linguistics.

Yue Zhang and Stephen Clark. 2007. Chinese segmentation with a word-based perceptron algorithm. In *ACL*.

Longkai Zhang, Houfeng Wang, Xu Sun, and Mairgup Mansur. 2013. Exploring representations from unlabeled data with co-training for Chinese word segmentation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*.

Xiaoqing Zheng, Hanyang Chen, and Tianyu Xu. 2013. Deep learning for chinese word segmentation and pos tagging. In *EMNLP*, pages 647–657.

Semi-supervised Chinese Word Segmentation based on Bilingual Information

Wei Chen
Institute of Automation,
Chinese Academy of Sciences,
Beijing, 100190, China
wei.chen.media@ia.ac.cn

Bo Xu
Institute of Automation,
Chinese Academy of Sciences,
Beijing, 100190, China
xubo@ia.ac.cn

Abstract

This paper presents a bilingual semi-supervised Chinese word segmentation (CWS) method that leverages the natural segmenting information of English sentences. The proposed method involves learning three levels of features, namely, character-level, phrase-level and sentence-level, provided by multiple sub-models. We use a sub-model of conditional random fields (CRF) to learn monolingual grammars, a sub-model based on character-based alignment to obtain explicit segmenting knowledge, and another sub-model based on transliteration similarity to detect out-of-vocabulary (OOV) words. Moreover, we propose a sub-model leveraging neural network to ensure the proper treatment of the semantic gap and a phrase-based translation sub-model to score the translation probability of the Chinese segmentation and its corresponding English sentences. A cascaded log-linear model is employed to combine these features to segment bilingual unlabeled data, the results of which are used to justify the original supervised CWS model. The evaluation shows that our method results in superior results compared with those of the state-of-the-art monolingual and bilingual semi-supervised models that have been reported in the literature.

1 Introduction

Chinese word segmentation (CWS) is generally accepted to be a necessary first step in most Chinese NLP tasks because Chinese sentences are written in continuous sequences of characters with no explicit delimiters (e.g., the spaces in English). Many studies have been conducted in this area, resulting in extensive investigation of the problem of CWS using machine learning techniques in recent years. However, the reliability of CWS that can be achieved using machine learning techniques relies heavily on the availability of a large amount of high-quality, manually segmented data. Because hand-labeling individual words and word boundaries is very difficult (Jiao et al., 2006), producing segmented Chinese texts is very time-consuming and expensive. Although a number of manually segmented datasets have been constructed by various organizations, it is not feasible to combine them into a single complete dataset because of their incompatibility due to the use of various segmenting standards. Thus, it is difficult to build a large-scale manually segmented corpus, and the resulting lack of such a corpus is detrimental to further enhancement of the accuracy of CWS.

To address the scarcity of manually segmented corpora, a number of semi-supervised CWS approaches have been intensively investigated in recent years. These approaches attempt to either learn the predicted label distribution (Jiao et al., 2006) or extract mutual information ((Liang et al., 2005); (Sun and Xu, 2011); (Zeng et al., 2013a)) from large-scale monolingual unlabeled data to update the baseline model (from manually segmented corpora). In addition to these techniques, several co-training approaches (Zeng et al., 2013b) using character-based and word-based models have also been employed. However, because monolingual unlabeled data contain limited natural segmenting information, in most semi-supervised methods, the objective function tends to be optimized based on the personal experience and knowledge of the researchers. This practice means that these methods can typically yield high performance in certain specialized domains, but they lack generalizability. In contrast with these methods, we propose to leverage bilingual unlabeled data, i.e., a Chinese-English corpus with sentence alignment. Because English sentences

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1207–1216,
Lisbon, Portugal, 17-21 September 2015.

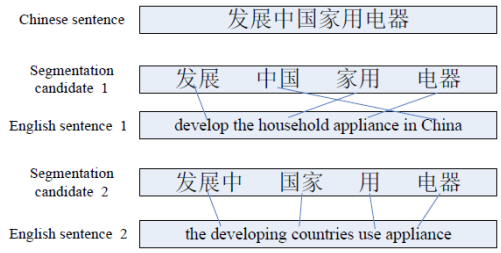


Figure 1: The examples of different segmentation on the same Chinese sentences guided by the English sentences

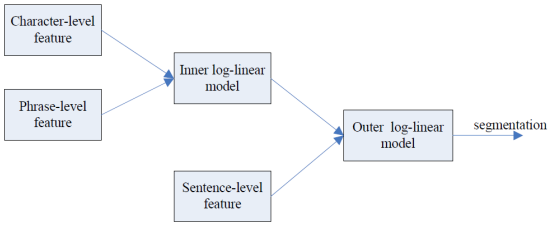


Figure 2: The structure of cascaded log-linear model with multilevel features

are naturally segmented, extracting information from a bilingual corpus is a much more objective task. As the example presented in Fig 1 shows, the English sentences that correspond to Chinese text can easily help guide better segmentation, and thus, the learning of segmenting information from bilingual data is a very promising approach.

In this paper, to obtain high-quality segmenting information from bilingual unlabeled data, we leverage multilevel features using the following steps: first, we integrate character-level features calculated using a conditional random field (CRF) model, which is used to capture the monolingual grammars. Then, we employ a statistical aligner to perform character-based alignment. Given the results of this character-based alignment, we apply several phrase-level features to extract explicit and implicit segmenting information: (1) we use two types of English-Chinese co-occurrence features (one-to-many and many-to-many) to learn the explicit segmenting information of the English sentences, (2) we use the transliteration similarity feature to detect out-of-vocabulary (OOV) words using a phrase-based translation model, and (3) we employ a neural network to calculate the semantic gap between the Chinese and English words to ensure that the Chinese segmentation follows the semantic meanings of the corresponding English sentences as closely as possible. Finally, we employ another phrase-based translation model to perform a sentence-level calculation of the translation probability of the Chinese segmentation and its corresponding English sentences. After obtaining these multilevel features, we normalize them and combine them into two log-linear models in a cascaded structure, which is illustrated in Fig 2. Finally, we segment the bilingual unlabeled data using the proposed model and use the segmentation of those data to justify the original supervised CWS model, which was trained on a standard manually segmented corpus.

In fact, several semi-supervised CWS methods have previously been proposed that leverage bilingual unlabeled data ((Xu et al., 2008); (Chang et al., 2008); (Ma and Way, 2009);(Chung et al., 2009); (Xi et al., 2012)). However, most were developed for statistical machine translation (SMT), causing them to focus on decreasing the perplexity of the bilingual data and the word alignment process rather than on achieving more accurate segmentation. These methods achieve significant improvement in SMT performance but are not very suitable for common NLP tasks because in many situations, they ignore the standard grammars to satisfy the needs of SMT. By contrast, we employ various types of features to capture both monolingual standard grammars and bilingual segmenting information, which allows our semi-supervised CWS model to be very efficient at other NLP tasks and endows it with higher generalizability.

Our evaluation also shows that our method significantly outperforms the state-of-the-art monolingual and bilingual semi-supervised approaches.

2 Related Work

First, we review related work on monolingual supervised and semi-supervised CWS methods. Then, we review bilingual semi-supervised CWS.

2.1 Monolingual Supervised and Semi-supervised CWS Methods

Considerable efforts have been made in the NLP community in the study of Chinese word segmentation. The most popular supervised approach treats word segmentation as a sequence labeling problem, as first proposed by (Xue et al., 2003). Most previous systems have addressed this task using linear statistical models with carefully designed features ((Peng et al., 2004); (Asahara et

al., 2005); (Zhang and Clark, 2007); (Zhao et al., 2010)). However, the primary shortcoming of these approaches is that they rely heavily on a large amount of labeled data, which is very time-consuming and expensive to produce. Thus, the scale of available manually labeled data has placed considerable limitations on the further enhancement of supervised CWS methods.

To address this problem, a number of semi-supervised CWS approaches have been intensively investigated in recent years. For example, (Sun and Xu, 2011) enhanced their segmentation results by interpolating statistics-based features derived from unlabeled data into a CRF model. (Zeng et al., 2013a) introduced a graph-based semi-supervised joint model of Chinese word segmentation and part-of-speech tagging and regularized the learning of a linear CRF model based on the label distributions derived from unlabeled data. However, because monolingual unlabeled data lack natural segmenting information, most previous semi-supervised CWS methods have required certain assumptions to be made regarding their objective functions based on the researchers' personal experiences. By contrast, we leverage bilingual unlabeled data that contain the natural segmentation that is present in English sentences and can therefore extract linguistic knowledge without any manual assumptions or bias.

2.2 Bilingual Semi-supervised CWS Methods

Some previous work ((Xu et al., 2008); (Chang et al., 2008); (Ma and Way, 2009);(Chung et al., 2009); (Xi et al., 2012)) has been performed on leveraging bilingual unlabeled data to achieve better segmentation, although most such studies have focused on statistical machine translation (SMT). These approaches leverage the mappings of individual English words to one or more consecutive Chinese characters either to construct a Chinese word dictionary for maximum-matching segmentation (Xu et al., 2004) or to form a labeled dataset for training a sequence labeling model (Peng et al., 2004). (Zeng et al., 2014) also used such mappings to bias a supervised segmentation model toward a better solution for SMT. However, because most of these approaches focus on SMT performance, they emphasize decreasing the perplexity of the bilingual data and word alignment rather than improving the CWS accuracy. Thus, they sometimes ignore the standard grammars during segmentation in favor of satisfying the needs of SMT, thereby causing these methods to be rather unsuitable for other NLP tasks. By contrast, we propose to use various types of features to capture syntactic and semantic information and a cascaded log-linear model to maintain balance between the monolingual grammars and the bilingual knowledge.

3 Multilevel Features

In this section, we describe the three levels of features used in our approach. We propose to use character-level features to capture monolingual grammars and phrase-level and sentence-level features to obtain bilingual segmenting information. Moreover, we describe a cascaded log-linear model by proposing both inner and outer log-linear models.

3.1 Character-level Feature

The conditional random field (CRF) (Lafferty et al., 2001) model was first used for CWS tasks by (Xue et al., 2003) who treated the CWS task as a sequence tagging problem and demonstrated this model's effectiveness in detecting OOV words.

In this paper, we score the character-level feature in the same manner defined by (Xue et al., 2003). For the jth character c_j in the sentence $c_1^J = c_1...c_J$, the score can be calculated as follows:

$$f_{CRF}(j) = \sum_k \lambda_k f_k(y_{j-1}, y_j, c_1^J, j) \quad (1)$$

where $f_k(y_{j-1}, y_j, c_1^J, j)$ is a feature function and λ_k is a learned weight that corresponds to the feature f_k. j represents the index of the character in the sentence. y_{j-1} and y_j represent the tags of the previous and current characters, respectively.

We do not introduce the CRF-based CWS model in detail here, but more information can be obtained from (Lafferty et al., 2001) and (Xue et al., 2003).

3.2 Phrase-level Features

In this section, we first describe English-Chinese character-based alignment. Then, we propose several phrase-level features to obtain explicit and implicit segmenting information from the character-based alignment. Finally, we describe the inner log-linear model that is used to combine the character-level and phrase-level features.

3.2.1 English-Chinese Character-based Alignment

To avoid introducing omissions and mistakes into the linguistic information in the initial segmentations of the bilingual data, we perform a statistical character-based alignment: First, every Chinese character in the bitexts is separated by white spaces so that individual characters are recognized as unique "words" or alignment targets. Then, they are associated with English words using a statistical word aligner.

By representing the English and Chinese sentences as $e_1^I = e_1 e_2 ... e_I$ and $c_1^J = c_1 c_2 ... c_J$, respectively, where e_i and c_j represent single elements of the sentences, we define their alignment as a_1^K, of which each element is a span $a_k =< s, t >$ and represents the alignment of the English word e_s with the Chinese character c_t. Then, the corpus of unlabeled bilingual data can be represented as the set of sentence tuples $<e_1^I, c_1^J, a_1^K>$

To obtain the character-based alignment, we employ an open-source toolkit Pialign[1] ((Neubig et al., 2011); (Neubig et al., 2012)) which uses Bayesian learning and inversion transduction grammars.

3.2.2 Features Obtained from the Character-based Alignment

Given the English-Chinese character-based alignment a_1^K, we extract several phrase-level features to optimize the segmentation. For the jth character in c_1^J, we assume that one of the segmentations of the substring c_1^j can be represented as $w_1^{N+1} = w_1 w_2 w_3 ... w_{N+1} = c_1^{j_1} c_{j_1+1}^{j_2} ... c_{j_N+1}^{j}$. Then, we calculate the scores of each Chinese word $w_n = c_{j_m}^{j_n}$ $(j_m = j_{n-1} + 1)$ in w_1^{N+1} using the following features.

English-Chinese One-to-Many Alignment

To evaluate the probability that a sequence of Chinese characters $c_{j_m}^{j_n} = c_{j_m} c_{j_m+1} ... c_{j_n}$ should be combined into a word w_n based on the corresponding English sentence, we integrate the feature of English-Chinese one-to-many alignment (one English word is aligned with multiple Chinese characters). First, for any English word e_i in e_1^I, the phrase tuple $< e_i, c_{j_m}^{j_n} >$ can be defined as an aligned One-to-Many phrase tuple if it satisfies the following conditions:

(1) $< i, j_m > \in a_1^K, < i, j_n > \in a_1^K$

(2) $\forall j' \notin [j_m, j_n], < i, j' > \notin a_1^K$

(3) $\forall i' \neq i,\ \forall j' \in [j_m, j_n], < i', j' > \notin a_1^K$

Then, for any phrase tuple $< \overline{e_i}, c_{j_m}^{j_n} >$ that satisfies these conditions, the span $< \overline{i}, j_m, j_n >$ is defined as a One-to-Many span and as a member of the set A_{One}.

Thus, for each span $< i, j_m, j_n >$, the One-to-Many score can be calculated as follows:

$$s(< i, j_m, j_n >) = \begin{cases} t(c_{j_m}^{j_n} | e_i) & if < i, j_m, j_n > \in A_{One} \\ 0 & else \end{cases} \quad (2)$$

where $t(c_{j_m}^{j_n} | e_i)$ represents the translation probability of the phrase tuple $c_{j_m}^{j_n} | e_i$.

Finally, the score for the feature of English-Chinese One-to-Many alignment for $w_n = c_{j_m}^{j_n}$ is derived as follows:

$$f_{One-to-Many}(n) = \underset{i \in [1,I]}{\operatorname{argmax}}\, s(< i, j_m, j_n >) \quad (3)$$

English-Chinese Many-to-Many Alignment

The second phrase-level feature, called English-Chinese Many-to-Many Alignment (multiple English words are aligned with multiple Chinese characters), is used to evaluate the probability that a space should be inserted between c_n and c_{n+1}. Similar to One-to-Many alignment, for any sequence of English words $e_{i_1}^{i_2}$ and the Chinese word $w_n = c_{j_m}^{j_n}$, the phrase tuple $< e_{i_1}^{i_2}, c_{j_1}^{j_n} >$ is defined as an aligned Many-to-Many phrase tuple if it satisfies the following conditions:

(1) $j_1 \leq j_m$, and j_1 is the beginning character of a word in w_1^n

(1) $< i_1, j_1 > \in a_1^K, < i_2, j_m > \in a_1^K$

(2) $\forall j' \notin [j_1, j_m], \forall i' \in [i_1, i_2], < i', j' > \notin a_1^K$

(3) $\forall j' \in [j_1, j_m], \forall i' \notin [i_1, i_2], < i', j' > \notin a_1^K$

Then, for any phrase tuple $< \overline{e_{i_1}^{i_2}}, c_{j_m}^{j_n} >$ that satisfies these conditions, the span $< \overline{i_1, i_2}, j_1, j_n >$ is defined as a Many-to-Many span and as a member of the set A_{Many}.

Thus, for each span $< i_1, i_2, j_1, j_n >$, the Many-to-Many score can be calculated as follows:

$$s(< i_1, i_2, j_1, j_n >) = \begin{cases} t(c_{j_1}^{j_n} | e_{i_1}^{i_2}) \\ \quad if < i_1, i_2, j_1, j_n > \in A_{Many} \\ 0 \quad else \end{cases} \quad (4)$$

where $t(c_{j_1}^{j_n} | e_{i_1}^{i_2})$ represents the translation probability of the phrase tuple $< e_{i_1}^{i_2}, c_{j_1}^{j_n} >$.

[1] http://www.phontron.com/pialign/

Finally, the score for the feature of English-Chinese Many-to-Many alignment for $w_n = c_{j_m}^{j_n}$ is derived as follows:

$$f_{Many-to-Many}(n) = \underset{i_1 \in [1,I] i_2 \in [i_1,I] j_1 \leq j_m}{\operatorname{argmax}} s(i_1, i_2, j_1, j_n) \tag{5}$$

Transliteration Feature

To account for named entities (NEs), which suffer from sparsity and thus make it difficult to calculate the probabilities discussed above, we introduce a transliteration feature to evaluate the similarities between the pronunciations of Chinese and English words because many NEs are translated via transliteration. To perform this task, we first introduce an initial NE dictionary and convert each dictionary item—for example, we convert "爱丽斯/Alice" into "ai l i s i/a l i c e" —by transforming the Chinese word into its pronunciation (represented by the function $F_{py}(\cdot)$) and splitting the English word into its constituent letters (represented by the function $F_{let}(\cdot)$). Then, we train two phrase-based translation models (Chinese-English and English-Chinese) on the data obtained from the converted NE dictionary.

Specifically, we apply two standard log-linear phrase-based SMT models. The GIZA++ aligner is adopted to obtain word alignments (Och and Ney, 2000) from the converted NE dictionary. The heuristic strategy of grow-diag-final-and (Koehn et al., 2003) is used to combine the bidirectional alignments to extract phrase translations and to reorder tables. A 5-gram language model with Kneser-Ney smoothing is trained using SRILM (Stolcke et al., 2002) on the target language. Moses (Koehn et al., 2007) is used as a decoder. Minimum error rate training (MERT) (Och et al., 2003) is applied to tune the feature parameters on the development dataset.

Given these two phrase-based translation models, we calculate each span $< i, j_m, j_n >$ in A_{One} for the Chinese word w_n using the following formula:

$$\begin{aligned} S_{tr}(< i, j_m, j_n >) = S_{ch-en}(< i, j_m, j_n >) \\ + S_{en-ch}(< i, j_m, j_n >) \end{aligned} \tag{6}$$

where $S_{ch-en}(<i, j_m, j_n>) = D_{Lev}(F_{let}e_i, PT_{ch-en}(F_{py}(c_{j_m}^{j_n})))$ means that the pronunciation conversion in the Chinese-English direction is performed as follows: First, the English word e_i is split into its constituent letters; Second, the sequence of Chinese characters $c_{j_m}^{j_n}$ is converted into its pronunciation; Third, this pronunciation is input into the Chinese-English phrase-based translation model, and the corresponding translation result is obtained; And finally, the Levenshtein distance between the English letters and the translation result is returned.

$S_{en-ch}(<i, j_m, j_n>)$ can be calculated in exactly the same way.

We set any span that does not belong to A_{One} to zero, and the transliteration feature score of a word $w_n = c_{j_m}^{j_n}$ is derived as follows:

$$f_{transliteration}(n) = \underset{i \in [1,I]}{\operatorname{argmax}} S_{tr}(< i, j_m, j_n >) \tag{7}$$

English-Chinese semantic gap feature

To guarantee that the semantic meanings of the Chinese segmentation match those of the corresponding English sentences as closely as possible, we propose to use a feature based on the English-Chinese semantic gap to ensure the retention of semantic meaning during the segmentation process.

First, we pre-train word embeddings using the open-source toolkit Word2Vec (Mikolov et al., 2013) on the Chinese (segmented using character-level features only) and English sentences separately, thereby obtaining the vocabularies V_{ch} and V_{en} and their corresponding embedding matrixes $L_{ch} \in R^{n \times |V_{ch}|}$ and $L_{en} \in R^{n \times |V_{en}|}$. Given a Chinese word w_n with an index i in the vocabulary, it is then straightforward to retrieve the word's vector representation via simple multiplication with a binary vector d that is equal to zero at all positions except that with index i:

$$X_i = L_{ch} d_i \in R^n \tag{8}$$

Because the word embeddings for the two languages (L_{ch} and L_{en}) are learned separately and located in different vector spaces, we suppose that a transformation exists between these two semantic embedding spaces. Thus, we collect all the One-to-Many phrase tuples $< e_1, c_{j_1}^{j_2} >$ that satisfy $e_1 \in V_{en}$ and $c_{j_1}^{j_2} \in V_{ch}$ from the entire corpus of bilingual data. Then, we insert the word embedding tuple of each One-to-Many phrase tuple into the set A_{embed}. Let us consider a word embedding tuple $< p_s, p_t >$ in A_{embed} as an example. We define a bidirectional semantic distance using the parameter θ as follows:

$$E_{sem}(p_s, p_t; \theta) = E_{sem}(p_s|p_t, \theta) + E_{sem}(p_t|p_s, \theta) \quad (9)$$

Here, $E_{sem}(p_s|p_t, \theta) = E_{sem}(p_t, f(W_{en}^{ch} p_s + b_{en}^{ch}))$ represents the transformation of p_s and is performed as follows: We first multiply a parameter matrix W_{en}^{ch} by p_s, and after adding a bias term b_{en}^{ch}, we apply an element-wise activation function $f = tanh(\cdot)$. Finally, we calculate their Euclidean distance:

$$E_{sem}(p_s|p_t, \theta) = \frac{1}{2}||p_t - f(W_{en}^{ch} p_s + b_{en}^{ch})||^2 \quad (10)$$

$E_{sem}(p_t|p_s, \theta)$ can be calculated in exactly the same way.

Given the definition of the semantic distance of each word-embedding tuple in A_{embed}, we wish to minimize the following objective function:

$$J = \sum_{<p_s, p_t> \in A_{embed}} E_{sem}(p_s, p_t; \theta) \quad (11)$$

We apply the Stochastic Gradient Descent (SGD) algorithm to optimize each parameter and ultimately obtain the optimized parameters θ^*.

Using θ^*, we can calculate the semantic gap for any possible span for w_n, such as $< i, j_m, j_n >$, as follows:

$$S_{gap}(< i, j_m, j_n >) = \begin{cases} \frac{1}{E_{sem}(p'_s|p'_t, \theta^*)} \\ \quad if \quad e_i \in V_{en} \quad c_{j_m}^{j_n} \in V_{ch} \\ \quad\quad\quad < i, j_m, j_n > \in A_{One} \\ 0 \quad else \end{cases} \quad (12)$$

where p'_s and p'_t are the word vector representation of e_i and $c_{j_m}^{j_n}$, respectively. Thus, the semantic gap feature score of the word $w_n = c_{j_m}^{j_n}$ is derived as follows:

$$f_{sem}(w_n) = \underset{i \in [1,I]}{\operatorname{argmax}} S_{gap}(< i, j_m, j_n >) \quad (13)$$

3.2.3 Normalization and the Inner Log-Linear Model

Because the output scores of each sub-model described above are not probabilistic and they vary by orders of magnitude, we must first normalize the output scores of each sub-model. After normalization, the scores have means and standard deviations of zero. We represent the normalization function by $Norm(\cdot)$.

Thus, for the substring c_1^j ($j \in [1, J)$) in c_1^J of the sentence tuple $<e_1^I, c_1^J, a_1^K>$, assuming that one of its candidate segmentations is $w_1^{N+1} = w_1 w_2 w_3 ... w_{N+1} = c_1^{j_1} c_{j_1+1}^{j_2} ... c_{j_N+1}^{j}$, the feature score of the inner log-linear model is derived as follows:

$$f_{inner} = \sum_{j' \in [1,j]} Norm(f_{CRF}(j')) + \lambda_1 \sum_{n \in [1, N+1]} (\sum_k Norm(f_k(n))) \quad (14)$$

where $f_k(n)$ represents the phrase-level features.

Then, we tune the weight λ_1 from 0 to 1 in equal increments of 0.1 to optimize its value.

3.3 Sentence-level Features

In this section, we describe the sentence-level features calculated using the phrase-based translation model and the outer log-linear model that is used to combine the sentence-level features with the features in the inner log-linear model.

3.3.1 Features Obtained from the Phrase-based Translation Model

Let us consider the last character c_J in c_1^J and assume that its candidate segmentation (according to the inner log-linear model only) is $w_1^{N+1} = w_1 w_2 w_3 ... w_{N+1}$. We now add a sentence-level feature to incorporate into the inner log-linear model. This sentence-level feature is obtained using a phrase-based translation model. We segment the Chinese sentences from the bilingual unlabeled data using character-level features only and train a phrase-based translation model on the bilingual data that is similar to the phrase-based translation model used for the transliteration features.

Unlike the usage of the phrase-based translation model in the case of the transliteration features, here, we input both the source and target sentences and achieve the output of translation probability. Thus, we perform a force decoding for the sentence tuple $<w_1^{N+1}, e_1^I>$ and obtain the set of decoding paths P(w_1^{N+1}), where each element acts as a decoding path that can translate w_1^{N+1} into e_1^I. Finally, we define the sentence-level feature score of $<w_1^{N+1}, e_1^I>$ as follows:

$$f_{sent}(w_1^{N+1}) = \underset{p(w_1^{N+1}) \in P(w_1^{N+1})}{\operatorname{argmax}} F_{trans}(p(w_1^{N+1})) \quad (15)$$

where $F_{trans}(\cdot)$ returns the translation score of the given decoding path based on the phrase-based translation model.

3.3.2 The Outer Log-Linear Model

Finally, we normalize the sentence-level features in a manner similar to that described previously and construct the outer log-linear model by combining the inner log-linear model and the sentence-level features as follows:

$$f_{outer} = f_{inner} + \lambda_2 Norm(f_{sent}(w_1^{N+1})) \quad (16)$$

Then, we also tune the weight λ_2 from 0 to 1 in equal increments of 0.1 to optimize its value.

3.3.3 Decoder

A traditional viterbi beam search procedure is applied in the decoder to seek the segmented sequence with the highest score. Given a sentence tuple $< e_1^I, c_1^J, a_1^K >$, the decoding procedure will proceed in a left-right fashion using a dynamic programming approach. At each position j in the sequence c_1^J, we maintain a vector of size N to store the top N candidate segmentations of subsequence c_1^j which are scored using the inner log-linear model ($j \in [1, J)$) or the outer log-linear model ($j = J$). Finally, we return the best segmentation.

4 Justifying the Original CWS Model

We justify the original CWS model (the CRF-based model trained on manually segmented data) using the new CRF model trained on the segmentation of unlabeled bilingual data. To avoid over-weakening the influence of the small-scale manually segmented data, we again utilized a log-linear model to balance their weights. The formula can be described as follows:

$$\begin{aligned} f_{new\_mono} = & \sum_{k_1} \lambda_{k_1} f_{k_1}(y_{j-1}, y_j, c_1^J, j) \\ & + \theta_3 \sum_{k_2} \lambda_{k_2} f_{k_2}(y_{j-1}, y_j, c_1^J, j) \end{aligned} \quad (17)$$

where θ_3 represents the weights of the second CRF model, which are set via minimum error rate training using the developing dataset, and λ_{k_i} (i =1, 2) represents the learned weights of the features of the CRF models.

5 The Datasets

In this paper, we conduct our experiments on the corpus of People's daily of 1998 (from January to June) as the standard (manually segmented) training corpus, the corpus of Bakeoff-2 CWS evaluation as the developing and testing dataset. As the corpus of Bakeoff-2 is made up of several sets provided by different organizations, we only select two sets whose segmenting standards are similar to the training corpus. For each set, we take 3000 sentences as the developing dataset and the others as the testing dataset. The statistics of every set and the standard training corpus are shown in Table 1.

Data Set	of sent.	of words
Training	120K	7.28M
AS	708K	5.45M
PKU	19K	1.1M

Table 1: Statistics of training and testing datasets

Moreover, the bilingual unlabeled data is formed by a large in-house Chinese-English parallel corpus (Tian et al., 2014). There are in total 2,215,000 Chinese-English sentence pairs crawled from online resources, concentrated in 5 different domains including laws, novels, spoken, news and miscellaneous.

6 Experiments

In our evaluation, the F-score was used as the accuracy measure. The precision p is defined as the percentage of words in the decoder output that are segmented correctly, and the recall r is the percentage of gold-standard output words that are correctly segmented by the decoder. The balanced F- score is calculated as 2pr/(p + r). We also report the recall of OOV words in our experiments. In the following, we refer to our methods as "SLBD" (segmenter leveraging bilingual data).

Initially, we evaluated state-of-the-art supervised CWS methods, i.e., those of (Peng et al., 2004) (Peng); (Asahara et al., 2005) (Asahara); (Zhang and Clark, 2007) ($Z\&C$); (Zhao et al., 2010) (Zhao), whose models are trained only on manually segmented data. Moreover, we also evaluated the performance of our sub-models by

methods	AS		PKU	
	F	OOV	F	OOV
Peng	91.6	52.5	91.1	59
Asahara	92.2	63.1	91.4	61.6
$Z\&C$	92.9	69.9	91.6	67.9
Zhao	93.1	72	92.3	60.6
character-level	92.3	58.6	92.9	60.8
Inner log-linear	95.9	78.8	96.1	81
Outer log-linear	96.7	80.8	97.1	85

Table 2: Word segmentation performance of SLBD and supervised CWS methods[%]

segmenting the bilingual unlabeled dataset using character-level features only, the inner log-linear model (which includes character-level and phrase-level features) and the outer log-linear model (the full SLBD approach). After applying these three segmentations using the different sub-models, we trained the new CRF models on the results of the three segmentations to justify the original CWS model. The evaluation results for the supervised CWS methods and the sub-models are presented in Table 2.

It can be seen that we achieved significant improvement in performance when we combined the character-level and phrase-level features in the inner log-linear model, demonstrating that the proposed phrase-level features can be used to efficiently obtain bilingual segmenting information. Moreover, the outer log-linear model achieves a further enhancement, thereby demonstrating that the sentence-level features can be used to effectively re-rank the candidate segmentations produced by the inner log-linear model.

Next, we compared the SLBD method with several state-of-the-art monolingual semi-supervised methods, including those of (Sun et al., 2012) (Sun); (Sun and Xu, 2011) ($S\&X$); (Zeng et al., 2013b) (Zeng). To ensure a fair comparison, we performed the evaluation in two steps. First, we input the entire bilingual unlabeled dataset into the SLBD method and input only the Chinese sentences from the bilingual unlabeled dataset into the other semi-supervised methods. Then, because the available monolingual unlabeled dataset was much larger than the bilingual unlabeled dataset in natural, we used the XIN_CMN portion of Chinese Gigaword 2.0 as an additional unlabeled dataset for the monolingual semi-supervised methods. which contains 204 million words, more than ten times

methods	Bilingual data		Monolingual data	
	F	OOV	F	OOV
Sun	93.9	63.1	94.6	67.9
$S\&X$	94.1	66	94.4	71
Zeng	94.0	64.5	94.8	63.2
SLBD	96.7	80.8	-	-

Table 3: Word segmentation performance of SLBD and other monolingual semi-supervised CWS methods[%]

methods	AS		PKU	
	F	OOV	F	OOV
Xu	92.8	70.5	92.1	66
Ma	93.1	73	92.6	71.1
Xi	90.2	63	90.9	67.2
Zeng2014	93.5	76	93.2	73.3
SLBD	96.7	80.8	97.1	85

Table 4: Word segmentation performance of SLBD and other bilingual semi-supervised CWS methods[%]

the number of words in the bilingual unlabeled dataset. The testing data was the set of AS only. The evaluation is summarized in Table 3.

The results demonstrate that either leveraging the same unlabeled data or providing a much larger unlabeled dataset for the monolingual semi-supervised methods, the SLBD method can significantly outperform the evaluated monolingual semi-supervised methods, which indicates that the segmenting information obtained using SLBD is much more efficient at optimizing segmentation.

Finally, we evaluated SLBD in comparison with other bilingual semi-supervised methods, including (Xu et al., 2008) (Xu); (Ma and Way, 2009) (Ma); (Xi et al., 2012) (Xi);(Zeng et al., 2014) (Zeng2014). The results presented in Table 4 indicate that SLBD demonstrates much stronger performance, primarily because these other methods were developed with a focus on SMT, which causes them to preferentially decrease the perplexity of the subsequent SMT steps rather than producing a highly accurate segmentation. In contrast to these methods, the SLBD method exhibits greater generalizability.

7 Conclusion

In this paper, we propose a cascaded log-linear model to involve learning three levels of bilingual linguistic features to semi-supervisedly learn a new CWS model. Different from other monolingual and bilingual semi-supervised approaches, we employ various types of features to capture both monolingual grammars and bilingual segmenting information, which allows our model to be very efficient at other NLP tasks and endows it with higher generalizability. The evaluation shows that our method significantly outperforms the state-of-the-art monolingual and bilingual semi-supervised approaches.

References

Masayuki Asahara, Kenta Fukuoka, Ai Azuma, ChooiLing Goh, Yotaro Watanabe, Yuji Matsumoto, and Takahashi Tsuzuki. 2005. *Combination of machine learning methods for optimum chinese word segmentation*. In Proceedings of The Fourth SIGHAN Workshop, pages 134-137.

Pi-Chuan Chang, Michel Galley, and Christopher D Manning. 2008. *Optimizing Chinese word segmentation for machine translation performance*. In Proceedings of WMT, pages 224-232. Association for Computational Linguistics.

Tagyoung Chung and Daniel Gildea. 2009. *Unsupervised Tokenization for Machine Translation*. Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2-Volume 2. Association for Computational Linguistics, 2009: 718-726.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. *Conditional random fields: Probabilistic models for segmenting and labeling sequence data*. In Proc. 18th International Conf. on Machine Learning.

Percy Liang. 2005. *Semi-supervised learning for natural language.*. Master' s thesis.

Yanjun Ma and Andy Way. 2009. *Bilingually motivated domain-adapted word segmentation for statistical machine translation*. Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics. Association for Computational Linguistics, 2009: 549-557.

Mikolov, Tomas, et al. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems. 2013.*

Graham Neubig, Taro Watanabe et al. 2011. *An Unsupervised Model for Joint Phrase Alignment and Extraction*. Proceedings of ACL 2011.

Graham Neubig, Taro Watanabe et al. 2012. *Machine Translation without Words through Substring Alignment*. Proceedings of ACL 2012.

Feng Jiao, Shaojun Wang, and Chi-Hoon Lee. 2006. Semi-supervised conditional random fields for improved sequence segmentation and labeling. In Proceedings of ACL, pages 209-216, Sydney, Australia.

Koehn, Philipp, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1. Association for Computational Linguistics, 2003.*

Koehn, Philipp, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th annual meeting of the ACL on interactive poster and demonstration sessions. Association for Computational Linguistics, 2007: 177-180.*

Franz Josef Och and Hermann Ney. 2000. Improved statistical alignment models. In *Proceedings of ACL*, pages 440-447.

Och, Franz Josef. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1. Association for Computational Linguistics, 2003: 160-167.*

Fuchun Peng, Fangfang Feng and Andrew McCallum. 2004. *Chinese segmentation and new word detection using conditional random fields*. Proceedings of the 20th international conference on Computational Linguistics.

Stolcke, Andreas. 2002. SRILM-an extensible language modeling toolkit. In *INTERSPEECH. 2002*

Weiwei Sun and Jia Xu. 2011. *Enhancing chinese word segmentation using unlabeled data.* . In Proceedings of EMNLP 2011.

Xu Sun, Houfeng Wang, Wenjie Li. 2012. Fast Online Training with Frequency-Adaptive Learning Rates for Chinese Word Segmentation and New Word Detection. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics, pages 253-262*

Tian, Liang, et al. 2014. UM-Corpus: a large English-Chinese parallel corpus for statistical machine translation. In *Proceedings of the 9th International Conference on Language Resources and Evaluation. ELRA Reykjavik, Iceland, 2014.*

Ning Xi, Guangchao Tang, Xinyu Dai, Shujian Huang, and Jiajun Chen. 2012. *Enhancing statistical machine translation with character alignment*. In Proceedings of ACL, pages 285-290. Association for Computational Linguistics.

Jia Xu, Richard Zens, and Hermann Ney. 2004. *Do we need Chinese word segmentation for statistical machine translation?*. Proceedings of the Third SIGHAN Workshop on Chinese Language Learning, pages 122 – 128.

Jia Xu, Jianfeng Gao, Kristina Toutanova, and Herman Ney. 2008. *Bayesian Semi-Supervised Chinese Word Segmentation for Statistical Machine Translation*. Proceedings of Coling 2008.

Nianwen Xue. 2003. *Chinese word segmentation as character tagging*. Computational Linguistics and Chinese Language Processing, pages 29 – 48.

Zhang and Clark. 2007. *Chinese segmentation with a word-based perceptron algorithm.*. Proceedings of ACL 2007.

Hai Zhao, Chang-Ning Huang, Mu Li, and Bao-Liang Lu. 2010. *A unified character-based tagging framework for chinese word segmentation*. ACM Transactions on Asian Language Information Processing, 9(2):5:1-5:32, June.

Xiaodong Zeng, Derek F. Wong, Lidia S. Chao and Isabel Trancoso. 2013a. *Graph-based Semi-Supervised Model for Joint Chinese Word Segmentation and Part-of-Speech Tagging*. Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics.

Xiaodong Zeng, Derek F. Wong, Lidia S. Chao and Isabel Trancoso 2013b. *Co-regularizing character-based and word-based models for semi-supervised Chinese word segmentation*. Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics.

Xiaodong Zeng, Derek F. Wong et al. 2014. *Toward Better Chinese Word Segmentation for SMT via Bilingual Constraints*. Proceedings of the Annual Meeting of the Association for Computational Linguistics. Association for Computational Linguistics. 2014: 1360-1369.

Hierarchical Back-off Modeling of Hiero Grammar based on Non-parametric Bayesian Model

Hidetaka Kamigaito[1] kamigaito@lr.pi.titech.ac.jp
Taro Watanabe[2] tarow@google.com
Hiroya Takamura[1] takamura@pi.titech.ac.jp
Manabu Okumura[1] oku@pi.titech.ac.jp
Eiichiro Sumita[3] eiichiro.sumita@nict.go.jp
[1]Tokyo Institute of Technology [2]Google Japan Inc.
[3]National Institute of Information and Communication Technology

Abstract

In hierarchical phrase-based machine translation, a rule table is automatically learned by heuristically extracting synchronous rules from a parallel corpus. As a result, spuriously many rules are extracted which may be composed of various incorrect rules. The larger rule table incurs more run time for decoding and may result in lower translation quality. To resolve the problems, we propose a hierarchical back-off model for Hiero grammar, an instance of a synchronous context free grammar (SCFG), on the basis of the hierarchical Pitman-Yor process. The model can extract a compact rule and phrase table without resorting to any heuristics by hierarchically backing off to smaller phrases under SCFG. Inference is efficiently carried out using two-step synchronous parsing of Xiao et al., (2012) combined with slice sampling. In our experiments, the proposed model achieved higher or at least comparable translation quality against a previous Bayesian model on various language pairs; German/French/Spanish/Japanese-English. When compared against heuristic models, our model achieved comparable translation quality on a full size German-English language pair in Europarl v7 corpus with significantly smaller grammar size; less than 10% of that for heuristic model.

1 Introduction

Hierarchical phrase-based statistical machine translation (HPBSMT) (Chiang, 2007) is a popular alternative to phrase-based SMT (PBSMT), in which synchronous context free grammar (SCFG) is used as the basis of the machine translation model. With HPBSMT, a restricted form of an SCFG, i.e., Hiero grammar, is usually used and is especially suited for linguistically divergent language pairs, such as Japanese and English. However, a rule table, i.e., a synchronous grammar, may be composed of spuriously many rules with potential errors especially when it was automatically acquired from a parallel corpus. As a result, the increase in the rule table incurs a large amount of time for decoding and may result in lower translation quality.

Pruning a rule table either on the basis of significance test (Johnson et al., 2007) or entropy (Ling et al., 2012; Zens et al., 2012) used in PBSMT can be easily applied for HPBSMT. However, these methods still rely on a heuristically determined threshold parameter. Bayesian SCFG methods (Blunsom et al., 2009) solve the spurious rule extraction problem by directly inducing a compact rule table from a parallel corpus on the basis of a non-parametric Bayesian model without any heuristics. Training for Bayesian SCFG models infers a derivation tree for each training instance, which demands the time complexity of $O(|\mathbf{f}|^3|\mathbf{e}|^3)$ when we use dynamic programming SCFG biparsing (Wu, 1997). Gibbs sampling without biparsing (Levenberg et al., 2012) can avoid this problem, though the induced derivation trees may strongly depend on initial derivation trees. Even though we may learn a statistically sound model on the basis of non-parametric Bayesian methods, current approaches for an SCFG still rely on exhaustive heuristic rule extraction from the word-alignment decided by derivation trees since the learned models cannot handle rules and phrases of various granularities.

We propose a model on the basis of the previous work on the non-parametric Inversion Transduction Grammar (ITG) model (Neubig et al., 2011) wherein phrases of various granularities are

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1217–1227,
Lisbon, Portugal, 17-21 September 2015.

learned in a hierarchical back-off process. We extend it by incorporating arbitrary Hiero rules when backing off to smaller spans. For efficient inference, we use a fast two-step bi-parsing approach (Xiao et al., 2012) which basically runs in a time complexity of $O(|\mathbf{f}|^3)$. Slice sampling for an SCFG (Blunsom and Cohn, 2010) is used for efficiently sampling a derivation tree from a reduced space of possible derivations.

Our model achieved higher or at least comparable BLEU scores against the previous Bayesian SCFG model on language pairs; German/French/Spanish-English in the News-Commentary corpus, and Japanese-English in the NTCIR10 corpus. When compared against heuristically extracted model through the GIZA++ pipeline, our model achieved comparable score on a full size Germany-English language pair in Europarl v7 corpus with significantly less grammar size.

2 Related Work

Various criteria have been proposed to prune a phrase table without decreasing translation quality, e.g., Fisher's exact test (Johnson et al., 2007) or relative entropy (Ling et al., 2012; Zens et al., 2012). Although those methods are easily applied for pruning a rule table, they heavily rely on the heuristically determined threshold parameter to trade off the translation quality and decoding speed of an MT system.

Previously, EM-algorithm based generative models were exploited for generating compact phrase and rule tables. Joint phrase alignment model (Marcu and Wong, 2002) can directly express many-to-many word aligments without heuristic phrase extraction. DeNero et al. (2006) proposed IBM Model 3 based many-to-many alignment model. Rule arithmetic method (Cmejrek and Zhou, 2010) can generate SCFG rules by combining other rule pairs through an inside-outside algorithm. However, those previous attempts were restricted in that the rules and phrases were induced by heuristic combination.

Bayesian SCFG models can induce a compact model by incorporating sophisticated non-parametric Bayesian models for an SCFG, such as a dirichlet process (DeNero et al., 2008; Blunsom et al., 2009; Chung et al., 2014) or Pitman-Yor process (Levenberg et al., 2012; Peng and Gildea, 2014). A model is learned by sampling derivation trees in a parallel corpus and by accumulating the rules in the sampled trees into the model. Due to the $O(|\mathbf{f}|^3|\mathbf{e}|^3)$ time complexity for bi-parsing a bilingual sentence, previous studies relied on bi-parsing at the initialization step, and conducted Gibbs sampling by local operators (Blunsom et al., 2009; Levenberg et al., 2012) or sampling on fixed word alignments (Chung et al., 2014; Peng and Gildea, 2014). As a result, the inference can easily result in local optimum, wherein induced derivation trees may strongly depend on the initial trees.

Xiao et al. (2012) proposed a two-step approach for bi-parsing a bilingual sentence in $O(|\mathbf{f}|^3)$ in the context of inducing SCFG rules discriminatively; however, their approach violates the detailed balance due to its heuristic k-best pruning. Blunsom and Cohn (2010) proposed a slice sampling for an SCFG, in the same manner as that for Infinite Hiden Markov Model (iHMM) (Van Gael et al., 2008), which can efficiently prune a space of possible derivations on the basis of dynamic programming. Although slice sampling can prune spans without violating the detailed balance, its time complexity of $O(|\mathbf{f}|^3|\mathbf{e}|^3)$ is still impractical for a large-scale experiment. We efficiently carried out large-scale experiments on the basis of the two-step bi-parsing of Xiao et al. combined with slice sampling of Blunsom and Cohn.

After learning a Bayesian model, it is not directly used in a decoder since it is composed of only minimum rules without considering phrases of various granularities. As a consequence, it is a standard practice to obtain word alignment from derivation trees and to extract SCFG rules heuristically from the word-aligned data (Cohn and Haffari, 2013). The work by Neubig et al. (2011) was the first attempt to directly use the learned model on the basis of a Bayesian ITG in which phrases of many granularities were encoded in the model by employing a hierarchical back-off procedure. Our work is strongly motivated by their work, but greatly differs in that our model can incorporate many arbitrary Hiero rules, not limited to ITG-style binary branching rules.

3 Model

We use Hiero grammar (Chiang, 2007), an instance of an SCFG, which is defined as a context-free grammar for two languages. Let Σ denote a set of terminal symbols in the source language, Δ a set of terminal symbols in the target language,

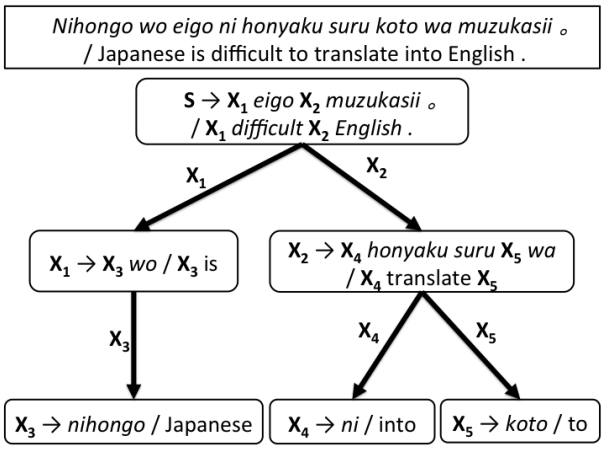


Figure 1: Derivation tree generated from Bayesian SCFG model

V a set of non-terminal symbols, S a start symbol and R a set of rewrite rules. An SCFG is denoted as a tuple of $\langle \Sigma, \Delta, V, S, R \rangle$. Each rewrite rule in R is represented as $X \rightarrow \langle \alpha / \beta \rangle$ in which α is a string of non-terminals and source side terminals $(V \cup \Sigma)^*$ and β is a string of non-terminals and target side terminals $(V \cup \Delta)^*$. An example derivation in an SCFG for the sentence pair "*nihongo wo eigo ni honyaku suru koto wa muzukasii* 。 / Japanese is difficult to translate into English ." is represented as follows:
$S \rightarrow X_1$ *eigo* X_2 *muzukasii* 。 / X_1 difficult X_2 English .
$X_1 \rightarrow X_3$ *wo* / X_3 is
$X_2 \rightarrow X_4$ *honyaku suru* X_5 *wa* / X_4 translate X_5
$X_3 \rightarrow$ *nihongo* / Japanese
$X_4 \rightarrow$ *ni* / into
$X_5 \rightarrow$ *koto* / to .

A Hiero grammar has additional constraints over a general SCFG; the number of terminal symbols in each rule for both source and target sides is limited to 5. Each rule may contain at most two non-terminal symbols; adjacent non-terminal symbols in the source side are prohibited. For details, refer to (Chiang, 2007).

3.1 Bayesian SCFG Models

Previous Bayesian SCFG Models, for instance a model proposed by Levenberg et al. (2012), are based on the Pitman-Yor process (Pitman and Yor, 1997) and learn SCFG rules by sampling a derivation tree for each bilingual sentence. Figure 1 shows an example derivation tree for our running example sentence pair under the model. The generative process is represented as follows:

$$\begin{aligned} G_X &\sim P_{rule}(d_r, \theta_r, G_{r_0}), \\ X \rightarrow \langle \alpha / \beta \rangle &\sim G_X, \end{aligned} \tag{1}$$

where G_X is a derivation tree and $P_{rule}(d_r, \theta_r, G_{r_0})$ is a Pitman-Yor process (Pitman and Yor, 1997), which is a generalization of a Dirichlet process parametrized by a discount parameter d_r, a strength parameter θ_r and a base measure G_{r_0}. The output probability of a Pitman-Yor process obeys the power-law distribution with the discount parameter, which is very common in standard NLP tasks.

The probability that a rule r_k is drawn from a model $P_{rule}(d_r, \theta_r, G_{r_0})$ is determined by a Chinese restaurant process which is decomposed into two probability distributions. If r_k already exists in a table, we draw r_k with probability

$$\frac{c_k - d_r \cdot |\varphi_{r_k}|}{\theta_r + n_r}, \tag{2}$$

where c_k is the number of customers of r_k, n_r is the number of all customers and φ_{r_k} is a number of r_k's tables. On the other hand, if r_k is a new rule, we draw r_k with probability

$$\frac{\theta_r + d_r \cdot |\varphi_r|}{\theta_r + n_r} \cdot G_{r_0}, \tag{3}$$

where $|\varphi_r|$ is the number of tables in the model.

3.2 Hierarchical Back-off Model

In the previous models, the generative process is represented as a rewrite process starting from the symbol S, which can incorporate only minimal rules. Following Neubig et al. (2011), our model reverses the process by recursively backing off to smaller phrase pairs as shown in Figure 2. First, our model attempts to generate a phrase pair, i.e., a sentence pair, as a derivation tree. If the model successfully generates the phrase pair, we will finish the generation process. Otherwise, a Hiero rule is generated to fallback to smaller spans represented in each non-terminal symbol X in the rule. Then, each phrase pair corresponding to each smaller span is recursively generated through our model. In Figure 2, a phrase pair with "nil" indicates those not in our model; therefore the phrase pair is forced to back-off either by generating a new phrase pair from a base measure (base) or by falling back to smaller phrases using a Hiero rule (back-off). The recursive procedure is done until

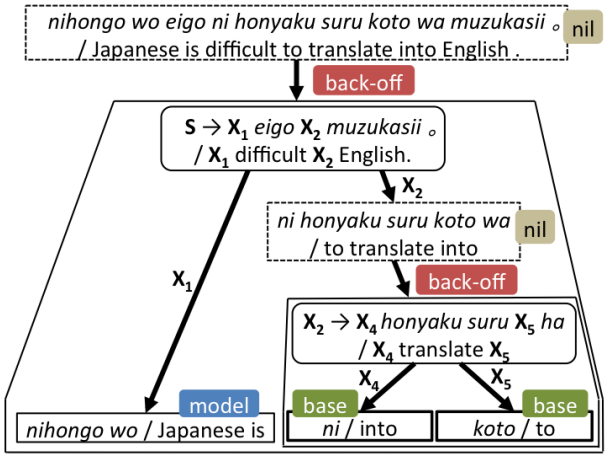


Figure 2: Derivation tree generated from the hierarchical back-off model

we reach phrase pairs which are generated without any back-offs. Let a discount parameter be d_p, a strength parameter be θ_p, and a base measure be G_{p_0}. More formally, the generative process is represented as follows:

$$\begin{aligned} G_X &\sim P_{rule}(d_r, \theta_r, G_{phrase}), \\ G_{phrase} &\sim P_{phrase}(d_p, \theta_p, G_X), \\ X \rightarrow \langle \mathbf{s}/\mathbf{t} \rangle &\sim G_{phrase}, \\ X \rightarrow \langle \alpha/\beta \rangle &\sim G_X, \end{aligned} \quad (4)$$

where $\mathbf{s}$ is source side terminals and $\mathbf{t}$ is target side terminals in phrase pair $\langle \mathbf{s}/\mathbf{t} \rangle$. P_{phrase} is composed of three states, i.e., model, back-off, and base, and follows a hierarchical Pitman-Yor process (Teh, 2006).

model: We draw a phrase pair $\langle \mathbf{s}/\mathbf{t} \rangle$ with the probability similar to Equation (2):

$$\frac{c_k - d_p \cdot |\varphi_{p_k}|}{\theta_p + n_p}, \quad (5)$$

where c_k is the numbers of customers of a phrase pair p_k and n_p is the number of all customers Note that this state is reachable when the phrase pair $\langle \mathbf{s}/\mathbf{t} \rangle$ exists in the model in the same manner as Equation (2).

back-off: We will back off to smaller phrases using a rule generated by P_{rule} as follows:

$$\begin{aligned} &\frac{\theta_p + d_p \cdot |\varphi_p|}{\theta_p + n_p} \cdot \frac{c_{back} + \gamma_b \cdot G_b}{c_{back} + c_{base} + \gamma_b} \\ &\cdot P_{rule}(d_r, \theta_r, G_{phrase}) \\ &\cdot \prod_{X \in \langle \alpha/\beta \rangle} P_{phrase}(d_p, \theta_p, G_X), \end{aligned} \quad (6)$$

where c_{back} and c_{base} are the number of customers sampled from the back-off and base phrases, respectively, with a base measure G_b and hyper-parameter γ_b. We use a uniform distribution for $G_b = 0.5$ since we consider only two states, back-off and base. Unlike the model state, P_{phrase} may reach this state even when a phrase pair is not in the model. The phrase pair is backed-off to smaller phrase pairs using P_{phrase} through the non-terminals in the generated rule $X \in \langle \alpha/\beta \rangle$.

base: As an alternative to the back-off state, we may reach the base state which follows the probability distribution on the basis of the base measure G_{p_0},

$$\frac{\theta_p + d_p \cdot |\varphi_p|}{\theta_p + n_p} \cdot \frac{c_{base} + \gamma_b \cdot G_b}{c_{back} + c_{base} + \gamma_b} \cdot G_{p_0}. \quad (7)$$

In summary, $P_{phrase}(d_p, \theta_p, G_X)$ is defined as a joint probability of Equations (5) through (7).

3.3 Base Measure

Similar to Levenberg et al. (2012), the base measure for rule probability G_{r_0} is composed of four generative processes. First, a number of symbols in a source side of a rule $|\alpha|$ is generated from a Poisson distribution, i.e., $|\alpha| \sim Poisson(0.1)$. Let $t(x)$ denote a function that returns terminals from a string x. The number of target side terminal symbols $|t(\beta)|$ is also generated from a Poisson distribution and represented as $|t(\beta)| \sim Poisson(\alpha + \lambda_0)$[1]. The type of symbol α_i in the source side, $type_i$, either terminal or non-terminal symbol, is determined by $type_i \sim Bernoulli(\phi^{|\alpha|})$ where ϕ is a hyper-parameter taking $0 < \phi < 1$. $\phi^{|\alpha|}$ is based on an intuition that shorter rules should be relatively more likely to contain terminal symbols than longer rules. Source and target terminal symbol pair $\langle t(\alpha), t(\beta) \rangle$ are generated from the geometric means of two directional IBM Model 1 word alignment probabilities and monolingual unigram probabilities for two languages, and represented as:

$$\begin{aligned} \langle t(\alpha), t(\beta) \rangle \sim (&P_{uni}(t(\alpha)) P_{\overrightarrow{M1}}(t(\alpha), t(\beta)) \cdot \\ &P_{uni}(t(\beta)) P_{\overleftarrow{M1}}(t(\alpha), t(\beta)))^{\frac{1}{2}}. \end{aligned} \quad (8)$$

When the $t(\alpha)$ or $t(\beta)$ is empty, we use the constant 0.01 instead of the Model1 probabilities.

[1]Note that λ_0 is a small constant for the input distribution greater than zero.

The base measure for phrases G_{p_0} is composed of three generative processes, in a similar manner as Levenberg et al. (2012), the number of terminal symbols in a phrase pair in the source side, $|\mathbf{s}|$, is generated from a Poisson distribution $|\mathbf{s}| \sim Poisson(0.1)$. The length for the target side $|\mathbf{t}|$ is generated in the same manner as the source side of the phrase pair. The alignments between $\mathbf{s}$ and $\mathbf{t}$ are also generated in the same manner as those for the base measure in a rule.

4 Inference

In inference, we use a sentence-wise block sampling of Blunsom and Cohn (2010), which has a better convergence property when compared with a step-wise Gibbs sampling. We repeat the following steps given a sentence pair.

1. Decrement customers of the rules and phrase pairs used in the current derivation for the sentence pair.
2. Bi-parse the sentence pair in a bottom up manner.
3. Sample a new derivation tree in a top-down manner.
4. Increment customers of the rules and phrase pairs in the sampled derivation tree.

The most time-consuming step during the inference procedure is bi-parsing of a sentence pair which essentially takes $O(|\mathbf{f}|^3|\mathbf{e}|^3)$ time using a bottom up dynamic programming algorithm (Wu, 1997). When a span is very large, it can easily suffer combinatorial explosion. To avoid this problem, we use a two-step slice sampling by performing the two-step bi-parsing of Xiao et al. (2012) and by pruning possible derivation space (Blunsom and Cohn, 2010) in each step (Algorithm 1). From lines 1 to 7, a set of word alignment is enumerated and put into $cube_a$. In addition to the arbitrary word alignment of $source_i$ to $target_j$, null word alignment is also merged into $cube_a$ (line 5). Note that word alignment considered in the algorithm is restricted to one-to-many. The set of word alignments in $cube_a$ is pruned and added to the $chart_a$ by SliceSampling. From lines 8 to 15, all possible phrases and rules for each span constrained by the pruned word alignment are enumerated and temporally stored into $cube$. The phrases and rules in $cube$ are pruned by SliceSampling and the remainders are added to $chart$. The

Algorithm 1 Two-step slice sampling

```
1:  for i ← 1, ··· , |source| do
2:    for j ← 1, ··· , |target| do
3:      cube_a ← {souce_i, target_j}
4:    end for
5:    cube_a ← {souce_i, null}
6:    chart_a ← SliceSampling(cube_a)
7:    clear cube_a
8:  end for
9:  for h ← 1, ··· , |source| do
10:   for all the i, j s.t j − i = h do
11:     for inferable rule, phrase from the sub-
        spans of [i, j] of all charts do
12:       cube ← rule, phrase
13:     end for
14:     chart ← SliceSampling(cube)
15:     clear cube
16:   end for
17: end for
```

time complexity for the word alignment enumeration from lines 1 to 7 is $O(|f||e|)$ and that for the phrase and rule enumeration from lines 8 to 15 is $O(|f|^3)$.

The key difference to the slice sampling of Blunsom and Cohn (2010) lies in lines 6 and 3 of Algorithm 1. Let $\mathbf{d}$ denote a set of derivation trees d and $\mathbf{u}$ be a set of slice variables u. In slice sampling, we prune the rules $\mathbf{r_{sp}}$ in each source span sp based on a slice variable u_{sp} corresponding to that sp. After pruning, we sample trees from the pruned space of $\mathbf{r}$. The above process is formally represented as:

$$\begin{aligned} \mathbf{u} &\sim P(\mathbf{u}|\mathbf{d}), \\ \mathbf{d} &\sim P(\mathbf{d}|\mathbf{u}), \end{aligned} \tag{9}$$

where $P(\mathbf{d}|\mathbf{u})$ is computed through sampling in a top-down manner after parsing in a bottom-up manner with Algorithm 1, and is equal to $\prod_d P(d|\mathbf{u})$. The probability $P(\mathbf{u}|\mathbf{d})$ is equal to $\prod_{sp} P(u_{sp}|\mathbf{d})$. Let r^*_{sp} denote a currently adopted rule in the span sp and $P(u_{sp}|d)$ be defined using a pruning score $Score(r^*_{sp})$ as follows:

$$Score(r_{sp_i}) = Inside(r_{sp_i}) \cdot Future(r_{sp_i}), \tag{10}$$

where $Inside(r_{sp})$ and $Future(r_{sp})$ are inside and outside probabilities for sp, respectively. Let $\mathbf{s}_{r_{sp}}$ denote a set of source side words in r_{sp}, $\mathbf{t}_{r_{sp}}$ a set of target side words in r_{sp}, $\overline{\mathbf{s}_{sp}}$ a set of words in a source sentence without $\mathbf{s}_{r_{sp}}$ and $\overline{\mathbf{t}_{sp}}$, a set of

words in a target sentence without $\mathbf{t}_{r_{sp}}$. By using IBM Model 1 probabilities in two directions, $Inside(r_{sp})$ is calculated by

$$(P_{\overrightarrow{M1}}(\mathbf{s}_{sp}, \mathbf{t}_{sp}) \cdot P_{\overleftarrow{M1}}(\mathbf{s}_{sp}, \mathbf{t}_{sp}))^{\frac{1}{2}}. \quad (11)$$

We use IBM Model1 outside probability for future score $Future(r_{sp})$. Similarly, the future score $Future(r_{sp})$ is computed using the two directional models:

$$(P_{\overrightarrow{M1}}(\overline{\mathbf{s}_{sp}}, \overline{\mathbf{t}_{sp}}) \cdot P_{\overleftarrow{M1}}(\overline{\mathbf{s}_{sp}}.\overline{\mathbf{t}_{sp}}))^{\frac{1}{2}}. \quad (12)$$

When sp is used in the current derivation $\mathbf{d}$, slice variable u_{sp} is sampled from a uniform distribution[2]:

$$P(u_{sp}|d) = \frac{\mathbb{I}(u_{sp} < Score(r^*_{sp}))}{Score(r^*_{sp})}, \quad (13)$$

otherwise, u_{sp} is sampled from a beta distribution if sp is not in the current derivation $\mathbf{d}$:

$$P(u_{sp}|d) = Beta(u_{sp}; a, 1.0), \quad (14)$$

where $a < 1$ is a parameter for the beta distribution. If the $Score(r_{sp_i})$ is less than u_{sp}, we prune the r_{sp_i} from $cube$. Similar to Blunsom and Cohn (2010), if the span sp is not in the current derivation, the rules with low probability are pruned according to Equation (14). Let r^d_{sp} denotes a rule in d with span sp, $P(d|\mathbf{u})$ is calculated by:

$$\prod_{sp \in d} \frac{P(r^d_{sp})}{\sum_{r_j \in \mathbf{r_{sp}}} P(r_j)\mathbb{I}(u_{sp} < Score(r_j))}. \quad (15)$$

In our experiments discussed in Section 6, slice sampling parameter a was set to 0.02 when incorporating the future score of Equation (12). In contrast, we used $a = 0.1$ when performing slice sampling without the future score. We empirically found that setting a lower value for a led to slower progress in learning due to a combinatorial explosion when inferencing a derivation for each sentence pair.

In the beginning of training, we do not have any derivation trees for given training data, although the derivation trees are required for estimating parameters for Bayesian models. We use the two-step parsing for generating initial derivation trees from only base measures. The k-best pruning is conducted against the score denoted by the equation 10 , which is very similar to Xiao et al. (2012).[3]

For faster bi-parsing, we run sampling in parallel in the same way as Zhao and Huang (2013), in which bi-parsing is performed in parallel among the bilingual sentences in a mini-batch. The updates to the model are synchronized by incrementing and decrementing customers for the bilingual sentences in the mini-batch. Note that the bi-parsing for each mini-batch is conducted on the fixed model parameters after the synchronised parameter updates.

In addition to the model parameters, hyperparameters are re-sampled after each training iteration following the discount and strength hyperparameter resampling in a hierarchical Pitman-Yor process (Teh, 2006). In particular, we resample $\langle d_p, \theta_p \rangle$, the pair of discount and strength parameters for phrases from a distribution:

$$\frac{[\theta_p]^{|\varphi_p|}_{d_p}}{[\theta_p]^{n_p}_1} \prod_{\langle \mathbf{s},\mathbf{t} \rangle} \prod_{k=1}^{|\varphi_p|} [1 - d_p]^{(c_{\langle \mathbf{s},\mathbf{t} \rangle} - 1)}_1 \quad (16)$$

where $[\,]$ denotes a generalized Pochhammer symbol, and $c_{\langle \mathbf{s},\mathbf{t} \rangle}$ the number of customers of phrase pair $\langle \mathbf{s}, \mathbf{t} \rangle$. We resample the pair $\langle d_r, \theta_r \rangle$ in the same way as $\langle d_p, \theta_p \rangle$. The hyperparameter γ_b is resampled from distribution:

$$\frac{(c_{back} + \gamma_b \cdot G_b)(c_{base} + \gamma_b \cdot G_b)}{(c_{back} + c_{base} + \gamma_b)^2}, \quad (17)$$

where ϕ, used in the generative process for either terminal or non-terminal symbol $type_i \sim Bernoulli(\phi^\alpha)$, is resampled from a distribution:

$$\prod_{\langle \alpha/\beta \rangle \in Base} Bernoulli(\phi^{|\alpha|})^{c_{\langle \alpha/\beta \rangle}}, \quad (18)$$

where $c_{\langle \alpha/\beta \rangle}$ denotes the number of customers of rule $\langle \alpha/\beta \rangle$, and $Base$ denotes a set of rules generated from the base measure. All the hyperparameters are inferred by slice sampling (Neal, 2003).

5 Extraction of Translation Model

In the previous work on Bayesian approaches (Blunsom and Cohn, 2010; Levenberg et al., 2012), it is a standard practice to heuristically extract rules and phrase pairs from the word alignment derived from the derivation trees sampled

[2] $\mathbb{I}(\cdot)$ is a function returns 1 if the condition is satisfied and 0 otherwise

[3] Note that we use $k = 30$ for k-best pruning.

from the Bayesian models. Instead of the heuristic method, we directly extract rules and phrase pairs from the learned models which are represented as Chinese restaurant tables. To limit grammar size, we include only phrase pairs that are selected at least once in the sample. During this extraction process, we limit the source or target terminal symbol size of phrase pairs to 5.

For each extracted rule or phase pair, we compute a set of feature scores used for a HPBSMT decoder; a weighted combination of multiple features is necessary in SMT since the model learned from training data may not fit well to translate an unseen test data (Och, 2003). We use the following six features; the joint model probability P_{model} is calculated by Equation (2) for rules and by Equation (5) for phrase pairs. The joint posterior probability $P_{posterior}(f, e)$ is estimated from the posterior probabilities for every rule and phrase pair in derivation trees through relative count estimation, motivated by Neubig et al. (2011) [4]. The joint posterior probability is considered as an approximation for those back-off scores. The conditional model probabilities in two directions, $P_{model}(f|e)$ and $P_{model}(e|f)$, are estimated by marginalizing the joint probability $P_{model}(f, e)$:

$$P_{model}(f|e) = \frac{P_{model}(f, e)}{\sum_{f'} P_{model}(f', e)}. \quad (19)$$

The inverse direction $P_{model}(e|f)$ is estimated, similarly. The lexical probabilities in two directions, $P_{lex}(f|e)$ and $P_{lex}(e|f)$, are scored by IBM Model probabilities between the source and target terminal symbols in rules and phrase pairs. In addition to the above features, we use Word penalty for each rule and phrase pair used in the cdec decoder (Dyer et al., 2010).

As indicated in previous studies (Koehn et al., 2003; DeNero et al., 2006), the translation quality of generative models is lower than that of models with heuristically extracted rules and phrase pairs. DeNero et al. (2006) reported that considering multiple phrase boundaries is important for improving translation quality. The generative models, in particular Bayesian models, are strict in determining phrase boundaries since their models are usually estimated from sampled derivations. As a result, translation quality is poorer when compared with a model estimated using a heuristic method. The Hiero grammar severely suffers from the phrase granularity problem and can overfit to the training data due to the flexibility of the rules.

To alleviate this problem, Neubig et al. (2011) combined the derivation trees across training iterations by averaging the features for each rule and phrase pair. During the sampling process, each training iteration draws a different derivation tree for each sentence pair, and the combination of those different derivation trees can provide multiple possible phrase boundaries to the model. Inspired by the averaging over the models from different iterations, we combine them as a part of a sampling process; we treat the derivation trees acquired from different iterations as additional training data, and increment the corresponding customers into our model. Hyperparameters are resampled after the merging process. The new features are directly computed from the merged model.

6 Experiments

6.1 Comparison with Previous Bayesian Model

First, we compared the previous Bayesian model (Gen) with our hierarchical back-off model (Back). We used the first 100K sentence pairs of the WMT10 News-Commentary corpus for German/Spanish/French-to-English pairs (Callison-Burch et al., 2010) and NTCIR10 corpus for Japanese-English (Goto et al., 2013) for the translation model. All sentences are lowercased and filtered to preserve at most 40 words on both source and target sides. We sampled 20 iterations for Gen and Back and combined the last 10 iterations for extracting the translation model.[5] The batch size was set to 64. The language models were estimated from the all-English side of the WMT News-Commentary and europarl-v7. In NTCIR10, we simply used the all-English side of the training data. All the 5-gram language models were estimated using SRILM (Stolcke and others, 2002) with interpolated Kneser-Ney smoothing. The details of the corpus are presented in Table 2. For detailed analysis, we also evaluate Hiero grammars extracted from GIZA++ (Och and Ney, 2003) grow-diag-final bidirectional alignments using Moses (Koehn et al., 2007) with Hiero options.

[4] Note that the correct way to decode from our model is to score every phrase pair created during decoding with back-off states, which is computationally intractable

[5] Gen and Back took 1 day, Back+future took 1.5 days for inference.

		News-Commentary						NTCIR10	
		de-en		es-en		fr-en		ja-en	
Model	Sample	BLEU	SIZE	BLEU	SIZE	BLEU	SIZE	BLEU	SIZE
∗GIZA++	-	16.66	7.07M	23.16	6.07M	20.79	6.25M	26.08	3.45M
Gen	1	15.36	397.63k	21.10	295.69k	19.45	311.76k	25.73	262.45k
	10	15.39	529.46k	20.83	384.55k	19.24	419.33k	25.79	344.67k
Back	1	15.30	410.92k	*21.43*	314.95k	*19.74*	362.22k	25.69	294.90k
	10	15.42	563.80k	*21.53*	420.15k	19.51	497.51k	25.63	388.87k
Back + future	1	**15.49**	384.69k	**21.63**	296.30k	**19.97**	340.70k	**25.82**	268.38k
	10	**15.55**	579.12k	**21.74**	429.33k	**19.97**	513.41k	25.41	390.23k

Table 1: Results of translation evaluation in 100k corpus

	de-en	es-en	fr-en	ja-en
TM(en)	1.85M	1.67M	1.54M	1.80M
TM(other)	1.86M	1.86M	1.83M	2.03M
LM(en)	55.6M	55.6M	55.6M	27.8M
Dev(en)	65.5k	65.5k	65.5k	67.3k
Dev(other)	62.7k	68.1k	72.5k	73.0k
Test(en)	61.9k	61.9k	61.9k	310k
Test(other)	61.3k	65.5k	70.5k	333k

Table 2: The number of words in training data

	TM	LM	Dev	Test
de	31.3M	-	55.1k	59.4k
en	32.8M	50.5M	58.8k	55.5k

Table 3: The number of words in training data

We use GIZA++ and Moses default parameters for training. Decoding was carried out using the cdec decoder (**?**). Feature weights were tuned on the development data by running MIRA (Chiang, 2012) for 20 iterations with 16 parallel. For other parameters, we used cdec's default values. The numbers reported here are the average of three tuning runs (Hopkins and May, 2011).

Table 1 lists the results measured using BLEU (Papineni et al., 2002).The term Sample denotes the combination size for each model. The term SIZE in the table denotes the number of the extracted grammar types composed of Hiero rules and phrase pairs. The numbers in italic denotes the score of Back, significantly improved from the score of 1 sampled combinated Gen. The numbers in bold denotes the score of Back + future, significantly improved from the score of 1 sampled combinated Back. All significance test are performed using Clark et al. (2011) under p-value of 0.05. Back performed better than Gen on Spanish-English and French-English language pairs. Note that the gains were achieved with the comparable grammar size. When comparing German-English and Japanese-English language pairs, there are no significant differences between Back and Gen. The combination of our Back with future score during slice sampling (+future) achieved further gains over the slice sampling without future scores, and slightly decrese the grammar size, compared to Back. However, there are still no significant difference between Back+future and Gen on German-English and Japanese-English language pairs. Sample combination has no or slight gain on BLEU score, in spite of the increase in grammar size. From the results, using last one sample as a grammar is sufficient for translation quality. The performance of the Bayesian model did not match with that for the GIZA++ pipeline heuristic approach. In general, complex model, such as Gen and Back, demands larger corpus size for training, and the evaluation on such smaller corpus may not be a fair comparison, since the sampling approach can rely on only sampled derivations. Thus, we evaluate these methods on large size corpus in the next section.

6.2 Comparison with Heuristic Extraction

As reported in (Koehn et al., 2003; DeNero et al., 2006), the comparison against heuristic extraction is a challenging task. We compare the Back+future and a baseline extracted from grow-diag-final alignments of GIZA++ using Moses with Hiero options. We use GIZA++ and Moses default parameters for training. In addition, we present heuristic extraction from the last 1 sample of Back+future in +Exhaustive.

We used the full europarl-v7 German-English corpus as presented in Table 3. The experimental set up was similar to that in Section 6.1 with the following exceptions; Slice sampling parameter a was set to 0.05. Mini-batch size was set to 1024 and sampling was performed 5 iterations.[6] The translation model was extracted by last 1 iterations.

Table 4 lists the results[7]. Our Back+future can

[6]Inference took 5 days.

[7]The row mark up with ∗ indicate the model using word

Model	BLEU	SIZE
∗ GIZA++ Model 4	27.21	73.24M (×14.0)
GIZA++ Model 3	26.78	59.26M (×11.3)
Back + future	26.83	5.25M (×1.0)
Back + future + exhasustive	26.73	90.42M (×17.2)

Table 4: Results of translation evaluation in de-en full size corpus.

Gen	*gin* X *kamera* / silver X camera *en* / salt
Back + future	*gin en kamera* / silver salt camera

Table 5: Example of a grammar

decrease the grammar size against GIZA++ with comparable BLEU score. Surprisingly, exhaustive extraction had no gains, probably because of the word alignment in each Hiero rules relied on the IBM Model 1.

7 Analysis

Intuitively, the use of the hierarchical back-off increases the Hiero grammar size, since the phrases of all the granularities in the derivation trees are incorporated in the grammar. In contrast, our hierarchical back-off model achieved gains in translation quality without increasing the size of the extracted grammar when compared to the previous generative model. The major differences were the use of the minimal phrase pairs used in the previous work in which only minimal phrase pairs in the leaves of derivation trees were included in the model. As a result, larger phrase pairs were forced to be constructed from those minimal rules. On the other hand, our back-off model could directly express phrase pairs of multiple granularities. In particular, a complex noun may be composed of several Hiero rules in the previous model, but it can be directly expressed by a single phrase pair in our model. Table 5 gives an example of a Japanese-English phrase pair which is represented by two Hiero rules in the previous model; it is directly expressed by a single phrase pair in our model.

The BLEU score of Back+future was higher than the generative baseline with comparable grammar size. We observed that a very different word alignment was sampled in every training iteration; the tendency was very frequent for function words. Our future score for inferring the slice variables may take into account the context in a sentence better than those without the future score. As a result, Back+future infers better models by avoiding over pruning spans.

The BLEU score of our back-off model did not achieve gains over the heuristic baselines. The detail analysis of the learned Hiero grammar's CRP tables reveals that the grammar is very sparse and may have little generalization capacity. The expansion of back-off process and the use of word classes will solve the sparsity and increase the translation quality.

class informations. Model 3 and our Back-off model dose not use any word class informations.

8 Conclusion

We proposed a hierarchical back-off model for Hiero grammar. Our back-off model achieved higher or equal translation quality against a previous Bayesian model under BLEU score on various language pairs;German/French/Spanish/Japanese-English. In addition to the hierarchical back-off model, we also proposed a two-step slice sampling approach. We showed that the two-step slice sampling approach can avoid over-pruning by incorporating a future score for estimating slice variables, which led to increase in translation quality through the experiments. The joint use of hierarchical back-off model and two step slice sampling approach achieved comparable translation quality on a full size Germany-English language pair in Europarl v7 corpus with with significantly smaller grammar size; 10% less than that for he heuristic baseline.

For future work, we plan to embed a back-off feature to decoder which is computed for all the phrase pairs constructed in a derivation during the decoding process. We will reflect the change of a probability as a statefull feature for decoding step.

References

Phil Blunsom and Trevor Cohn. 2010. Inducing synchronous grammars with slice sampling. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 238–241, Los Angeles, California, June. Association for Computational Linguistics.

Phil Blunsom, Trevor Cohn, Chris Dyer, and Miles Osborne. 2009. A gibbs sampler for phrasal synchronous grammar induction. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 782–790, Suntec, Singapore, August. Association for Computational Linguistics.

Chris Callison-Burch, Philipp Koehn, Christof Monz, Kay Peterson, Mark Przybocki, and Omar F Zaidan. 2010. Findings of the 2010 joint workshop on statistical machine translation and metrics for machine translation. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, pages 17–53. Association for Computational Linguistics.

David Chiang. 2007. Hierarchical phrase-based translation. *computational linguistics*, 33(2):201–228.

David Chiang. 2012. Hope and fear for discriminative training of statistical translation models. *The Journal of Machine Learning Research*, 13(1):1159–1187.

Tagyoung Chung, Licheng Fang, Daniel Gildea, and Daniel Štefankovič. 2014. Sampling tree fragments from forests. *Computational Linguistics*, 40(1):203–229.

Jonathan H. Clark, Chris Dyer, Alon Lavie, and Noah A. Smith. 2011. Better hypothesis testing for statistical machine translation: Controlling for optimizer instability. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 176–181, Portland, Oregon, USA, June. Association for Computational Linguistics.

Martin Cmejrek and Bowen Zhou. 2010. Two methods for extending hierarchical rules from the bilingual chart parsing. In *Coling 2010: Posters*, pages 180–188, Beijing, China, August. Coling 2010 Organizing Committee.

Trevor Cohn and Gholamreza Haffari. 2013. An infinite hierarchical bayesian model of phrasal translation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 780–790, Sofia, Bulgaria, August. Association for Computational Linguistics.

John DeNero, Dan Gillick, James Zhang, and Dan Klein. 2006. Why generative phrase models underperform surface heuristics. In *Proceedings on the Workshop on Statistical Machine Translation*, pages 31–38, New York City, June. Association for Computational Linguistics.

John DeNero, Alexandre Bouchard-Côté, and Dan Klein. 2008. Sampling alignment structure under a Bayesian translation model. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 314–323, Honolulu, Hawaii, October. Association for Computational Linguistics.

Chris Dyer, Adam Lopez, Juri Ganitkevitch, Jonathan Weese, Ferhan Ture, Phil Blunsom, Hendra Setiawan, Vladimir Eidelman, and Philip Resnik. 2010. cdec: A decoder, alignment, and learning framework for finite-state and context-free translation models. In *Proceedings of the ACL 2010 System Demonstrations*, pages 7–12, Uppsala, Sweden, July. Association for Computational Linguistics.

Isao Goto, Ka Po Chow, Bin Lu, Eiichiro Sumita, and Benjamin K Tsou. 2013. Overview of the patent machine translation task at the ntcir-10 workshop. In *Proceedings of the 10th NTCIR Workshop Meeting on Evaluation of Information Access Technologies: Information Retrieval, Question Answering and Cross-Lingual Information Access, NTCIR-10*.

Mark Hopkins and Jonathan May. 2011. Tuning as ranking. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1352–1362, Edinburgh, Scotland, UK., July. Association for Computational Linguistics.

Howard Johnson, Joel Martin, George Foster, and Roland Kuhn. 2007. Improving translation quality by discarding most of the phrasetable. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 967–975, Prague, Czech Republic, June. Association for Computational Linguistics.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 48–54. Association for Computational Linguistics.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th annual meeting of the ACL on interactive poster and demonstration sessions*, pages 177–180. Association for Computational Linguistics.

Abby Levenberg, Chris Dyer, and Phil Blunsom. 2012. A bayesian model for learning scfgs with discontiguous rules. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 223–232, Jeju Island, Korea, July. Association for Computational Linguistics.

Wang Ling, João Graça, Isabel Trancoso, and Alan Black. 2012. Entropy-based pruning for phrase-based machine translation. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 962–971, Jeju Island, Korea, July. Association for Computational Linguistics.

Daniel Marcu and Daniel Wong. 2002. A phrase-based,joint probability model for statistical machine translation. In *Proceedings of the 2002 Conference*

on Empirical Methods in Natural Language Processing, pages 133–139. Association for Computational Linguistics, July.

Radford M Neal. 2003. Slice sampling. *Annals of statistics*, pages 705–741.

Graham Neubig, Taro Watanabe, Eiichiro Sumita, Shinsuke Mori, and Tatsuya Kawahara. 2011. An unsupervised model for joint phrase alignment and extraction. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 632–641, Portland, Oregon, USA, June. Association for Computational Linguistics.

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

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 160–167, Sapporo, Japan, July. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July. Association for Computational Linguistics.

Xiaochang Peng and Daniel Gildea. 2014. Type-based mcmc for sampling tree fragments from forests. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1735–1745, Doha, Qatar, October. Association for Computational Linguistics.

Jim Pitman and Marc Yor. 1997. The two-parameter poisson-dirichlet distribution derived from a stable subordinator. *The Annals of Probability*, pages 855–900.

Andreas Stolcke et al. 2002. Srilm-an extensible language modeling toolkit. In *INTERSPEECH*.

Yee Whye Teh. 2006. A hierarchical bayesian language model based on pitman-yor processes. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 985–992, Sydney, Australia, July. Association for Computational Linguistics.

Jurgen Van Gael, Yunus Saatci, Yee Whye Teh, and Zoubin Ghahramani. 2008. Beam sampling for the infinite hidden markov model. In *Proceedings of the 25th international conference on Machine learning*, pages 1088–1095. ACM.

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

Xinyan Xiao, Deyi Xiong, Yang Liu, Qun Liu, and Shouxun Lin. 2012. Unsupervised discriminative induction of synchronous grammar for machine translation. In *Proceedings of COLING 2012*, pages 2883–2898, Mumbai, India, December. The COLING 2012 Organizing Committee.

Richard Zens, Daisy Stanton, and Peng Xu. 2012. A systematic comparison of phrase table pruning techniques. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 972–983, Jeju Island, Korea, July. Association for Computational Linguistics.

Kai Zhao and Liang Huang. 2013. Minibatch and parallelization for online large margin structured learning. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 370–379, Atlanta, Georgia, June. Association for Computational Linguistics.

Consistency-Aware Search for Word Alignment

Shiqi Shen[†], **Yang Liu**[†‡*], **Huanbo Luan**[†] and **Maosong Sun**[†‡]
[†]State Key Laboratory of Intelligent Technology and Systems
Tsinghua National Laboratory for Information Science and Technology
Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China
[‡]Jiangsu Collaborative Innovation Center for Language Competence, Jiangsu, China
{vicapple22,liuyang.china,luanhuanbo}@gmail.com, sms@tsinghua.edu.cn

Abstract

As conventional word alignment search algorithms usually ignore the consistency constraint in translation rule extraction, improving alignment accuracy does not necessarily increase translation quality. We propose to use *coverage*, which reflects how well extracted phrases can recover the training data, to enable word alignment to model consistency and correlate better with machine translation. This can be done by introducing an objective that maximizes both alignment model score and coverage. We introduce an efficient algorithm to calculate coverage on the fly during search. Experiments show that our consistency-aware search algorithm significantly outperforms both generative and discriminative alignment approaches across various languages and translation models.

1 Introduction

Word alignment, which aims to identify the correspondence between words in two languages, plays an important role in statistical machine translation (Brown et al., 1993). Word alignment and translation rule extraction often constitute two consecutive steps in the training pipeline. Word-aligned bilingual corpora serve as a fundamental resource for translation rule extraction, not only for phrase-based models (Koehn et al., 2003; Och and Ney, 2004), but also for syntax-based models (Chiang, 2005; Galley et al., 2006). Dividing alignment and extraction into two separate steps significantly improves the efficiency and scalability of parameter estimation as compared with directly learning translation models from bilingual corpora (Marcu and Wong, 2002; DeNero and Klein, 2008; Cohn and Blunsom, 2009).

However, separating word alignment from translation rule extraction suffers from a major problem: maximizing the accuracy of word alignment does not necessarily lead to the improvement of translation quality. A number of studies show that alignment error rate (AER) only has a loose correlation with BLEU (Callison-Burch et al., 2004; Goutte et al., 2004; Ittycheriah and Roukos, 2005). Ayan and Dorr (2006) find that precision-oriented alignments result in better translation performance than recall-oriented alignments. Fraser and Marcu (2007) show that using AER and balanced F-measure can only partially explain the effect of alignment quality on BLEU for several language pairs.

We believe that the correlation problem arises from the discrepancy between word alignment and translation rule extraction. On one hand, aligners seek to find the alignment with the highest alignment model score, without regard to structural constraints. Consequently, sensible translation rules may not be extracted because they violate consistency constraints required by translation rule extraction (Och and Ney, 2004). Wang et al. (2010) find that the standard alignment tools are not optimal for training syntax-based models. As a result, they have to resort to re-aligning. On the other hand, the consistency constraint used in most translation rule extraction algorithms tolerate wrong links within consistent phrase pairs. Chiang (2007) uses the union of two unidirectional alignments, which usually has a low precision, for extracting hierarchical phrases. Therefore, it is important to include both alignment model score and the consistency constraint in the optimization objective of word alignment.

In this work, we propose to use *coverage*, which measures how well extracted phrases can

*Corresponding author: Yang Liu.

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1228–1237,
Lisbon, Portugal, 17-21 September 2015.

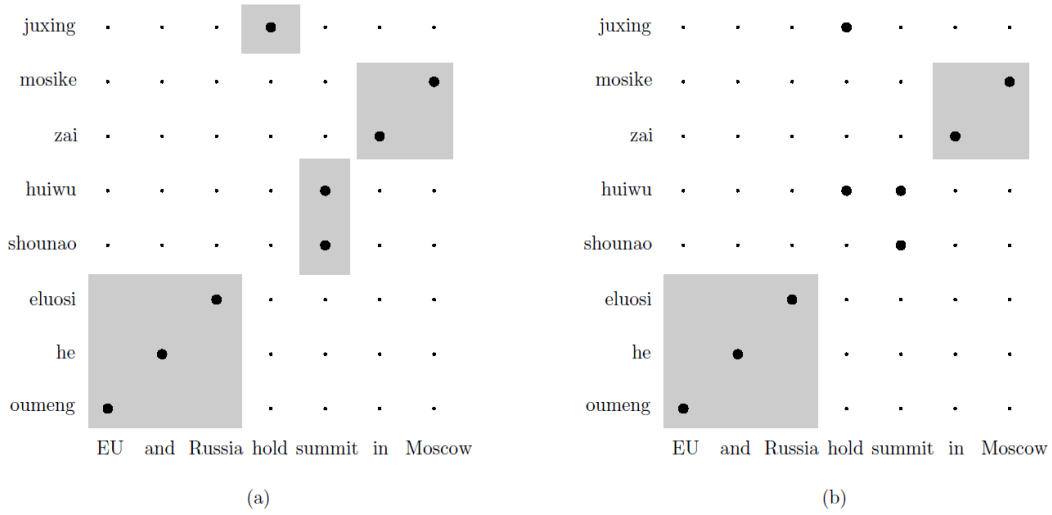


Figure 1: (a) An alignment resulting in a set of bilingual phrases (highlighted by shading) that can recover the training example, and (b) an alignment resulting in a set of bilingual phrases that fails to fully recover the training example. We assume the maximum phrase length $w = 3$. Our approach aims to avoid adding links that both have low posterior probabilities and hurt the recovery (e.g., the link between "huiwu" and "hold").

recover the training data, to bridge word alignment and (hierarchical) phrase-based translation. We introduce a new alignment search algorithm with an objective that maximizes both alignment model score and coverage while keeping the training algorithm unchanged. The coverage of an alignment is calculated on the fly during search using a local phrase extraction algorithm. Experiments show that our approach achieves significant improvements over state-of-the-art baselines across various languages and translation models.

2 Background

We begin by introducing the preliminaries of word alignment and phrase-based translation.

Definition 1 Given a source-language sentence $\mathbf{f} = f_1^J = f_1 \dots f_J$ and a target-language sentence $\mathbf{e} = e_1^I = e_1 \dots e_I$, an **alignment** $\mathbf{a}$ is a subset of the Cartesian product of the word positions of two sentences: $\mathbf{a} \subseteq \{(j, i) : j = 1, \dots, J; i = 1, \dots, I\}$.

Figure 1(a) shows an alignment for a Chinese sentence "oumeng he eluosi shounao huiwu zai mosike juxing" and an English sentence "EU and Russia hold summit in Moscow". We use black circles to denote links. The link $(1, 1)$ indicates that the first Chinese word "oumeng" and the first English word "EU" are translations of each other.

Definition 2 Given a training example $\langle \mathbf{f}, \mathbf{e}, \mathbf{a} \rangle$, a **bilingual phrase** B is a pair of source and target phrases: $B = (f_{j_1}^{j_2}, e_{i_1}^{i_2})$ such that $1 \le j_1 \le j_2 \le J \wedge 1 \le i_1 \le i_2 \le I$.

For example, ("zai mosike", "in Moscow") in Figure 1 can be denoted as a bilingual phrase $B = (f_6^7, e_6^7)$. For convenience, We use $B.j_1$ and $B.j_2$ to denote the beginning and ending positions of the source phrase in B, respectively. $B.i_1$ and $B.i_2$ are defined likewise for the target side.

Definition 3 A bilingual phrase $B = (f_{j_1}^{j_2}, e_{i_1}^{i_2})$ is said to be **tight** if and only if all boundary words (i.e., f_{j_1}, f_{j_2}, e_{i_1}, and e_{i_2}) are aligned. Otherwise, it is a **loose** bilingual phrase.

For example, in Figure 1, while (f_1^3, e_1^3) is a tight bilingual phrase, (f_1^4, e_1^4) is a loose bilingual phrase.

Definition 4 (Och and Ney, 2004) Given a training example $\langle \mathbf{f}, \mathbf{e}, \mathbf{a} \rangle$, a bilingual phrase $B = (f_{j_1}^{j_2}, e_{i_1}^{i_2})$ is said to be **consistent** with the word alignment $\mathbf{a}$ if and only if:

1. No words in the source phrase are aligned with words outside the target phrase and vice versa: $\forall (j, i) \in \mathbf{a} : j_1 \le j \le j_2 \leftrightarrow i_1 \le i \le i_2$,
2. At least one word in the source phrase is aligned with at least one word in the target

phrase: $\exists(j,i) \in \mathbf{a} : j_1 \leq j \leq j_2 \wedge i_1 \leq i \leq i_2$.

Alignment consistency forms the basis of translation rule extraction in modern SMT systems (Koehn and Hoang, 2007; Chiang, 2007; Galley et al., 2006; Liu et al., 2006). In Figure 1, (f_1^3, e_1^3) is consistent with the alignment because all words in "oumeng he eluosi" are aligned with all words in "EU and Russia". In contrast, in Figure 1(b), "huiwu shounao" and "hold summit" are not consistent with the alignment because "hold" is also aligned to a word "juxing" outside.

However, alignment consistency only defines a loose relationship between alignment and translation. A phrase pair consistent with alignment tolerates wrong inside links. For example, even if "oumeng" is aligned with "Russia", (f_1^3, e_1^3) is still consistent. This is one possible reason that maximizing alignment accuracy does not necessarily lead to improved translation performance.

3 Modeling Consistency in Word Alignment

Our intuition is that including the consistency constraint in word alignment can hopefully reduce the discrepancy between alignment and translation. While this idea has been suggested by a number of authors (e.g., (Deng and Zhou, 2009; DeNero and Klein, 2010)), our goal is to optimize arbitrary alignment models with respect to end-to-end translation in the search phase without labeled data (see Related Work for detailed comparison).

A natural way is to include consistency in the optimization objective as a regularization term. However, as consistency is only defined at the phrase level (see **Definition 4**), we need a sentence-level measure to reflect how well an alignment conforms to the consistency constraint. A straightforward measure is the number of bilingual phrases consistent with the alignment (*phrase count* for short), which is easy and efficient to calculate during search (Deng and Zhou, 2009). Unfortunately, optimizing with respect to *phrase count* is prone to yield alignments with very few links in a biased way, which result in a large number of bilingual phrases extracted from a small fraction of the training data. Another alternative is *reachability* (Liang et al., 2006a; Yu et al., 2013) that indicates whether there exists a full derivation to recover the training data. However, calculating reachability faces a major problem: a large portion of training data cannot be fully recovered due to noisy alignments and the distortion limit (Yu et al., 2013).

In this work, we propose *coverage*, which reflects how well extracted phrases can recover the training data, to measure the sentence-level consistency. In the following, we will introduce a number of definitions to facilitate the exposition.

Definition 5 A source word f_j is said to be **covered** by a bilingual phrase $B = (f_{j_1}^{j_2}, e_{i_1}^{i_2})$ if and only if $j_1 \leq j \leq j_2 : cov(f_j, B) = [\![j_1 \leq j \leq j_2]\!]$. Similarly, a target word e_i is covered by B if and only if $i_1 \leq i \leq i_2$.

The indicator function $[\![expr]\!]$ returns 1 if the boolean expression $expr$ is true and returns 0 otherwise. For example, in Figure 1(a), "oumeng" and "EU" are covered by the bilingual phrase $B = (f_1^3, e_1^3)$.

Definition 6 Given a set of bilingual phrases $\mathbf{B} = \{B^{(k)}\}_{k=1}^{K}$, a source word f_j is said to be **covered** by the bilingual phrase set $\mathbf{B}$ if and only if it is covered by at least one phrase in $B : cov(f_j, \mathbf{B}) = [\![\sum_{k=1}^{K} cov(f_j, B^{(k)}) > 0]\!]$. The definition for a target word is similar.

For example, in Figure 1(a), all source and target words are covered by the bilingual phrase set. In Figure 1(b), the source words "shounao", "huiwu", "juxing" and the target words "hold" and "summit" are not covered.

Definition 7 Given a sentence pair $\langle \mathbf{f}, \mathbf{e} \rangle$ and a phrase length limit w [1], the **hard coverage** of an alignment $\mathbf{a}$ is defined as a boolean value:

$$C_h(\mathbf{f}, \mathbf{e}, \mathbf{a}, w) = \Big[\!\Big[\delta\Big(\sum_{j=1}^{J} cov(f_j, \mathbf{B}), J \Big) \wedge \delta\Big(\sum_{i=1}^{I} cov(e_i, \mathbf{B}), I \Big) \Big]\!\Big] \quad (1)$$

where $\mathbf{B} = \textsc{Extract}(\mathbf{f}, \mathbf{e}, \mathbf{a}, 1, J, 1, I, w)$ is the set of consistent bilingual phrases extracted from the sentence pair using a standard phrase extraction algorithm (Och and Ney, 2004). The function δ returns true if the two parameters are same and returns false otherwise.

[1] The phrase length limit w is essential in defining coverage, restricting that the sentence pair must be covered by bilingual phrases no longer than w words. Otherwise, a very long bilingual phrase (e.g., the entire sentence pair) can achieve full coverage in a biased way.

Algorithm 1 A consistency-aware search algorithm for word alignment.

1: **procedure** ALIGN($\mathbf{f}, \mathbf{e}, \boldsymbol{\theta}, w, \beta, b, n$)
2: $open \leftarrow \emptyset$
3: $\mathcal{N} \leftarrow \emptyset$
4: $\langle \mathbf{a}, \mathbf{B} \rangle \leftarrow \langle \emptyset, \emptyset \rangle$
5: ADD($open, \langle \mathbf{a}, \mathbf{B} \rangle, \beta, b$)
6: **while** $open \neq \emptyset$ **do**
7: $closed \leftarrow \emptyset$
8: **for all** $\langle \mathbf{a}, \mathbf{B} \rangle \in open$ **do**
9: **for all** $l \in J \times I - \mathbf{a}$ **do**
10: $\mathbf{a}' \leftarrow \mathbf{a} \cup \{l\}$
11: $\mathbf{B}' \leftarrow$ UPDATE($\mathbf{f}, \mathbf{e}, \mathbf{a}, l, \mathbf{B}, w$)
12: **if** GAIN($\mathbf{f}, \mathbf{e}, \mathbf{a}, \mathbf{a}', w, \boldsymbol{\theta}$) > 0 **then**
13: ADD($closed, \langle \mathbf{a}', \mathbf{B}' \rangle, \beta, b$)
14: **end if**
15: ADD($\mathcal{N}, \langle \mathbf{a}', \mathbf{B}' \rangle, \beta, n$)
16: **end for**
17: **end for**
18: $open \leftarrow closed$
19: **end while**
20: **return** $\mathcal{N}$
21: **end procedure**

Depending on the tightness of extracted phrases (see **Definition 3**), we further distinguish between $C_{h+t}(\mathbf{f}, \mathbf{e}, \mathbf{a}, w)$ and $C_{h+l}(\mathbf{f}, \mathbf{e}, \mathbf{a}, w)$, which denote hard coverage calculated with tight and loose phrases, respectively.

Hard coverage denotes whether extracted phrases can fully recover the training data. For example, the values of hard coverage for Figures 1(a) and 1(b) are 1 and 0, respectively. As most training examples can hardly be fully recovered, we introduce *soft coverage* to better account for partially recoverable training data.

Definition 8 Given a sentence pair $\langle \mathbf{f}, \mathbf{e} \rangle$ and a phrase length limit w, the **soft coverage** of an alignment $\mathbf{a}$ is defined as

$$C_s(\mathbf{f}, \mathbf{e}, \mathbf{a}, w) = \frac{\sum_{j=1}^{J} cov(f_j, \mathbf{B}) + \sum_{i=1}^{I} cov(e_i, \mathbf{B})}{J + I} \quad (2)$$

Similarly, we also distinguish between C_{s+t} and C_{s+l} depending on the tightness of extracted phrases.

Definition 9 Given a word-aligned bilingual corpus $D = \{\langle \mathbf{f}^{(s)}, \mathbf{e}^{(s)}, \mathbf{a}^{(s)} \rangle\}_{s=1}^{S}$ and a phrase length limit w, the **corpus-level soft coverage** is defined as

$$C_s(D, w) = \frac{\sum_{j=1}^{|\mathbf{f}^{(s)}|} cov(f_j^{(s)}, \mathbf{B}^{(s)})}{\sum_{s=1}^{S} |\mathbf{f}^{(s)}| + |\mathbf{e}^{(s)}|} + \frac{\sum_{i=1}^{|\mathbf{e}^{(s)}|} cov(e_i^{(s)}, \mathbf{B}^{(s)})}{\sum_{s=1}^{S} |\mathbf{f}^{(s)}| + |\mathbf{e}^{(s)}|} \quad (3)$$

Algorithm 2 Updating the set of extracted bilingual phrases after adding a link.

1: **procedure** UPDATE($\mathbf{f}, \mathbf{e}, \mathbf{a}, l, \mathbf{B}, w$)
2: $\mathbf{B}' \leftarrow \mathbf{B}$
3: **for all** $B \in \mathbf{B}'$ **do**
4: **if** $B.j_1 \leq l.j \leq B.j_2 \vee B.i_1 \leq l.i \leq B.i_2$ **then**
5: $\mathbf{B}' \leftarrow \mathbf{B}' - \{B\}$
6: **end if**
7: **end for**
8: $j_1 \leftarrow l.j - w + 1$
9: $j_2 \leftarrow l.j + w - 1$
10: $i_1 \leftarrow l.i - w + 1$
11: $i_2 \leftarrow l.i + w - 1$
12: $\mathbf{a}' \leftarrow \mathbf{a} \cup \{l\}$
13: $\mathbf{B}'' \leftarrow$ EXTRACT($\mathbf{f}, \mathbf{e}, \mathbf{a}', j_1, j_2, i_1, i_2, w$)
14: $\mathbf{B}' \leftarrow \mathbf{B}' \cup \mathbf{B}''$
15: **return** $\mathbf{B}'$
16: **end procedure**

The corpus-level hard coverage is defined likewise.

4 Consistency-Aware Search

While Deng and Zhou (2009) focus on introducing an effectiveness function such as phrase count into alignment symmetrization, we are interested in guiding the search algorithms of arbitrary alignment models using coverage. Therefore, the objective of our search algorithm is defined as

$$score(\mathbf{f}, \mathbf{e}, \mathbf{a}, w, \boldsymbol{\theta}) = M(\mathbf{f}, \mathbf{e}, \mathbf{a}, \boldsymbol{\theta}) + \lambda C(\mathbf{f}, \mathbf{e}, \mathbf{a}, w) \quad (4)$$

where $M(\mathbf{f}, \mathbf{e}, \mathbf{a}, \boldsymbol{\theta})$ is alignment model score, $\boldsymbol{\theta}$ is a set of model parameters, $C(\mathbf{f}, \mathbf{e}, \mathbf{a}, w)$ is coverage (either hard or soft), and λ is a hyper-parameter that controls the preference between alignment model score and coverage. [2]

Therefore, the decision rule is given by

$$\hat{\mathbf{a}} = \underset{\mathbf{a} \in \mathcal{A}(\mathbf{f}, \mathbf{e})}{\operatorname{argmax}} \left\{ score(\mathbf{f}, \mathbf{e}, \mathbf{a}, w, \boldsymbol{\theta}) \right\} \quad (5)$$

where $\mathcal{A}(\mathbf{f}, \mathbf{e})$ is a set of all possible alignments for the sentence pair.

Algorithm 1 shows the consistency-aware search algorithm for word alignment. The input of the algorithm includes a source sentence $\mathbf{f}$, a target sentence $\mathbf{e}$, a set of model parameters $\boldsymbol{\theta}$, phrase length limit w, pruning parameters β and b, and the number of most likely alignments to be retaind n (line 1). Inspired by Liu et al. (2010),

[2] Note that training algorithms are unchanged. We only introduce a new search algorithm that takes coverage into consideration. We leave consistency-aware training algorithms for arbitrary alignment models for future work.

the algorithm starts with an empty alignment $\mathbf{a}$ together with an empty phrase set $\mathbf{B}$. We use $open$ to store active alignments during search and $\mathcal{N}$ to store top-n alignments after search (lines 2-4). The procedure ADD($open, \langle \mathbf{a}, \mathbf{B} \rangle, \beta, b$) adds $\langle \mathbf{a}, \mathbf{B} \rangle$ to $open$ and discards any alignment that has a score worse than β multiplied by the best score in the list or the score of the b-th best alignment (line 5). For each iteration (line 6), we use a list $closed$ to store promising alignments that have higher scores than the current alignment (line 8). For every possible link l (line 9), the algorithm produces a new alignment $\mathbf{a}'$ and updates the phrase set by calling a procedure UPDATE($\mathbf{f}, \mathbf{e}, \mathbf{a}, l, \mathbf{B}, w$) (lines 10-11). Then, the algorithm calls a procedure GAIN($\mathbf{f}, \mathbf{e}, \mathbf{a}, \mathbf{a}', w, \boldsymbol{\theta}$) to calculate the difference of model score after adding the link l:

$$score(\mathbf{f}, \mathbf{e}, \mathbf{a}', w, \boldsymbol{\theta}) - score(\mathbf{f}, \mathbf{e}, \mathbf{a}, w, \boldsymbol{\theta})$$

If $\mathbf{a}'$ has a higher score, it is added to $closed$ (line 13). We also update $\mathcal{N}$ to retain the top n alignment explored during the search (line 15). This process iterates until the model score does not increase.

Algorithm 2 describes how to update the set of extracted bilingual phrases after adding a link. Our idea is to only update the phrases near the added link l and keep other phrases unchanged. This strategy improves the efficiency by avoiding extracting phrases from the entire sentence pair. The algorithm first removes bilingual phrases that are either in the same row or in the same column with l (lines 2-7). For example, in Figure 1, the following bilingual phrases are removed after adding the link between "huiwu" and "hold" because the link breaks the consistency:

("shounao huiwu", "summit")
("juxing", "hold")

Other phrases out of the reach of the added link remain unchanged.

Then, the algorithm extracts bilingual phrases near l by calling the procedure EXTRACT. Note that the phrase extraction is restricted to a local region (j_1, j_2, i_1, i_2) by the phrase length limit w. We use $l.i$ and $l.j$ to denote the source and target positions of the link, respectively.

coverage	BLEU
C_{h+l}	24.89
C_{h+t}	23.16
C_{s+l}	24.69
C_{s+t}	25.41

Table 1: Comparison of different settings of coverage on the Chinese-English dataset using Moses. "h" denotes "hard", "s" denotes "soft", "l" denotes "loose", and "t" denotes "tight". The BLEU scores were calculated on the development set. For quick validation, we used a small fraction of the training data to train the phrase-based model.

5 Experiments

5.1 Setup

5.1.1 Languages and Datasets

We evaluated our approach in terms of alignment and translation quality on five language pairs: Chinese-English (ZH-EN), Czech-English (CS-EN), German-English (DE-EN), Spanish-English (ES-EN), and French-English (FR-EN). The evaluation metrics for alignment and translation are alignment error rate (AER) (Och and Ney, 2003) and case-insensitive BLEU (Papineni et al., 2002), respectively.

For Chinese-English, the training data consists of 1.2M pairs of sentences with 30.9M Chinese words and 35.5M English words. We used the SRILM toolkit (Stolcke, 2002) to train a 4-gram language model on the Xinhua portion of the English GIGAWORD corpus, which contains 398.6M words. For alignment evaluation, we used the Tsinghua Chinese-English word alignment evaluation data set (Liu and Sun, 2015). [3] For translation evaluation, we used the NIST 2006 dataset as the development set and the NIST 2002, 2003, 2004, 2005 and 2008 datasets as the test sets.

For other languages, the training data is Europarl v7. The English language model trained on the Xinhua portion of the English GIGAWORD corpus was also used for translation from European languages to English. For translation evaluation, we used the "news-test2012" dataset that contains 3,003 sentences as the development set and the "news-test2013" dataset that contains 3,000 sentences as the test set.

[3] http://nlp.csai.tsinghua.edu.cn/~ly/systems/TsinghuaAligner/TsinghuaAligner.html

method	# bp	# sp	# tp	# sw	# tw	C_{s+t}	C_{s+l}	AER	BLEU
C → E	49.8M	33.0M	14.1M	80.4K	89.5K	73.76	82.82	29.21	30.49
E → C	66.0M	14.9M	43.1M	80.0K	82.2K	74.58	86.49	33.04	29.76
Intersection	465.6M	64.3M	72.5M	133.1K	165.0K	72.46	98.52	28.01	29.90
Union	11.8M	7.4M	7.8M	51.3K	50.5K	53.17	54.34	32.80	30.24
GDF	15.9M	9.6M	10.0M	64.7K	62.2K	63.12	64.45	30.56	30.40
phrase count	388.7M	58.5M	63.7M	133.4K	164.5K	78.42	99.52	25.70	30.16
this work	46.0M	20.6M	21.7M	130.8K	159.6K	91.25	98.34	25.77	31.33

Table 2: Comparison of different alignment methods on the Chinese-English dataset. "GDF" denotes the grow-diag-final heuristic. "phrase count" denotes optimizing with respect to maximizing the number of extracted tight phrases. We used Moses to extract loose phrases from word-aligned training data for all methods. "# bp" denotes the number of extracted bilingual phrases, "# sp" denotes the number of source phrases, "# tp" denotes the number of target phrases, "# sw" denotes the source vocabulary size, "# tw" denotes the target vocabulary size. We report BLEU scores on the NIST 2005 test set.

alignment	NIST06	NIST02	NIST03	NIST04	NIST05	NIST08
generative	29.60	31.84	31.68	31.80	30.40	24.53
+*coverage*	30.63**	32.89**	32.77**	32.96**	31.33**	25.25**
discriminative	28.98	32.31	31.69	31.89	30.65	23.20
+*coverage*	29.98**	32.93**	32.45**	32.45**	31.10**	24.67**

Table 3: Translation evaluation on different alignment models. We apply our approach to both generative and discriminative alignment models. "generative" denotes applying the grow-diag-final heuristic to the alignments produced by IBM Model 4 in two directions. "discriminative" denotes the log-linear alignment model (Liu et al., 2010). Adding coverage leads to significant improvements. We use "**" to denote that the difference is statistically significant at $p < 0.01$ level.

5.1.2 Alignment Models

We apply our approach to both generative and discriminative alignment models. For generative models, we used GIZA++ (Och and Ney, 2003) to train IBM Model 4 in two directions. To calculate a model score for symmetrized alignments, we follow Liang et al. (2006b) to leverage link posterior marginal probabilities. For discriminative models, we used the open-source toolkit TsinghuaAligner (Liu and Sun, 2015) that implements the log-linear alignment model as described in (Liu et al., 2010). The model score for the log-linear model is also defined using link posteriors.

5.1.3 Translation Models

Two kinds of translation models, phrase-based (Koehn et al., 2003) and hierarchical phrase-based (Chiang, 2007), are used to evaluate whether our approach improves the correlation between alignment and translation. For the phrase-based model, we used the open-source toolkit Moses (Koehn and Hoang, 2007). For the hierarchical phrase-based model, we used an in-house re-implementation on par with state-of-the-art open-source decoders.

5.2 Comparison of Different Settings

We first investigate the optimal setting for coverage (hard vs. soft, tight vs. loose) on the Chinese-English dataset. For quick validation, we used a subset of the training data to train the phrase-based model using Moses. We used the development set to optimize the scaling factor λ (see Eq. (4)) and set it to 0.3 in our experiments.

Table 1 compares C_{h+l}, C_{h+t}, C_{s+l}, and C_{s+t}. We find that the "*soft + tight*" combination (i.e., C_{s+t}) yields the highest BLEU score on the development set. One possible reason is that tight phrases are usually of high quality and soft coverage allows for taking full advantage of the training data. On the contrary, C_{h+t} yields the lowest BLEU score because hard coverage fails to distinguish between partially recoverable training examples as it assigns zero to all partially recoverable data.

Then, we investigate the effect of the phrase length limit w in Algorithm 1 on translation quality. We find $w = 7$ achieves the best result,

which is consistent with the default setting in Moses. As a result, we used C_{s+t} and set $w = 7$ in the following experiments.

5.3 Comparison of Different Alignment Methods

We compare our approach with a number of alignment methods in terms of AER and BLEU, including IBM Model 4 in two directions (C $\rightarrow$ E and E $\rightarrow$ C), symmetrization heuristics (Intersection, Union, grow-diag-final), and consistency-aware models (tight phrase count and coverage). We used Moses to extract loose bilingual phrases from word-aligned bilingual corpora from all methods. Note that our approach uses C_{s+t} for finding alignments, from which Moses extracts loose phrases.

Table 2 lists the numbers of extracted bilingual phrases ("# bp"), source phrases ("# sp"), target phrases ("# tp"), source vocabulary size ("# sw"), and target vocabulary size ("# tw"). We find that a very large number of loose phrases can be extracted from the Intersection alignments, which also have the highest vocabulary sizes. However, a large portion of words in these phrases are actually unaligned, resulting in low translation quality.

We observe that adding consistency, either in terms of phrase count or coverage, significantly improves alignment accuracy by a large margin, suggesting that imposing structural constraint helps to reduce alignment errors. Our approach outperforms all methods in terms of BLEU significantly. Note that the coverage itself does not correlate well with BLEU. It is important to achieve a balance between model score and coverage. As mentioned in Section 5.2, we set $\lambda = 0.3$ in our experiments.

5.4 Translation Evaluation on Different Alignment Models

We apply our approach to both generative (Brown et al., 1993) and discriminative (Liu et al., 2010) alignment models. As shown in Table 3, we find that adding coverage to the optimization objective significantly improves the BLEU scores. All differences are statistically significant at $p < 0.01$ level. This finding suggests that our approach generalizes well to various alignment models.

5.5 Translation Evaluation on Different Translation Models

We also evaluated our approach on both phrase-based and hierarchical phrase-based models. As shown in Table 4, adding coverage to generative models leads to significant improvements for both models. All the differences are statistically significant at $p < 0.01$ level.

Although coverage is designed for extracting phrases, using coverage is still beneficial to hierarchical phrase-based models because hierarchical phrases are derived from phrases consistent with word alignment. [4]

5.6 Translation Evaluation on Different Language Pairs

Finally, we report BLEU scores across five language pairs in Table 5: Chinese-English (ZH-EN), Czech-English (CS-EN), German-English (DE-EN), Spanish-English (ES-EN), and French-English (FR-EN). ZH-EN uses four references and other language pairs only use single references.

We find that our approach outperforms the baseline statistically significantly at $p < 0.01$ for four language pairs and $p < 0.05$ for one language pair. Therefore, using coverage to bridge word alignment and machine translation can hopefully benefit more languages.

6 Related Work

Our work is inspired by three lines of research: (1) reachability in discriminative training of translation models, (2) structural constraints for alignment, and (3) learning with constraints.

6.1 Reachability in Discriminative Training of Translation Models

Discriminative training algorithms for statistical machine translation often need *reachable* training examples to find full derivations for updating model parameters (Liang et al., 2006a; Yu et al., 2013). Yu et al. (2013) report that only 32.1% sentences in the Chinese-English training data that contain 12.7% words are fully reachable

[4] We also tested our approach on syntax-based models (Galley et al., 2006; Liu et al., 2006) but failed to achieve significant improvements. The reason is that extracting syntactic translation rules often imposes an additional constraint: a phrase must be a constituent that is subsumed by a subtree. We believe that appending such constraint to the optimization objective will hopefully benefit syntax-based translation models. We leave this for future work.

translation	alignment	NIST06	NIST02	NIST03	NIST04	NIST05	NIST08
phrase	generative	29.60	31.84	31.68	31.80	30.40	24.53
	+*coverage*	30.63**	32.89**	32.77**	32.96**	31.33**	25.25**
hierarchical	generative	30.43	33.36	32.58	32.72	31.57	24.21
	+*coverage*	31.60**	34.67**	34.14**	34.24**	32.73**	24.89**

Table 4: Translation evaluation on different translation models. For translation, We used both phrase-based and hierarchical phrase-based models. For alignment, we used the generative model. "generative" denotes applying the grow-diag-final heuristic to the alignments produced by IBM Model 4 in two directions. Adding coverage leads to significant improvements. We use "**" to denote that the difference is statistically significant at $p < 0.01$ level.

alignment	ZH-EN	CS-EN	DE-EN	ES-EN	FR-EN
generative	30.40	19.89	21.13	26.39	26.22
+*coverage*	31.33**	20.04*	21.63**	26.79**	26.76**

Table 5: Translation evaluation on five language pairs. "generative" denotes applying the grow-diag-final heuristic to the alignments produced by IBM Model 4 in two directions. We use "*" and"**" to denote that the difference is statistically significant at $p < 0.05$ and $p < 0.01$, respectively. Note that ZH-EN uses four references and other language pairs only use single references.

due to noisy alignments and distortion limit. They find that most reachable sentences are short and generally literal.

We borrow the idea of measuring the degree of recovering training data from reachability but ignore the dependency between bilingual phrases for efficiency. To calculate reachability, one needs to figure out a full derivation, in which the bilingual phrases cover the training data and do not intersect with each other. Yu et al. (2013) indicate that using forced decoding to select reachable sentences with an unlimited distortion limit runs in $O(2^n n^3)$ time. In contrast, calculating coverage is much easier and more efficient by ignoring the dependency between phrases but still retains the spirit of measuring recovery.

6.2 Structural Constraints for Alignment

Modeling structural constraints in alignment has received intensive attention in the community, either directly modeling phrase-to-phrase alignment (Marcu and Wong, 2002; DeNero and Klein, 2008; Cohn and Blunsom, 2009) or intersecting synchronous grammars with alignment (Wu, 1997; Zhang and Gildea, 2005; Haghighi et al., 2009).

Our work is in spirit most close to (Deng and Zhou, 2009) and (DeNero and Klein, 2010). Deng and Bowen (2009) cast combining IBM Model 4 alignments in two directions as an optimization problem driven by an effectiveness function. They evaluate the impact of adding or removing a link with respect to phrase extraction using the effectiveness function of phrase count. The major difference is that we generalize their idea to arbitrary alignment models in the search phase rather than bidirectional alignment combination in the post-processing phase. In addition, we find that using coverage instead of phrase count results in better translation performance (see Table 2).

DeNero and Klein (2010) develop a discriminative model of extraction sets and optimize an extraction-based loss function with respect to translation. Their model is capable of predicting the extracted phrase set. While their approach relies on annotated data for training the discriminative model, our method only needs to tune the scaling factor λ on the development set. In addition, our approach is very general and can easily apply to arbitrary alignment models by appending a term to the optimization objective.

6.3 Learning with Constraints

Our work is also related to learning with constraints such as constraint-driven learning (Chang et al., 2007) and posterior regularization (Ganchev et al., 2010). The basic idea is to inject prior knowledge to the model as a regularization term. The major difference is that our coverage regularizer is independent of model parameters. As a result, alignment models can still be trained independently.

7 Conclusion

In this work, we have presented a general framework for optimizing word alignment with respect to machine translation. We introduce coverage to measure how well extracted bilingual phrases can recover the training data. We develop a consistency-aware search algorithm that calculates coverage on the fly during search efficiently. Experiments show the our approach is effective in both alignment and translation tasks across various alignment models, translation models, and language pairs.

In the future, we plan to apply our approach to syntax-based models (Galley et al., 2006; Liu et al., 2006; Shen et al., 2008) and include the constituency constraint in the optimization objective. It is also interesting to develop consistency-aware training algorithms for word alignment.

Acknowledgements

Yang Liu and Maosong Sun are supported by the 863 Program (2015AA011808) and the National Natural Science Foundation of China (No. 61331013 and No. 61432013). Huanbo Luan is supported by the National Natural Science Foundation of China (No. 61303075). This research is also supported by the Singapore National Research Foundation under its International Research Centre@Singapore Funding Initiative and administered by the IDM Programme. Many thanks go to Chunyang Liu and Meng Zhang for their discussions. We also thank the anonymous reviewers for their valuable feedback.

References

Necip Fazil Ayan and Bonnie J. Dorr. 2006. Going beyond aer: An extensive analysis of word alignments and their impact on mt. In *Proceedings of COLING-ACL 2006*, pages 9–16, Sydney, Australia, July.

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

Chris Callison-Burch, David Talbot, and Miles Osborne. 2004. Statistical machine translation with word- and sentence aligned parallel corpora. In *Proceedings of ACL 2004*.

Ming-Wei Chang, Lev Ratinov, and Dan Roth. 2007. Guiding semi-supervision with constraint-driven learning. In *Proceedings of ACL 2007*.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of ACL 2005*, pages 263–270, Ann Arbor, Michigan, June.

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.

Trevor Cohn and Phil Blunsom. 2009. A bayesian model of syntax-directed tree to string grammar induction. In *Proceedings of EMNLP 2009*.

John DeNero and Dan Klein. 2008. The complexity of phrase alignment problems. In *Proceedings of ACL 2008*.

John DeNero and Klein Klein. 2010. Discriminative modeling of extraction sets for machine translation. In *Proceedings of ACL 2010*.

Yonggang Deng and Bowen Zhou. 2009. Optimizing word alignment combination for phrase table training. In *Proceedings of ACL 2009*.

Alexander Fraser and Daniel Marcu. 2007. Measuring word alignment quality for statistical machine translation. *Computational Linguistics, Squibs and Discussions*, 33(3):293–303.

Michel Galley, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang, and Ignacio Thayer. 2006. Scalable inference and training of context-rich syntactic translation models. In *Proceedings of COLING-ACL 2006*, pages 961–968, Sydney, Australia, July.

Kuzmann Ganchev, João Graça, Jennifer Gillenwater, and Ben Taskar. 2010. Posterior regularization for structured latent variable models. *Journal of Machine Learning Research*.

Cyril Goutte, Kenji Yamada, and Eric Gaussier. 2004. Aligning words using matrix factorisation. In *Proceedings of ACL 2004*.

Aria Haghighi, John Blitzer, John DeNero, and Dan Klein. 2009. Better word alignments with supervised itg models. In *Proceedings of ACL 2009*.

Abraham Ittycheriah and Salim Roukos. 2005. A maximum entropy word aligner for arabic-english machine translation. In *Proceedings of EMNLP 2005*.

Philipp Koehn and Hieu Hoang. 2007. Factored translation models. In *Proceedings of EMNLP-CoNLL 2007*, pages 868–876, Prague, Czech Republic, June.

Philipp Koehn, Franz J. Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of HLT-NAACL 2003*, pages 127–133, Edmonton, Canada, May.

Percy Liang, Alexandre Bouchard-Cote, Dan Klein, and Ben Taskar. 2006a. An end-to-end discriminative approach to machine translation. In *Proceedings of ACL 2006*.

Percy Liang, Ben Taskar, and Dan Klein. 2006b. Alignment by agreement. In *Proceedings of HLT-NAACL 2006*, pages 104–111, New York City, USA, June.

Yang Liu and Maosong Sun. 2015. Contrastive unsupervised word alignment with non-local features. In *Proceedings of AAAI 2015*.

Yang Liu, Qun Liu, and Shouxun Lin. 2006. Tree-to-string alignment template for statistical machine translation. In *Proceedings of COLING/ACL 2006*.

Yang Liu, Qun Liu, and Shouxun Lin. 2010. Discriminaitve word alignment by linear modeling. *Computational Linguistics*, 36(3):303–339.

Daniel Marcu and William Wong. 2002. A phrase-based, joint probability model for statistical machine translation. In *Proceedings of EMNLP 2002*.

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

Franz J. Och and Hermann Ney. 2004. The alignment template approach to statistical machine translation. *Computational Linguistics*, 30(4):417–449.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a methof for automatic evaluation of machine translation. In *Proceedings of ACL 2002*.

Libin Shen, Jinxi Xu, and Ralph Weischedel. 2008. A new string-to-dependency machine translation algorithm with a target dependency language model. In *Proceedings of ACL 2008*.

Andreas Stolcke. 2002. Srilm - an extensible language modeling toolkit. In *Proceedings of ICSLP 2002*.

Wei Wang, Jonathan May, Kevin Knight, and Daniel Marcu. 2010. Re-structuring, re-labeling, and re-aligning for syntax-based machine translation. *Computational Linguistics*.

Dekai Wu. 1997. Stochastic inversion transaction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23:377–404.

Heng Yu, Liang Huang, Haitao Mi, and Kai Zhao. 2013. Max-violation perceptron and forced decoding for scalable mt training. In *Proceedings of EMNLP 2013*.

Hao Zhang and Danieal Gildea. 2005. Stochastic lexicalized inversion transduction grammars for alignment. In *Proceedings of ACL 2005*.

Graph-Based Collective Lexical Selection for Statistical Machine Translation

Jinsong Su[1,2], **Deyi Xiong**[3]*, **Shujian Huang**[2], **Xianpei Han**[4], **Junfeng Yao**[1]
Xiamen University, Xiamen, P.R. China[1]
State Key Laboratory for Novel Software Technology, Nanjing University, P.R. China[2]
Soochow University, Suzhou, P.R. China[3]
Institute of Software, Chinese Academy of Sciences, Beijing, P.R. China[4]
{jssu, yao0010}@xmu.edu.cn huangsj@nlp.nju.edu.cn
dyxiong@suda.edu.cn xianpei@nfs.iscas.ac.cn

Abstract

Lexical selection is of great importance to statistical machine translation. In this paper, we propose a graph-based framework for collective lexical selection. The framework is established on a *translation graph* that captures not only local associations between source-side content words and their target translations but also target-side global dependencies in terms of relatedness among target items. We also introduce a random walk style algorithm to collectively identify translations of source-side content words that are strongly related in translation graph. We validate the effectiveness of our lexical selection framework on Chinese-English translation. Experiment results with large-scale training data show that our approach significantly improves lexical selection.

1 Introduction

Lexical selection, which selects appropriate translations for lexical items on the source side, is a crucial task in statistical machine translation (SMT). The task is closely related to two factors: 1) associations of selected translations with lexical items on the source side, including corresponding source items and their neighboring words, and 2) dependencies[1] between selected target translations and other items on the target side.

As translation rules and widely-used n-gram language models can only capture local associations and dependencies, we have witnessed increasing efforts that attempt to incorporate non-local associations/dependencies into lexical selection. Efforts using source-side associations mainly focus on the exploitation of either sentence-level context (Chan et al., 2007; Carpuat and Wu, 2007; Hasan et al., 2008; Mauser et al., 2009; He et al., 2008; Shen et al., 2009) or the utilization of document-level context (Xiao et al., 2011; Ture et al., 2012; Xiao et al., 2012; Xiong et al., 2013). In contrast, target-side dependencies attract little attention, although they have an important impact on the accuracy of lexical selection. The most common practice is to use language models to estimate the strength of target-side dependencies (Koehn et al., 2003; Shen et al., 2008; Xiong et al., 2011). However, conventional n-gram language models are not good at capturing long-distance dependencies. Consider the example shown in Figure 1. As the translations of polysemous words "*wèntí*", "*chíyǒu*" and "*lìchǎng*" are far from each other, our baseline can only correctly translate "*lìchǎng*" as "*stance*". It inappropriately translates the other two words as "*problem*" and *null*, respectively, even with the support of an n-gram language model. If we could model long-distance dependencies among target translations of source words "*wèntí*"(*issue*), "*chíyǒu*"(*hold*) and "*lìchǎng*"(*stance*), these translation errors could be avoided.

In order to model target-side global dependencies, we propose a novel graph-based collective lexical selection framework for SMT. Specifically,

- First, we propose a translation graph to model not only local associations between source-side content words and their target translations but also global relatedness among target-side items.

*Corresponding author.

[1]Please note that dependencies in this paper are not necessarily syntactic dependencies.

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1238–1247,
Lisbon, Portugal, 17-21 September 2015.

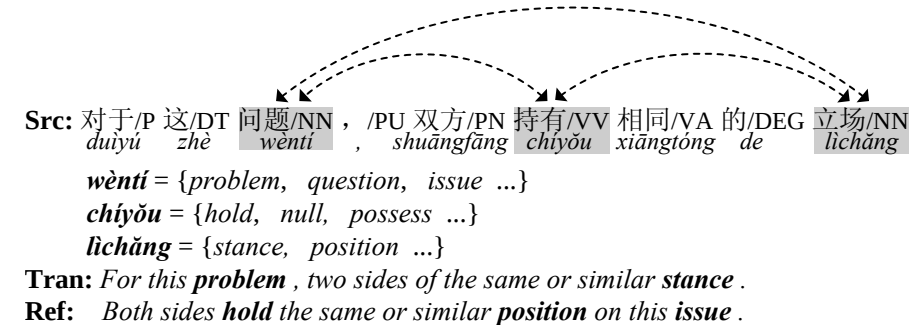


Figure 1: A Chinese-English translation example to illustrate the importance of target-side long-distance dependencies for lexical selection. Dotted lines show long-distance dependencies of source content words. Three content words "*wèntí*", "*chíyǒu*", "*lìchǎng*", and their candidate translations with high translation probabilities are also presented. **Src**: A Chinese sentence with part-of-speech tags. **Tran**: system output. **Ref**: reference translation.

- Second, we introduce a collective lexical selection algorithm, which can jointly identify translations of all source-side content words in the translation graph.

- Finally, we incorporate confidence scores of candidate translations in the translation graph, which are derived by the collective selection algorithm, into SMT to improve lexical selection.

We validate the effectiveness of our graph-based lexical selection framework on a hierarchical phrase-based system (Chiang, 2007). Experiment results on the NIST Chinese-English test sets show that our approach significantly improves translation quality.

We begin in Section 2 with the construction of translation graph for each translated sentence. Then, we propose a graph-based collective lexical selection framework for SMT in Section 3. Experiment results are reported in Section 4. We summarize and compare related work in Section 5. Finally, Section 6 presents conclusions and directions for future research.

2 Translation Graph

Formally, a translation graph is a weighted graph G=(N, E). In the node set N, each node represents either a source word or a target translation that contains one or multiple target words. In the edge set E, an edge linking a source word to a target translation is referred to as a *source-target association edge*, and an edge connecting two target translations is called as a *target-target relatedness edge*. In Section 2.1 and 2.2, we will answer the following two questions on the translation graph:

- Which source words and their translations should be included in the translation graph?

- How can we measure the strength of the above two types of relations in the graph with edge weights?

2.1 Graph Nodes

For a source sentence, the most ideal translation graph is a graph that includes all source words and their candidate translations. However, this ideal graph has two problems: intensive computation for graph inference and difficulty in modeling dependencies between function and content words. In order to get around these two issues, we only consider lexical selection for source content words[2].

We first identify source-side content word pairs using statistical metrics, and then keep word pairs with a high relatedness strength in the translation graph. To be specific, we use pointwise mutual information (PMI) (Church and Hanks, 1990) and co-occurrence frequency to measure the relatedness strength of two source-side words s and s' within a window d_s. Content word pairs will be kept when their co-occurrence frequencies are more than ϵ_{cf} times in our training corpus and PMI values are larger than ϵ_{pmi}. In this process, we remove noisy word pairs using the following heuristic rules: (1) As an adjective only has relations with its head nouns or dependent adverbs, we remove all word pairs where an adjective is paired with words other than its head nouns or dependent adverbs; (2) We apply a similar constraint to adverbs too, since the same thing happens to an adverb and its head verb or head ad-

[2]In this work, we consider nouns, verbs, adjectives and adverbs as content words in the source/target language.

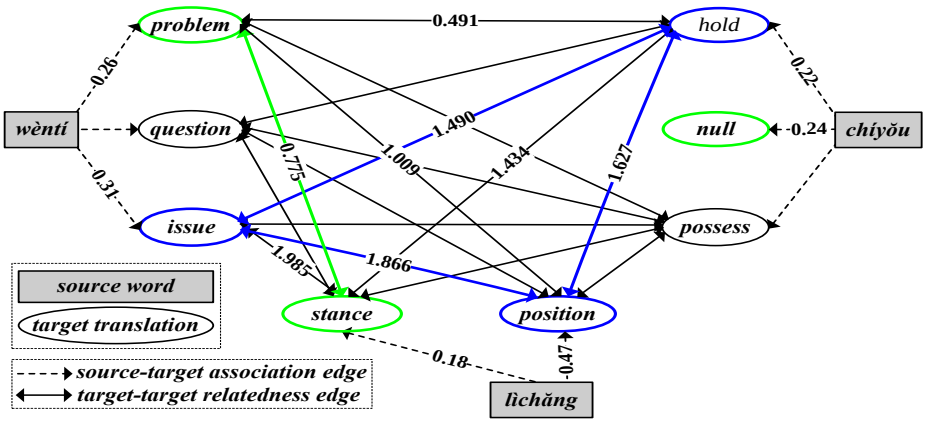


Figure 2: Translation graph of the example shown in Figure 1. Relatedness scores on edges are shown for two group of translations {"*problem*", *null*, "*stance*"} (**green**) and {"*issue*", "*hold*", "*position*"} (**blue**), which are estimated with PMI. Note that the *null* node does not have any relations with other nodes. Besides, two translation combinations {"*problem*", *null*, "*stance*"} (green) and {"*issue*", "*hold*", "*position*"} (blue) have different strengths of relatedness.

jective. For example, in the Chinese sentence in Figure 1, the adjective "*xiāngtóng*" is only related to the noun "*lìchǎng*" although it also frequently co-occur with "*wèntí*".

After identifying source-side content word pairs, we collect all target translations of these content words from extracted bilingual rules according to word alignments. These content words and target translations are used to build a translation graph, where each node represents a source-side content word or a candidate target translation. Note that there may be hundreds of different translations for a source word. For simplicity, we only consider target translations from translation options that are adopted by the decoder after rule filtering. Let's revisit the example in Figure 2, we include the following target translations in the translation graph: "*problem*", "*question*", "*issue*", "*hold*", *null*, "*possess*", "*stance*" and "*position*".

2.2 Edges and Weights

In this section, we introduce how we calculate weights for two kinds of edges in a translation graph.

2.2.1 Source-Target Association Edge

Connecting a source-side content word and its candidate target translations, a source-target association edge provides a way to propagate translation association evidence from a source word to its candidate translations. Obviously, the stronger the association between a source word and its candidate translation, the more evidence the corresponding edge will propagate. For each source-side content word, we obtain its candidate translations via the kept word alignments. Following Xiong et al. (2014), we allow a target translation to be either a phrase of length up to 3 words or *null* when s is not aligned to any word in the corresponding bilingual rule. We define the weight of the edge from a source-side content word s to its target translation $\tilde{t}$ as follows:

$$Weight(s \rightarrow \tilde{t}) = \frac{TP(s,\tilde{t})}{\sum_{\tilde{t}' \in N(s)} TP(s,\tilde{t}')} \quad (1)$$

where $N(s)$ denotes the set of candidate target translations of s kept on the translation graph, and $TP(s,\tilde{t})$ measures the probability of s being translated to $\tilde{t}$. It is very important to note that there is no evidence propagated from a target translation to a source word, as source-target association edges only go from a source-side content word to its translations.

We compute $TP(s,\tilde{t})$ according to the principle of maximal likelihood as follows:

$$TP(s,\tilde{t}) = \frac{count(s,\tilde{t})}{count(s)} \quad (2)$$

where $count(s,\tilde{t})$ indicates how often s is aligned to $\tilde{t}$ in the training corpus. Using this method, we can compute the translation probabilities of the *source-target association edges* in Figure 2 as follows: *TP*("*wèntí*", "*issue*")=0.31, *TP*("*chíyǒu*", "*hold*")=0.22 and *TP*("*lìchǎng*", "*position*")=0.47.

2.2.2 Target-Target Relatedness Edge

Connecting two target translations of different related source content words, a target-target relatedness edge enables translation graph to capture

dependencies between translations of any two different source words.

Computing the weight of a target-target relatedness edge is crucial for our method. Intuitively, the stronger co-occurrence strength two translations have, the more evidence should be propagated between them. Therefore we calculate the weight of a target-target related edge based on the co-occurrence strength of two translations linked by the edge. Formally, given the translation $\tilde{t}$ of source-side content word s and the translation $\tilde{t}'$ of source-side content word s', the weight of the edge from $\tilde{t}$ to $\tilde{t}'$ is defined as follows:

$$Weight(\tilde{t} \rightarrow \tilde{t}') = \frac{RS(\tilde{t}, \tilde{t}')}{\sum_{\tilde{t}'' \in N(\tilde{t})} RS(\tilde{t}, \tilde{t}'')} \quad (3)$$

where $N(\tilde{t})$ denotes the set of candidate translations that link to $\tilde{t}$, and $RS(\tilde{t}, \tilde{t}')$ measures the strength of relatedness between $\tilde{t}$ and $\tilde{t}'$ which is calculated as the average word-level relatedness over all content words in these two translations $\tilde{t}$ and $\tilde{t}'$.

As for the word-level relatedness $RS(t, t')$ for a content word pair (t, t'), we estimate it with the following two approaches over collected co-occurring word pairs within a window of size d_t: (1) $RS(t, t')$ is computed as a bigram conditional probability $p_{lm}(t'|t)$ via the language model; (2) Following (Xiong et al., 2011) and (Liu et al., 2014), we employ PMI to define $RS(t, t')$ as $\ln \frac{p(t,t')}{p(t)p(t')}$.

3 Collective Lexical Selection Algorithm

Based on the translation graph, we propose a collective lexical selection algorithm to jointly identify translations of all source words in the graph.

3.1 Problem Statement and Solution Method

As stated previously, the translation of a source-side content word s should be: 1) associated with s; 2) related to the translations of other source-side content words. Thus, in the translation graph, the translation of s should be a target-side node which has: 1) an association edge with the node of s; 2) many relatedness edges with other target-side nodes that represent translations of other source words.

Let's revisit Figure 2. If we know that the translation of "*wèntí*" is "*issue*", the relatedness between ("*issue*", "*hold*") and between ("*issue*", "*position*") can provide evidences that "*hold*" and "*position*" are the correct translations of "*chíyǒu*" and "*lìchǎng*", respectively. On the other hand, the candidate translation "*problem*" is less related to "*hold*" and "*position*", which may suggest that it is not likely to be the correct translation of "*wèntí*", even if it has a strong source-target association relation with "*wèntí*". However, in the translation graph, the correct target translation of a source word depends on correct translations of other source words in the graph, and vice versa. So how do we find these correct translations?

We propose a *Random Walk* (Gobel and Jagers, 1974) style algorithm to solve this problem, aiming to use both local source-target associations and global target-target relatedness simultaneously during translation. In our algorithm, we assign each node an *evidence score* in the translation graph, which indicates either the importance of a source word (for a source word node) or the confidence of a target translation being a correct translation (for a target word node). Specifically, we perform collective inference on the translation graph as follows:

- First, we set initial evidence scores for nodes in the translation graph.
- Second, evidence scores are simultaneously updated by propagating evidences along edges in the translation graph.

In the following sub-section, we describe the two steps in detail.

3.2 Details of Our Algorithm

Using the algorithm shown in Algorithm 1, we iteratively derive evidence scores for candidate translations.

3.2.1 Notations

For a translation graph with $\boldsymbol{n}$ nodes, we assign each node an index from 1 to $\boldsymbol{n}$ and use this index to represent the node. We also use the following two notations:

- **The evidence vector $\boldsymbol{V}$**: an n-dimensional vector where the ith component $\boldsymbol{V}_i$ is the evidence score contained in this node (if node i corresponds to a source word), or the evidence score from the related translations (if node i corresponds to a target translation). In particular, we use $\boldsymbol{V}^{(0)}$ to denote the initial

Algorithm 1 Collective Inference in Translation Graph.

Input: $\boldsymbol{S}$: the set of source-side content words, and $S(i)$ denotes the source word of node i;
$\boldsymbol{k}$: the number of source-side content words ;
$\boldsymbol{T}$: the set of all candidate target translations, and $T(j)$ denotes the target translation of node j;
$\boldsymbol{l}$: the number of candidate target translations;
λ: the reallocation weight;
maxIter: the maximum iteration number;
ϵ: the difference threshold;

1: **for** $i = 1, 2..., \boldsymbol{k}$
2: **for** $j = 1, 2..., \boldsymbol{l}$
3: **if** $\boldsymbol{S}(i)$ is linked to $\boldsymbol{T}(j)$
4: $\boldsymbol{M}_{\boldsymbol{k}+j,i} \leftarrow \textbf{\textit{Weight}}(\boldsymbol{S}(i) \rightarrow \boldsymbol{T}(j))$
5: **for** $j_1 = 1, 2..., \boldsymbol{l}$
6: **for** $j_2 = 1, 2..., \boldsymbol{l}$
7: **if** $\boldsymbol{T}(j_1)$ is linked to $\boldsymbol{T}(j_2)$
8: $\boldsymbol{M}_{\boldsymbol{k}+j_2,\boldsymbol{k}+j_1} \rightarrow \textbf{\textit{Weight}}(\boldsymbol{T}(j_1) \rightarrow \boldsymbol{T}(j_2))$
9: **for** $i = 1, 2..., \boldsymbol{k}$
10: $\boldsymbol{V}_i^{(0)} \leftarrow \textbf{\textit{Importance}}(\boldsymbol{S}(i))$
11: **for** $j = 1, 2..., \boldsymbol{l}$
12: $\boldsymbol{V}_{\boldsymbol{k}+j}^{(0)} \leftarrow 0$
13: $\delta \leftarrow \infty$
14: $r \leftarrow 1$
15: **while** $r \leq maxIter$ && $\delta > \epsilon$ **do**
16: $\boldsymbol{V}^{(r)} \leftarrow (1-\lambda) \times \boldsymbol{M} \times \boldsymbol{V}^{(r-1)} + \lambda \times \boldsymbol{V}^{(0)}$
17: $\delta \leftarrow \|\boldsymbol{V}^{(r)} - \boldsymbol{V}^{(r-1)}\|_2$
18: $r \leftarrow r + 1$
19: **end while**
20: **for** $i = 1, 2..., \boldsymbol{k}$
21: **for** $j = 1, 2..., \boldsymbol{l}$
22: **if** $\boldsymbol{S}(i)$ is linked to $\boldsymbol{T}(j)$
23: $\textbf{\textit{LexiTable}}(\boldsymbol{S}(i), \boldsymbol{T}(j)) \leftarrow normalize(\boldsymbol{V}_{\boldsymbol{k}+j}^{(r)})$

Return: ***LexiTable***;

evidence vector, and $\boldsymbol{V}^{(r)}$ to represent the evidence vector we obtain at the rth iteration.

- **The evidence propagation matrix $\boldsymbol{M}$**: an $\boldsymbol{n} \times \boldsymbol{n}$ matrix where $\boldsymbol{M}_{ij}$ is the evidence propagation ratio from node j to node i, and its value is the weight of the edge from node j to node i.

3.2.2 Algorithm

In Algorithm 1, we jointly infer the evidence scores of all candidate translations in the following three steps.

In **Step 1**, we calculate the evidence propagation matrix $\boldsymbol{M}$ according to the method described in Section 2.2 (equations (1) and (3)) (**Lines 1-8**).

In **Step 2**, we adopt different methods to set the value of $\boldsymbol{V}^{(0)}$ according to the node type. If the node corresponds to a source word, we set the initial value using its importance score in the translation graph, as implemented in (Han et al. 2011) (**Lines 9-10**). We calculate the importance score of the source word s using *tf.idf* as follows:

$$\textbf{\textit{Importance}}(s) = \frac{tf.idf(s)}{\sum_{s' \in N_{src}} tf.idf(s')} \quad (4)$$

where N_{src} is the set of source words in the translation graph. If the node corresponds to a target translation, its initial evidence score is 0 (**Lines 11-12**).

In **Step 3**, evidences are simultaneously reinforced by propagating them among semantically related translations (**Lines 13-19**). Specific to our algorithm, we update them by propagating evidences according to different types of relations in the evidence propagation matrix $\boldsymbol{M}$. Formally, the recursive update of the evidence vector is defined as follows:

$$\boldsymbol{V}^{(r)} = \boldsymbol{M} \times \boldsymbol{V}^{(r-1)} \quad (5)$$

where r is the number of iterations.

One problem with the above equation is that some nodes in the translation graph do not have evidence outgoing edges, such as translation nodes containing only function words or the *null* node. The evidence will disappear when passing through these nodes. To solve this problem, we propagate evidence in the form of reallocation: we reallocate a fraction of evidence to the initial evidence vector $\boldsymbol{V}^{(0)}$ at each step. The new recursive update of the evidence vector is formulated as follows:

$$\boldsymbol{V}^{(r)} = (1-\lambda) \times \boldsymbol{M} \times \boldsymbol{V}^{(r-1)} + \lambda \times \boldsymbol{V}^{(0)} \quad (6)$$

where $\lambda \in (0, 1)$ is the fraction of the reallocated evidence. We keep updating the evidence vector according to this equation (**Line 16**), until the maximal number of iteration $maxIter$ is reached or the Euclidean distance (**Line 17**) between evidence vectors calculated in two consecutive iterations is less than a pre-defined threshold ϵ (**Line 15**).

In this way, we jointly infer the evidence scores of all candidate target translations in the translation graph. Table 1 gives the evidence scores of the example in Figure 2. We can find that our system enhanced with target translation dependencies is able to select correct translations.

3.2.3 Integration of Derived Evidence Score

For each translated sentence, we may build multiple translation graphs. For each translation graph,

	scores of the source words			scores of the target translations							
	wèntí	***chíyǒu***	***lìchǎng***	***problem***	***question***	***issue***	***hold***	***null***	***possess***	***stance***	***position***
$\boldsymbol{V}^{(0)}$	0.2015	0.3989	0.3996	0	0	0	0	0	0	0	0
$\boldsymbol{V}^{(\tau)}$	0.0302	0.0598	0.0599	0.0533	0.0774	**0.1071**	**0.1218**	0.0244	0.0604	0.1186	**0.1486**

Table 1: The initial and final evidence scores of some source words and their target translations in Figure 2. Here we set the reallocation weight λ as 0.15. Note that the translations "*issue*", "*hold*" and "*position*" are given high evidence scores.

we infer evidence scores of translations represented by graph nodes using the above-mentioned algorithm before decoding. Then, for each candidate translation of a source-side content word, we normalize its evidence score over the corresponding translation graph to form an additional lexical translation probability (**Lines 20-23**). For instance, the normalized evidence score of "*chíyǒu*" translated into "*hold*" is calculated as $0.1218/(0.1218+0.0244+0.0604) \approx 0.5895$. In this way, for each bilingual rule with word alignments, we will obtain a new lexical weight which can be used together with the original translation probabilities and lexical weight to improve lexical selection in SMT.

4 Experiments

4.1 Setup

Our bilingual training corpus is the combination of the FBIS corpus and Hansards part of LDC2004T07 corpus ($1M$ parallel sentences, $54.6K$ documents, with $25.2M$ Chinese words and $29M$ English words). We word-aligned them using *GIZA++* (Och and Ney, 2003) with the option "*grow-diag-final-and*". We chose the NIST evaluation set of 2005 (MT05) as the development set, and the sets of MT06/MT08 as test sets. We used *SRILM* Toolkit (Stolcke, 2002) to train one 5-gram language model on the Xinhua portion of Gigaword corpus.

To construct translation graphs, we first used the *ZPar* toolkit[3] and the Stanford toolkit[4] to preprocess (word segmentation, PoS tagging and so on) Chinese and English sentences, respectively. We used the Chinese part of our bilingual corpus and an additional Chinese LDC Xinhua news corpus ($10.2M$ sentences with $279.9M$ words) as training data to collect Chinese word pairs. We set window size d_s=15, thresholds ϵ_{pmi}=0, ϵ_{cf}=5 to identify Chinese related word pairs in the NIST translated sentences. Averagely, these three sets contain 13.5, 10.3 and 9.5 content words used to build translation graphs per sentence, respectively. Using the English part of our bilingual corpus and the Xinhua portion of Gigaword corpus as training data, we set window size d_t=20, and used the SRILM toolkit with Witten-Bell smoothing and PMI to calculate relatedness strengths for target-side translations. To avoid data sparseness, we build the graph using the surface forms of words while calculating the word relatedness at the lemma level. To achieve this, we converted each word into its corresponding lemma with the exception of adjectives and adverbs. In the procedure of collective lexical selection, the difference threshold ϵ was set as 10^{-10}, and the maximal iteration number *maxIter* 100.

We reimplemented the decoder of Hiero (Chiang, 2007), a famous hierarchical phrase-based (HPB) system. HPB system is a formally syntax-based system and delivers good performance in various translation evaluations. During decoding, we set the *ttable-limit* as 20, the *stack-size* as 100. The translation quality is evaluated by case-insensitive BLEU-4 metric (Papineni et al., 2002). To alleviate the impact of the instability of MERT (Och, 2003), we ran it three times for each experiment and reported the average BLEU scores as suggested in (Clark et al., 2011). Finally, we conducted *paired bootstrap sampling* (Koehn, 2004) to test the significance in BLEU score differences.

4.2 Our Method vs Other Methods

In the first group of experiments, we investigated the effectiveness of our model by comparing it against the baseline as well as two additional models: (1) *lexicalized rule selection model* (He et al., 2008) (**LRSM**), which employs local context to improve rule selection in the HPB system; (2) *topic similarity model* (Xiao et al., 2012)[5] (**TSM**), which explores document-level topic information for translation rule selection in the HPB system. Furthermore, we combined our model with the two models to see if we could obtain further improvements. For this, we integrated the new lexi-

[3]http://people.sutd.edu.sg/∼yue_zhang/doc/index.html

[4]http://nlp.stanford.edu/software

[5]We used 30 topics following (Xiao et al., 2012).

System	MT06	MT08	Avg
Baseline	30.25	21.25	25.75
LRSM	31.12	21.98	26.55
TSM	30.79	21.90	26.35
GM(LM)	30.64	21.78	26.21
GM(PMI)	31.02	21.77	26.40
LRSM +GM(PMI)	31.66	22.23	26.95
TSM +GM(PMI)	31.34	22.26	26.80

Table 2: Experiment results on the test sets with λ=0.15. **Avg** = average BLEU scores, **GM(LM)** and **GM(PMI)** denote our model using the measure based on language model and PMI, respectively.

cal weight learned by our model as a new feature into the **LRSM/TSM** system.

Table 2 reports the results. All models outperform the baseline. Especially, our graph-based lexical selection model GM(PMI) achieves an average BLEU score of **26.40** on the two test sets, which is higher than that of the baseline by **0.65** BLEU points. This improvement is statistically significant at $p<0.01$. The BLEU score of our model is close to those of LRSM and TSM, which achieve an average BLEU score of 26.55 and 26.35 on the two test sets, respectively. As PMI is slightly better than LM in our model, we use PMI in experiments hereafter.

The combination of our model and LRSM is able to further improve translation quality in terms of BLEU. In this case, the average BLEU score of the improved system is 26.95, with 0.4 BLEU points higher than LRSM. When combining our model with TSM, we obtain an average BLEU score of 26.80, which is better than TSM by 0.45 BLEU points. The two improvements over LRSM and TSM are also statistically significant at $p<0.05$. These experiment results suggest that exploring long-distance dependencies among target translations is complementary to the previous lexical selection methods which focus on source-side context information.

In order to know how our approach improves the performance of the HPB system, we compared the best translations of the HPB system using different models. We find that our approach really improves translation quality by utilizing target-side long-distance dependencies which are, on the contrary, ignored in previous methods.

For example, the source sentence "... 穆沙拉夫 去年 9月 和 [巴] 北方 地区... 塔利班 残余 势力..." is translated as follows:

- **Ref**: ... *musharraf and some tribal leaders in the northern region of [pakistan] last september ... the remnant forces of the taliban ...*
- **Baseline**: ... *musharraf last september and [palestine] north of tribal leaders ... the remnants of the taliban ...*
- **LRSM**: ... *musharraf last september and some tribal chiefs of the northern region of [palestine] ... the remnants of the taliban ...*
- **LRSM+GM(PMI)**: ... *last september musharraf and some tribal chiefs of the northern region of [pakistan] ... the remnants of the taliban ...*

Here both the baseline and LRSM fail to obtain the right translation for the word "巴" because "*palestine*" has a higher probability than "*pakistan*" (0.0374 vs 0.0285). However, in our model, the long-distance dependencies between (*"musharraf", "pakistan"*) and (*"taliban", "pakistan"*) help the decoder correctly choose the translation "*pakistan*" for "巴".

In yet another example, the source sentence "美 希望 朝 核 问题 [协议] 得到 全面 执行" is translated as follows:

- **Ref**: *us hopes agreement on north korean nuclear issue be fully implemented*
- **Baseline**: *us hoped that the dprk nuclear issue is the full implementation*
- **TSM**: *us hope that the full implementation of the nuclear issue*
- **TSM+GM(PMI)**: *us hope that the dprk nuclear issue [agreement] to be fully implemented*

Even with TSM, the HPB system did not translate "协议" at all because translation rules "X_1 协议 得到 ||| X_1 is" and "X_1 协议 X_2 执行 ||| X_2 implementation of X_1" are used to translate the source sentence by the baseline and TSM systems respectively. However, in the combined model TSM+GM(PMI), the differences in relatedness scores between (*"nuclear", "agreement"*), (*"issue", "agreement"*) and (*"agreement", "implemented"*) encourage the enhanced system to select right translation for this word.

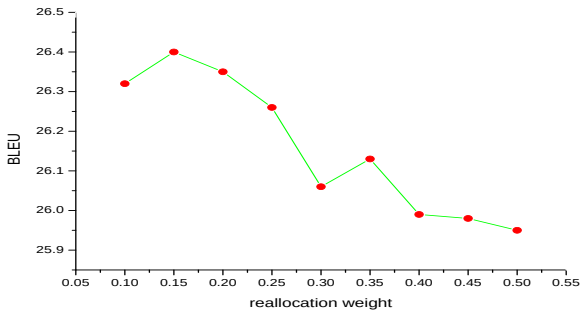


Figure 3: Experiment results on the test sets using different reallocation weights.

4.3 Effect of Reallocation Weight λ.

In Eq. (6), the reallocation weight λ determines which part plays a more important role in our method. In order to investigate the effect of λ on our method, we tried different values for λ: from 0.1 to 0.5 with an increment of 0.05 each time. The experimental setup is the same as the previous experiments. Figure 3 shows the average BLEU scores on the two test sets. Our system performs well when λ ranges from 0.1 to 0.25. The performance drops when λ is larger than 0.25. A small reallocation weight λ reduces the impact of initial evidences and local source-side associations in the collective lexical selection algorithm, but increases the impact of global dependencies of target-side translation, which are normally not considered in previous lexical selection methods. This performance curve on the values of λ suggests that target-side global dependencies are important for lexical selection.

5 Related Work

The collective inference algorithm is partially inspired by Han et al. (2011) who propose a graph-based collective entity linking (EL) method to model global interdependences among different EL decisions. We successfully adapt this algorithm to lexical selection in SMT. Other related work mainly includes the following two strands.

(1) Lexical selection in SMT. In order to capture source-side context for lexical selection, some researchers propose *trigger-based lexicon models* to capture long-distance dependencies (Hasan et al., 2008; Mauser et al., 2009), and many more researchers build classifiers with rich context information to select desirable translations during decoding (Chan et al., 2007; Carpuat and Wu, 2007; He et al., 2008; Liu et al., 2008). Shen et al. (2009) introduce four new linguistic and contextual features for HPB system. We have also witnessed increasing efforts in the exploitation of document-level context information. Xiao et al. (2011) impose a hard constraint to guarantee the translation consistency in document-level translation. Ture et al. (2012) soften this consistency constraint by integrating three counting features into decoder. Hardmeier et al. (2012, 2013) introduce a document-wide phrase-based decoder and integrate a semantic language model that cross sentence boundaries into the decoder. Based on topic models, Xiao et al. (2012) present a topic similarity model for HPB system, where each rule is assigned with a topic distribution. Also relevant is the work of Xiong et al. (2013), who use three different models to capture lexical cohesion for document-level machine translation. Compared with the above-mentioned studies, our method focuses on the exploitation of global dependencies among target translations, which has attracted little attention before.

Different from exploring source-side context, other researchers pay attention to the utilization of target-side context information. The common practice in SMT is to use an n-gram language model to capture local dependencies between translations (Koehn et al., 2003; Xiong et al., 2011). Yet another approach exploring target-side context information is proposed by Shen et al. (2008), who use a dependency language model to capture long-distance relations on the target side. Moreover, Zhang et al. (2014) treat translation as an unconstrained target sentence generation task, using soft features to capture lexical and syntactic correspondences between the source and target language. Recently, many researcher have proposed to use deep neural networks to model long-distance dependencies of arbitrary length for SMT (Auli et al., 2013; Kalchbrenner and Blunsom, 2013; Devlin et al., 2014; Hu et al., 2014; Liu et al., 2014; Sundermeyer et al., 2014). Our work is significantly different from these methods. We use a graph representation to capture local and global context information, which, to the best of our knowledge, is the first attempt to explore graph-based representations for lexical selection. Furthermore, our model do not resort to any syntactic resources such as dependency parsers of the target language.

(2) Random walk for SMT. Because of the advantage of global consistency, random walk al-

gorithm has been applied in SMT. For example, Cui et al. (2013) develop an effective approach to optimize phrase scoring and corpus weighting jointly using graph-based random walk. Zhu et al. (2013) apply a random walk method to discover implicit relations between the phrases of different languages. Aiming to better evaluate translation quality at the document level, Gong and Li (2013) run PageRank algorithm to assign weights to words in translation evaluation. Different from these studies, the key interest of our research lies in the lexical selection with random walk.

6 Conclusion and Future Work

This paper has presented a novel graph-based collective lexical selection method for SMT. We build translation graphs to capture local source-side associations and global target-side dependencies, and propose a purely collective inference algorithm to jointly identify target translations of source-side content words in translation graphs. Our method capitalizes on capabilities of translation graphs to represent both local and global relations on the source/target side. Experiment results demonstrate the effectiveness of our method.

In the future, we plan to further improve our model by capturing semantic relatedness among source words. Additionally, we also want to jointly model different levels of context information in a unified framework for SMT.

Acknowledgments

The authors were supported by National Natural Science Foundation of China (Grant Nos 61303082 and 61403269), Natural Science Foundation of Jiangsu Province (Grant No. BK20140355), Research Fund for the Doctoral Program of Higher Education of China (Grant No. 20120121120046), Research fund of the State Key Laboratory for Novel Software Technology in Nanjing University (Grant No. KFKT2015B11), the Special and Major Subject Project of the Industrial Science and Technology in Fujian Province 2013 (Grant No. 2013HZ0004-1), and 2014 Key Project of Anhui Science and Technology Bureau (Grant No. 1301021018). We also thank the anonymous reviewers for their insightful comments.

References

Michael Auli, Michel Galley, Chris Quirk, and Geoffrey Zweig. 2013. Joint language and translation modeling with recurrent neural networks. In *Proc. of EMNLP 2013*, pages 1044–1054.

Marine Carpuat and Dekai Wu. 2007. Improving statistical machine translation using word sense disambiguation. In *Proc. of EMNLP 2007*, pages 61–72.

Yee Seng Chan, Hwee Tou Ng, and David Chiang. 2007. Word sense disambiguation improves statistical machine translation. In *Proc. of ACL 2007*, pages 33–40.

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, pages 201–228.

Kenneth Ward Church and Patrick Hanks. 1990. Word association norms, mutual information, and lexicography. *Computational Linguistics*, 16(1):22–29.

Jonathan H. Clark, Chris Dyer, Alon Lavie, and Noah A. Smith. 2011. Better hypothesis testing for statistical machine translation: Controlling for optimizer instability. In *Proc. of ACL 2011, short papers*, pages 176–181.

Lei Cui, Dongdong Zhang, Shujie Liu, Mu Li, and Ming Zhou. 2013. Bilingual data cleaning for smt using graph-based random walk. In *Proc. of ACL 2013*, pages 340–345.

Jacob Devlin, Rabih Zbib, Zhongqiang Huang, Thomas Lamar, Richard Schwartz, and John Makhoul. 2014. Fast and robust neural network joint models for statistical machine translation. In *Proc. of ACL 2014*, pages 1370–1380.

F. Gobel and A.A. Jagers. 1974. Random walks on graphs. *Stochastic Processes and Their Applications*, 2(4):331–336.

Zhengxian Gong and Liangyou Li. 2013. Document-level automatic machine translation evaluation based on weighted lexical cohesion. In *Proc. of NLPCC 2013*.

Xianpei Han, Le Sun, and Jun Zhao. 2011. Collective entity linking in web text: A graph-based method. In *Proc. of SIGIR 2011*, pages 765–774.

Saša Hasan, Juri Ganitkevitch, Hermann Ney, and Jesús Andrés-Ferrer. 2008. Triplet lexicon models for statistical machine translation. In *Proc. of EMNLP 2008*, pages 372–381.

Zhongjun He, Qun Liu, and Shouxun Lin. 2008. Improving statistical machine translation using lexicalized rule selection. In *Proc. of COLING 2008*, pages 321–328.

Yuening Hu, Michael Auli, Qin Gao, and Jianfeng Gao. 2014. Minimum translation modeling with recurrent neural networks. In *Proc. of EACL 2014*, pages 20–29.

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

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proc. of NAACL-HLT 2003*, pages 127–133.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proc. of EMNLP 2004*, pages 388–395.

Qun Liu, Zhongjun He, Yang Liu, and Shouxun Lin. 2008. Maximum entropy based rule selection model for syntax-based statistical machine translation. In *Proc. of EMNLP 2008*, pages 89–97.

Kang Liu, Liheng Xu, and Jun Zhao. 2014. Extracting opinion targets and opinionwords from online reviews with graph co-ranking. In *Proc. of ACL 2014*, pages 314–324.

Arne Mauser, Saša Hasan, and Hermann Ney. 2009. Extending statistical machine translation with discriminative and trigger-based lexicon models. In *Proc. of EMNLP 2009*, pages 210–218.

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

Franz Joseph Och. 2003. Minimum error rate training in statistical machine translation. In *Proc. of ACL 2003*, pages 160–167.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: A method for automatic evaluation of machine translation. In *Proc. of ACL 2002*, pages 311–318.

Libin Shen, Jinxi Xu, and Ralph Weischedel. 2008. A new string-to-dependency machine translation algorithm with a target dependency language model. In *Proc. of ACL 2008*, pages 577–585.

Libin Shen, Jinxi Xu, Bing Zhang, Spyros Matsoukas, and Ralph Weischedel. 2009. Effective use of linguistic and contextual information for statistical machine translation. In *Proc. of EMNLP 2009*, pages 72–80.

Andreas Stolcke. 2002. Srilm - an extensible language modeling toolkit. In *Proc. of ICSLP 2002*, pages 901–904.

Martin Sundermeyer, Tamer Alkhouli, Joern Wuebker, and Hermann Ney. 2014. Translation modeling with bidirectional recurrent neural networks. In *Proc. of EMNLP 2014*, pages 14–25.

Ferhan Ture, DouglasW. Oard, and Philip Resnik. 2012. Encouraging consistent translation choices. In *Proc. of NAACL-HLT 2012*, pages 417–426.

Tong Xiao, Jingbo Zhu, Shujie Yao, and Hao Zhang. 2011. Document-level consistency verification in machine translation. In *Proc. of MT SUMMIT 2011*, pages 131–138.

Xinyan Xiao, Deyi Xiong, Min Zhang, Qun Liu, and Shouxun Lin. 2012. A topic similarity model for hierarchical phrase-based translation. In *Proc. of ACL 2012*, pages 750–758.

Deyi Xiong and Min Zhang. 2014. A sense-based translation model for statistical machine translation. In *Proc. of ACL 2014*, pages 1459–1469.

Deyi Xiong, Min Zhang, and Haizhou Li. 2011. Enhancing language models in statistical machine translation with backward n-grams and mutual information triggers. In *Proc. of ACL 2011*, pages 1288–1297.

Deyi Xiong, Guosheng Ben, Min Zhang, Yajuan Lü, and Qun Liu. 2013. Modeling lexical cohesion for document-level machine translation. In *Proc. of IJCAI 2013*, pages 2183–2189.

Yue Zhang, Kai Song, Linfeng Song, Jingbo Zhu, and Qun Liu. 2014. Syntactic smt using a discriminative text generation model. In *Proc. of EMNLP 2014*, pages 177–182.

Xiaoning Zhu, Zhongjun He, Hua Wu, Haifeng Wang, Conghui Zhu, and Tiejun Zhao. 2013. Improving pivot-based statistical machine translation using random walk. In *Proc. of EMNLP 2013*, pages 524–534.

Bilingual Correspondence Recursive Autoencoders for Statistical Machine Translation

Jinsong Su[1], **Deyi Xiong**[2]*, **Biao Zhang**[1], **Yang Liu**[3], **Junfeng Yao**[1], **Min Zhang**[2]
Xiamen University, Xiamen, P.R. China[1]
Soochow University, Suzhou, P.R. China[2]
Tsinghua University, Beijing, P.R. China[3]
{jssu, biaozhang, yao0010}@xmu.edu.cn
{dyxiong, minzhang}@suda.edu.cn
liuyang2011@tsinghua.edu.cn

Abstract

Learning semantic representations and tree structures of bilingual phrases is beneficial for statistical machine translation. In this paper, we propose a new neural network model called Bilingual Correspondence Recursive Autoencoder (BCorrRAE) to model bilingual phrases in translation. We incorporate word alignments into BCorrRAE to allow it freely access bilingual constraints at different levels. BCorrRAE minimizes a joint objective on the combination of a recursive autoencoder reconstruction error, a structural alignment consistency error and a cross-lingual reconstruction error so as to not only generate alignment-consistent phrase structures, but also capture different levels of semantic relations within bilingual phrases. In order to examine the effectiveness of BCorrRAE, we incorporate both semantic and structural similarity features built on bilingual phrase representations and tree structures learned by BCorrRAE into a state-of-the-art SMT system. Experiments on NIST Chinese-English test sets show that our model achieves a substantial improvement of up to 1.55 BLEU points over the baseline.

1 Introduction

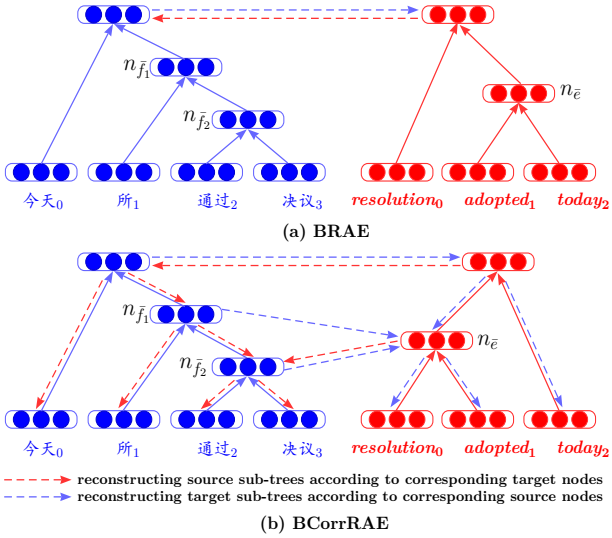
Recently a variety of "deep architecture" approaches, including autoencoders, have been successfully used in statistical machine translation (SMT) (Yang et al., 2013; Liu et al., 2013; Zou et al., 2013; Devlin et al., 2014; Tamura et al., 2014; Sundermeyer et al., 2014; Wang et al., 2014; Kočiský et al., 2014). Typically, these approaches represent words as dense, low-dimensional and real-valued vectors, i.e., word embeddings. However, translation units in machine translation have long since shifted from words to phrases (sequence of words), of which syntactic and semantic information cannot be adequately captured and represented by word embeddings. Therefore, learning compact vector representations for phrases or even longer expressions is more crucial for successful "deep" SMT.

To address this issue, many efforts have been initiated on learning representations for bilingual phrases in the context of SMT, inspired by the success of work on monolingual phrase embeddings (Socher et al., 2010; Socher et al., 2011a; Socher et al., 2013b; Chen and Manning, 2014; Kalchbrenner et al., 2014; Kim, 2014). The learning process of bilingual phrase embeddings in these efforts is normally interacted and mingled with single or multiple essential components of SMT, e.g., with reordering models (Li et al., 2013), translation models (Cui et al., 2014; Zhang et al., 2014; Gao et al., 2014), or both language and translation models (Liu et al., 2014). In spite of their success, these approaches center around capturing relations between entire source and target phrases. They do not take into account internal phrase structures and bilingual correspondences of sub-phrases within source and target phrases. The neglect of these important clues may be due to the big challenge imposed by the integration of them into the learning process of bilingual phrase representations. However, we believe such internal structures and correspondences can help us learn better phrase representations since they provide multi-level syntactic and semantic constraints.

In this paper, we propose a Bilingual Correspondence Recursive Autoencoder (BCorrRAE) to learn bilingual phrase embeddings. BCorrRAE substantially extends the Bilingually-constrained Recursive Auto-encoder (BRAE) (Zhang et al., 2014) to exploit both inner structures and corre-

*Corresponding author.

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1248–1258,
Lisbon, Portugal, 17-21 September 2015.



(a) BRAE

(b) BCorrRAE

Figure 1: BRAE vs BCorrRAE models for generating of a bilingual phrase ("今天 所 通过 决议", "*resolution adopted today*") with word alignments ("0-2 2-1 3-0"). The subscript number of each word indicates its position within phrase. Solid lines depict the generation procedure of phrase structures, while dash lines illustrate the reconstruction procedure from one language to the other. In this paper, the dimensionality of vector d in all figures is set to 3 for better illustration.

spondences within bilingual phrases. The intuitions behind BCorrRAE are twofold: 1) bilingual phrase structure generation should satisfy word alignment constraints as much as possible; and 2) corresponding sub-phrases on the source and target side of bilingual phrases should be able to reconstruct each other as they are semantic equivalents. In order to model the first intuition, BCorrRAE punishes bilingual structures that violate word alignment constraints and rewards those in consistent with word alignments. This enables BCorrRAE to produce desirable bilingual phrase structures from the perspective of word alignments. With regard to the second intuition, BCorrRAE reconstructs structures of sub-phrases of one language according to aligned nodes in the other language and minimizes the gap between original and reconstructed structures. In doing so, BCorrRAE is capable of capturing semantic relations at different levels.

To better illustrate our model, let us consider the example in Figure 1. Similar to the conventional recursive antoencoder (RAE), BRAE neglects bilingual correspondences of sub-phrases. Thus, it may combine "*adopted*" and "*today*" together to generate an undesirable target tree structure which violates word alignments. In contrast, BCorrRAE aligns source-side nodes (e.g. ("通过", "决 议")) to their corresponding target-side nodes (accordingly ("*resolution*", "*adopted*")) according to word alignments. Furthermore, in BCorrRAE, each subtree on the target side can be reconstructed from the corresponding source node that aligns to the target-side node dominating the subtree and vice versa. These advantages allow us to obtain improved bilingual phrase embeddings with better inner correspondences of sub-phrases and word alignment consistency.

We conduct experiments with a state-of-the-art SMT system on large-scale data to evaluate the effectiveness of BCorrRAE model. Results on the NIST 2006 and 2008 datasets show that our system achieves significant improvements over baseline methods. The main contributions of our work lie in the following three aspects:

- We learn both embeddings and tree structures for bilingual phrases using cross-lingual RAE reconstruction that minimizes semantic distances between original and reconstructed subtrees. To the best of our knowledge, this has not been investigated before.
- We incorporate word alignment information to guide phrase structure generation and establish internal semantic associations of sub-phrases within bilingual phrases.
- We integrate two similarity features based on BCorrRAE to enhance translation candidate selection, and achieve an improvement of 1.55 BLEU points on Chinese-English translation.

2 RAE and BRAE

In this section, we briefly introduce the RAE and its bilingual variation BRAE. This will provide background knowledge on our proposed BCorrRAE.

2.1 RAE

The component in the dash box of Figure 2 illustrates an instance of an RAE applied to a three-word phrase. The input to the RAE is $x = (x_1, x_2, x_3)$, which are the d-dimensional vector representations of the ordered words in a phrase. For two children $c_1 = x_1$ and $c_2 = x_2$, the parent vector y_1 can be computed in the following way:

$$p = f(W^{(1)}[c_1; c_2] + b^{(1)}) \quad (1)$$

where $[c_1; c_2] \in \mathbb{R}^{2d \times 1}$ is the concatenation of c_1 and c_2, $W^{(1)} \in \mathbb{R}^{d \times 2d}$ is a parameter matrix, $b^{(1)} \in \mathbb{R}^{d \times 1}$ is a bias term, and f is an element-

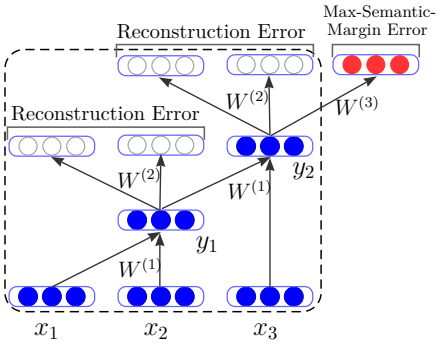


Figure 2: An illustration of the BRAE architecture.

wise activation function such as $tanh(\cdot)$, which is used for all activation functions in BRAE and our model. The learned parent vector p is also a d-dimensional vector. In order to measure how well p represents its children, we reconstruct the original children nodes in a reconstruction layer:

$$[c_1'; c_2'] = f(W^{(2)}p + b^{(2)}) \quad (2)$$

where c_1' and c_2' are reconstructed children vectors, $W^{(2)} \in \mathbb{R}^{2d \times d}$ and $b^{(2)} \in \mathbb{R}^{2d \times 1}$.

We can set $y_1 = p$ and then further use Eq. (1) again to compute y_2 by setting $[c_1; c_2] = [y_1; x_3]$. This combination and reconstruction process of auto-encoder repeats at each node until the vector of the entire phrase is generated. To obtain the optimal binary tree and phrase representation for x, we employ a greedy algorithm (Socher et al., 2011c) to minimize the sum of *reconstruction error* at each node in the binary tree $T(x)$:

$$E_{rec}(x;\theta) = \sum_{n \in T(x)} \frac{1}{2} \parallel [c_1; c_2]_n - [c_1'; c_2']_n \parallel^2 \quad (3)$$

where θ denotes model parameters and n represents a node in $T(x)$.

2.2 BRAE

BRAE jointly learns two RAEs for source and target phrase embeddings as shown in Figure 1(a). The core idea behind BRAE is that a source phrase and its target correct translation should share the same semantic representations, while non-equivalent pairs should have different semantic representations. Zhang et al. (2014) use this intuition to constrain semantic pharse embedding learning.

As shown in Figure 2, in addition to the above-mentioned reconstruction error, BRAE introduces a *max-semantic-margin error* to minimize the semantic distance between translation equivalents and maximize the semantic distance between non-equivalent pairs simultaneously. Formally, the max-semantic-margin error of a bilingual phrase (f, e) is defined as

$$E_{sem}(f, e; \theta) = E^*_{sem}(f|e, \theta) + E^*_{sem}(e|f, \theta) \quad (4)$$

where $E^*_{sem}(f|e, \theta)$ is used to ensure that the semantic error for an equivalent pair is much smaller than that for a non-equivalent pair (the source phrase f and a bad translation e'):

$$E^*_{sem}(f|e, \theta) = max\{0, \; E_{sem}(f|e, \theta) - E_{sem}(f|e', \theta) + 1\} \quad (5)$$

where $E_{sem}(f|e, \theta)$ is defined as the semantic distance between the learned vector representations of f and e, denoted by p_f and p_e, respectively. Since phrase embeddings for the source and target language are learned separately in different vector spaces, a transformation matrix $W^{(3)}_f \in \mathbb{R}^{d \times d}$ is introduced to capture this semantic transformation in the source-to-target direction. Thus, $E_{sem}(f|e, \theta)$ is calculated as

$$E_{sem}(f|e, \theta) = \frac{1}{2} \parallel p_e - f(W^{(3)}_f p_f + b^{(3)}_f) \parallel^2 \quad (6)$$

where $b^{(3)}_f \in \mathbb{R}^{d \times 1}$ is a bias term. $E^*_{sem}(e|f, \theta)$ and $E_{sem}(e|f, \theta)$ can be computed in a similar way. The joint error of (f, e) is therefore defined as follows:

$$E(f, e; \theta) = \alpha(E_{rec}(f, \theta) + E_{rec}(e, \theta)) + (1 - \alpha)(E^*_{sem}(f|e, \theta) + E^*_{sem}(e|f, \theta)) \quad (7)$$

The final BRAE objective function over the training instance set $\mathcal{D}$ becomes:

$$J_{BRAE} = \sum_{(f,e) \in D} E(f, e; \theta) + \frac{\lambda}{2} \|\theta\|^2 \quad (8)$$

Model parameters can be optimized over the total errors on training bilingual phrases in a co-training style algorithm (Zhang et al., 2014).

3 The BCorrRAE Model

As depicted above, the learned embeddings using BRAE may be unreasonable due to the neglect of bilingual constraints at different levels. To address this drawback, we propose the BCorrRAE for bilingual phrase embeddings, which incorporates bilingual correspondence information into the learning process of structures and embeddings via word alignments. In our model, we explore word alignments in two ways: (1) ensuring that a learned bilingual phrase structure is consistent with word alignments as much as possi-

ble; (2) identifying corresponding sub-phrases in the source language for reconstructing sub-phrases in the target language, and vice versa. More specifically, the former is to encourage alignment-consistent generation of sub-structures, while the latter is to minimize semantic distances between bilingual sub-phrases.

In this section, we first formally introduce a concept of structural alignment consistency encoded in bilingual phrase structure learning, which is the basis of our model. Then, we describe the objective function which is composed of three types of errors. Finally, we provide details on the training of our model.

3.1 Structural Alignment Consistency

We adapt word alignment to structural alignment and introduce some related concepts. Given a bilingual phrase (f, e) with its binary tree structures (T_f, T_e), if the source node $n_{\bar{f}} \in T_f$ covers a source-side sub-phrase $\bar{f}$, and there exists a target-side sub-phrase $\bar{e}$ such that $(\bar{f}, \bar{e})$ are consistent with word alignments (Och and Ney, 2003), we say $n_{\bar{f}}$ satisfies the *structural alignment consistency*, and it is referred to as a *structural-alignment-consistent* (SAC) *node*. Further, if $\bar{e}$ is covered by a target node $n_{\bar{e}} \in T_e$, we say $n_{\bar{e}}$ is the *aligned node* of $n_{\bar{f}}$. In this way, several different target nodes may be all aligned to the same source node because of null alignments. For this, we choose the target node with the smallest span as the aligned one for the considered source node. This is because a smaller span reflects a stronger semantic relevance in most situations.

Likewise, we have similar definitions for target nodes. Note that alignment relations between source- and target-side nodes may not be symmetric. For example, in Figure 1(b), node $n_{\bar{e}}$ is the aligned node of node $n_{\bar{f}_1}$, while node $n_{\bar{f}_2}$ rather than $n_{\bar{f}_1}$ is the aligned node of $n_{\bar{e}}$.

3.2 The Objective Function

We elaborate the three types of errors defined for a bilingual phrase (f, e) with its binary tree structures (T_f, T_e) on both sides below.

3.2.1 Reconstruction Error

Similar to RAE, the first error function is used to estimate how well learned phrase embeddings represent corresponding phrases. The *reconstruction error* $E_{rec}(f, e; \theta)$ of (f, e) is defined as follows:

$$E_{rec}(f, e; \theta) = E_{rec}(f; \theta) + E_{rec}(e; \theta) \quad (9)$$

where both $E_{rec}(f; \theta)$ and $E_{rec}(e; \theta)$ can be calculated according to Eq. (3).

3.2.2 Consistency Error

This metric corresponds to the first way in which we exploit word alignments mentioned before, which enables our model to generate as many SAC nodes as possible to respect word alignments.

Formally, the *consistency error* $E_{con}(f, e; \theta)$ of (f, e) is defined in the following way:

$$E_{con}(f, e; \theta) = E_{con}(T_f; \theta) + E_{con}(T_e; \theta) \quad (10)$$

where $E_{con}(T_f; \theta)$ and $E_{con}(T_e; \theta)$ denote the consistency error score for T_f and T_e, given word alignments. Here we only describe the calculation of the former while the latter can be calculated in exactly the same way.

To calculate $E_{con}(T_f; \theta)$, we first judge whether a source node $n_{\bar{f}}$ is an SAC node according to word alignments. Let $p_{n_{\bar{f}}}$ be the vector representation of $n_{\bar{f}}$. Following Socher et al. (2010), who use a simple inner product to measure how well the two words are combined into a phrase, we use inner product to calculate the consistency/inconsistency score for $n_{\bar{f}}$:

$$s(n_{\bar{f}}) = W^{score} p_{n_{\bar{f}}} \quad (11)$$

where $W^{score} \in \mathbb{R}^{1 \times d}$ is the score parameter. We calculate W^{score} by distinguishing SAC from non-SAC nodes defined as follows:

$$W^{score} = \begin{cases} W^{score}_{cns} & \text{if } n_{\bar{f}} \text{ is an SAC node} \\ W^{score}_{inc} & \text{otherwise} \end{cases}$$

where the subscript *cns* and *inc* represent consistency and inconsistency, respectively. For example, in Figure 3, as $n_{\bar{f}_3}$ is a non-SAC node, we calculate the inconsistency score using W^{score}_{inc} for it.

We expect T_f to satisfy structural alignment consistency as much as possible. Therefore we encourage the consistency score for T_f to be larger than its inconsistency score using a max-margin consistency error function:

$$\begin{aligned} E_{con}(T_f; \theta) = & max\{0, 1 - s(T_f)_{cns} \\ & + s(T_f)_{ins}\} \end{aligned} \quad (12)$$

where $s(T_f)_{cns}$ denotes the sum of consistency scores over all SAC nodes and $s(T_f)_{ins}$ the sum of inconsistency scores over all non-SAC nodes in T_f. Minimizing this error function will maximize the sum of consistency scores of SAC nodes and minimize (up to a margin) the sum of inconsis-

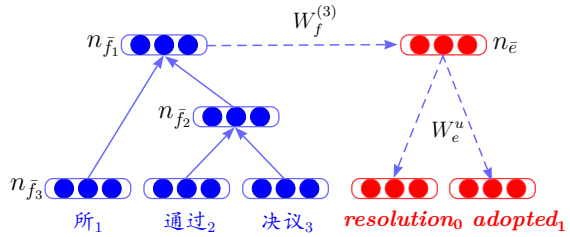


Figure 3: The structure generation procedure of the source sub-phrase "所 通过 决议" and the structure reconstruction procedure of the target sub-phrase "*resolution adopted*". According to word alignments ("2-1 3-0"), the node $n_{\bar{f}_1}$ and $n_{\bar{f}_2}$ are SAC ones while the node $n_{\bar{f}_3}$ is a non-SAC node.

tency scores of non-SAC nodes.

3.2.3 Cross-Lingual Reconstruction Error

This metric corresponds to the second way in which we exploit word alignments. The assumption behind this is that a source/target node should be able to reconstruct the entire subtree rooted at its target/source aligned node as they are semantically equivalent. Based on this, for the considered node, we calculate the cross-lingual reconstruction error along the entire subtree rooted at its aligned node in the other language and use the error to measure how well the learned vector represents this node.

Similarly, the *cross-lingual reconstruction error* $E_{clrec}(f, e; \theta)$ of (f, e) can be decomposed into two parts as follows:

$$E_{clrec}(f, e; \theta) = E_{f2e\cdot rec}(T_f, T_e; \theta) + E_{e2f\cdot rec}(T_f, T_e; \theta) \quad (13)$$

where $E_{f2e\cdot rec}(T_f, T_e; \theta)$ denotes the error score using T_f to reconstruct T_e. Note that in this process, the structure and the original node vector representations of T_e have been already generated. $E_{e2f\cdot rec}(T_f, T_e; \theta)$ denotes the reconstruction error score using T_e to reconstruct T_f. Here we still only describe the method of computing the former, which also applies to the latter.

To calculate $E_{f2e\cdot rec}(T_f, T_e; \theta)$, we first collect all source nodes $(n_{\bar{f}})$ in T_f and their aligned nodes $(n_{\bar{e}})$ in T_e to form a set of aligned node pairs $S = \{\langle n_{\bar{f}}, n_{\bar{e}}\rangle\}$ according to word alignments. We then calculate $E_{f2e\cdot rec}(T_f, T_e; \theta)$ as the sum of error scores over all node pairs in S. Given a source node $n_{\bar{f}}$ with its aligned node $n_{\bar{e}}$ on the target side, we use $n_{\bar{f}}$ to reconstruct the sub-tree structure $T_{\bar{e}}$ rooted at $n_{\bar{e}}$ and compute the error score based on the semantic distance between the original and reconstructed vector representations of nodes in $T_{\bar{e}}$. As source and target phrase embeddings are separately learned, we first introduce a transformation matrix $W_f^{(3)}$ and a bias term $b_f^{(3)}$ to transform source phrase embeddings into the target-side semantic space, following Zhang et al. (2014) and Hermann and Blunsom (2014):

$$p'_{n_{\bar{e}}} = f(W_f^{(3)} p_{n_{\bar{f}}} + b_f^{(3)}) \quad (14)$$

here $p'_{n_{\bar{e}}}$ denotes the reconstructed vector representation of $n_{\bar{e}}$, which is transformed from the vector representation $p_{n_{\bar{f}}}$ of $n_{\bar{f}}$. Then, we repeat the reconstruction procedure in a top-down manner along the corresponding target tree structure until leaf nodes are reached, following Socher et al. (2011a). Specifically, given the vector representation $p'_{n_{\bar{e}}}$, we reconstruct vector representations of its two children nodes:

$$[c_{e1}^u; c_{e2}^u] = f(W_e^u p'_{n_{\bar{e}}} + b_e^u) \quad (15)$$

where c_{e1}^u and c_{e2}^u are the reconstructed vector representations of the children nodes, $W_e^u \in \mathbb{R}^{2d\times d}$, and $b_e^u \in \mathbb{R}^{2d\times 1}$. Eventually, given the original and reconstructed target phrase representations, we calculate $E_{f2e\cdot rec}(T_f, T_e; \theta)$ as follows:

$$E_{f2e\cdot rec}(T_f, T_e; \theta) = \frac{1}{2} \sum_{\langle n_{\bar{f}}, n_{\bar{e}}\rangle \in \mathcal{S}} \sum_{n \in T_{\bar{e}}} \| p_n - p'_n \|^2 \quad (16)$$

where p_n and p'_n are the original and reconstructed vector representations of node n in the sub-tree structure $T_{\bar{e}}$ rooted at $n_{\bar{e}}$. This error function will be minimized so that semantic differences between original and reconstructed structures are minimal.

Figure 3 demonstrates the structure reconstruction from a generated source sub-tree to its target counterpart. In this way, BCorrRAE propagates semantic information along dash lines sequentially until leaf nodes in the generated structure of the target phrase.

3.2.4 The Final Objective

Similar to Eq. (8), we define the final objective function of our model based on the three types of errors described above

$$J_{BCorrRAE} = \sum_{(f,e)\in\mathcal{D}} \{\alpha (E_{rec}(f;\theta) + E_{rec}(e;\theta)) + \beta (E_{con}(T_f;\theta) + E_{con}(T_e;\theta)) + \gamma (E_{f2e\cdot rec}(T_f, T_e;\theta) + E_{e2f\cdot rec}(T_f, T_e;\theta))\} + R(\theta) \quad (17)$$

where weights α, β, γ (s.t. $\alpha+\beta+\gamma=1$) are used to balance the preference among the three errors, and $R(\theta)$ is the regularization term. Parameters θ are divided into four sets[1]:

1. θ_L: the word embedding matrix;
2. θ_{rec}: the RAE parameter matrices $W^{(1)}$, $W^{(2)}$ and bias terms $b^{(1)}$, $b^{(2)}$ (Section 2.1);
3. θ_{con}: the consistency/inconsistency score parameter matrices W_{cns}^{score}, W_{inc}^{score} (Section 3.2.2);
4. θ_{clrec}: the cross-lingual RAE semantic transformation parameter matrices $W^{(3)}$, W^{u} and bias terms $b^{(3)}$, b^{u} (Section 3.2.3).

For regularization, we assign each parameter set a unique weight:

$$\begin{aligned} R(\theta) = &\frac{\lambda_L}{2}\|\theta_L\|^2 + \frac{\lambda_{rec}}{2}\|\theta_{rec}\|^2 \\ &+ \frac{\lambda_{con}}{2}\|\theta_{con}\|^2 + \frac{\lambda_{lcrec}}{2}\|\theta_{lcrec}\|^2 \end{aligned} \tag{18}$$

Additionally, in order to prevent the hidden layer from being very small, we normalize all output vectors of the hidden layer to have length 1, $p = \frac{p}{\|p\|}$, following Socher et al. (2011c).

3.3 Model Training

Similar to Zhang et al. (2014), we adopt a co-training style algorithm to train model parameters in the following two steps:

First, we use a normal distribution ($\mu = 0, \sigma = 0.01$) to randomly initialize all model parameters, and adopt the standard RAE to pre-train source- and target-side phrase embeddings and tree structures (Section 2.1).

Second, for each bilingual phrase, we update its source-side parameters to obtain the fine-tuned vector representation and binary tree of the source phrase, given the target-side phrase structure and node representations, and vice versa. In this process, we apply L-BFGS to tune parameters based on gradients over the joint error, as implemented in (Socher et al., 2011c).

We repeat the procedure of the second step until either the joint error (shown in Eq. (17)) reaches a local minima or the number of iterations is larger than a pre-defined number (25 is used in experiments).

[1]Note that the source and target languages have different four sets of parameters.

4 Decoding with BCorrRAE

Once the model training is completed, we incorporate two different phrasal similarity features built on the trained BCorrRAE into the standard log-linear framework of SMT. Given a bilingual phrase (f, e), we first obtain their semantic phrase representations (p_f, p_e). Then we transform p_f into p'_e in the target semantic space and p_e into p'_f in the source semantic space via transformation matrixes. Finally, we reconstruct sub-trees of p'_f along the source structure T_f learned by BCorrRAE, sub-trees of p'_e along the target structure T_e.

We exploit two kinds of phrasal similarity features based on the learned phrase representations and their tree structures as follows:

- *Semantic Similarity* measures the similarity between original and transformed phrase representations of (f, e):

$$\begin{aligned} Sim_{SM}(p_f, p'_f) &= \frac{1}{2}\|p_f - p'_f\|^2 \\ Sim_{SM}(p_e, p'_e) &= \frac{1}{2}\|p_e - p'_e\|^2 \end{aligned} \tag{19}$$

- *Structural Similarity* calculates the similarity between original and reconstructed tree structures learned by BCorrRAE for (f, e):

$$\begin{aligned} Sim_{ST}(p_f, p'_f) &= \frac{1}{2C_f}\sum_{n\in T_f}\|p_n - p'_n\|^2 \\ Sim_{ST}(p_e, p'_e) &= \frac{1}{2C_e}\sum_{n\in T_e}\|p_n - p'_n\|^2 \end{aligned} \tag{20}$$

where p_n and p'_n represent vector representations of original and reconstructed node n, and C_f and C_e count the number of nodes in the source and target tree structure respectively. Note that if we only compute the similarity for root nodes in the bilingual tree of (f, e), the structural similarity equals to the semantic similarity in Eq. (19).

5 Experiments

We conducted experiments on NIST Chinese-English translation task to validate the effectiveness of BCorrRAE.

5.1 System Overview

Our baseline decoder is a state-of-the-art phrase-based translation system equipped with a maximum entropy based reordering model (MEBTG). It adopts three bracketing transduction grammar rules (Wu, 1997; Xiong et al., 2006): merging

rules $A \rightarrow [A_1, A_2]|\langle A_1, A_2\rangle$ which are used to merge two neighboring blocks[2] A_1 and A_2 in a straight|inverted order, and lexical rule $A \rightarrow f/e$ used to translate a source phrase f into a target phrase e.

The MEBTG system features a maximal entropy classifier based reordering model that predicts orientations of neighboring blocks. During training, we extract bilingual phrases containing up to 7 words on the source side from the training corpus. With the collected reordering examples, we adopt the maximal entropy toolkit[3] developed by Zhang to train the reordering model with the following parameters: iteration number $iter$=200 and gaussian prior g=1.0. Following Xiong et al. (2006), we use only boundary words of blocks to trigger the reordering model.

The whole translation model is organized in a log-linear framework (Och and Ney, 2002). The adopted sub-models mainly include: (1) rule translation probabilities in two directions, (2) lexical weights in two directions, (3) targets-side word number, (4) phrase number, (5) language model score, and (6) the score of maximal entropy based reordering model. We perform *minimum error rate training* (Och, 2003) to tune various feature weights. During decoding, we set $ttable\text{-}limit$=20 for translation candidates kept for each source phrase, $stack\text{-}size$=100 for hypotheses in each span, and $swap\text{-}span$=15 for the length of the maximal reordering span.

5.2 Setup

Our bilingual data is the combination of the FBIS corpus and Hansards part of LDC2004T07 corpus, which contains 1.0M parallel sentences (25.2M Chinese words and 29M English words). Following Zhang et al. (2014), we collected 1.44M bilingual phrases using forced decoding (Wuebker et al., 2010) to train BCorrRAE from the training data. We used a 5-gram language model trained on the Xinhua portion of Gigaword corpus using *SRILM* Toolkits[4]. Translation quality is evaluated by case-insensitive BLEU-4 metric (Papineni et al., 2002). We performed paired bootstrap sampling (Koehn, 2004) to test the significance in BLEU score differences. In our experiments, we used NIST MT05 and MT06/MT08 data set as the development and test set, respectively.

Parameter	BRAE	BCorrRAE
α	0.119	0.121
β	-	0.6331
γ	-	0.2459
λ_L	4.95 $\times 10^{-5}$	3.13 $\times 10^{-5}$
λ_{rec}	2.64 $\times 10^{-7}$	2.05 $\times 10^{-5}$
λ_{con}	-	7.32 $\times 10^{-6}$
λ_{lcrec}	9.31 $\times 10^{-5}$	5.25 $\times 10^{-6}$

Table 1: Hyper-parameters for BCorrRAE and BRAE model.

Method	d	MT06	MT08	AVG
$\mathbf{BCorrRAE}_{SM}$	25	30.81	$22.68^{\Downarrow}$	26.75
	50	$30.58^{\downarrow}$	$22.72^{\Downarrow}$	26.65
	75	30.50	$22.53^{\Downarrow}$	26.52
	100	$30.34^{\Downarrow}$	$22.61^{\Downarrow}$	26.48
$\mathbf{BCorrRAE}_{ST}$	25	30.56	23.28	26.92
	50	30.94	23.33	27.14
	75	30.73	23.40	27.07
	100	30.90	23.50	27.20

Table 2: Experiment results for different dimensions (d). BCorrRAE_{SM} and BCorrRAE_{ST} are our systems that are enhanced with the semantic and structural similarity features learned by BCorrRAE, respectively. $\downarrow/\Downarrow$: significantly worse than the BCorrRAE_{ST} with the same dimensionality ($p < 0.05/p < 0.01$).

In addition to the baseline described below, we also compare our method against the BRAE model, which focuses on modeling relations of source and target phrases as a whole unit. Word embeddings in BRAE are pre-trained with toolkit *Word2Vec*[5] (Mikolov et al., 2013) on large-scale monolingual data that contains 0.83B words for Chinese and 0.11B words for English.

Hyper-parameters in all neural models are optimized by *random search* (Bergstra and Bengio, 2012) based on related joint errors. We randomly extracted $250,000$ bilingual phrases from the above-mentioned training data as training set, $5,000$ as development set and another $5,000$ as test set. We drew α, β, γ uniformly from 0.10 to 0.50, and $\lambda_L, \lambda_{rec}, \lambda_{con}$ and λ_{lcrec} exponentially from 10^{-8} to 10^{-2}. Final parameters are shown in Table 1 for both BRAE and BCorrRAE.

5.3 Dimensionality of Embeddings

To investigate the impact of embedding dimensionality on our BCorrRAE, we tried four different dimensions from 25 to 100 with an increment of 25 each time. The results are displayed in Table 2. We can observe that the performance of our model is not consistently improved with the increment of dimensionality. This may be because a

[2]A block is a bilingual phrase without maximum length limitation.

[3]http://homepages.inf.ed.ac.uk/lzhang10/maxent_toolkit.html

[4]http://www.speech.sri.com/projects/srilm/download.html

[5]https://code.google.com/p/word2vec/

larger dimension brings in much more parameters, and therefore makes parameter tuning more difficult. In practice, setting the dimension d to 50, we can get satisfactory results without much computation effort, which has also been found by Zhang et al. (2014).

5.4 Structural Similarity vs. Semantic Similarity

Table 2 also shows that the performance of BCorrRAE$_{ST}$, the system with the structural similarity feature in Eq. (20), is always superior to that of BCorrRAE$_{SM}$ with the semantic similarity feature in Eq. (19). BCorrRAE$_{ST}$ is better than BCorrRAE$_{SM}$ by 0.483 BLEU points on average. In most cases, differences between BCorrRAE$_{ST}$ and BCorrRAE$_{SM}$ with the same dimensionality are statistically significant. This suggests that digging into structures of bilingual phrases (BCorrRAE$_{ST}$) can obtain further improvements over only modeling bilingual phrases as whole units (BCorrRAE$_{SM}$).

5.5 Overall Performance

Table 3 summarizes the comparison results of different models on the test sets. The BCorrRAE$_{SM}$ outperforms the baseline and BRAE by 1.06 and 0.25 BLEU points on average respectively, while BCorrRAE$_{ST}$ gains 1.55 and 0.74 BLEU points on average over the baseline and BRAE. The improvements of BCorrRAE$_{ST}$ over the baseline, BRAE and BCorrRAE$_{SM}$ are statistically significant at different levels. This demonstrates the advantage of our BCorrRAE over BRAE in that BCorrRAE is able to explore sub-structures of bilingual phrases.

5.6 Analysis

We compute a ratio of aligned nodes (Section 3.1) over all nodes to estimate how well tree structures of bilingual phrases generated by BRAE and BCorrRAE are consistent with word alignments. We consider two factors when computing the ratio: the length of the source side of a bilingual phrase l_s and the length of a span covered by an aligned node l_a. The result is illustrated in Table 4.[6] We find that BCorrRAE significantly outperforms BRAE model by 7.22% on average in terms of the aligned node ratio. This strongly demonstrates that the proposed BCorrRAE is able to generate tree structures that are more consistent with word alignments than those generated by BRAE.

[6]We only give ratios for bilingual phrases with source-side length from 3 to 4 words because 1) ratios of BRAE and BCorrRAE in the case of $l_a < 3$ are very close and 2) phrases with length > 4 are rarely used during decoding (accounting for $< 0.5\%$).

Method	MT06	MT08	AVG
Baseline	$29.66^{\Downarrow}$	$21.52^{\Downarrow}$	25.59
BRAE	$30.27^{\Downarrow}$	$22.53^{\Downarrow}$	26.40
BCorrRAE$_{SM}$	$30.58^{\downarrow}$	$22.72^{\Downarrow}$	26.65
BCorrRAE$_{ST}$	30.94	23.33	27.14

Table 3: Experiment results on the test sets. **AVG** = average BLEU scores for test sets. For both BRAE and BCorrRAE, we set d=50. $\downarrow/\Downarrow$: significantly worse than the BCorrRAE$_{ST}$ with d=50 ($p < 0.05$/$p < 0.01$, respectively).

$[l_s, l_a]$	[3,2]	[4,2]	[4,3]
BRAE	52.70%	39.88%	46.58%
BCorrRAE	60.08%	46.32%	54.43%

Table 4: Aligned node ratio for source phrases of different lengths.

We further show example source phrases in Table 5 with their most semantically similar translations learned by BRAE and BCorrRAE in the training corpus. Both models can select correct translations for content words. However, they are different in dealing with function words. Compared to our model, the BRAE model prefers longer target phrases surrounded with function words. Take the source phrase "严峻 挑战" as an example, the BRAE model learns both "*a serious challenge to*" and "*a serious challenge from*" as its semantically similar target phrases. Although the content words "严峻" and "挑战" are translated correctly into "*serious*" and "*challenge*", the function words "*to*" and "*from*" express exactly the opposite meanings. In contrast, our model, especially the BCorrRAE$_{ST}$ model, tends to choose shorter translations that are consistent with word alignments.

6 Related Work

A variety of efforts have been devoted to learning vector representations for words/phrases with deep neural networks. According to the difference of learning contexts, previous work mainly include the following two strands.

(1) *Monolingual Word/Phrase Embeddings*. The straightforward approach to represent word/phrases is to learn their hidden representations with traditional feature vectors, which requires manual and task-dependent feature engineering (Cui et al., 2014; Wu et al., 2014;

Source Phrase	BRAE	BCorrRAE$_{SM}$	BCorrRAE$_{ST}$
鼓吹 (advocate)	to advocate the in preaching the the promotion of	out to advocate been encouraging an advocate	encouraging claimed advocate
严峻 挑战 (serious challenge)	as well as severe challenges a serious challenge to a serious challenge from	of rigorous challenges as well as severe challenges of severe challenges	rigorous challenge enormous challenge severe challenge
公布 的 数据 (data released)	by the figures published by the the statistics released by data published by the	to the estimates announced at the figures published the statistics released by	published data released figures the estimates announced

Table 5: Semantically similar target phrases in the training set for example source phrases.

Chen and Manning, 2014). To avoid exploiting manually input features, Bengio et al. (2003) convert words to dense, real-valued vectors by learning probability distributions of n-grams. Mikolov et al. (2013) generate word vectors by predicting their limited context words. Instead of exploiting outside context information, recursive auto-encoder is usually adopted to learn the composition of internal words (Socher et al., 2010; Socher et al., 2011b; Socher et al., 2013b; Socher et al., 2013a). Recently, convolution architecture has drawn more and more attention due to its ability to explicitly capture short and long-range relations (Collobert et al., 2011; Kalchbrenner and Blunsom, 2013; Kalchbrenner et al., 2014; Kim, 2014).

(2) *Bilingual Word/Phrase Embeddings*. In the field of machine translation and cross-lingual information processing, bilingual embedding learning has become an increasingly important study. The bilingual embedding research origins in the word embedding learning, upon which Zou et al. (2013) utilize word alignments to constrain translational equivalence. Kočiský et al. (2014) propose a probability model to capture more semantic information by marginalizing over word alignments. More specifically to SMT, its main components have been exploited to learn better bilingual phrase embeddings in different aspects: language models (Wang et al., 2014; Garmash and Monz, 2014), reordering models (Li et al., 2013) and translation models (Tran et al., 2014; Zhang et al., 2014). Instead of exploiting a single model, Liu et al. (2014) combine the recursive and recurrent neural network to incorporate the language and translation model.

Different from the methods mentioned above, our model considers both the cross-language consistency of phrase structures and internal correspondence relations inside bilingual phrases. The most related works include Zhang et al. (2014) and Socher et al. (2011a). Compared with these works, our model exploits different levels of correspondence relations inside bilingual phrases instead of only the top level of entire phrases, and reconstructs tree structures of sub-phrases in one language according to aligned nodes in the other language, which, to the best of our knowledge, has never been investigated before.

7 Conclusions and Future Work

In this paper, we have presented the BCorrRAE to learn phrase embeddings and tree structures of bilingual phrases for SMT. Punishing structural-alignment-inconsistent sub-structures and minimizing the gap between original and reconstructed structures, our approach is able to not only generate alignment-consistent phrase structures, but also capture different levels of semantic relations within bilingual phrases. Experiment results demonstrate the effectiveness of our model.

In the future, we would like to derive more features from BCorrRAE, e.g., consistency/inconsistency scores of bilingual phrases, to further enhance SMT. Additionally, we also want to apply our model to other bilingual tasks, e.g., learning bilingual terminology or paraphrases.

Acknowledgments

The authors were supported by National Natural Science Foundation of China (Grant Nos 61303082 and 61403269), Natural Science Foundation of Jiangsu Province (Grant No. BK20140355), National 863 program of China (No. 2015AA011808), the Special and Major Subject Project of the Industrial Science and Technology in Fujian Province 2013 (Grant No. 2013HZ0004-1), and 2014 Key Project of Anhui Science and Technology Bureau (Grant No. 1301021018). We also thank the anonymous reviewers for their insightful comments.

References

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1139–1155.

James Bergstra and Yoshua Bengio. 2012. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13:22–29.

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

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research.*, 16:2493–2537.

Lei Cui, Dongdong Zhang, Shujie Liu, Qiming Chen, Mu Li, Ming Zhou, and Muyun Yang. 2014. Learning topic representation for smt with neural networks. In *Proc. of ACL 2014*, pages 133–143.

Jacob Devlin, Rabih Zbib, Zhongqiang Huang, Thomas Lamar, Richard Schwartz, and John Makhoul. 2014. Fast and robust neural network joint models for statistical machine translation. In *Proc. of ACL 2014*, pages 1370–1380.

Jianfeng Gao, Xiaodong He, Wen-tau Yih, and Li Deng. 2014. Learning continuous phrase representations for translation modeling. In *Proc. of ACL 2014*, pages 699–709.

Ekaterina Garmash and Christof Monz. 2014. Dependency-based bilingual language models for reordering in statistical machine translation. In *Proc. of EMNLP 2014*, pages 1689–1700.

Karl Moritz Hermann and Phil Blunsom. 2014. Multilingual models for compositional distributed semantics. In *Proc. of ACL 2014*, pages 58–68.

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

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proc. of ACL 2014*, pages 655–665.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proc. of EMNLP 2014*, pages 1746–1751.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proc. of EMNLP 2004*, pages 388–395.

Tomáš Kočiský, Karl Moritz Hermann, and Phil Blunsom. 2014. Learning bilingual word representations by marginalizing alignments. In *Proc. of ACL 2014*, pages 224–229.

Peng Li, Yang Liu, and Maosong Sun. 2013. Recursive autoencoders for ITG-based translation. In *Proc. of EMNLP 2013*, pages 567–577.

Lemao Liu, Taro Watanabe, Eiichiro Sumita, and Tiejun Zhao. 2013. Additive neural networks for statistical machine translation. In *Proc. of ACL 2013*, pages 791–801.

Shujie Liu, Nan Yang, Mu Li, and Ming Zhou. 2014. A recursive recurrent neural network for statistical machine translation. In *Proc. of ACL 2014*, pages 1491–1500.

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

Franz Josef Och and Hermann Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *Proc. of ACL 2002*, pages 295–302.

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

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proc. of ACL 2003*, pages 160–167. Association for Computational Linguistics.

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

Richard Socher, Christopher D. Manning, and Andrew Y. Ng. 2010. Learning continuous phrase representations and syntactic parsing with recursive neural networks. In *Proc. of NIPS 2010*.

Richard Socher, Eric H. Huang, Jeffrey Pennington, Andrew Y. Ng, and Christopher D. Manning. 2011a. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Proc. of NIPS 2011*.

Richard Socher, Cliff Chiung-Yu Lin, Andrew Y. Ng, and Christopher D. Manning. 2011b. Parsing natural scenes and natural language with recursive neural networks. In *Proc. of ICML 2011*.

Richard Socher, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher D. Manning. 2011c. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proc. of EMNLP 2011*, pages 151–161.

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

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013b. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proc. of EMNLP 2013*, pages 1631–1642.

Martin Sundermeyer, Tamer Alkhouli, Joern Wuebker, and Hermann Ney. 2014. Translation modeling with bidirectional recurrent neural networks. In *Proc. of EMNLP 2014*, pages 14–25.

Akihiro Tamura, Taro Watanabe, and Eiichiro Sumita. 2014. Recurrent neural networks for word alignment model. In *Proc. of ACL 2014*, pages 1470–1480.

Ke M. Tran, Arianna Bisazza, and Christof Monz. 2014. Word translation prediction for morphologically rich languages with bilingual neural networks. In *Proc. of EMNLP 2014*, pages 1676–1688.

Rui Wang, Hai Zhao, Bao-Liang Lu, Masao Utiyama, and Eiichiro Sumita. 2014. Neural network based bilingual language model growing for statistical machine translation. In *Proc. of EMNLP 2014*, pages 189–195.

Haiyang Wu, Daxiang Dong, Xiaoguang Hu, Dianhai Yu, Wei He, Hua Wu, Haifeng Wang, and Ting Liu. 2014. Improve statistical machine translation with context-sensitive bilingual semantic embedding model. In *Proc. of EMNLP 2014*, pages 142–146.

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

Joern Wuebker, Arne Mauser, and Hermann Ney. 2010. Training phrase translation models with leaving-one-out. In *Proc. of ACL 2010*, pages 475–484.

Deyi Xiong, Qun Liu, and Shouxun Lin. 2006. Maximum entropy based phrase reordering model for statistical machine translation. In *Proc. of ACL 2006*, pages 521–528.

Nan Yang, Shujie Liu, Mu Li, Ming Zhou, and Nenghai Yu. 2013. Word alignment modeling with context dependent deep neural network. In *Proc. of ACL 2013*, pages 166–175.

Jiajun Zhang, Shujie Liu, Mu Li, Ming Zhou, and Chengqing Zong. 2014. Bilingually-constrained phrase embeddings for machine translation. In *Proc. of ACL 2014*, pages 111–121.

Will Y. Zou, Richard Socher, Daniel Cer, and Christopher D. Manning. 2013. Bilingual word embeddings for phrase-based machine translation. In *Proc. of EMNLP 2013*, pages 1393–1398.

How to Avoid Unwanted Pregnancies: Domain Adaptation using Neural Network Models

Shafiq Joty, Hassan Sajjad, Nadir Durrani,
Kamla Al-Mannai, Ahmed Abdelali and Stephan Vogel
Qatar Computing Research Institute - Hamad Bin Khalifa University
{sjoty,hsajjad,ndurrani,kmannai,aabdelali,svogel}@qf.org.qa

Abstract

We present novel models for domain adaptation based on the neural network joint model (NNJM). Our models maximize the cross entropy by regularizing the loss function with respect to in-domain model. Domain adaptation is carried out by assigning higher weight to out-domain sequences that are similar to the in-domain data. In our alternative model we take a more restrictive approach by additionally penalizing sequences similar to the out-domain data. Our models achieve better perplexities than the baseline NNJM models and give improvements of up to 0.5 and 0.6 BLEU points in Arabic-to-English and English-to-German language pairs, on a standard task of translating TED talks.

1 Introduction

Rapid influx of digital data has galvanized the use of empirical methods in many fields including Machine Translation (MT). The increasing availability of bilingual corpora has made it possible to automatically learn translation rules that required years of linguistic analysis previously. While additional data is often beneficial for a general purpose Statistical Machine Translation (SMT) system, a problem arises when translating new domains such as lectures (Cettolo et al., 2014), patents (Fujii et al., 2010) or medical text (Bojar et al., 2014), where either the bilingual text does not exist or is available in small quantity. All domains have their own vocabulary and stylistic preferences which cannot be fully encompassed by a system trained on the general domain.

Machine translation systems trained from a simple concatenation of small in-domain and large out-domain data often perform below par because the out-domain data is distant or overwhelmingly larger than the in-domain data. Additional data increases lexical ambiguity by introducing new senses to the existing in-domain vocabulary. For example, an Arabic-to-English SMT system trained by simply concatenating in- and out-domain data translates the Arabic phrase "عن مشكلة الحمل الزائد للاختيار" to "about the problem of unwanted pregnancy". This translation is incorrect in the context of the in-domain data, where it should be translated to "about the problem of choice overload". The sense of the Arabic phrase taken from out-domain data completely changes the meaning of the sentence. In this paper, we tackle this problem by proposing domain adaptation models that make use of all the data while preserving the in-domain preferences.

A significant amount of research has been carried out recently in domain adaptation. The complexity of the SMT pipeline, starting from corpus preparation to word-alignment, and then training a wide range of models opens a wide horizon to carry out domain specific adaptations. This is typically done using either *data selection* (Matsoukas et al., 2009) or *model adaptation* (Foster and Kuhn, 2007). In this paper, we further research in model adaptation using the neural network framework.

In recent years, there has been a growing interest in deep neural networks (NNs) and word embeddings with application to numerous NLP problems. A notably successful attempt on the SMT frontier was recently made by Devlin et al. (2014). They proposed a neural network joint model (NNJM), which augments streams of source with target n-grams and learns a NN model over vector representation of such streams. The model is then integrated into the decoder and used as an additional language model feature.

Our aim in this paper is to advance the state-of-the-art in SMT by extending NNJM for domain adaptation to leverage the huge amount of out-

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1259–1270,
Lisbon, Portugal, 17-21 September 2015.

domain data coming from heterogeneous sources. We hypothesize that the distributed vector representation of NNJM helps to bridge the lexical differences between the in-domain and the out-domain data, and adaptation is necessary to avoid deviation of the model from the in-domain data, which otherwise happens because of the large out-domain data.

To this end, we propose two novel extensions of NNJM for domain adaptation. Our first model minimizes the cross entropy by regularizing the loss function with respect to the in-domain model. The regularizer gives higher weight to the training instances that are similar to the in-domain data. Our second model takes a more conservative approach by additionally penalizing data instances similar to the out-domain data.

We evaluate our models on the standard task of translating Arabic-English and English-German language pairs. Our adapted models achieve better perplexities (Chen and Goodman, 1999) than the models trained on in- and in+out-domain data. Improvements are also reflected in BLEU scores (Papineni et al., 2002) as we compare these models within the SMT pipeline. We obtain gains of up to 0.5 and 0.6 on Arabic-English and English-German pairs over a competitive baseline system.

The remainder of this paper is organized as follows: Section 2 gives an account on related work. Section 3 revisits NNJM model and Section 4 discusses our models. Section 5 presents the experimental setup and the results. Section 6 concludes.

2 Related Work

Previous work on domain adaptation in MT can be broken down broadly into two main categories namely *data selection* and *model adaptation*.

2.1 Data Selection

Data selection has shown to be an effective way to discard poor quality or irrelevant training instances, which when included in an MT system, hurts its performance. The idea is to score the out-domain data using a model trained from the in-domain data and apply a cut-off based on the resulting scores. The MT system can then be trained on a subset of the out-domain data that is closer to in-domain. Selection based methods can be helpful to reduce computational cost when training is expensive and also when memory is constrained. Data selection was done earlier for language modeling using information retrieval techniques (Hildebrand et al., 2005) and perplexity measures (Moore and Lewis, 2010). Axelrod et al. (2011) further extended the work of Moore and Lewis (2010) to translation model adaptation by using both source- and target-side language models. Duh et al. (2013) used a recurrent neural language model instead of an ngram-based language model to do the same. Translation model features were used recently by (Liu et al., 2014; Hoang and Sima'an, 2014) for data selection. Durrani et al. (2015a) performed data selection using operation sequence model (OSM) and NNJM models.

2.2 Model Adaptation

The downside of data selection is that finding an optimal cut-off threshold is a time consuming process. An alternative to completely filtering out less useful data is to minimize its effect by down-weighting it. It is more robust than selection since it takes advantage of the complete out-domain data with intelligent weighting towards the in-domain.

Matsoukas et al. (2009) proposed a classification-based sentence weighting method for adaptation. Foster et al. (2010) extended this by weighting phrases rather than sentence pairs. Other researchers have carried out weighting by merging phrase-tables through linear interpolation (Finch and Sumita, 2008; Nakov and Ng, 2009) or log-linear combination (Foster and Kuhn, 2009; Bisazza et al., 2011; Sennrich, 2012) and through phrase training based adaptation (Mansour and Ney, 2013). Durrani et al. (2015a) applied EM-based mixture modeling to OSM and NNJM models to perform model weighting. Chen et al. (2013b) used a vector space model for adaptation at the phrase level. Every phrase pair is represented as a vector, where every entry in the vector reflects its relatedness with each domain. Chen et al. (2013a) also applied mixture model adaptation for reordering model.

Other work on domain adaptation includes but not limited to studies focusing on topic models (Eidelman et al., 2012; Hasler et al., 2014), dynamic adaptation without in-domain data (Sennrich et al., 2013; Mathur et al., 2014) and sense disambiguation (Carpuat et al., 2013).

In this paper, we do model adaptation using a neural network framework. In contrast to previous work, we perform it at the (bilingual) n-gram level, where n is sufficiently large to capture long-range cross-lingual dependencies. The

generalized vector representation of the neural network model reduces the data sparsity issue of traditional Markov-based models by learning better word classes. Furthermore, our specially designed loss functions for adaptation help the model to avoid deviation from the in-domain data without losing the ability to generalize.

3 Neural Network Joint Model

In recent years, there has been a great deal of effort dedicated to neural networks (NNs) and word embeddings with applications to SMT and other areas in NLP (Bengio et al., 2003; Auli et al., 2013; Kalchbrenner and Blunsom, 2013; Gao et al., 2014; Schwenk, 2012; Collobert et al., 2011; Mikolov et al., 2013a; Socher et al., 2013; Hinton et al., 2012). Recently, Devlin et al. (2014) proposed a neural network joint model (NNJM) and integrated it into the decoder as an additional feature. They showed impressive improvements in Arabic-to-English and Chinese-to-English MT tasks. Let us revisit the NNJM model briefly.

Given a source sentence S and its corresponding target sentence T, the NNJM model computes the conditional probability $P(T|S)$ as follows:

$$P(T|S) \approx \prod_{i}^{|T|} P(t_i|t_{i-1} \dots t_{i-p+1}, \mathbf{s_i}) \quad (1)$$

where, $\mathbf{s_i}$ is a q-word source window for the target word t_i based on the one-to-one (non-NULL) alignment of T to S. As exemplified in Figure 1, this is essentially a $(p+q)$-gram neural network LM (NNLM) originally proposed by Bengio et al. (2003). Each input word i.e. source or target word in the context is represented by a D dimensional vector in the shared look-up layer $L \in \mathbb{R}^{|V_i| \times D}$, where V_i is the input vocabulary.[1] The look-up layer then creates a context vector $\mathbf{x_n}$ representing the context words of the $(p+q)$-gram sequence by concatenating their respective vectors in L. The concatenated vector is then passed through non-linear hidden layers to learn a high-level representation, which is in turn fed to the output layer. The output layer has a `softmax` activation over the output vocabulary V_o of target words. Formally, the probability of getting k-th word in the output given the context $\mathbf{x_n}$ can be written as:

$$P(y_n = k|\mathbf{x_n}, \theta) = \frac{exp\ (\mathbf{w}_k^T \phi(\mathbf{x_n}))}{\sum_{m=1}^{|V_o|} exp\ (\mathbf{w}_m^T \phi(\mathbf{x_n}))} \quad (2)$$

[1]Note that L is a model parameter to be learned.

where $\phi(\mathbf{x_n})$ defines the transformations of $\mathbf{x_n}$ through the hidden layers, and $\mathbf{w}_k$ are the weights from the last hidden layer to the output layer. For notational simplicity, henceforth we will use $(\mathbf{x_n}, y_n)$ to represent a training sequence.

By setting p and q to be sufficiently large, NNJM can capture long-range cross-lingual dependencies between words, while still overcoming the data sparseness issue by virtue of its distributed representations (i.e., word vectors). A major bottleneck, however, is to surmount the computational cost involved in training the model and applying it for MT decoding. Devlin et al. (2014) proposed two tricks to speed up computation in decoding. The first one is to pre-compute the hidden layer computations and fetch them directly as needed during decoding. The second technique is to train a *self-normalized* NNJM to avoid computation of the softmax normalization factor (i.e., the denominator in Equation 2) in decoding. However, self-normalization does not solve the computational cost of training the model. In the following, we describe a method to address this issue.

3.1 Training by Noise Contrastive Estimation

The standard way to train NNLMs is to maximize the log likelihood of the training data:

$$J(\theta) = \sum_{n=1}^{N} \sum_{k=1}^{|V_o|} y_{nk}\ log\ P(y_n = k|\mathbf{x_n}, \theta) \quad (3)$$

where, $y_{nk} = I(y_n = k)$ is an indicator variable (i.e., y_{nk}=1 when y_n=k, otherwise 0). Optimization is performed using first-order online methods, such as stochastic gradient ascent (SGA) with standard *backpropagation* algorithm. Unfortunately, training NNLMs are impractically slow because for each training instance $(\mathbf{x_n}, y_n)$, the softmax output layer (see Equation 2) needs to compute a summation over all words in the output vocabulary.[2] Noise contrastive estimation or NCE (Gutmann and Hyvärinen, 2010) provides an efficient and stable way to avoid this repetitive computation as recently applied to NNLMs (Vaswani et al., 2013; Mnih and Teh, 2012). We can re-write Equation 2 as follows:

$$P(y_n = k|\mathbf{x_n}, \theta) = \frac{\sigma(y_n = k|\mathbf{x_n}, \theta)}{Z(\phi(\mathbf{x_n}), \mathbf{W})} \quad (4)$$

where $\sigma(.)$ is the un-normalized score and $Z(.)$ is the normalization factor. In NCE, we consider

[2]This would take few weeks for a modern CPU machine to train a single NNJM model on the whole data.

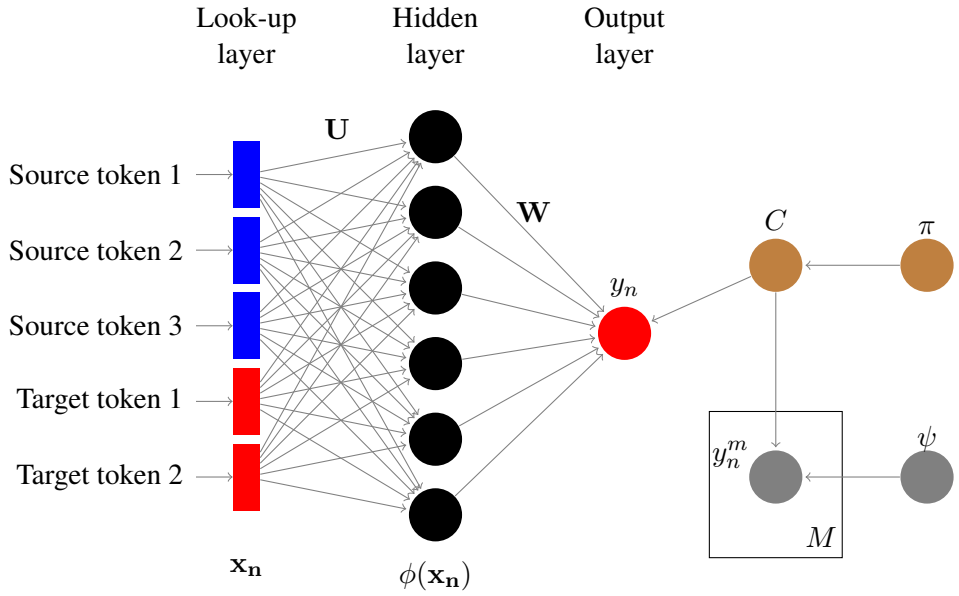


Figure 1: A simplified neural network joint model with noise contrastive loss, where we use 3-gram target words (i.e., 2-words history) and a source context window of size 3. For illustration, the output y_n is shown as a single categorical variable (scalar) as opposed to the traditional one-hot vector representation.

$Z(.)$ as an additional model parameter along with the regular parameters, i.e., weights, look-up vectors. However, it has been shown that fixing $Z(.)$ to 1 instead of learning it in training does not affect the model performance (Mnih and Teh, 2012).

For each training instance $(\mathbf{x_n}, y_n)$, we add M noise samples $(\mathbf{x_n}, y_n^m)$ by sampling y_n^m from a known noise distribution ψ (e.g., `unigram`, `uniform`) M many times (i.e., $m = 1 \ldots M$); see Figure 1. NCE loss is then defined to discriminate a *true* instance from a *noisy* one. Let $C \in \{0, 1\}$ denote the class of an instance with $C = 1$ indicating *true* and $C = 0$ indicating *noise*. NCE maximizes the following conditional log likelihood:

$$J(\theta) = \sum_{n=1}^{N} \Big[log[P(C=1|y_n, \mathbf{x}_n, \theta)] + \sum_{m=1}^{M} log[P(C=0|y_n^m, \mathbf{x}_n, \psi)] \Big] \quad (5)$$

$$= \sum_{n=1}^{N} \Big[log\left[P(y_n|C=1, \mathbf{x}_n, \theta) P(C=1|\pi)\right] + \sum_{m=1}^{M} log\left[\left(P(y_n^m|C=0, \mathbf{x}_n, \psi)\right) P(C=0|\pi)\right] - (M+1)\, log\, Q \Big] \quad (6)$$

where $Q = P(y_n, C=1|\mathbf{x}_n, \theta, \pi) + P(y_n^m, C=0|\mathbf{x}_n, \psi, \pi)$ is a normalization constant. After removing the constant terms, Equation 6 can be further simplified as:

$$J(\theta) = \sum_{n=1}^{N} \sum_{k=1}^{|V_o|} \Big[y_{nk}\, log\, \sigma_{nk} + \sum_{m=1}^{M} y_{nk}^m\, log\, \psi_{nk} \Big] \quad (7)$$

where $\psi_{nk} = P(y_n^m = k|\mathbf{x}_n, \psi)$ is the noise distribution, $\sigma_{nk} = \sigma(y_n = k|\mathbf{x}_n, \theta)$ is the unnormalized score at the output layer (Equation 4), and y_{nk} and y_{nk}^m are indicator variables as defined before. NCE reduces the number of computations needed at the output layer from $|V_o|$ to $M + 1$, where M is a small number in comparison with $|V_o|$. In all our experiments we use NCE loss with $M = 100$ samples as suggested by Mnih and Teh (2012).

4 Neural Domain Adaptation Models

The ability to generalize and learn complex semantic relationships (Mikolov et al., 2013b) and its compelling empirical results gives a strong motivation to use the NNJM model for the problem of domain adaptation in machine translation. However, the vanilla NNJM described above is limited in its ability to effectively learn from a large and diverse out-domain data in the best favor of an in-domain data. To address this, we propose two neural domain adaptation models (NDAM) extending the NNJM model. Our models add regularization to its loss function either with respect to in-domain or both in- and out-domains. In both cases, we first present the regularized loss function for the normalized output layer with the standard softmax,

followed by the corresponding un-normalized one using the noise contrastive estimation.

4.1 NDAM$_{v1}$

To improve the generalization of word embeddings, NNLMs are generally trained on very large datasets (Mikolov et al., 2013a; Vaswani et al., 2013). Therefore, we aim to train our neural domain adaptation models (NDAM) on in- plus out-domain data, while restricting it to drift away from in-domain. In our first model NDAM$_{v1}$, we achieve this by biasing the model towards the in-domain using a regularizer (or prior) based on the in-domain model. Let θ_i be an NNJM model already trained on the in-domain data. We train an adapted model θ_a on the whole data, but regularizing it with respect to θ_i. We redefine the normalized loss function of Equation 3 as follows:

$$J(\theta_a) = \sum_{n=1}^{N}\sum_{k=1}^{|V_o|}\Big[\lambda\, y_{nk}\, logP(y_n = k|\mathbf{x_n}, \theta_a) + (1-\lambda)\, y_{nk}\, P(y_n = k|\mathbf{x_n}, \theta_i)\, logP(y_n = k|\mathbf{x_n}, \theta_a)\Big] \quad (8)$$

$$= \sum_{n=1}^{N}\sum_{k=1}^{|V_o|}\Big[\lambda\, y_{nk}\, log\, \hat{y}_{nk}(\theta_a) + (1-\lambda)\, y_{nk}\, p_{nk}(\theta_i)\, log\, \hat{y}_{nk}(\theta_a)\Big] \quad (9)$$

where $\hat{y}_{nk}(\theta_a)$ is the softmax output and $p_{nk}(\theta_i)$ is the probability of the training instance according to the in-domain model θ_i. Notice that the loss function minimizes the cross entropy of the current model θ_a with respect to the gold labels y_n and the in-domain model θ_i. The mixing parameter $\lambda \in [0, 1]$ determines the relative strength of the two components.[3] Similarly, we can re-define the NCE loss of Equation 7 as:

$$J(\theta_a) = \sum_{n=1}^{N}\sum_{k=1}^{|V_o|}\Big[\lambda\, y_{nk}\, log\, \sigma_{nk} + (1-\lambda)\, y_{nk}\, p_{nk}(\theta_i)\, log\, \sigma_{nk} + \sum_{m=1}^{M} y_{nk}^{m} log\, \psi_{nk}\Big] \quad (10)$$

We use SGA with backpropagation to train this model. The derivatives of $J(\theta_a)$ with respect to the final layer weight vectors $\mathbf{w}_j$ turn out to be:

$$\nabla_{\mathbf{w}_j} J(\theta_a) = \sum_{n=1}^{N}\Big[\lambda\,(y_{nj} - \sigma_{nj}) + (1-\lambda)\,[p_{nj}(\theta_i) - \sum_{k} y_{nk}\, p_{nk}(\theta_i)\, \sigma_{nj}]\Big] \quad (11)$$

[3] We used a balanced value $\lambda = 0.5$ for our experiments.

4.2 NDAM$_{v2}$

The regularizer in NDAM$_{v1}$ is based on an in-domain model θ_i, which puts higher weights to the training instances (i.e., n-gram sequences) that are similar to the in-domain ones. This might work better when the out-domain data is similar to the in-domain data. In cases where the out-domain data is different, we might want to build a more conservative model that penalizes training instances for being similar to the out-domain ones.

Let θ_i and θ_o be the two NNJMs already trained from the in- and out-domains, respectively, and θ_o is trained using the same vocabulary as θ_i. We define the new normalized loss function as follows:

$$J(\theta_a) = \sum_{n=1}^{N}\sum_{k=1}^{|V_o|}\Big[\lambda\, y_{nk}\, log\, \hat{y}_{nk}(\theta_a) + (1-\lambda)\, y_{nk}\, [p_{nk}(\theta_i) - p_{nk}(\theta_o)]\, log\, \hat{y}_{nk}(\theta_a)\Big] \quad (12)$$

where y_{nk}, $\hat{y}_{nk}(\theta_a)$, $p_{nk}(\theta_i)$ and $p_{nk}(\theta_o)$ are similarly defined as before. This loss function minimizes the cross entropy of the current model θ_a with respect to the gold labels y_n and the difference between the in-domain model θ_i and the out-domain model θ_o. Intuitively, the regularizer assigns higher weights to training instances that are not only similar to the in-domain but also dissimilar to the out-domain. The parameter $\lambda \in [0, 1]$ determines the strength of the regularization. The corresponding NCE loss can be defined as follows:

$$J(\theta_a) = \sum_{n=1}^{N}\sum_{k=1}^{|V_o|}\Big[\lambda\, y_{nk}\, log\, \sigma_{nk} + (1-\lambda)\, y_{nk}\, log\, \sigma_{nk}\, (p_{nk}(\theta_i) - p_{nk}(\theta_o)) + \sum_{m=1}^{M} y_{nk}^{m}\, log\, \psi_{nk}\Big] \quad (13)$$

The derivatives of the above cost function with respect to the final layer weight vectors $\mathbf{w}_j$ are:

$$\nabla_{\mathbf{w}_j} J(\theta_a) = \sum_{n=1}^{N}\Big[\lambda\,(y_{nj} - \sigma_{nj}) + (1-\lambda)[p_{nj}(\theta_i) - p_{nj}(\theta_o) - \sum_{k} y_{nk}\, \sigma_{nj}\; (p_{nk}(\theta_i) - p_{nk}(\theta_o))]\Big] \quad (14)$$

In a way, the regularizers in our loss functions are inspired from the data selection methods of Axelrod et al. (2011), where they use cross entropy between the in- and the out-domain LMs to score out-domain sentences. However, our approach is quite different from them in several aspects. First

and most importantly, we take the scoring inside model training and use it to bias the training towards the in-domain model. Both the scoring and the training are performed at the bilingual n-gram level rather than at the sentence level. Integrating scoring inside the model allows us to learn a robust model by training/tuning the relevant parameters, while still using the complete data. Secondly, our models are based on NNs, while theirs utilize the traditional Markov-based generative models.

4.3 Technical Details

In this section, we describe some implementation details of NDAM that we found to be crucial, such as: using *gradient clipping* to handle vanishing/exploding gradient problem in SGA training with backpropagation, selecting appropriate *noise distribution* in NCE, and special handling of out-domain words that are unknown to the in-domain.

4.3.1 Gradient Clipping

Two common issues with training deep NNs on large data-sets are the *vanishing* and the *exploding* gradients problems (Pascanu et al., 2013). The error gradients propagated by the backpropagation may sometimes become very small or very large which can lead to undesired (nan) values in weight matrices, causing the training to fail. We also experienced the same problem in our NDAM quite often. One simple solution to this problem is to truncate the gradients, known as *gradient clipping* (Mikolov, 2012). In our experiments, we limit the gradients to be in the range $[-5; +5]$.

4.3.2 Noise Distribution in NCE

Training with NCE relies on sampling from a noise distribution (i.e., ψ in Equation 5), and the performance of the NDAM models varies considerably with the choice of the distribution. We explored *uniform* and *unigram* noise distributions in this work. With uniform distribution, every word in the output vocabulary has the same probability to be sampled as noise. The *unigram* noise distribution is a multinomial distribution over words constructed by counting their occurrences in the output (i.e., n-th word in the n-gram sequence). In our experiments, unigram distribution delivered much lower perplexity and better MT results compared to the uniform one. Mnih and Teh (2012) also reported similar findings on perplexity.

4.3.3 Handling of Unknown Words

In order to reduce the training time and to learn better word representations, NNLMs are often trained on most frequent vocabulary words only and low frequency words are represented under a class of unknown words, `unk`. This results in a large number of n-gram sequences containing at least one `unk` word and thereby, makes `unk` a highly probable word in the model.[4]

Our NDAM models rely on scoring out-domain sequences (of word Ids) using models that are trained based on the in-domain vocabulary. To score out-domain sequences using a model, we need to generate the sequences using the same vocabulary based on which the model was trained. In doing so, the out-domain words that are unknown to the in-domain data map to the same `unk` class. As a result, out-domain sequences containing `unks` get higher probability although they are distant from the in-domain data.

A solution to this problem is to have an in-domain model that can differentiate between its own `unk` class, resulted from the reduced in-domain vocabulary, and actual unknown words that come from the out-domain data. We introduce a new class $\texttt{unk}_o$ to represent the latter. We train the in-domain model by adding a few dummy sequences containing $\texttt{unk}_o$ occurring on both source and target sides. This enables the model to learn `unk` and $\texttt{unk}_o$ separately, where $\texttt{unk}_o$ is a less probable class according to the model. Later, the n-gram sequences of the out-domain data contain both `unk` and $\texttt{unk}_o$ classes depending on whether a word is unknown to only pruned in-domain vocabulary (i.e., `unk`) or is unknown to full in-domain vocabulary (i.e., $\texttt{unk}_o$).

5 Evaluation

In this section, we describe the experimental setup (i.e., data, settings for NN models and MT pipeline) and the results. First we evaluate our models intrinsically by comparing the perplexities on a held-out in-domain testset against the baseline NNJM model. Then we carry out an extrinsic evaluation by using the NNJM and NDAM models as features in machine translation and compare the BLEU scores. Initial developmental experiments were done on the Arabic-to-English language pair.

[4]For our Arabic-English in-domain data, 30% of n-gram sequences contain at least one `unk` word.

We carried out further experiments on the English-to-German pair to validate our models.

5.1 Data

We experimented with the data made publicly available for the translation task of the International Workshop on Spoken Language Translation (IWSLT) (Cettolo et al., 2014). We used TED talks as our in-domain corpus. For Arabic-to-English, we used the QCRI Educational Domain (QED) – A bilingual collection of educational lectures[5] (Abdelali et al., 2014), the News, and the multiUN (UN) (Eisele and Chen, 2010) as our out-domain corpora. For English-to-German, we used the News, the Europarl (EP), and the Common Crawl (CC) corpora made available for the 9^{th} Workshop of Statistical Machine Translation.[6] Table 1 shows the size of the data used.

Training NN models is expensive. We, therefore, randomly selected subsets of about 300K sentences from the bigger domains (UN, CC and EP) to train the NN models.[7] The systems were tuned on concatenation of the dev. and test2010 and evaluated on test2011-2013 datasets. The tuning set was also used to measure the perplexities of different models.

AR-EN			EN-DE		
Corpus	Sent.	Tok.	Corpus	Sent.	Tok.
IWSLT	150k	2.8/3.0	IWSLT	177K	3.5/3.3
QED	150k	1.4/1.5	CC	2.3M	57/53
NEWS	203k	5.6/6.3	NEWS	200K	2.8/3.4
UN	3.7M	129/125	EP	1.8M	51/48

Table 1: Statistics of the Arabic-English and English-German training corpora in terms of Sentences and Tokens (Source/Target). Tokens are represented in millions.

5.2 System Settings

NNJM & NDAM: The NNJM models were trained using NPLM[8] toolkit (Vaswani et al., 2013) with the following settings. We used a target context of 5 words and an aligned source window of 9 words, forming a joint stream of 14-grams for training. We restricted source and target side vocabularies to the 20K and 40K most frequent words. The word vector size D and the hidden layer size were set to 150 and 750, respectively. Only one hidden layer is used to allow faster decoding. Training was done by the standard stochastic gradient ascent with NCE using 100 noise samples and a mini-batch size of 1000. All models were trained for 25 epochs. We used identical settings to train the NDAM models, except for the special handling of `unk` tokens.

Machine Translation System: We trained a Moses system (Koehn et al., 2007), with the following settings: a maximum sentence length of 80, Fast-Aligner for word-alignments (Dyer et al., 2013), an interpolated Kneser-Ney smoothed 5-gram language model with KenLM (Heafield, 2011), lexicalized reordering model (Galley and Manning, 2008), a 5-gram operation sequence model (Durrani et al., 2015b) and other default parameters. We also used an NNJM trained with the settings described above as an additional feature in our baseline system. In adapted systems, we replaced the NNJM model with the NDAM models. We used ATB segmentation using the Stanford ATB segmenter (Green and DeNero, 2012) for Arabic-to-English and the default tokenizer provided with the Moses toolkit (Koehn et al., 2007) for the English-to-German pair. Arabic OOVs were translated using an unsupervised transliteration module in Moses (Durrani et al., 2014). We used k-best batch MIRA (Cherry and Foster, 2012) for tuning.

5.3 Intrinsic Evaluation

In this section, we compare the NNJM model and our NDAM models in terms of their perplexity numbers on the in-domain held-out dataset (i.e., dev+test2010). We choose Arabic-English language pair for the development experiments and train domain-wise models to measure the relatedness of each domain with respect to the in-domain. We later replicated selective experiments for the English-German language pair.

The first part of Table 2 summarizes the results for Arabic-English. The perplexity numbers in the second column (NNJM$_b$) show that NEWS is the

[5]Guzmán et al. (2013) showed that the QED corpus is similar to IWSLT and adding it improves translation quality.

[6]http://www.statmt.org/wmt14/translation-task.html

[7]Concatenating all the data results in a corpus of approximately 4.5 million sentences which requires roughly 18 days of wall-clock time (18 hours/epoch on a Linux Ubuntu 12.04.5 LTS running on a 16 Core Intel Xeon E5-2650 2.00Ghz and 64Gb RAM) to train NNJM models on our machines. We ran one baseline experiment with all the data and did not find it better than the system trained on randomly selected subset of the data. In the interest of time, we therefore reduced the NN training to a subset (800K and 1M sentences for AR-EN and EN-DE respectively).

[8]http://nlg.isi.edu/software/nplm/

Domain	NNJM$_b$	NNJM$_{cat}$	NDAM$_{v1}$	NDAM$_{v2}$
Arabic-English				
IWSLT	12.55	-	-	-
QED	61.34	11.72	**11.14**	11.15
NEWS	42.88	10.88	10.67	**10.59**
UN	111.11	11.25	10.83	**10.74**
ALL	-	10.31	**10.08**	10.22
English-German				
IWSLT	10.20	–	–	–
ALL	-	6.71	**6.21**	**6.37**

Table 2: Comparing the perplexity of NNJM and NDAM models. NNJM$_b$ represents the model trained on each individual domain separately.

most related domain from the perspective of in-domain data, whereas UN is the farthest having the worst perplexity. The third column (NNJM$_{cat}$) shows results of the models trained from concatenating each domain to the in-domain data. The perplexity numbers improved significantly in each case showing that there is useful information available in each domain which can be utilized to improve the baseline. It also shows the robustness of neural network models. Unlike the n-gram model, the NN-based model improves generalization with the increase in data without completely skewing towards the dominating part of the data.

Concatenating in-domain with the NEWS data gave better perplexities than other domains. Best results were obtained by concatenating all the data together (See row *ALL*). The third and fourth columns show results of our models (NDAM$_{v*}$). Both give better perplexities than NNJM$_{cat}$ in all cases. However, it is unclear which of the two is better. Similar observations were made for the English-to-German pair, where we only did experiments on the concatenation of all domains.

5.4 Extrinsic Evaluation

Arabic-to-English: For most language pairs, the conventional wisdom is to train the system with all available data. However, previously reported MT results on Arabic-to-English (Mansour and Ney, 2013) show that this is not optimal and the results are often worse than only using in-domain data. The reason for this is that the UN domain is found to be distant and overwhelmingly large as compared to the in-domain IWSLT data. We carried out domain-wise experiments and also found this to be true.

We considered three baseline systems: (*i*) B$_{in}$, which is trained on the in-domain data, (*ii*) B$_{cat}$, which is trained on the concatenation of in- and out-domain data, and (*ii*) B$_{cat,in}$, where the MT pipeline was trained on the concatenation but the NNJM model is trained only on the in-domain data. Table 3 reports average BLEU scores across three test sets on all domains. Adding QED and NEWS domains gave improvements on top of the in-domain IWSLT baseline. Concatenation of UN with in-domain made the results worse. Concatenating all out-domain and in-domain data achieves +0.4 BLEU gain on top of the baseline in-domain system. We will use B$_{cat}$ systems as our baseline to compare our adapted systems with.

SYS	IWSLT	QED	NEWS	UN	ALL
B$_{in}$	26.1	-	-	-	-
B$_{cat}$	-	26.2	26.7	25.8	26.5
B$_{cat,in}$	-	26.2	26.3	25.9	26.5

Table 3: Results of the baseline Arabic-to-English MT systems. The numbers are averaged over tst2011-2013.

Table 4 shows results of the MT systems S$_{v1}$ and S$_{v2}$ using our adapted models NDAM$_{v1}$ and NDAM$_{v2}$. We compare them to the baseline system B$_{cat}$, which uses the non-adapted NNJM$_{cat}$ as a feature. S$_{v1}$ achieved an improvement of up to +0.4 and S$_{v2}$ achieved an improvement of up to +0.5 BLEU points. However, S$_{v2}$ performs slightly worse than S$_{v1}$ on individual domains. We speculate this is because of the nature of the NDAM$_{v2}$, which gives high weight to out-domain sequences that are liked by the in-domain model and disliked by the out-domain model. In the case of individual domains, NDAM$_{v2}$ might be over penalizing out-domain since the out-domain model is only built on that particular domain and always prefers it more than the in-domain model. In case of *ALL*, the out-domain model is more diverse and has different level of likeness for each domain.

We analyzed the output of the baseline system (S$_{cat}$) and spotted several cases of lexical ambiguity caused by out-domain data. For example, the Arabic phrase الحمل الزائد للاختيار can be translated to *choice overload* or *unwanted pregnancy*. The latter translation is incorrect in the context of in-domain. The bias created due to the out-domain data caused S$_{cat}$ to choose the contextually incorrect translation *unwanted pregnancy*. However, the adapted systems S$_{v*}$ were able to translate it

	QED			NEWS			UN			ALL		
	tst11	tst12	tst13	tst11	tst12	tst13	tst11	tst12	tst13	tst11	tst12	tst13
B_{cat}	25.0	27.3	26.2	25.4	27.6	27.1	24.7	27.0	25.8	25.0	27.5	27.0
S_{v1}	25.2	27.7	26.2	25.8	27.8	27.3	24.7	27.5	26.1	25.3	27.8	27.0
Δ	**+0.2**	**+0.4**	0.0	**+0.4**	**+0.2**	**+0.2**	0.0	**+0.5**	**+0.3**	**+0.3**	**+0.2**	0.0
S_{v2}	25.1	27.6	26.2	25.6	27.9	27.2	24.6	27.2	26.1	25.5	27.9	26.9
Δ	+0.1	**+0.3**	0.0	**+0.2**	**+0.3**	+0.1	-0.1	**+0.2**	**+0.3**	**+0.5**	**+0.4**	-0.1

Table 4: Arabic-to-English MT Results

SYS	tst11	tst12	tst13	Avg
	Baselines			
B_{in}	25.0	22.5	23.2	23.6
B_{cat}	25.7	22.9	24.1	24.2
$B_{cat,in}$	26.0	22.4	23.6	24.0
	Comparison against NDAM			
B_{cat}	25.7	22.9	24.1	24.2
S_{v1}	26.3	23.1	24.5	24.6
Δ	**+0.6**	**+0.2**	**+0.4**	**+0.4**
S_{v2}	26.2	23.0	24.6	24.6
Δ	**+0.5**	+0.1	**+0.5**	**+0.4**

Table 5: English-to-German MT Results

SYS	tst11	tst12	tst13	Avg
	Arabic-to-English			
B_{cat}	25.0	27.5	27.0	26.5
S_{v1}	25.3	27.8	27.0	26.7
B_{mml}	25.5	27.8	26.8	26.7
S_{v1+mml}	25.5	28.2	27.2	27.0
	English-to-German			
B_{cat}	25.7	22.9	24.1	24.2
S_{v1}	26.3	23.1	24.5	24.6
B_{mml}	25.1	22.7	23.9	23.9
S_{v1+mml}	25.4	22.8	23.9	24.0

Table 6: Comparison with Modified Moore-Lewis

correctly. In another example ماذا عن لياقة البدن؟ (*How about fitness?*), the word لياقة is translated to *proprietary* by S_{cat}, a translation frequently observed in the out-domain data. S_{v*} translated it correctly to *fitness*, as preferred by the in-domain.

English-to-German: Concatenating all training data to train the MT pipeline has been shown to give the best results for English-to-German (Birch et al., 2014). Therefore, we did not do domain-wise experiments, except for training a system on the in-domain IWSLT data for the sake of completeness. We also tried $B_{cat,in}$ variation, i.e. training an MT system on the entire data and using in-domain data to train the baseline NNJM. The baseline system B_{cat} gave better results and was used as our reference for comparison.

Table 5 shows the results of our systems, S_{v1} and S_{v2}, compared to the baselines, B_{in} and B_{cat}. Unlike Arabic-to-English, the baseline system B_{in} is much worse than B_{cat}. Our adapted MT systems S_{v1} and S_{v2} both outperformed the best baseline system (B_{cat}) with an improvement of up to 0.6 points. S_{v2} performed slightly better than S_{v1} on one occasion and slightly worse in others.

Comparison with Data Selection: We also compared our results with the MML-based data selection approach as shown in Table 6. The MML-based baseline systems (B_{mml}) used 20% selected data for training the MT system and the NNJM. On Arabic-English, both MML-based selection and our model (S_{v1}) gave similar gains on top of the baseline system (B_{cat}). Further results showed that both approaches are complementary. We were able to obtain an average gain of +0.3 BLEU points by training an $NDAM_{v1}$ model over the selected data (see S_{v1+mml}).

However, on English-German, the MML-based selection caused a drop in the performance (see Table 6). Training an adapted $NDAM_{v1}$ model over selected data gave improvements over MML in two test sets but could not restore the baseline performance, probably because the useful data has already been filtered by the selection process.

6 Conclusion

We presented two novel models for domain adaptation based on NNJM. Adaptation is performed by regularizing the loss function towards the in-domain model and away from the unrelated out-of-domain data. Our models show better perplexities than the non-adapted baseline NNJM models. When integrated into a machine translation system, gains of up to 0.5 and 0.6 BLEU points were obtained in Arabic-to-English and English-to-German systems over strong baselines.

References

Ahmed Abdelali, Francisco Guzman, Hassan Sajjad, and Stephan Vogel. 2014. The AMARA corpus: Building parallel language resources for the educational domain. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, Reykjavik, Iceland, May.

Michael Auli, Michel Galley, Chris Quirk, and Geoffrey Zweig. 2013. Joint language and translation modeling with recurrent neural networks. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, Seattle, Washington, USA, October.

Amittai Axelrod, Xiaodong He, and Jianfeng Gao. 2011. Domain adaptation via pseudo in-domain data selection. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, Edinburgh, United Kingdom.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3:1137–1155, March.

Alexandra Birch, Matthias Huck, Nadir Durrani, Nikolay Bogoychev, and Philipp Koehn. 2014. Edinburgh SLT and MT system description for the IWSLT 2014 evaluation. In *Proceedings of the 11th International Workshop on Spoken Language Translation*, IWSLT '14, Lake Tahoe, CA, USA.

Arianna Bisazza, Nick Ruiz, and Marcello Federico. 2011. Fill-up versus interpolation methods for phrase-based SMT adaptation. In Marcello Federico, Mei-Yuh Hwang, Margit Rödder, and Sebastian Stüker, editors, *Proceedings of the seventh International Workshop on Spoken Language Translation (IWSLT)*, pages 136–143.

Ondrej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Johannes Leveling, Christof Monz, Pavel Pecina, Matt Post, Herve Saint-Amand, Radu Soricut, Lucia Specia, and Aleš Tamchyna. 2014. Findings of the 2014 workshop on statistical machine translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, Baltimore, Maryland, USA, June.

Marine Carpuat, Hal Daume III, Katharine Henry, Ann Irvine, Jagadeesh Jagarlamudi, and Rachel Rudinger. 2013. Sensespotting: Never let your parallel data tie you to an old domain. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Sofia, Bulgaria, August.

Mauro Cettolo, Jan Niehues, Sebastian Stüker, Luisa Bentivogli, and Marcello Federico. 2014. Report on the 11th IWSLT Evaluation Campaign. *Proceedings of the International Workshop on Spoken Language Translation, Lake Tahoe, US*.

Stanley F. Chen and Joshua Goodman. 1999. An empirical study of smoothing techniques for language modeling. *Computer Speech & Language*, 13(4):359–393.

Boxing Chen, George Foster, and Roland Kuhn. 2013a. Adaptation of reordering models for statistical machine translation. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Atlanta, Georgia, June.

Boxing Chen, Roland Kuhn, and George Foster. 2013b. Vector space model for adaptation in statistical machine translation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Sofia, Bulgaria, August.

Colin Cherry and George Foster. 2012. Batch tuning strategies for statistical machine translation. In *Proceedings of the 2012 Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL-HLT '12, Montréal, Canada.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. volume 12, pages 2493–2537. JMLR. org.

Jacob Devlin, Rabih Zbib, Zhongqiang Huang, Thomas Lamar, Richard Schwartz, and John Makhoul. 2014. Fast and robust neural network joint models for statistical machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Kevin Duh, Sudoh Katsuhito Neubig, Graham, and Hajime Tsukada. 2013. Adaptation data selection using neural language models: Experiments in machine translation. In *Proceedings of the 51th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, Sofia, Bulgaria, August.

Nadir Durrani, Hassan Sajjad, Hieu Hoang, and Philipp Koehn. 2014. Integrating an Unsupervised Transliteration Model into Statistical Machine Translation. In *Proceedings of the 15th Conference of the European Chapter of the ACL (EACL 2014)*, Gothenburg, Sweden, April.

Nadir Durrani, Hassan Sajjad, Shafiq Joty, Ahmed Abdelali, and Stephan Vogel. 2015a. Using Joint Models for Domain Adaptation in Statistical Machine Translation. In *Proceedings of the Fifteenth Machine Translation Summit (MT Summit XV)*, Florida, USA, To Appear. AMTA.

Nadir Durrani, Helmut Schmid, Alexander Fraser, Philipp Koehn, and Hinrich Schütze. 2015b. The Operation Sequence Model – Combining N-Gram-based and Phrase-based Statistical Machine Translation. *Computational Linguistics*, 41(2):157–186.

Chris Dyer, Victor Chahuneau, and Noah A. Smith. 2013. A simple, fast, and effective reparameterization of ibm model 2. In *Proceedings of NAACL'13*.

Vladimir Eidelman, Jordan Boyd-Graber, and Philip Resnik. 2012. Topic models for dynamic translation model adaptation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, Jeju Island, Korea, July.

Andreas Eisele and Yu Chen. 2010. MultiUN: A Multilingual Corpus from United Nation Documents. In *Proceedings of the Seventh conference on International Language Resources and Evaluation*, Valleta, Malta, May.

Andrew Finch and Eiichiro Sumita. 2008. Dynamic model interpolation for statistical machine translation. In *Proceedings of the Third Workshop on Statistical Machine Translation*, Columbus, Ohio, June.

George Foster and Roland Kuhn. 2007. Mixture-model adaptation for smt. In *Proceedings of the Second Workshop on Statistical Machine Translation*, StatMT '07.

George Foster and Roland Kuhn. 2009. Stabilizing minimum error rate training. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, StatMT '09, Athens, Greece.

George Foster, Cyril Goutte, and Roland Kuhn. 2010. Discriminative instance weighting for domain adaptation in statistical machine translation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, Cambridge, MA, October.

Atsushi Fujii, Masao Utiyama, Mikio Yamamoto, and Takehito Utsuro. 2010. Overview of the patent translation task at the ntcir-8 workshop. In *In Proceedings of the 8th NTCIR Workshop Meeting on Evaluation of Information Access Technologies: Information Retrieval, Question Answering and Cross-lingual Information Access*, pages 293–302.

Michel Galley and Christopher D. Manning. 2008. A Simple and Effective Hierarchical Phrase Reordering Model. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 848–856, Honolulu, Hawaii, October.

Jianfeng Gao, Xiaodong He, Wen-tau Yih, and Li Deng. 2014. Learning continuous phrase representations for translation modeling. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Baltimore, Maryland, June.

Spence Green and John DeNero. 2012. A class-based agreement model for generating accurately inflected translations. In *Proceedings of the Association for Computational Linguistics (ACL'12)*, Jeju Island, Korea.

Michael Gutmann and Aapo Hyvärinen. 2010. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In Y.W. Teh and M. Titterington, editors, *Proc. Int. Conf. on Artificial Intelligence and Statistics (AISTATS)*, volume 9 of *JMLR W&CP*, pages 297–304.

Francisco Guzmán, Hassan Sajjad, Stephan Vogel, and Ahmed Abdelali. 2013. The AMARA corpus: Building resources for translating the web's educational content. In *Proceedings of the 10th International Workshop on Spoken Language Technology (IWSLT-13)*, December.

Eva Hasler, Phil Blunsom, Philipp Koehn, and Barry Haddow. 2014. Dynamic topic adaptation for phrase-based mt. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, Gothenburg, Sweden, April.

Kenneth Heafield. 2011. KenLM: Faster and Smaller Language Model Queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 187–197, Edinburgh, Scotland, United Kingdom, July.

Almut Silja Hildebrand, Matthias Eck, Stephan Vogel, and Alex Waibel. 2005. Adaptation of the translation model for statistical machine translation based on information retrieval. In *Proceedings of the 10th Conference of the European Association for Machine Translation (EAMT)*, Budapest, May.

Geoffrey Hinton, Li Deng, Dong Yu, George Dahl, Abdel rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara Sainath, and Brian Kingsbury. 2012. Deep neural networks for acoustic modeling in speech recognition. *Signal Processing Magazine*.

Cuong Hoang and Khalil Sima'an. 2014. Latent domain translation models in mix-of-domains haystack. In *COLING 2014, 25th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, August 23-29, 2014, Dublin, Ireland*.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, Seattle, Washington, USA, October.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the Association for Computational Linguistics (ACL'07)*, Prague, Czech Republic.

Le Liu, Yu Hong, Hao Liu, Xing Wang, and Jianmin Yao. 2014. Effective selection of translation model

training data. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, Baltimore, Maryland, June.

Saab Mansour and Hermann Ney. 2013. Phrase training based adaptation for statistical machine translation. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Atlanta, Georgia, June.

Prashant Mathur, Sriram Venkatapathy, and Nicola Cancedda. 2014. Fast domain adaptation of smt models without in-domain parallel data. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, Dublin, Ireland, August.

Spyros Matsoukas, Antti-Veikko I. Rosti, and Bing Zhang. 2009. Discriminative corpus weight estimation for machine translation. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2*, EMNLP '09.

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

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013b. Linguistic regularities in continuous space word representations. HLT-NAACL, pages 746–751.

Tomas Mikolov, 2012. *Statistical Language Models based on Neural Networks*. PhD thesis, Brno University of Technology.

Andriy Mnih and Yee Whye Teh. 2012. A fast and simple algorithm for training neural probabilistic language models. In *Proceedings of the International Conference on Machine Learning*.

Robert C. Moore and William Lewis. 2010. Intelligent selection of language model training data. In *Proceedings of the Association for Computational Linguistics (ACL'10)*, Uppsala, Sweden.

Preslav Nakov and Hwee Tou Ng. 2009. Improved statistical machine translation for resource-poor languages using related resource-rich languages. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP'09)*, Singapore.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the Association for Computational Linguistics (ACL'02)*, Philadelphia, PA, USA.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*.

Holger Schwenk. 2012. Continuous space translation models for phrase-based statistical machine translation. In *Proceedings of COLING 2012: Posters*, Mumbai, India, December.

Rico Sennrich, Holger Schwenk, and Walid Aransa. 2013. A multi-domain translation model framework for statistical machine translation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Sofia, Bulgaria, August.

Rico Sennrich. 2012. Perplexity minimization for translation model domain adaptation in statistical machine translation. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, Avignon, France, April.

Richard Socher, John Bauer, Christopher D. Manning, and Ng Andrew Y. 2013. Parsing with compositional vector grammars. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 455–465, Sofia, Bulgaria, August.

Ashish Vaswani, Yinggong Zhao, Victoria Fossum, and David Chiang. 2013. Decoding with large-scale neural language models improves translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*.

Detecting Content-Heavy Sentences: A Cross-Language Case Study

Junyi Jessy Li
University of Pennsylvania
ljunyi@seas.upenn.edu

Ani Nenkova
University of Pennsylvania
nenkova@seas.upenn.edu

Abstract

The information conveyed by some sentences would be more easily understood by a reader if it were expressed in multiple sentences. We call such sentences content heavy: these are possibly grammatical but difficult to comprehend, cumbersome sentences. In this paper we introduce the task of detecting content-heavy sentences in cross-lingual context. Specifically we develop methods to identify sentences in Chinese for which English speakers would prefer translations consisting of more than one sentence. We base our analysis and definitions on evidence from multiple human translations and reader preferences on flow and understandability. We show that machine translation quality when translating content heavy sentences is markedly worse than overall quality and that this type of sentence are fairly common in Chinese news. We demonstrate that sentence length and punctuation usage in Chinese are not sufficient clues for accurately detecting heavy sentences and present a richer classification model that accurately identifies these sentences.

1 Introduction

To generate text, people and machines need to decide how to package the content they wish to express into clauses and sentences. There are multiple possible renderings of the same information, with varying degrees of ease of comprehension, compactness and naturalness. Some sentences, even though they are grammatical, would be more accessible to a reader if expressed in multiple sentences. We call such sentences *content heavy* sentences, or *heavy sentences* for brevity.

In the established areas of language research, text simplification and sentence planning in dialog and generation systems are clearly tasks in which identification of content-heavy sentences is of great importance. In this paper we introduce a novel flavor of the task in the cross-lingual setting, which in the long term may guide improvements in machine translation. We seek to identify sentences in Chinese that would result in heavy sentences in English if translated to a single sentence.

Example I in Table 1 shows a Chinese sentence and its two English translations A and B. Translator A used three English sentences to express all the information. Translator B, on the other hand, used a single sentence, which most readers would find more difficult to read. Example II illustrates a case where a translator would be hard pressed to convey all the content in a sentence in Chinese into a single grammatical English sentence.

Here we provide an operational characterization of content-heavy sentences in the context of Chinese-English translation. Instead of establishing guidelines for standalone annotation, we repurpose datasets developed for evaluation of machine translation consisting of multiple reference translations for each Chinese sentence. In this cross-lingual analysis, sentences in Chinese are considered content-heavy if their content would be more felicitously expressed in multiple sentences in English.

We first show that, with respect to English, content-heavy Chinese sentences are common. A fifth to a quarter of the sentences in the Chinese news data that we analyze are translated to multiple sentences in English. Moreover our experiments with reader preference indicate that for these sentences, readers strongly prefer multi-sentence translation to a single-sentence translation (§ 4.1). We also compare the difference in machine translation quality for heavy sentences and find that it is considerably lower than overall system performance (§ 4.2). Then we study the connection between heavy sentences and the factors

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1271–1281,
Lisbon, Portugal, 17-21 September 2015.

[Example I] 虽然菲军方在南部的巴西兰岛上部署了5000多兵力，并在美军的帮助下围剿阿布沙耶夫分子，但迄今收效不大。*Although the Philippine army on the southern Basilan island deployed over 5,000 troops, and with the US army's help are hunting down ASG members, but so far achieved little.* **[A]** The Philippine army has already deployed over 5 thousand soldiers on the southern island of Basilan. With the help of U.S. army, these soldiers are searching and suppressing members of Abu Sayyaf. However, there is not much achievement this far. **[B]** The Philippine military has stationed over 5,000 troops on Basilan Island in the southern Philippines and also tried to hunt down ASG members with the help of the United States, yet so far it has little success.	**[Example II]** 端粒是染色体末端的结构，随着细胞老化和失去分裂能力，端粒会逐渐缩短长度，换言之，端粒愈长显示细胞老化愈慢。*Telomeres are chromosome ends structures, with cell aging and losing division ability, telomeres will gradually decrease length, in other words, telomeres the longer shows cell aging the slower.* **[A]** Telomeres are structures at the ends of chromosomes, which gradually reduce in length with the aging of the cells and their loss of the ability to divide. In other words, longer telomeres indicate the slower aging of the cells. **[B]** Telomeres are the physical ends of chromosomes. As cells age and lose the ability to divide, the telomeres shrink gradually. That is to say, longer telomeres indicate that cells are aging more slowly.

Table 1: Examples of Chinese sentences expressed in multiple English sentences.

used in prior work to split a Chinese sentence into multiple sentences, showing that they do not fully determine the empirically defined content-heavy status (§ 5). Finally, we present an effective system to automatically identify content-heavy sentences in Chinese (§ 6, 7, 8).

2 Related work

The need for identifying content-heavy sentences arises in many specialized domains, including dialog systems, machine translation, text simplification and Chinese language processing but it is usually addressed in an implicit or application specific way. In contrast, we focus on identifying heavy sentences as a standalone task, providing a unifying view of the seemingly disparate strands of prior work. We now overview the literature which motivated our work.

Sentence planning. In text generation, a sentence planner produces linguistic realizations of a list of propositions (Rambow and Korelsky, 1992). One subtask is to decide whether to package the same content into one or more sentences. In the example below (Pan and Shaw, 2005), the multi-sentence expression B is much easier to process:

[A] This is a 1 million dollar 3 bedroom, 2 bathroom, 2000 square foot colonial with 2 acre of land, 2 car garage, annual taxes 8000 dollars in Armonk and in the Byram Hills school district.

[B] This is a 3 bedroom, 2 bathroom, 2000 square foot colonial located in Armonk with 2 acres of land. The asking price is 1 million dollar and the annual taxes are 8000 dollars. The house is located in the Byram Hills School District.

Identifying sentence [A] as heavy would be useful in selecting the best realization.

A crucial difference between our task and its counterpart in sentence planning is that traditional text generation systems have access to rich semantic information about the type of propositions the system needs to convey, while in our task we have access only to Chinese text. In some dialog systems, content selection is treated as an optimization problem, balancing the placement of full-stops and the insertion or deletion of propositions with the similarity of the resulting output and an existing corpus of acceptable productions (Pan and Shaw, 2005). Others formulate the problem as a supervised ranking task, in which different possible content realizations are generated, including variation in the number of sentences (Walker et al., 2001; Stent et al., 2004). With the introduction of the concept of content-heavy sentences, we can envision dialog systems addressing the sentence realization task in two steps, first predicting if the semantic content will require multiple sentences, then having different rankers for expressing the content in one or multiple sentences. In that case the ranker will need to capture only sentence-level information and the discourse-level decision to use multiple sentences will be treated separately.

Text simplification. "Text simplification, defined narrowly, is the process of reducing the linguistic complexity of a text, while still retaining the original information content and meaning" (Siddharthan, 2014). An important aspect of simplification is syntactic transformation in which sentences deemed difficult are re-written as multiple sentences (Chandrasekar et al., 1996; Aluísio et al., 2008). Our task may be viewed as identifying sentences in one language that will require simplification when translated, for the benefit of the speakers of the target language. In rule-based simplification systems, splitting is performed al-

ways when a given syntactic construction such as relative clause, apposition or discourse connective are detected (Chandrasekar et al., 1996; Siddharthan, 2006; De Belder and Moens, 2010). Most recently, text simplification has been addressed as a monolingual machine translation task from complex to simple language (Specia, 2010; Coster and Kauchak, 2011; Wubben et al., 2012). However simplification by repackaging the content into multiple sentences is not naturally compatible with the standard view of statistical MT in which a system is expected to produce a single output sentence for a single input sentence. Some of the recent systems using MT techniques separately model the need for sentence splitting (Zhu et al., 2010; Woodsend and Lapata, 2011; Narayan and Gardent, 2014). Identifying heavy sentences in simplification is equivalent to identifying sentences that require syntactic simplification.

Sentence structure and MT. Prior work in machine translation has discussed the existence of sentences in Chinese which would result in a poor translation if translated in one sentence in English. The main factors proposed to characterize such problematic sentences are sentence length (Xu and Tan, 1996) and the presence of given syntactic constructions (Xu et al., 2005; Yin et al., 2007; Jin and Liu, 2010). Mishra et al. (2014) used rules involving similar factors to distinguish sentences in Hindi that need simplification prior to translation.

In each of these approaches, the identified sentences are segmented into smaller units. Similar to work in text simplification, the simplification rules are applied to all sentences meeting certain criteria, normally to all sentences longer than a predefined threshold or where certain conjunctions or coordinations are present. In contrast, the model we propose here can be used to predict when segmentation is at all necessary.

Our approach to the problem is more compatible with the empirical evidence we presented in our prior work (Li et al., 2014) where we analyzed the output of Chinese to English machine translation and found that there is no correlation between sentence length and MT quality. Rather we showed that the quality of translation was markedly inferior, compared to overall translation quality, for sentences that were translated into multiple English sentences. This prior work was carried over a dataset containing a single reference translation for each Chinese sentence. In the work presented in this paper, we strengthen our findings by examining multiple reference translations for each Chinese sentence. We define heavy sentences based on agreement of translator choices and reader preferences.

Commas in Chinese. Often a comma in a sentence can be felicitously replaced by a full stop. Such commas offer a straightforward way to split a long sentence into multiple shorter ones by replacing the comma with a full stop. Monolingual text simplification systems often try to identify such commas. They are particularly common in Chinese and replacing them with full stops leads to improvements in the accuracy of syntactic parsing (Jin et al., 2004; Li et al., 2005). Moreover, existing syntactically parsed corpora conveniently provide numerous examples of these full-stop commas, and thus training data for systems to identify them (Xue and Yang, 2011; Yang and Xue, 2012). In this paper, we systematically study the relationship between the presence of full-stop commas in the sentence and whether it is content-heavy for Chinese to English translation.

3 Data

In this work we use three news datasets: the newswire portion of the NIST 2012 Open Machine Translation Evaluation (OpenMT) (Group, 2013), Multiple-Translation Chinese (MTC) parts 1-4 (Huang et al., 2002; Huang et al., 2003; Ma, 2004; Ma, 2006), and the Chinese Treebank (Xue et al., 2005). In OpenMT and MTC, multiple reference translations in English are available for each Chinese segment (sentence).

To study the relationship between content-heavy sentences and reader preference for multi-sentence translations (§ 4.1), we use OpenMT (688 segments) and MTC parts 2-4 (2,439 segments), both of which provide four English translations for each Chinese segment. This analysis forms the basis for labeling heavy sentences for supervised training and evaluation (§ 5, 6, 7).

The Chinese Treebank (CTB) has been used in prior work as data for identifying full-stop commas. Moreover, 52 documents in MTC part 1 were drawn from the CTB. The intersection of the two datasets allows us to directly analyze the relationship between heavy sentences and full-stop commas in Chinese (§ 5). Furthermore we use this intersection as test set to identify heavy sentences so we can directly compare with models developed

for comma disambiguation. To be consistent with the rest of the MTC data, we use 4 out of the 11 translators in part 1 in these experiments.[1]

Our model for Chinese full-stop comma recognition is trained following the features and training sets specified in Xue and Yang (2011)[2], excluding the overlapping MTC/CTB documents mentioned above. There are 12,291 sentences in training that contain at least one comma. A classifier for detecting heavy sentences is trained on OpenMT and MTC (excluding the test set). A quick inspection of both datasets reveals that Chinese sentences without a comma were never translated into multiple sentences by more than one translator. Therefore in our experiments we consider only sentences that contain at least one comma. There are 301 test sentences, 511 training sentences in OpenMT and 2418 in MTC. Sentences are processed by the Stanford NLP packages[3]. CTB gold-standard parses are used to obtain full-stop commas and to train comma disambiguation models.

4 Content-heavy sentences: definition

In this section we provide an operational definition for which sentences should be considered content-heavy, based on the choices made by translators and the fluency preferences of readers when a sentence is translated into a single or multiple sentences. We further demonstrate the difference in machine translation quality when translating content-heavy sentences compared to other sentences.

4.1 Content-heaviness and multi-sentence translations

First we quantify how often translators choose to translate a Chinese sentence into multiple English sentences. Content-heavy Chinese sentences are those for which there is a strong preference to produce multiple sentences when translating to English (at the end of the section we present specific criteria).

Obviously, splitting a sentence into multiple ones is often possible but is not necessarily preferred. In Table 2, we show in the "%data" columns the percentage of source sentences split in translation by 0, 1, 2, 3 and all 4 translators. For about 20% of segments in OpenMT and 15% in MTC, at least three of the translators produce a multi-sentence translation, a rate high enough to warrant closer inspection of the problem.

	OpenMT		MTC	
#ref multi	%data	%best multi	%data	%best multi
0	65.4	0	58.9	0
1	7.4	23.5	20.4	20.1
2	7.0	66.7	8.3	56.7
3	9.2	88.9	7.9	89.6
4	11.0	100	4.6	100

Table 2: Percentage of sentences for which a given number of translators prefer to use multiple sentences in English, along with percentage of times a multi-sentence translation was selected as most fluent and comprehensible by readers.

Next, we conduct a study to find out what level of translator agreement leads to strong reader preference for the same information to be presented in multiple sentences.

For each Chinese segment with one, two or three multi-sentence reference translations, we ask five annotators on Mechanical Turk to rank the reference translations according to their general flow and understandability. The annotators saw only the four randomly ordered English translations and were not shown the Chinese original, with the following instruction:

Below are 1-2 sentence snippets that describe the same content. Some are more readable and easier to understand than others. Your task is to rank them from the best to worst in terms of wording or flow (organization). There can be ties, but you have to pick one that is the best.

We obtain reader preference for each segment in the following manner: for each annotator, we take the highest ranked translation and check whether it consists of multiple sentences. In this way we have five binary indicators. We say readers prefer a sentence to have a multi-sentence translation in terms of flow and comprehensibility if the majority of these five indicators are positive.

In the "%best multi" columns of Table 2, we tabulate the percentage of segments with majority preference for multi-sentence translation, stratified by the number of translators who split the content. Obviously the more multi-sentence translations there are, the higher the probability that the readers will select one as the best translation. We are interested in knowing for which conditions the

[1] We did not use translator IDs as parameters in any of our systems.

[2] Document IDs 41-325, 400-454, 500-554, 590-596, 600-885, 900, 1001-1078, 1100-1151.

[3] The Stanford segmenter (Tseng et al., 2005), parser (Levy and Manning, 2003) and the CoreNLP package (Manning et al., 2014)

Criteria	%data(Y)	Y	N	Δbleu
heavy	27.2	15.34	19.24	**3.9**

Table 3: Percentage of data for heavy sentences along with BLEU scores for heavy and non-heavy sentences and their difference.

preference for multi-sentence translation exceeds the probability of randomly picking one.

When only one (out of four) translations is multi-sentence, the best translations chosen by the majority of readers contain multiple sentences less often than in random selection from the available translations. When two out of the four reference translations are multi-sentence, the reader preference towards them beats chance by a good margin. The difference between chance selection and reader preference for multiple sentences grows steadily with the number of reference translations that split the content. These data suggest that when at least two translators perform a multi-sentence translation, breaking down information in the source sentence impacts the quality of the translation.

Hence we define content-heavy sentences in Chinese to be those for which at least two out of four reference translations consist of multiple sentences.

4.2 A challenge for MT

We now quantitatively show that heavy sentences are particularly problematic for machine translation. We collect translations for each segment in OpenMT and MTC from the Bing Translator. We split the sentences into two groups, heavy and other, according to the gold standard label explained in the previous section. We then compare the BLEU score for sentences in a respective group, where each group is in turn used as a test set. The difference in BLEU scores (Δbleu) is a strong indicator whether these sentences are challenging for MT systems.

In Table 3 we show the BLEU scores and Δbleu for sentences that are heavy (Y) and non-heavy (N). Also included in the table is the percentage of heavy sentences in all the data.

Translations for heavy sentences received a BLEU score that is 3.9 points lower than those that are not. This clearly illustrates the challenge and potential for improvement for MT systems posed by content-heavy sentences. Therefore the ability to reliably recognize them provides a first step towards developing a better translation approach for such sentences.

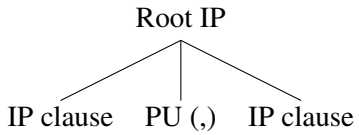


Figure 1: Coordinating IP structure at the root.

heavy	fs-comma	No fs-comma
N	19	180
Y	40	62

Table 4: Count of heavy and non-heavy sentences with and without full-stop commas.

5 Comma usage and heavy sentences

In Chinese, commas can sometimes act as sentence boundaries, similar to the function of an English period. In Xue and Yang (2011), the authors showed that these full-stop commas can be identified in the constituent parse tree as coordinating IPs at the root level, shown in Figure 1. Fancellu and Webber (2014) demonstrated that it is beneficial to split sentences containing negation on these types of commas, translate the resulting shorter sentences separately, then stitch the resulting translations together. They report that this approach prevented movement of negation particles beyond their scope. Here we study the degree to which the content-heavy status of a sentence is explained by the presence of a full-stop comma in the sentence. We show that they are interrelated but not equivalent.

Corpus analysis. First we study how often a heavy sentence contains a full-stop comma and vice versa, using the overlapping MTC/CTB documents. We show in Table 4 the number of heavy and non-heavy sentences with and without full-stop commas[4]. When there is a full-stop comma in the sentence, there is a higher chance that the sentence is content-heavy. Yet of the 102 heavy sentences in this data, fewer than 40% contain full-stop commas; of the 242 sentences without full-stop commas, more than a quarter are heavy. Therefore, although comma usage in the Chinese sentence may provide clues for detecting content heaviness, the two phenomena are not equivalent

[4]For the study we exclude sentences without a comma. A χ^2 test for the strength of association between the presence of full stop commas and heavy sentence status shows high significance.

and heavy sentences are not fully explained by the presence of full-stop commas.

Learning with full-stop commas. Here we evaluate the usefulness of using full-stop commas as training data to predict whether a sentence is content-heavy. From the analysis presented above we know that the two tasks are not equivalent. Nevertheless we would like to test directly if the Chinese Treebank—the large (but noisy for the task at hand) data available for comma function disambiguation—would lead to better results than learning on the cleaner but much smaller datasets for which multiple translations are available.

We use logistic regression as our classification model[5]. The performance of identifying heavy sentences on the MTC/CTB overlapping test set is compared using the following methods:

[Parallel] A classifier is trained using four English translations for each Chinese sentence (OpenMT and MTC training set). Following the definition in Section 4.1, content-heavy sentences are those translated into multiple English sentences by two or more translators.

[Oracle comma] A test sentence is assigned to class "heavy" if there is a full-stop comma in its corresponding gold standard parse tree.

[Predicted comma] We train a comma disambiguation system on CTB to predict if a comma is a full-stop comma. In testing, a sentence is marked "heavy" if it contains a predicted full-stop comma.

Features. We reimplemented the per-comma features used in Xue and Yang (2011)[6]. As in their best performing system, features are extracted from gold-standard parse trees during training and from automatic parsing during testing. These include: words and part-of-speech tags immediately before and after the comma; left- and right-sibling node labels of the parent of the comma; ordered ancestor node labels above the comma; punctuation tokens ordered from left to right of the sentence; whether the comma has a coordinating IP structure; whether the comma's parent is a child of the root of the tree; whether there is a subordination before the comma; whether the difference in number of words before and after the comma is greater than or equal to seven.

[5]We use the Liblinear package (Fan et al., 2008).

[6]For *predicted comma*, our reimplementation of Xue and Yang (2011) gave practically identical results to those reported in the original paper on the test set that they used.

Training	A	P	R
parallel	75.75	69.86	50
oracle comma	73.09	67.8	39.2
predicted comma	74.42	66.67	49.02

Table 5: Performance for identify heavy sentences using multiple reference data (parallel) vs. full-stop comma oracle labels (oracle comma) and predicted full-stop commas (predicted comma).

For *parallel*, feature values are accumulated from all the commas in the sentence. For binary features, we use an *or* operation on the feature values for each individual comma.

Results and comparison. In Table 5, we show the accuracy, precision and recall for identifying content-heavy sentences using the three methods described above. We do not include the majority baseline here because it assumes no sentences are content heavy.

Interestingly, the system using oracle information in each test sentence for full-stop commas performs the worst. The system trained to identify full-stop commas outperform the oracle system with about 10% better in recall and less than 1% lower in precision. This finding strongly suggests that the features used for learning capture certain characteristics of heavy sentences even with non-ideal training labels. The best performance is obtained learning directly on parallel corpora with multiple reference translations. Note that we try to provide the best possible setting for full-stop comma prediction, using much more training data, gold-standard parses, same-domain training and testing, as well as the reimplementation of state-of-the-art system. These settings allow us to conservatively interpret the results listed here, which confirm that content-heaviness is different from using a full-stop comma in the Chinese sentence. It is more advantageous—leading to higher precision and overall accuracy—to learn from data where translators encode their interpretation in the form of multi-sentence translations.

6 Features to characterize content-heavy sentences

In this section, we experiment with a wide range of features from the sentence string, part-of-speech tags and dependency parse trees.

Baseline. Intuitively, sentence length can be an indication of too much content that needs to be

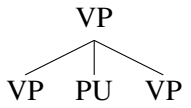


Figure 2: Multiple VP structures

repackaged into multiple sentences. Therefore as our baseline we train a decision tree using the number of words[7] in a Chinese sentence.

Sentence structure cues. We collect potential signals for structural complexity: punctuation, conjunctions, prepositional phrases and relative clauses. As features we count the number of commas, conjunction, preposition and postposition part-of-speech tags. In Chinese "DE" often marks prepositional phrases or relative clauses among other functions (Chang et al., 2009a). Here we include a simple count the number of "DEG" tags in the sentence.

Dependencies. Dependency grammar captures both syntactic and semantic relationship between words and are shown to improve reordering in MT (Chang et al., 2009b). To account for such relational information we include two feature classes: the percentage of each dependency type and the typed dependency pairs themselves. For the latter we use the universal part-of-speech tags (Petrov et al., 2012) for each word rather than the word itself to avoid too detailed and sparse representations. For example, the relation *dobj(处理/handle, 事情/matter)* becomes feature *dobj(verb, noun)*.

Furthermore, we use dependency trees to extract four features for potentially complex constructions. First, we indicate the presence of noun phrases with heavy modifiers on the left. These are frequently used in Chinese and would require a relative clause or an additional sentence in English. Specifically we record the maximum number of dependents for the nouns in the sentence. The second type of construction is the use of serial verb phrases, illustrated in Figure 2. We record the number of dependents of the head verb of the sentence. The third feature class is the typed dependencies (over universal POS tags) whose edge crosses a comma. Finally, we also record the maximum number of dependents in the sentence to capture the general phrasal complexity in the sentence.

[7] obtained using the Stanford Chinese Word Segmenter (Tseng et al., 2005)

Features	Training	A	P	R
baseline	MTC+OpenMT	71.43	73.5	24.5
full set	OpenMT	76.41	66.67	**60.78**
full set	MTC	78.41	74.03	55.9
full set	MTC+OpenMT	**80.73**	**79.73**	57.84

Table 6: Accuracy, precision and recall (for the content heavy class) of binary classification using proposed features to identify content-heavy sentences.

Parts-of-speech. POS information captures numerous aspects of the sentence such as the frequency of different classes of words used and the transition between them. Historically they are also shown to be helpful for phrase boundary detection (Taylor and Black, 1998). Here, we first convert all Chinese POS tags into their corresponding universal tags. We then use the percentage of each tag and tag bigram as two feature classes. To capture the transition of each phrase and clause in the sentence, we construct functional POS trigrams for each sentence by removing all nouns, verbs, adjectives, adverbs, numbers and pronouns in the sentence. Percentages of these sequences are used as feature values.

Comma disambiguation features. We also incorporate most of the features proposed by Xue and Yang (2011), aggregated in the same way as the *parallel* method (cf. Section 5). These include: POS tags immediately before and after the comma; left- and right-sibling node labels of the parent of the comma; the punctuation tokens ordered from left to right in the sentence, whether the comma has a coordinating IP structure; whether the comma's parent is a child of the root of the tree; whether there is a subordination before the comma; whether the difference in number of words before and after the comma is greater than or equal to seven.

7 Results

7.1 Recognizing content-heavy sentences

We train a logistic regression model as in the *parallel* method in Section 5 using features illustrated above. In Table 6, we show the performance of detecting heavy sentences using four systems: the baseline system using the number of words in the sentence and three systems using our full feature set, trained on MTC, OpenMT and both.

The baseline performance is characterized by a remarkably poor recall. It becomes appar-

#ref multi	≥ 0	≥ 1	≥ 2	≥ 3	4
#seg	301	187	102	58	25
precision	79.73	84.29	100	100	100
recall	57.85	57.84	57.84	68.98	76
posterior	0.29	0.40	0.53	0.61	0.67

Table 7: Number of segments, precision, recall and posterior probability (for the content-heavy class) for examples where at least 0-4 translators split the sentence.

ent that length alone cannot characterize content-heaviness. On the other hand, using the full feature set achieves an accuracy of above 80%, a precision close to 80% and a recall about 58%. The improvement in precision and recall over using oracle full-stop commas (Table 5) are about 12% and 19%. When compared with using features tuned for comma disambiguation from Xue and Yang (2011) (Table 5), our full feature set achieved a 5% increase in accuracy, about 10% increase in precision and 8% increase in recall.

We also demonstrate the usefulness of having more multi-reference translation data by comparing training using MTC and OpenMT individually and both. Remarkably, using only the very small dataset of OpenMT is sufficient to produce a predictor that is more accurate than all of the methods listed in Section 5. Adding these examples to MTC drastically improves precision by more than 13% with a less than 3% drop on recall.

Finally, we consider the portions of our test set for which at least n translators provided a multi-sentence translation (n ranges from 0 to 4). In Table 7 we show the respective precision, recall and the average posterior probability from the classifier for marking a sentence as content-heavy. There is a clear trend that the classifier is more confident and has higher precision for sentences where more translators produce multi-sentence translations. Although the model is not highly confident in all groups, the precision of the predictions are remarkably high. Miss rate also decreases when more translators translate the source into multiple sentences.

7.2 Post-hoc feature analysis

Here we identify which of the feature classes from our full set are most helpful by performing forward feature selection: in each iteration, the feature class that improves accuracy the most is selected. The process is repeated until none of the remaining feature classes lead to improvement when added to the model evaluated at the previous iteration. We use our test data as the evaluation set for forward selection, but we do so only to evaluate features, not to modify our system.

Five feature classes are selected using this greedy procedure. The first selected class is the typed dependencies over universal POS tags. Remarkably, this single feature class achieves 76.6% accuracy, a number already reasonably high and better than features used in Xue and Yang (2011). The second feature added is whether there is a comma of coordinating IP structure in the automatic parse tree of the sentence. It gives a further 1.7% increase in accuracy, showing that the comma structure provide useful information as features for detecting heavy sentences. Note that this feature does not represent full stop commas, i.e., it does not record whether the comma is under the root level of the parse tree. The next selected class is typed dependencies over universal POS tags that have an edge across commas in the sentence, with an 1% increase in accuracy. The fourth feature selected is the number of prepositions and postposition POS tags in the sentence, improving the accuracy about 1%. Finally, part-of-speech tags before each comma are added, with a 0.3% improvement of accuracy.

The results from forward selection analysis reveal that the dependency structure of a sentence captures the most helpful information for heavy sentence identification. The interplay between punctuation and phrase structure gives further important enhancements to the model. The final accuracy, precision and recall after forward selection are 0.804, 0.8209, 0.5392, respectively. This overall performance shows that forward selection yields a sub-optimal feature set, suggesting that the other features are also informative.

8 A challenge for MT: revisited

It is important to know whether a predictor for content-heavy sentences is good at identifying challenging sentences for applications such as machine translation. Here, we would like to revisit Section 4.2 and see if *predicted* heavy sentences are harder to translate.

For all the source sentences in OpenMT and MTC, we compare five criteria for dividing the test data in two subsets: whether the sentence contains a full-stop comma or not; whether the sentence is longer than the baseline decision tree threshold

Criteria	%data(Y)	Y	N	Δbleu
fs-comma	21.6	16.01	18.43	2.42
length threshold	8.6	15.38	18.3	2.92
pred-heavy (0.5)	22.72	15.81	18.77	2.96
pred-heavy (0.55)	19.72	15.47	18.76	3.29
pred-heavy (0.6)	16.67	14.95	18.77	3.82
oracle heavy	27.4	15.34	19.24	3.9

Table 8: Data portion, BLEU scores and differences for sentences with/without a full-stop comma, are/are not longer than the length threshold, are/are not content heavy.

(47 words) or not; whether the sentence is predicted to be content-heavy with posterior probability threshold of 0.5, 0.55 and 0.6. Predictions for the training portion is obtained using 10-fold cross-validation. In the same manner as Table 3, Table 8 shows the percentage of data that satisfies each criterion, BLEU scores of Bing translations for sentences that satisfy a criterion and those that do not, as well as the difference of BLEU between the two subsets (Δbleu). As reference we also include numbers listed in Table 3 using oracle content-heavy labels.

First, notice that regardless of the posterior probability threshold, the numbers of sentences predicted to be content-heavy are much larger than that using the length cutoff. These sentences are also collectively translated much worse than the sentences in the other subset. Sentences that contain a predicted full-stop comma are also harder to translate, but show smaller difference in BLEU than when sentence heaviness or length are used as separation criterion. As the posterior probability threshold goes up and the classifier becomes more confident when it identifies heavy sentences, there is a clear trend that system translations for these sentences become worse. These BLEU score comparisons indicate that our proposed model identifies sentences that pose a challenge for MT systems.

9 Conclusion and future work

In this work, we propose a cross-lingual task of detecting content-heavy sentences in Chinese, which are best translated into multiple sentences in English. We show that for such sentences, a multi-sentence translation is preferred by readers in terms of flow and understandability. Content-heavy sentences defined in this manner present practical challenges for MT systems. We further demonstrate that these sentences are not fully explained by sentence length or syntactically defined full-stop commas in Chinese. We propose a classification model using a rich set of features that effectively identify these sentences.

The findings in this paper point out a definite issue in different languages currently underinvestigated in text-to-text generation systems. One possible way to improve MT systems is to incorporate sentence simplification before translation (Mishra et al., 2014). Future work could use our proposed model to detect heavy sentences that needs such pre-processing. Our findings can also inspire informative features for sentence quality estimation, in which the task is to predict the sentence-level fluency (Beck et al., 2014). We have shown that heavy Chinese sentences are likely to lead to hard to read, disfluent sentences in English. Another important future direction lies in text simplification. In our inspection of parallel Wikipedia/Simple Wikipedia data (Kauchak, 2013), around 23.6% of the aligned sentences involve a single sentence on one side and multiple sentences on another. A similar analysis using ideas from this work can be useful in identifying sentences that needs simplification in the first place.

Acknowledgements

We would like to express our gratitude to Bonnie Webber for her detailed comments on earlier versions of this paper. Her encouragement and support were essential in seeing the work through to publication. We would also like the reviewers for their thoughtful suggestions which we have tried to incorporate in the final version. The work was partially supported by NSF CAREER grant IIS-0953445.

References

Sandra M. Aluísio, Lucia Specia, Thiago A.S. Pardo, Erick G. Maziero, and Renata P.M. Fortes. 2008. Towards Brazilian Portuguese automatic text simplification systems. In *Proceedings of the Eighth ACM Symposium on Document Engineering (DocEng)*.

Daniel Beck, Kashif Shah, and Lucia Specia. 2014. SHEF-Lite 2.0: Sparse multi-task gaussian processes for translation quality estimation. In *Ninth Workshop on Statistical Machine Translation (WMT)*.

R. Chandrasekar, Christine Doran, and B. Srinivas.

1996. Motivations and methods for text simplification. In *COLING 1996 Volume 2: The 16th International Conference on Computational Linguistics*.

Pi-Chuan Chang, Daniel Jurafsky, and Christopher D. Manning. 2009a. Disambiguating "DE" for Chinese-English machine translation. In *Proceedings of the Fourth Workshop on Statistical Machine Translation (WSMT)*.

Pi-Chuan Chang, Huihsin Tseng, Dan Jurafsky, and Christopher D. Manning. 2009b. Discriminative reordering with Chinese grammatical relations features. In *Proceedings of the Third Workshop on Syntax and Structure in Statistical Translation (SSST)*.

Will Coster and David Kauchak. 2011. Learning to simplify sentences using Wikipedia. In *Proceedings of the Workshop on Monolingual Text-To-Text Generation*.

Jan De Belder and Marie-Francine Moens. 2010. Text simplification for children. In *Prroceedings of the SIGIR workshop on accessible search systems*.

Federico Fancellu and Bonnie Webber. 2014. Applying the semantics of negation to SMT through n-best list re-ranking. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*.

NIST Multimodal Information Group. 2013. NIST 2008-2012 Open Machine Translation (OpenMT) Progress Test Sets LDC2013T07. Web Download. In *Philadelphia: Linguistic Data Consortium*.

Shudong Huang, David Graff, and George Doddington. 2002. Multiple-Translation Chinese Corpus LDC2002T01. Web Download. In *Philadelphia: Linguistic Data Consortium*.

Shudong Huang, David Graff, Kevin Walker, David Miller, Xiaoyi Ma, Christopher Cieri, and George Doddington. 2003. Multiple-Translation Chinese (MTC) Part 2 LDC2003T17. Web Download. In *Philadelphia: Linguistic Data Consortium*.

Yaohong Jin and Zhiying Liu. 2010. Improving Chinese-English patent machine translation using sentence segmentation. In *Proceedings of the 6th International Conference on Natural Language Processing and Knowledge Engineering (NLP-KE)*.

Meixun Jin, Mi-Young Kim, Dongil Kim, and Jong-Hyeok Lee. 2004. Segmentation of Chinese long sentences using commas. In *Proceedings of the Third SIGHAN Workshop on Chinese Language Processing (SIGHAN)*.

David Kauchak. 2013. Improving text simplification language modeling using unsimplified text data. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL)*.

Roger Levy and Christopher D. Manning. 2003. Is it harder to parse Chinese, or the Chinese Treebank? In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL)*.

Junyi Jessy Li and Ani Nenkova. 2015. Fast and accurate prediction of sentence specificity. In *Proceedings of the Twenty-Ninth Conference on Artificial Intelligence (AAAI)*, January.

Xing Li, Chengqing Zong, and Rile Hu. 2005. A hierarchical parsing approach with punctuation processing for long Chinese sentences. In *Proceedings of the Second International Joint Conference on Natural Language Processing (IJNLP): Companion Volume including Posters/Demos and Tutorial Abstracts*.

Junyi Jessy Li, Marine Carpuat, and Ani Nenkova. 2014. Assessing the discourse factors that influence the quality of machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL): Short Papers*.

Xiaoyi Ma. 2004. Multiple-Translation Chinese (MTC) Part 3 LDC2004T07. Web Download. In *Philadelphia: Linguistic Data Consortium*.

Xiaoyi Ma. 2006. Multiple-Translation Chinese (MTC) Part 4 LDC2006T04. Web Download. In *Philadelphia: Linguistic Data Consortium*.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics (ACL): System Demonstrations*.

Kshitij Mishra, Ankush Soni, Rahul Sharma, and Dipti Sharma. 2014. Exploring the effects of sentence simplification on Hindi to English machine translation system. In *Proceedings of the Workshop on Automatic Text Simplification - Methods and Applications in the Multilingual Society (ATS-MA)*.

Shashi Narayan and Claire Gardent. 2014. Hybrid simplification using deep semantics and machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*.

Shimei Pan and James Shaw. 2005. Instance-based sentence boundary determination by optimization for natural language generation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*.

Slav Petrov, Dipanjan Das, and Ryan McDonald. 2012. A universal part-of-speech tagset. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC)*.

Owen Rambow and Tanya Korelsky. 1992. Applied text generation. In *Proceedings of the Third Conference on Applied Natural Language Processing (ANLP)*.

Advaith Siddharthan. 2006. Syntactic simplification and text cohesion. *Research on Language and Computation*, 4(1):77–109.

Advaith Siddharthan. 2014. A survey of research on text simplification. *International Journal of Applied Linguistics*, 165:259–298.

Lucia Specia. 2010. Translating from complex to simplified sentences. In *Proceedings of the 9th International Conference on Computational Processing of the Portuguese Language*.

Amanda Stent, Rashmi Prassad, and Marilyn Walker. 2004. Trainable sentence planning for complex information presentations in spoken dialog systems. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL)*.

Paul Taylor and Alan W. Black. 1998. Assigning phrase breaks from part-of-speech sequences. *Computer Speech and Language*, 12:99–117.

Huihsin Tseng, Pichuan Chang, Galen Andrew, Daniel Jurafsky, and Christopher Manning. 2005. A conditional random field word segmenter. In *Prroceedings of the Fourth SIGHAN Workshop on Chinese Language Processing*.

Marilyn Walker, Owen Rambow, and Monica Rogati. 2001. Spot: A trainable sentence planner. In *Proceedings of the Second Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Kristian Woodsend and Mirella Lapata. 2011. Learning to simplify sentences with quasi-synchronous grammar and integer programming. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Sander Wubben, Antal van den Bosch, and Emiel Krahmer. 2012. Sentence simplification by monolingual machine translation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Donghua Xu and Chew Lim Tan. 1996. Automatic alignment of English-Chinese bilingual texts of CNS news. In *Proceedings of International Conference on Chinese Computing*.

Jia Xu, Richard Zens, and Hermann Ney. 2005. Sentence segmentation using IBM word alignment model 1. In *In Proceedings of the 10th Annual Conference of the European Association for Machine Translation (EAMT)*.

Nianwen Xue and Yaqin Yang. 2011. Chinese sentence segmentation as comma classification. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-HLT): Short Papers*.

Naiwen Xue, Fei Xia, Fu-dong Chiou, and Marta Palmer. 2005. The Penn Chinese TreeBank: Phrase structure annotation of a large corpus. *Natural Language Engineering*, 11(2):207–238, June.

Yaqin Yang and Nianwen Xue. 2012. Chinese comma disambiguation for discourse analysis. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Dapeng Yin, Fuji Ren, Peilin Jiang, and Shingo Kuroiwa. 2007. Chinese complex long sentences processing method for Chinese-Japanese machine translation. In *Proceedings of the International Conference on Natural Language Processing and Knowledge Engineering (NLP-KE)*.

Zhemin Zhu, Delphine Bernhard, and Iryna Gurevych. 2010. A monolingual tree-based translation model for sentence simplification. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING)*.

Search-Aware Tuning for Hierarchical Phrase-based Decoding

Feifei Zhai

Queens College
City University of New York
ffzhai2012@gmail.com

Liang Huang **Kai Zhao**

Queens College & Graduate Center
City University of New York
{lianghuang.sh,kzhao.hf}@gmail.com

Abstract

Parameter tuning is a key problem for statistical machine translation (SMT). Most popular parameter tuning algorithms for SMT are agnostic of decoding, resulting in parameters vulnerable to search errors in decoding. The recent research of "search-aware tuning" (Liu and Huang, 2014) addresses this problem by considering the partial derivations in every decoding step so that the promising ones are more likely to survive the inexact decoding beam. We extend this approach from phrase-based translation to syntax-based translation by generalizing the evaluation metrics for partial translations to handle tree-structured derivations in a way inspired by inside-outside algorithm. Our approach is simple to use and can be applied to most of the conventional parameter tuning methods as a plugin. Extensive experiments on Chinese-to-English translation show significant BLEU improvements on MERT, MIRA and PRO.

1 Introduction

Efforts in parameter tuning algorithms for SMT, such as MERT (Och, 2003; Galley et al., 2013), MIRA (Watanabe et al., 2007; Chiang et al., 2009; Cherry and Foster, 2012; Chiang, 2012), and PRO (Hopkins and May, 2011) have improved the translation quality considerably in the past decade. These tuning algorithms share the same characteristic that they treat the decoder as a black box. This decoding insensitiveness has two effects: 1) the parameter tuning algorithm can be more general to choose the most effective decoding paradigm for different language pairs; 2) however, it also means that the learned parameters may not fit the decoding algorithm well, so that promising partial translations might be pruned out due to the beam search decoding.

Recent researches reveal that the parameter tuning algorithms tailored for specific decoding algorithms can be beneficial in general structured prediction problems (Huang et al., 2012), and in machine translation (Yu et al., 2013; Zhao et al., 2014; Liu and Huang, 2014). Particularly, Liu and Huang (2014) show that by requiring the conventional parameter tuning algorithms to consider the final decoding results (full translations) as well as the intermediate decoding states (partial translations) at the same time, the inexact decoding can be significantly improved so that correct intermediate partial translations are more likely to survive the beam. However, the underlying phrase-based decoding model suffers from limited distortion, and thus, may not be flexible enough for translating language pairs that are syntactically different, which require long distance reordering.

In order to better handle long distance reordering which beyonds the capability of phrase-based MT, we extend the search-aware tuning framework from phrase-based MT to syntax-based MT, in particular the hierarchical phrase-based translation model (HIERO) (Chiang, 2007).

One key advantage of search-aware tuning for previous phrase-based MT is the minimal change to existing parameter tuning algorithms, which is achieved by defining BLEU-like metrics for the intermediate decoding states with sequence-structured derivations. To keep our approach simple, we generalize these BLEU-like metrics to handle intermediate decoding states with tree-structured derivations in HIERO, which are calculated by dynamic programming algorithms inspired by the inside-outside algorithm.

We make the following contributions:

1. We extend the framework of search-aware tuning methods from phrase-based translation to syntax-based translation. This extension is simple to be applied to most conventional parameter tuning methods, requiring minimal extra changes to existing algorithms.

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1282–1291,
Lisbon, Portugal, 17-21 September 2015.

2. We propose two BLEU metrics and their variants to evaluate partial derivations. In order to efficiently compute the new BLEU metrics, we investigate dynamic programming based algorithms to recover the good partial derivations for search-aware tuning.

3. Our method obtains significant improvements on large-scale Chinese-to-English translation on top of MERT, MIRA and PRO baselines.

2 Hierarchical Phrase-based Decoding

We first briefly describe the hierarchical phrase-based translation (Chiang, 2007), HIERO, by a Chinese-English translation example (Figure 1).

A HIERO decoder translates a source sentence by using a synchronous grammar (Figure 1 (a)) to parse it into a bilingual derivation tree, as shown in Figure 1 (b) and Figure 1 (c). An example rule of the synchronous grammar is

$$r_8 : \mathrm{X} \rightarrow \langle \mathrm{X}_1 \text{ qian } \mathrm{X}_2, \mathrm{X}_2 \text{ before } \mathrm{X}_1 \rangle,$$

where the left-hand-side is a non-terminal symbol, and the right-hand-side is a pair of source and target strings, each of which consists a sequence of lexical terminals and non-terminals. Specifically, the same subscript on both sides denotes a one-to-one correspondence between their non-terminals. We use $|r|$ to denote the arity of rule r, i.e., the number of non-terminals. Usually, the rule used in HIERO system has a maximum arity 2.

Let $\langle x, y \rangle$ be a Chinese-English sentence pair in tuning set, the HIERO decoder will generate a derivation tree for it (Figure 1 (b)), which can be defined recursively in a *functional* way:

$$d = \begin{cases} r & |r| = 0 \\ r(d_1) & |r| = 1 \\ r(d_1, d_2) & |r| = 2 \,, \end{cases} \quad (1)$$

where d is a (partial) derivation, i.e., the (sub-) derivation tree. When r is a fully lexicalized rule ($|r| = 0$), the decoder generates a tree node directly (e.g., $X_{[3,5]}$ in Figure 1 (b)). If $|r| > 0$, a new (partial) derivation will be created by applying r to its children nodes. In our notation, we denote this process as applying a function of rule r to its arguments. For example, node $X_{[1,5]}$ in Figure 1(b) is created by applying rule r_8 in Figure 1(a), where the arguments are $d_1 = X_{[1,2]}$ and $d_2 = X_{[3,5]}$ respectively.

Practically, we organize the partial derivations based on spans in HIERO decoder, and use a beam $B_{[i,j]}$ to keep the k-best partial derivations for each span $[i, j]$:

$$B_{[i,j]} = \mathbf{top}^k_{\mathbf{w}} \left(D_{[i,j]} \right),$$

where $D_{[i,j]}$ is the set of all possible partial derivations for span $[i, j]$, and $\mathbf{top}^k_{\mathbf{w}}(\cdot)$ returns the top k ones according to the current model $\mathbf{w}$. Figure 2 shows an example of the HIERO beam-search decoding.

3 Search-Aware Tuning for HIERO

As we discussed in Section 1, current tuning methods for HIERO system (MERT, MIRA, or PRO) are mostly *search-agnostic*. They only consider the *complete* translations in final beam $B_{[1,|x|]}$, but ignore the partial ones in the intermediate beams. However, because MT decoding is inexact (beam-search), many potentially promising derivations might be pruned before reaching the final beam (e.g., the partial derivation $X_{[1,5]}$ of Figure 1(b) is pruned in beam $B_{[1,5]}$ in Figure 2). Consequently, once we lose these good partial derivations in decoding, it is hard to promote them by these search-agnostic tuning methods.

In order to address this problem, search-aware tuning (Liu and Huang, 2014) aims to promote not only the accurate complete translations in the final beam, but more importantly those promising partial derivations in non-final beams.

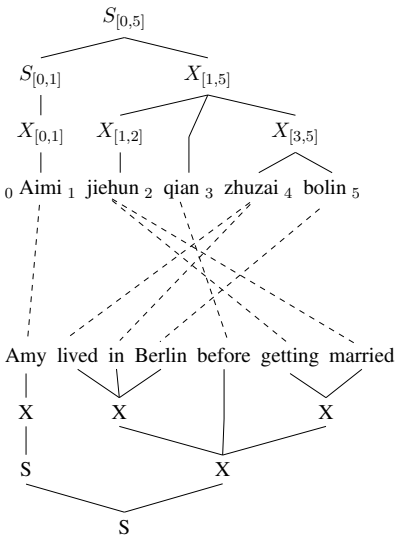
In this section, we apply search-aware tuning to HIERO system. Similar to the phrase-based one, the key challenge here is also the evaluation of partial derivations. Following (Liu and Huang, 2014), we design two different evaluation metrics, *partial* BLEU and *potential* BLEU respectively for HIERO decoding.

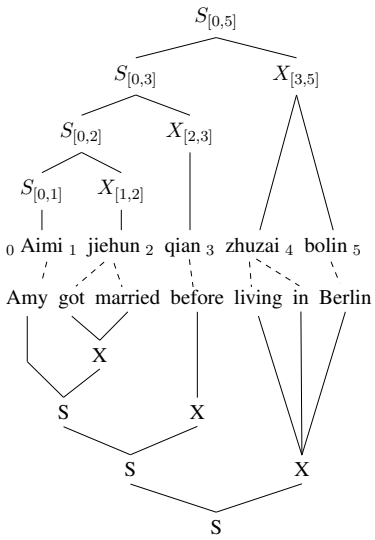
3.1 Partial BLEU

Given a partial derivation d of span $[i, j]$, *partial* BLEU evaluates it by comparing its partial translation $e(d)$ directly with the (full) reference. We explore two different partial BLEU measures here: *full partial* BLEU and *span partial* BLEU.

Full partial BLEU is similar to the one used in (Liu and Huang, 2014), which compares $e(d)$ with the full reference y and computes BLEU score using an adjusted effective reference length proportional to the length of the source span $[i, j]$ (i.e.,

id	rule
r_0	S → ⟨X$_1$, X$_1$⟩
r_1	S → ⟨S$_1$X$_1$, S$_1$X$_1$⟩
r_2	X → ⟨Aimi, Amy⟩
r_2	X → ⟨Aimi, Amy has⟩
r_3	X → ⟨zhuzai bolin, lived in Berlin⟩
r_3	X → ⟨zhuzai bolin, living in Berlin⟩
r_4	X → ⟨qian, before⟩
r_5	X → ⟨qian, former⟩
r_6	X → ⟨jiehun, got married⟩
r_7	X → ⟨jiehun, getting married⟩
r_8	X → ⟨X$_1$ qian X$_2$,X$_2$ before X$_1$⟩
r_9	X → ⟨X$_1$ qian X$_2$,X$_1$ before X$_2$⟩





(a) HIERO rules (b) good derivation (c) bad derivation

Figure 1: An example of HIERO translation.

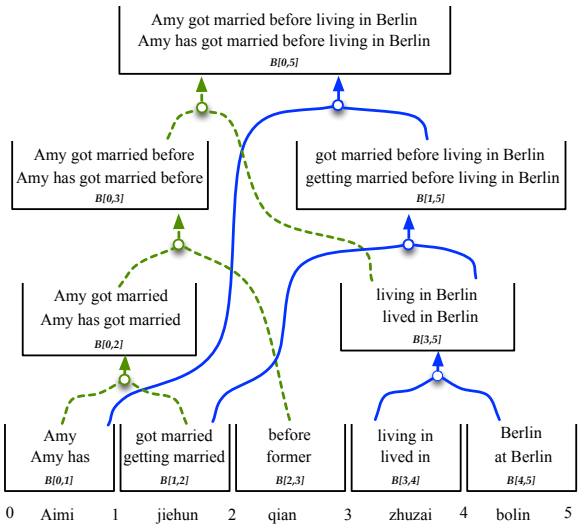


Figure 2: An example of HIERO beam search decoding (beam size = 2). We can see the good derivation (Figure 1(b)) is pruned very early, while the bad derivation (Figure 1(c)) ranks first at the final beam.

the number of translated source words). See (Liu and Huang, 2014) for more details.

Span partial BLEU differs from full partial BLEU by creating a *span reference* for span $[i, j]$ to evaluate its partial derivations, rather than using the full reference. We use word alignment to determine the span references of different spans. Specifically, we first add the tuning set into training data, and then run GIZA++ to get the word alignment. For a pair of source and target span which are consistent with word alignment[1], the string of target span is used as the span reference for the source span.[2] For example, in Figure 1(b), the span reference of source span $[1, 5]$ is "lived in Berlin before getting married".

Partial BLEU is quite an intuitive choice for evaluating partial derivations. However, as Liu and Huang (2014) discussed, partial BLEU only evaluates the partial derivation itself without considering any of its context information, leading to a performance degradation. Thus, in order to take the context information into account, we investigate a more reasonable metric, potential BLEU, in the next section.

3.2 Potential BLEU

Potential BLEU evaluates the partial derivation d by assessing its *potential*, which is a complete translation generated from d, rather than $e(d)$ as in partial BLEU. In phrase-based decoding, Liu

[1]Two spans are consistent with word alignment means that words in one span only align to words in the other span via word alignment, and vice versa.

[2]In practice, there might be several different consistent target spans for a source span due to the unaligned words. We use the longest target span as the span reference which shows the best performance in our experiments.

and Huang (2014) introduce a future string $f(d)$ to denote the translation of the untranslated part, and get the potential of d by concatenating $e(d)$ and $f(d)$.

Different from their work, we define an *outside string* for d in HIERO system. Suppose a complete translation generated from d is $\bar{e}(d)$, it can be decomposed as follows:

$$\bar{e}(d) = \eta(d) \circ e(d) \circ \xi(d) \quad (2)$$

where $\circ$ is an operator that concatenates strings monotonically, $\eta(d)$ and $\xi(d)$ are the left and right parts of $\bar{e}(d)$ apart from $e(d)$, which we call *left* and *right future strings* of d. The pair of $\eta(d)$ and $\xi(d)$ is defined as the outside string of d:

$$O(d) = \langle \eta(d),\ \xi(d) \rangle$$

An example of the outside string is shown in Figure 3. Assume we are evaluating the partial derivation which translates Chinese word "qian" to English word "before", and get a complete translation "Amy lived in Berlin before getting married" from it, then its outside string would be:

$$\langle \text{Amy lived in Berlin,}\ \ \text{getting married} \rangle$$

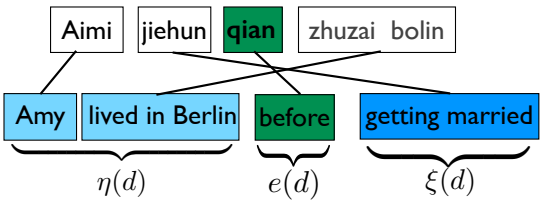


Figure 3: Example of outside string when we use"Amy lived in Berlin before getting married" as the potential for the partial derivation which translates word "qian" to "before".

Theoretically, different partial derivations of the same span could have different outside strings. To simplify the problem, we use the same outside string for all partial derivations of the same span. The outside string for span $[i, j]$ is defined as:

$$O([i,j]) = \langle \eta([i,j]),\ \xi([i,j]) \rangle$$

For a specific partial derivation d of span $[i, j]$, by combining $O([i, j])$ and $e(d)$, we can compute its potential BLEU against the reference. Apparently, different outside string will lead to different potential BLEU score. In the rest of this section, we will explore three different methods to generate outside strings.

3.2.1 Concatenation

In order to generate outside string, one simple and straightforward way is concatenation. For a specific span $[i, j]$, we first get the best translation of its adjacent spans $[0, i]$ and $[j, |x|]$ ($e([0, i])$ and $e([j, |x|])$ respectively).[3] Then for a partial derivation d of span $[i, j]$, we generate the outside string of d by *concatenation*

$$\eta([i,j]) = e([0,i])$$
$$\xi([i,j]) = e([j,|x|])$$
$$\bar{e}_x(d) = e([0,i]) \circ e(d) \circ e([j,|x|])$$

Also take the example of Figure 3, if we perform concatenation on it, the outside string of the corresponding partial derivation is:

$$\langle \text{Amy getting married,}\ \ \text{lived in Berlin} \rangle.$$

Obviously, this outside string is not good, since it does not consider the reordering between spans. In order to incorporate reordering into outside string, we propose a new top-down algorithm in the next subsection.

3.2.2 Top-Down

Top-down method (Algorithm 1) is defined over the derivation tree, and takes the complete translations in the final beam as the potential for computing potential BLEU.

Suppose we have a partial derivation $d = r(d_1, d_2)$ as shown in Formula 1, and the target side string of rule r is:

$$w_1 \cdots w_p\ X_1\ w_q \cdots w_m\ X_2\ w_n \cdots w_l$$

The corresponding *(partial) translation* $e(d)$ of d could be generated by applying an r-based function $e_r(\cdot)$ with d_1 and d_2 as the arguments:

$$e_r(d_1, d_2) = w_1 \cdots w_p\ e(d_1)\ w_q \cdots w_m\ e(d_2)\ w_n \cdots w_l$$

where $e(d_1)$ and $e(d_2)$ are the partial translations of d_1 and d_2. We can further decompose $e_r(d_1, d_2)$ based on $e(d_1)$:

$$\underbrace{w_1 \cdots w_p}_{\eta_d(d_1)}\ \ e(d_1)\ \ \underbrace{w_q \cdots w_m\ \ e(d_2)\ \ w_n \cdots w_l}_{\xi_d(d_1)}.$$

where we call $\eta_d(d_1)$ and $\xi_d(d_1)$ the *partial left and right future strings* of d_1 under d. Similar

[3] For the spans not having translations, we compute a best possible translation for each of them by concatenating the translations of its children spans monotonically.

Algorithm 1 Top-Down Outside String.

```
1: function TopDown(d)
2:   r ← the rule that generates d
3:   for each non-terminal x in rule r do
4:     η(d_x) = η(d) ∘ η_d(d_x)
5:     ξ(d_x) = ξ_d(d_x) ∘ ξ(d)
6:     TopDown(d_x)
7:   return
```

to the outside cost of inside-outside algorithm, we compute the outside string $\langle \eta(d_1), \xi(d_1) \rangle$ for d_1 based on the decomposition:

$$\eta(d_1) = \eta(d) \circ \eta_d(d_1)$$
$$\xi(d_1) = \xi_d(d_1) \circ \xi(d).$$

Similarly, if we decompose $e_r(d_1, d_2)$ based on $e(d_2)$ we have:

$$\underbrace{w_1 \cdots w_p \;\; e(d_1) \;\; w_q \cdots w_m}_{\eta_d(d_2)} \;\; e(d_2) \;\; \underbrace{w_n \cdots w_l}_{\xi_d(d_2)},$$

and the outside string $\langle \eta(d_2), \xi(d_2) \rangle$ for d_2 is:

$$\eta(d_2) = \eta(d) \circ \eta_d(d_2)$$
$$\xi(d_2) = \xi_d(d_2) \circ \xi(d).$$

The top-down method works based on the above decompositions. For a complete translation in the final beam $B_{[0,|x|]}$, the algorithm tracebacks on its derivation tree, and gets the outside string for all spans in the derivation. Taking the span $[1, 2]$ in Figure 1(b) as an example, if we do top-down, its outside string would be:

$$\langle \text{Amy lived in Berlin before}, \;\; \epsilon \rangle.$$

where ϵ denotes the empty string. Compared to the concatenation method, top-down is able to consider the reordering between spans, and thus would be much better. However, since it bases on the k-best list in the final beam, it can only handle the spans appearing in the final k-best derivations. In our experiments, top-down algorithm only covers about 30% of all spans.[4] In order to incorporate more spans into search-aware tuning, we enhance the top-down algorithm and propose guided backtrace algorithm in the next subsection.

3.2.3 Guided Backtrace

The *guided backtrace* algorithm is a variation of top-down method. In guided backtrace, we first introduce a container $s_{[i,j]}$ to keep a best partial derivation for each split point of $[i, j]$ during decoding.[5] Hence, there will be at most $j - i$ partial derivations in $s_{[i,j]}$.[6] For instance, suppose the span is $[2, 5]$, we will keep three partial derivations in $s_{[i,j]}$ for split point 2, 3, and 4 respectively. Different from the similar k-best list in the decoding beam, $s_{[i,j]}$ introduces more diverse partial derivations for backtracing, and thus could help to cover more spans.

After decoding, we employ algorithm 2 to do backtrace. The algorithm starts from the best translation in the final beam. At first, we get the corresponding span of the input partial derivation d (line 2), and the outside string for this span (line 5-6). We then sort the partial derivations in $s_{[i,j]}$ based on their potential Bleu^{+1} (line 12). Thus, the sorting will guide us to first backtrace the partial derivations with better potential Bleu^{+1} scores. Then we traverse all these partial derivations (line 13-14), and do guided backtrace recursively on its child partial derivations (line 16-19). In this process, about 90% of all spans are visited in our experiments. We demand each span will only be visited once (line 3-4),

During the above process, guided backtrace will collect good partial derivations which have better potential Bleu^{+1} scores than the best Bleu^{+1} score *MaxSenBleu* of the final beam (line 10-11) into *goodcands*. Meanwhile, we also collect the good partial derivations from descendant nodes (line 19), and apply rule r to them to form good partial derivations for span $[i, j]$ (line 20-21). At last, we add the top 50 good partial derivations to the beam $B_{[i,j]}$ for tuning (line 22).

The purpose of adding good partial derivations for tuning is to recover the good ones pruned out of the beam. For example, the partial derivation $X_{[1,5]}$ in Figure 1 has been pruned out of the beam $B_{[1,5]}$ in Figure 2. If we only consider the partial derivations in the beam, it is still hard to promote it. After adding good partial derivations, we will have more good targets to do better tuning.

From Algorithm 2, we can also get a new way to compute oracle BLEU score of a translation system. We memorize the string that has the maximum potential Bleu^{+1} score of all strings (line 9). We then collect the best string of all source sen-

[4] We do not consider the hypothesis recombination in top-down algorithm.

[5] If the partial derivation is generated by HIERO rules, we set the split point as the last position of the first non-terminal.

[6] Some split points might not have corresponding partial derivations.

Algorithm 2 Guided Backtrace Outside String.

function GUIDED(d)
  $[i, j] \leftarrow$ the source span that d belongs to
  if $[i, j]$ has been visited **then**
    return $\emptyset$
  $\eta([i, j]) = \eta(d)$
  $\xi([i, j]) = \xi(d)$
  goodcands $\leftarrow \emptyset$
  for each partial derivation d in $B_{[i,j]}$ **do**
    $bleu = \text{Bleu}^{+1}(\eta([i, j]) \circ e(d) \circ \xi([i, j]))$
    if $bleu > MaxSenBleu$ **then**
      add d to goodcands
  sort $s_{[i,j]}$ based on potential Bleu^{+1}
  for each partial derivation d of span $[i, j]$ **do**
    $r \leftarrow$ the rule that generates d
    $D_d \leftarrow$ the set of partial derivations r use to form d
    for each non-terminal x in rule r **do**
      $\eta(d_x) = \eta([i, j]) \circ \eta_d(d_x)$
      $\xi(d_x) = \xi_d(d_x) \circ \xi([i, j])$
      cands $\leftarrow$ GUIDED(d_x)
      for each cand d_g in cands **do**
        add $r(d_g, D_k \backslash d_x)$ to goodcands
  add top 50 partial derivations of goodcands to $B_{[i,j]}$
  return goodcands

tences in the tuning or test set, and use them to compute BLEU score of the set. This can be seen as an approximation upper bound of the current model and decoder, which we call **guided oracle**.

3.3 Implementation

The major process of search-aware tuning is similar to traditional tuning pipeline. We first decode the source sentence, and then output both the complete translations in the final beam and the partial derivations from the shorter spans as training instances. For potential BLEU, the outputs of partial derivations would be the corresponding complete translations generated by Equation (2). If we use partial BLEU, the outputs are the corresponding partial translations. Each span will serve as a single tuning instance. Different from (Liu and Huang, 2014), we only use the features from partial derivations for tuning.

For the spans that top-down or guided backtrace algorithm cannot get outside strings, we use concatenation for them to maintain consistent number of tuning instances between different tuning iterations. However, since we do not want to spent much effort on them, we only use the one-best partial derivation for each of them.

4 Experiments

To evaluate our method, we conduct experiments on Chinese-to-English translation. The training data includes 1.8M bilingual sentence pairs, with about 40M Chinese words and 48M English words. We generate symmetric word alignment using GIZA++ and the *grow-diag-final-and* strategy. We train a 4-gram language model on the Xinhua portion of English Gigaword corpus by SRILM toolkit (Stolcke, 2002). We use BLEU 4 with "average reference length" to evaluate the translation performance for all experiments.

We use the NIST MT 2002 evaluation data (878 sentences) as the tuning set, and adopt NIST MT04 (1,789 sentences), MT05 (1,082 sentences), MT06 (616 sentences from new portion) and MT08 (691 sentences from new portion) data as the test set.

Our baseline system is an in-house hierarchical phrase-based system. The translation rules are extracted with Moses toolkit (Koehn et al., 2007) by default settings. For the decoder, we set the beam size to 30, nbest list to 50, and 20 as the maximum length of spans for using non-glue rules.

The baseline tuning methods are batch tuning methods based on k-best translations, including MERT (Och, 2003), MIRA (Cherry and Foster, 2012) and PRO (Hopkins and May, 2011) from Moses. Another baseline tuning method is hypergraph-MERT from `cdec` toolkit (Dyer et al., 2010). To guarantee the tuning efficiency, we constrain the minimum length of spans for search-aware tuning to restrict the number of training instances. For the sentences with less than 20 words, we only use the spans longer than 0.75 times sentence length. For the ones with more than 20 words, the minimum span length is set to 18.[7] All results are achieved by averaging three independent runs for fair comparison (Clark et al., 2011).

4.1 Translation Results

Table 1 compares the main results of our MERT-based search-aware tuning with traditional tuning: MERT and hypergraph-MERT.

From the results, we can see that hypergraph-MERT is better than MERT by 0.5 BLEU points, verifying the result of (Kumar et al., 2009). For search-aware tuning, partial BLEU (both full and span one) only gets comparable results with baseline tuning method, confirming our previous analysis in section 3.1, and the results are also consistent with (Liu and Huang, 2014).

[7] As the decoder demands that spans longer than 20 can only be translated by glue rule, for these spans, we only need to consider the ones beginning with 0 in search-aware tuning.

	nist03	nist04	nist05	nist06	nist08	avg.
MERT	34.4	35.9	34.2	31.9	28.2	32.9
MERT-S	34.3	36.1	34.3	32.3	28.5	33.1
hypergraph-MERT	34.5	36.4	34.4	33.0	28.9	33.4
full partial BLEU	34.3	35.9	34.1	32.4	28.1	33.0
span partial BLEU	34.1	36.1	34.5	32.5	28.4	33.1
concatenation	34.3	36.4	34.6	33.1	28.5	33.4
top-down	34.5	36.6	34.5	33.1	28.9	33.5
guided backtrace	**34.9**	**36.9**	**35.2**	**33.7**	**29.1**	**34.0**

Table 1: MERT results: BLEU scores the test sets (nist03, nist04, nist05, nist06, and nist08). MERT-S is an enhanced version of MERT, which uses a beam size 200 and nbest list 200 for tuning, and beam size 30 for testing.

	methods	nist02	nist04
1-best	MERT	36.1	35.9
	guided backtrace	**+0.5**	**+1.0**
k-best Oracle	MERT	43.3	42.8
	guided backtrace	**+1.1**	**+1.2**
Guided Oracle	MERT	48.0	47.2
	guided backtrace	**+1.4**	**+1.4**

Table 2: The oracle BLEU comparison between baseline MERT and guided backtrace.

Compared to partial BLEU, potential BLEU is more helpful. Both concatenation and top-down method are better than MERT on all five test sets. Guided backtrace gets the best performance over all methods. It outperforms traditional MERT by 1.1 BLEU points on average, and better than hypergraph-MERT by 0.6 BLEU points.

4.2 Analysis

In this section, we analyze the results of search-aware tuning by comparing MERT-based baseline and guided backtrace in detail.

Oracle BLEU We first compare the oracle BLEU scores of baseline and guided backtrace in Table 2. In order to get the k-best oracle, we first look for the best Bleu^{+1} translation in the k-best list for each source sentence, and then use these best translations to compute the BLEU score of the entire set. To get the guided oracle, we use the weights from baseline MERT to run Algorithm 2 on tuning set (nist02) and nist04 test set, and generate the best oracle translation (section 3.2.3) for each source sentence for evaluation.

The final oracle BLEU comparison is shown in Table 2. On both nist02 tuning set and nist04 test set, guided backtrace method gains at lease 1.0 BLEU points improvements over traditional MERT on both k-best oracle and guided oracle. Moreover, k-best and guided oracle get more improvements than 1-best, indicating that by search-aware tuning, the decoder could generate a better k-best list, and has a higher upper bound.

	nist02	nist04
MERT	0.3960	0.3791
guided backtrace	0.4098	0.3932

Table 3: The diversity comparison based on k-best list in the final beam on both tuning set (nist02) and nist04 test set. The higher the value is, the more diverse the k-best list is.

Diversity As shown in (Gimpel et al., 2013; Liu and Huang, 2014), diversity is important for MT tuning. An k-best list with higher diversity can better represent the entire decoding space, and thus tuning on it will lead to a better performance.

Similar as (Liu and Huang, 2014), our method encourages the diversity of partial translations in each beam, including the ones in the final beam. We use the metric in (Liu and Huang, 2014) to compare the diversity of traditional MERT and guided backtrace. The metric is based on the n-gram matches between two sentences y and y':

$$\Delta(y, y') = -\sum_{i=1}^{|y|-q} \sum_{j=1}^{|y'|-q} \langle y_{i:i+q} = y'_{j:j+q} \rangle$$

$$d(y, y') = 1 - \frac{2 \times \Delta(y, y')}{\Delta(y, y) + \Delta(y', y')},$$

where $q = n - 1$ and $\langle x \rangle$ equals to 1 if x is true, 0 otherwise. The final diversity results are shown in Table 3. We can see guided backtrace gets a better diversity than traditional MERT.

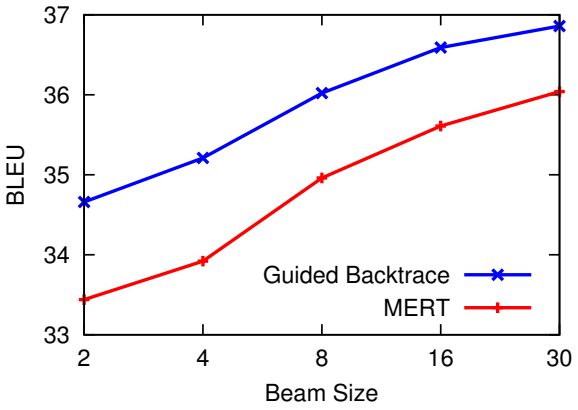


Figure 4: The BLEU comparison between MERT and guided backtrace on nist04 test set over different beam sizes.

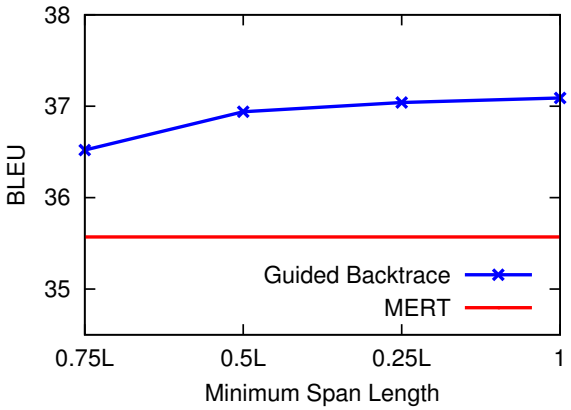


Figure 5: The BLEU comparison between MERT and guided backtrace on nist04 test set over different span sizes. L denotes the sentence length. The value of x-axis refers to the minimum length of spans used for search-aware tuning. More spans will be used in tuning with smaller minimum span length.

Beam Size As we discussed before, search-aware tuning helps to accommodate search errors in decoding, and promotes good partial derivations. Thus, we believe that even with a small beam, these good partial derivations can still survive with search-aware tuning, resulting in a good translation quality. Figure 4 compares the results of different beam sizes $(2, 4, 8, 16, 30)$ between traditional MERT and guided backtrace. The comparison shows that guided backtrace achieves better result than baseline MERT, and when the beam is smaller, the improvement is bigger. Moreover, guided backtrace method with a beam size 8 could achieve comparable BLEU score to traditional MERT with beam size 30.

Span Size For a big tuning set, in order to make the tuning tractable, we constrain the length of spans for search-aware tuning. Intuitively, towards search-aware tuning, more spans will get better results because we have more training instances to guide the tuning process, and accommodate search errors in early decoding stage (shorter spans). To verify this intuition, we perform experiments on a small tuning set, which is generated by selecting the sentences with less than 20 words from the original NIST MT 02 tuning set.

Figure 5 compares the results of traditional MERT and guided backtrace over different minimum span length. The curve in the figure shows that by using more spans for search-aware tuning, we can achieve better translation performance.

4.3 MIRA and PRO Results

Table 4 and Table 5 shows the final results of MIRA and PRO for traditional tuning and search-aware tuning using potential BLEU. We can see that potential BLEU is helpful for tuning. Guided backtrace is also the best one, which outperforms the baseline MIRA and PRO by 0.9 and 0.8 BLEU points on average.

4.4 Efficiency

Table 6 shows the training time comparisons between search-aware tuning and traditional tuning. From this Table, we can see that search-aware tuning slows down the training speed.[8] The slow training is due to three reasons.

Similar to (Liu and Huang, 2014), as search-aware tuning considers partial translations of spans besides the complete translations, it has much more training instances than traditional tuning. In our experiments, although we have constrained the length of spans, we get a total of 38,539 instances for search-aware tuning, while it is 878 for traditional tuning.

Another time consuming factor is the computation of Bleu^{+1} scores. In guided backtrace setting, we need to compute the Bleu^{+1} scores for all candidates of spans we consider. This is why the

[8] Since the tuning set is different for traditional PRO tuning and search-aware tuning, we do not compare them here.

	nist03	nist04	nist05	nist06	nist08	avg.
MIRA	34.5	36.1	34.4	32.1	28.5	33.1
concatenation	34.5	36.5	34.6	33.6	28.9	33.6
top-down	34.4	36.6	34.6	33.5	28.8	33.6
guided backtrace	**35.0**	**36.9**	**35.1**	**33.7**	**29.1**	**34.0**

Table 4: MIRA results: BLEU scores on test sets (nist03, nist04, nist05, nist06, and nist08).

	nist03	nist04	nist05	nist06	nist08	avg.
PRO	34.2	36.1	33.9	32.3	28.7	33.1
concatenation	34.6	36.2	34.7	32.3	28.5	33.2
top-down	34.8	36.4	34.7	32.7	29.0	33.5
guided backtrace	**35.0**	**36.8**	**34.7**	**33.4**	**29.6**	**33.9**

Table 5: PRO results: BLEU scores on test sets (nist03, nist04, nist05, nist06, and nist08). Similar to (Liu and Huang, 2014), there is also an monster phenomenon (Nakov et al., 2013) in our search-aware tuning setting. Therefore, here we perform search-aware tuning on only 109 short sentences (with less than 10 words) from nist02.

	MERT		MIRA	
	Total	Optim.	Total	Optim.
baseline	17	2	17	2
concatenation	59	44	38	23
top-down	31	14	23	8
guided backtrace	108	45	69	22

Table 6: Comparison on training efficiency. The time (in minutes) is measured at the last iteration of tuning. Column "Total" refers to the time for an entire iteration, while "Optim." is the time of optimization. We use a parallelized MERT for tuning by 24 cores.

decoding of guided backtrace is much slower than baseline or top-down.

Finally, adding good candidates enlarges the k-best lists of training instances, which further slows down the tuning process of guided backtrace.

It should be noted that although search-aware tuning is slower than traditional tuning method, since the decoding is all the same for them in testing time, it does not slow down the testing speed.

5 Conclusion

We have presented an extension of "search-aware tuning" to accommodate search errors in hierarchical phrase-based decoding and promote the promising partial derivations so that they are more likely to survive in the inexact beam search. In order to handle the tree-structured derivations for HIERO system, we generalize the BLEU metrics and propose corresponding partial BLEU and potential BLEU to evaluate partial derivations. Our approach can be used as a plugin for most popular parameter tuning algorithms including MERT, MIRA and PRO. Extensive experiments confirmed substantial BLEU gains from our method.

6 Acknowledgment

We thank the three anonymous reviewers for the valuable suggestions. This project was supported in part by DARPA FA8750-13-2-0041 (DEFT), NSF IIS-1449278, and a Google Faculty Research Award.

References

Colin Cherry and George Foster. 2012. Batch tuning strategies for statistical machine translation. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 427–436, Montréal, Canada, June. Association for Computational Linguistics.

David Chiang, Kevin Knight, and Wei Wang. 2009. 11,001 new features for statistical machine translation. In *Proceedings of NAACL 2009*.

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–208.

David Chiang. 2012. Hope and fear for discriminative training of statistical translation models. *J. Machine Learning Research (JMLR)*, 13:1159–1187.

Jonathan H. Clark, Chris Dyer, Alon Lavie, and Noah A. Smith. 2011. Better hypothesis testing for statistical machine translation: Controlling for optimizer instability. In *Proceedings of the 49th Annual Meeting of the Association for Computational*

Linguistics: Human Language Technologies, pages 176–181, Portland, Oregon, USA, June. Association for Computational Linguistics.

Chris Dyer, Adam Lopez, Juri Ganitkevitch, Johnathan Weese, Ferhan Ture, Phil Blunsom, Hendra Setiawan, Vladimir Eidelman, and Philip Resnik. 2010. cdec: A decoder, alignment, and learning framework for finite-state and context-free translation models. In *Proceedings of the ACL*.

Michel Galley, Chris Quirk, Colin Cherry, and Kristina Toutanova. 2013. Regularized minimum error rate training. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1948–1959, Seattle, Washington, USA, October. Association for Computational Linguistics.

Kevin Gimpel, Dhruv Batra, Chris Dyer, and Gregory Shakhnarovich. 2013. A systematic exploration of diversity in machine translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1100–1111, Seattle, Washington, USA, October. Association for Computational Linguistics.

Mark Hopkins and Jonathan May. 2011. Tuning as ranking. In *Proceedings of EMNLP*.

Liang Huang, Suphan Fayong, and Yang Guo. 2012. Structured perceptron with inexact search. In *Proceedings of NAACL*.

P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin, and E. Herbst. 2007. Moses: open source toolkit for statistical machine translation. In *Proceedings of ACL: Demonstrations*.

Shankar Kumar, Wolfgang Macherey, Chris Dyer, and Franz Och. 2009. Efficient minimum error rate training and minimum bayes-risk decoding for translation hypergraphs and lattices. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pages 163–171. Association for Computational Linguistics.

Lemao Liu and Liang Huang. 2014. Search-aware tuning for machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1942–1952, Doha, Qatar, October. Association for Computational Linguistics.

Preslav Nakov, Francisco Guzmán, and Stephan Vogel. 2013. A tale about pro and monsters. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 12–17, Sofia, Bulgaria, August. Association for Computational Linguistics.

Franz Joseph Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of ACL*, pages 160–167.

Andreas Stolcke. 2002. Srilm - an extensible language modeling toolkit. In *Proceedings of ICSLP*, volume 30, pages 901–904.

Taro Watanabe, Jun Suzuki, Hajime Tsukada, and Hideki Isozaki. 2007. Online large-margin training for statistical machine translation. In *Proceedings of EMNLP-CoNLL*.

Heng Yu, Liang Huang, Haitao Mi, and Kai Zhao. 2013. Max-violation perceptron and forced decoding for scalable MT training. In *Proceedings of EMNLP*.

Kai Zhao, Liang Huang, Haitao Mi, and Abe Ittycheriah. 2014. Hierarchical mt training using max-violation perceptron. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 785–790, Baltimore, Maryland, June. Association for Computational Linguistics.

Part-of-speech Taggers for Low-resource Languages using CCA Features

Young-Bum Kim[†] **Benjamin Snyder**[‡] **Ruhi Sarikaya**[†]

[†]Microsoft Corporation, Redmond, WA
[‡]University of Wisconsin-Madison, Madison, WI
{ybkim, ruhi.sarikaya}@microsoft.com
bsnyder@cs.wisc.edu

Abstract

In this paper, we address the challenge of creating accurate and robust part-of-speech taggers for low-resource languages. We propose a method that leverages existing parallel data between the target language and a large set of resource-rich languages without ancillary resources such as tag dictionaries. Crucially, we use CCA to induce latent word representations that incorporate cross-genre distributional cues, as well as projected tags from a full array of resource-rich languages. We develop a probability-based confidence model to identify words with highly likely tag projections and use these words to train a multi-class SVM using the CCA features. Our method yields average performance of 85% accuracy for languages with almost no resources, outperforming a state-of-the-art partially-observed CRF model.

1 Introduction

We address the challenge of creating accurate and robust part-of-speech taggers for low-resource languages. We aim to apply our methods to the hundreds, and potentially thousands, of languages with meager electronic resources. We do not assume the existence of a tag dictionary, or any other sort of prior knowledge of the target language. Instead, we base our methods entirely on the existence of parallel data between the target language and a set of resource-rich languages.

Fortunately, such parallel data exists for just about every written language, in the form of Bible translations. Around 2,500 languages have at least partial Bible translations, and somewhere between 500 and 1,000 languages have complete translations. We have collected such electronic Bible translations for 650 languages. Figure 1 breaks down the number of languages in our collection according to their token count. The majority of our languages have at least 200,000 tokens of Bible translations.

While previous studies (Täckström et al., 2013; Ganchev and Das, 2013) have addressed this general setting, they have typically assumed the existence of a partial tag dictionary as well as large quantities of non-parallel data in the target language. These assumptions are quite reasonable for the dozen most popular languages in the world, but are inadequate for the creation of a truly world-wide repository of NLP tools and linguistic data.

In fact, we argue that such ancillary sources of information are not really necessary once we take into account the vastly multilingual nature of our parallel data. Annotations projected from individual resource-rich languages are often noisy and unreliable, due to systematic differences between the languages in question, as well as word alignment errors. We can thus think of these languages as very lazy and unreliable annotators of our target language. Despite their incompetence, as the number of such annotators increases, their combined efforts converge upon the truth, as idiosyncratic biases and random noise are washed away.

Our assumption throughout will be that we have in our possession a single multilingual corpus (the Bible) consisting of about 200,000 tokens for several hundred languages languages, as well as reasonably accurate POS taggers for about ten "resource-rich" languages. We will tag the Bible data for the resource-rich languages, word-align them to one another, and also word-align them to

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1292–1302,
Lisbon, Portugal, 17-21 September 2015.

the remaining several hundred target languages.

Of course, our goal is not to produce a tagger restricted to the Biblical lexicon. We therefore assume a small unannotated monolingual sample of the target language in an entirely unrelated genre (e.g. newswire). We use this sample transductively to adapt our learned taggers from the Biblical genre. In our experiments, we use the CoNLL 2006 and 2007 shared-task test data for this purpose. Of course tagged data does not exist for truly resource-poor languages, so we evaluate our methodology on the resource-rich languages. Each such language takes a turn playing the role of the target language for testing purposes.

The goal of the paper is to introduce a general "recipe" for successful cross-lingual induction of accurate taggers using meager resources. We faced three major technical challenges:

- First, word alignments across languages are incomplete, and often do not preserve part-of-speech due to language differences.
- Second, when using multiple resource-rich languages, we need to resolve conflicting projections.
- Third, the parallel data at our disposal is of an idiosyncratic genre (the Bible), and we wish to induce a general-purpose tagger.

To address these challenges, we forgo the typical sequence-based learning technique of HMM's and CRF's and instead adopt an instance-learning approach using latent distributional features. To induce these features, we introduced a new method using Canonical Correlation Analysis (CCA) to generalize the aligned information to new words. This method views each word position as consisting of three fundamental views: (1) the token view (word context), (2) the type view, and (3) the projected tags in the local vicinity. We perform a CCA to induce latent continuous vector representations of each view that maximizes their correlations to one another. On the test data, a simple multi-class classifier then suffices to predict accurate tags, even for novel words. This approach outperform a state-of-the-art baseline (Täckström et al., 2013) to achieve average tag accuracy of 85% on newswire text.

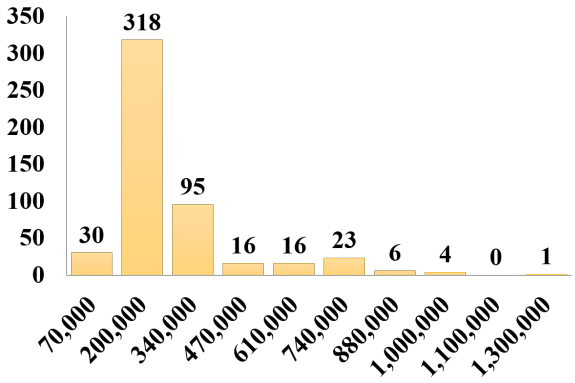


Figure 1: The breakdown of languages by the number of tokens in their available Bible translations. The horizontal axis gives the number of tokens, and the vertical axis gives the number of languages in each token range.

2 Related Work

2.1 Multilingual Projection

The idea of projecting annotated resources across languages using parallel data was first proposed by Yarowsky et al. (2001). This early work recognized the noisy nature of automatic word alignments and engineered smoothing and filtering methods to mitigate the effects of cross- lingual variation and alignment errors. More recent work in this vein has dealt with this by instead transferring information at the word type or model structure level, rather than on a token-by-token basis (Das and Petrov, 2011; Durrett et al., 2012). Current state-of-the-art results for indirectly supervised POS performance use a combination of token constraints as well as type constraints mined from Wiktionary (Li et al., 2012; Täckström et al., 2013; Ganchev and Das, 2013). As we argued above, the only widely available source of information for most low-resource languages is in fact their Bible translation. Perhaps surprisingly, our experiments show that this data source suffices to achieve state-of-the-art results.

Several previous authors have considered the advantage of using more than one resource-rich language to alleviate alignment noise.

Fossum and Abney (2005) found that using two source languages project-sources gave better results than simply using more data from one language. McDonald et al. (2011) also found advantages to using multiple language sources for projecting parsing constraints. In more of an unsu-

pervised context (but using small tag dictionaries), adding more languages to the mix has been shown to improve part-of-speech performance across all component languages (Naseem et al., 2009).

In our own previous multilingual work, we have developed the idea that supervised knowledge of some number of languages can help guide the unsupervised induction of linguistic structure, even in the absence of parallel text (Kim et al., 2011; Kim and Snyder, 2012; Kim and Snyder, 2013a; Kim and Snyder, 2013b). We have showed that cross-lingual supervised learning leads to significant performance gains over monolingual models. We point out that the previous tasks have considered as word-level structural analyses and our present case as a sentence-level analysis.

2.2 Word Alignment

Most of the papers surveyed above rely on automatic word alignments to guide the cross-lingual transfer of information. Given our desire to use highly multilingual information to improve projection accuracy, the question of word alignment performance becomes crucial. Our hypothesis is that multiple language projections are beneficial not only in weeding out random errors and idiosyncratic variations, but also in improving the linguistic consistency of the alignments themselves. Instead of simply aligning each source language to the target language in isolation, we will instead use a confidence model to synthesize information from multiple sources.

While there are not many well-known papers that have explored word alignment on a multilingual scale[1], there have been related efforts to *symmetrize* bilingual alignment models, using a variety of techniques ranging from modifications of EM (Liang et al., 2006), posterior-regularized objective function (Ganchev et al., 2010), and by considering relaxations of the hard combinatorial assignment problem (DeNero and Macherey, 2011).

2.3 Canonical Correlation Analysis (CCA)

Our method for generalizing the projections to unseen words and contexts is based on Canonical Correlation Analysis (CCA), a dimensionality reduction technique first introduced by Hotelling (1936). The key idea is to consider two groups of random variables with corresponding observations and to find linear subspaces with highest correlation between the two views. This can be seen as a kind of supervised version of Principal Components Analysis (PCA), where each view is providing supervision for the other. In fact, it can be shown that CCA directly generalizes both multiple linear regression and Fisher's Latent Discriminative Analysis (LDA) (Glahn, 1968).

From a learning theory perspective, CCA is interesting in that it allows us to prove regret-based learning bounds that depend on the "intrinsic" dimensionality of the problem rather than the apparent dimensionality (Kakade and Foster, 2007). This seems especially relevant to natural language processing scenarios, where the ambient dimension is extremely large and sparse, but reductions to dense lower-dimensional spaces may preserve nearly all the relevant semantic and syntactic information. In fact, CCA has recently been adapted to learning latent word representations in an interesting way: by dividing each word position into a token view (which only sees surrounding context) and a type view (which only sees the word itself) and performing a CCA between these two views (Dhillon et al., 2012; Kim et al., 2014; Stratos et al., 2014; Stratos et al., 2015; Kim et al., 2015c). CCA is also used to induce label representations (Kim et al., 2015d) and lexicon representations (Kim et al., 2015b).

Our technique will extend this idea by additionally considering a third *projected tag* view. Crucially, it is this view which pushes the latent representations into coherent part-of-speech categories, allowing us to simply apply multi-class SVM for unseen words in our test set.

3 Tag projection from resource-rich languages

In this section, we describe two methods for incorporating transferred tags from resource-rich languages: sequence-based learning (Täckström et al., 2013; Kim et al., 2015a) and instance-based learning. In the former, the transferred tags are used to train a partially-observed CRF (PO-CRF) by maximizing the probability of a constrained lattice. In contrast, instance-based learning views each word token as an independent classification task, but uses latent distributional information gleaned from surrounding words as features.

[1] Mayer and Cysouw (2012) used multilingual word alignment to compare languages

3.1 A sequence learning example of partially observed CRF (PO-CRF)

A first-order CRF parametrized by $\theta \in \mathbb{R}^d$ defines a conditional probability of a label sequence $y = y_1 \dots y_n$ given an observation sequence $x = x_1 \dots x_n$ as follows:

$$p_\theta(y|x) = \frac{\exp(\theta^\top \Phi(x, y))}{\sum_{y' \in \mathcal{Y}(x)} \exp(\theta^\top \Phi(x, y'))}$$

where $\mathcal{Y}(x)$ is the set of all possible label sequences for x and $\Phi(x, y) \in \mathbb{R}^d$ is a global feature function that decomposes into local feature functions $\Phi(x, y) = \sum_{j=1}^{n} \phi(x, j, y_{j-1}, y_j)$ by the first-order Markovian assumption. Given fully labeled sequences $\{(x^{(i)}, y^{(i)})\}_{i=1}^{N}$, the standard training method is to find θ that maximizes the log likelihood of the label sequences under the model with l_2-regularization:

$$\theta^* = \arg\max_{\theta \in \mathbb{R}^d} \sum_{i=1}^{N} \log p_\theta(y^{(i)}|x^{(i)}) - \frac{\lambda}{2} ||\theta||^2$$

We used an l_2 penalty weight λ of 1. Unfortunately, in our setting, we do not have fully labeled sequences. Instead, for each token x_j in sequence $x_1 \dots x_n$ we have the following two sources of label information:

- A set of allowed label types $\mathcal{Y}(x_j)$. (Label dictionary, type constraints)
- Labels $\tilde{y}_j$ transferred from resource rich languages. (transferred labels, token constraints)

Following previous work of Täckström et al. (2013), we first define a constrained *lattice* $\mathcal{Y}(x, \tilde{y}) = \mathcal{Y}(x_1, \tilde{y}_1) \times \dots \times \mathcal{Y}(x_n, \tilde{y}_n)$ where at each position j a set of allowed label types is given as:

$$\mathcal{Y}(x_j, \tilde{y}_j) = \begin{cases} \{\tilde{y}_j\} & \text{if } \tilde{y}_j \text{ is given} \\ \mathcal{Y}(x_j) & \text{otherwise} \end{cases}$$

And then we can define a conditional probability over label lattices for a given observation sequence x:

$$p_\theta(\mathcal{Y}(x, \tilde{y})|x) = \sum_{y \in \mathcal{Y}(x, \tilde{y})} p_\theta(y|x)$$

Given a label dictionary $\mathcal{Y}(x_j)$ for every token type x_j and training sequences $\{(x^{(i)}, \tilde{y}^{(i)})\}_{i=1}^{N}$ where $\tilde{y}^{(i)}$ is transferred labels for $x^{(i)}$ and, the new training method is to find θ that maximizes the log likelihood of the label lattices:

$$\theta^* = \arg\max_{\theta \in \mathbb{R}^d} \sum_{i=1}^{N} \log p_\theta(\mathcal{Y}(x^{(i)}, \tilde{y}^{(i)})|x^{(i)}) - \frac{\lambda}{2} ||\theta||^2$$

Since this objective is non-convex, we find a local optimum with a gradient-based algorithm. The gradient of this objective at each example $(x^{(i)}, \tilde{y}^{(i)})$ takes an intuitive form:

$$\begin{aligned} &\frac{\partial}{\partial \theta} \log p_\theta(\mathcal{Y}(x^{(i)}, \tilde{y}^{(i)})|x^{(i)}) - \frac{\lambda}{2} ||\theta||^2 \\ &= \sum_{y \in \mathcal{Y}(x^{(i)}, \tilde{y})} p_\theta(y|x^{(i)}) \Phi(x^{(i)}, y) \\ &- \sum_{y \in \mathcal{Y}(x^{(i)})} p_\theta(y|x^{(i)}) \Phi(x^{(i)}, y) - \lambda\theta \end{aligned}$$

This is the same as the standard CRF training except the first term where the gold features $\Phi(x^{(i)}, y^{(i)})$ are replaced by the expected value of features in the constrained lattice $\mathcal{Y}(x^{(i)}, \tilde{y})$.

An important distinction in our setting is that our token and type constraints are generated by only using the transferred tags whereas Täckström et al. (2013) generate type constraints induced from Wiktionary. Our setting is more realistic for at least two reasons; 1) Wiktionary is not always available. 2) transferable information is not limited, but Wiktionary is (e.g., semantic role and named entity).

3.2 Cross-lingual instance-based learning

The proposed method for cross-lingual instance-based learning has three steps:

1. Select training tokens based on the confidence of the projected tag information.
2. Induce distributional features over these words that incorporate all projected tags.
3. Train a multi-class classifier with these induced features to make local predictions for individual tokens.

We will describe each step below.

3.2.1 Selecting training words

Since transferred tags are not always reliable, all words in the parallel data are not necessary helpful in training. Since this method trains on words

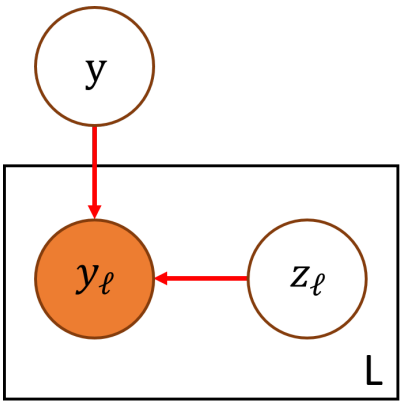


Figure 2: Graphical representation of the confidence model. Unobserved variable y denotes the true target-language tag for a token. Each of the L resource-rich languages displays a project of y, as y_ℓ, with an indicator variable z_ℓ determining the fidelity of the projection.

instead of sequences, it is easy to discard words which have unreliable or highly conflicting projections from different resource-rich languages.

To select our set of training tokens, we define a simple probability-based *confidence* model, illustrated in Figure 2. Suppose we have L resource-rich languages with alignments to the word in question. If the true tag is y, we assume that the projected tag for language ℓ will be identical to y with probability $1 - \epsilon_\ell$, where ϵ_ℓ is a language-specific corruption probability. With probability ϵ_ℓ, the projection will instead be chosen randomly (uniformly).

To make this explicit, we introduce a corruption indicator variable z_ℓ with:

$$P(z_\ell = 1) = \epsilon_\ell$$

Given z_ℓ, the probability of the projected tag y_ℓ is given by:

$$P(y_\ell|y, z_\ell) = \begin{cases} 1 & \text{if } z = 0 \text{ and } y = y_\ell, \\ \frac{1}{m} & \text{if } z = 1, \\ 0 & \text{otherwise.} \end{cases}$$

where m is the total number of possible tags. We can now compute a conditional distribution over the unknown tag y, marginalizing out the unknown corruption variables for each language:

$$p(y|y_1, \ldots, y_n) = \frac{\prod_{\ell=1}^{n} \left[\frac{\epsilon_\ell}{m} + (1 - \epsilon_\ell)\delta(y, y_\ell)\right]}{\frac{1}{m^{n-1}} \sum_{y' \in Y} \prod_{\ell=1}^{n} \left[\frac{\epsilon_\ell}{m} + (1 - \epsilon_\ell)\delta(y', y_\ell)\right]}$$

where Y is all possible tags. For simplicity, we simply set all ϵ_ℓ to 0.1 and use y as a training label when the conditional probability of the most likely value is greater than 0.9.

3.2.2 Inducing distributional features

In this section we discuss our approach for deriving latent distributional features. Canonical Correlation Analysis (CCA) is a general method for inducing new representations for a pair of variables X and Y (Hotelling, 1936). To derive word embeddings using CCA, a natural approach is to define X to represent a word and Y to represent the relevant information about a word, typically context words (Dhillon et al., 2012; Kim et al., 2015c). When they are defined as one-hot encodings, the CCA computation reduces to performing an SVD of the matrix Ω where each entry is

$$\Omega_{w,c} = \frac{\textbf{count}(w, c)}{\sqrt{\textbf{count}(w)\textbf{count}(c)}}$$

where $\textbf{count}(w, c)$ denotes co-occurrence count of word w and context c in the given corpus, $\textbf{count}(w) = \sum_c \textbf{count}(w, c)$, and $\textbf{count}(c) = \sum_w \textbf{count}(w, c)$.

The resulting word representation is given by $U^\top X$ where U is a matrix of the scaled left singular vectors of Ω (See Figure 3). In our work, we use a slightly modified version of this definition by taking square-root of each count:

$$\sqrt{\Omega}_{w,c} = \frac{\textbf{count}(w, c)^{1/2}}{\sqrt{\textbf{count}(w)^{1/2}\textbf{count}(c)^{1/2}}}$$

This has an effect of stabilizing the variance of each term in the matrix, leading to a more efficient estimator. The square-root transformation also transforms the distribution of the count data to look more Gaussian (Bartlett, 1936): since an interpretation of CCA is a latent-variable with normal distributions (Bach and Jordan, 2005), it makes the data more suitable for CCA. It has been observed in past works (e.g., Dhillon et al. (2012)) to significantly improve the quality of the resulting representations.

3.3 Feature Induction Algorithm

We now describe our algorithm for inducing latent distributional features both on the multilingual parallel corpus, as well as the monolingual, newswire test data. This algorithm is described

in detail in Figure 4. The key idea is to perform two CCA steps. The first step incorporates word-distributional information over both the multilingual corpus (the Bible) as well as the external domain monolingual corpus (CONLL data)[2]. This provides us with word representations that are general, and not overly specific to any single genre. However, it does not incorporate any projected tag information. We truncate this first SVD to the first 100 dimensions[3].

After this CCA step is performed, we then replace the words in the multilingual Bible data with their latent representations. We then perform a second CCA between these word representations and vectors representing the projected tags from all resource-rich languages. This step effectively *adapts* the first latent representation to the information contained in the tag projections. We truncate this second SVD to the first 50 dimensions.

We now have word embeddings that can be applied to any corpus, and are designed to maximize correlation both with typical surrounding word context, as well as typical projected tag context. These embeddings serve as our primary feature vectors for training the POS classifier (described in the next section). We concatenate this primary feature vector with the embeddings of the previous and subsequent words, in order to provide context-sensitive POS predictions.

3.3.1 Multi-class classifier

To train our POS tagger, we use a linear multi-class SVM (Crammer and Singer, 2002). It has a parameter $w_y \in \mathbb{R}^d$ for every tag $y \in \mathcal{T}$ and defines a linear score function $s(\mathbf{x}, j, y) := w_y^\top \Phi(\mathbf{x}, j)$. Given any sentence $\mathbf{x}$ and a position j, it predicts $\arg\max_{y \in \mathcal{T}} s(\mathbf{x}, j, y)$ as the tag of x_j. We use the implementation of Fan et al. (2008) with the default hyperparameter configurations for training.

4 Experiments

4.1 Datasets and Experimental Setup

There are more than 4,000 living languages in the world, and one of the most prevalently translated books is the Bible. We now describe the Bible dataset we collected.

[2]For context words, we use 5 words before and after the word occurrence.

[3]Embedding dimension was empirically determined by the singular values.

CCA-PROJ-SPARSE
Input: samples $(x^{(1)}, y^{(1)}) \ldots (x^{(n)}, y^{(n)}) \in \{0,1\}^d \times \{0,1\}^{d'}$, dimension k
Output: projections $A \in \mathbb{R}^{d \times k}$ and $B \in \mathbb{R}^{d' \times k}$

- Calculate $B \in \mathbb{R}^{d \times d'}$, $u \in \mathbb{R}^d$, and $v \in \mathbb{R}^{d'}$:

$$B_{i,j} = \sum_{l=1}^{n} [[x_i^{(l)} = 1]][[y_j^{(l)} = 1]]$$

$$u_i = \sum_{l=1}^{n} [[x_i^{(l)} = 1]] \quad v_i = \sum_{l=1}^{n} [[y_i^{(l)} = 1]]$$

- Define $\hat{\Omega} = \text{diag}(u)^{-1/2} B \text{diag}(v)^{-1/2}$.
- Calculate rank-k SVD $\hat{\Omega}$. Let $U \in \mathbb{R}^{d \times k}$ ($V \in \mathbb{R}^{d' \times k}$) be a matrix of the left (right) singular vector corresponding to the largest k singular values.
- Let $A = \text{diag}(u)^{-1/2} U$ and $B = \text{diag}(v)^{-1/2} V$.

Figure 3: Algorithm for deriving CCA projections from samples of two variables.

Lang	Tagger	Accuracy
BG	Treetagger	0.9909
CS	Hunpos	0.8969
DA	Hunpos	0.9756
DE	Hunpos	0.9855
EN	Hunpos	0.9854
ES	Treetagger	0.8785
IT	Treetagger	0.9059
NL	Treetagger	0.8781
PT	Hunpos	0.9770
AVG	-	0.9415

Table 1: Tagger accuracy on CoNLL data.

We first collect 893 bible volumes spanning several hundred languages that are freely available from three resources (www.bible.is, www.crosswire.org, www.biblegateway.com) and changed to UTF-8 format. The distribution of token in each bible in the unit of a language is in Figure 1.

Note that the Bible scripts are not exactly translated by sentences but by verses. We thus assume that each verse in a chapter has the same meaning if the number of verses is exactly same in a same chapter. We also assume that the whole chapters have the same meaning if the number of chapters in a book are exactly the same. In the same manner, we also assume the volumes that have the same number of chapters are the same. That is, their volume size should be as similar as possible

Input:

- N "labeled" tokens in the Bible domain: word $w^{(i)} \in \mathcal{V}$, corresponding context $C(w^{(i)}) \subset \mathcal{V}$ and (projected) tag set $P^{(i)} \subset \mathcal{T}$ for $i = 1 \dots N$
- N' tokens in data in the test domain: word $v^{(i)} \in \mathcal{V}'$ and corresponding context $C(v^{(i)}) \subset \mathcal{V}'$ for $i = 1 \dots N'$
- CCA dimensions k_1, k_2

Output: embedding $\mathbf{e}(w) \in \mathbb{R}^{k_2}$ for each word $w \in \mathcal{V} \cup \mathcal{V}'$

1. Combine the observed tokens and their context from the Bible and data in the test domain:

$$\mathcal{W}_1 := \left(w : w \in (w^{(i)})_{i=1}^{N} \cup (v^{(i)})_{i=1}^{N'}\right)$$
$$\mathcal{C}_1 := \left(C(w) : w \in (w^{(i)})_{i=1}^{N} \cup (v^{(i)})_{i=1}^{N'}\right)$$

2. Perform rank-k_1 **CCA-PROJ-SPARSE** on $(\mathcal{W}_1, \mathcal{C}_1)$ to derive a word projection matrix $\Phi_{\mathcal{W}_1}$ and a context projection matrix $\Phi_{\mathcal{C}_1}$.

3. Project all word examples in the Bible domain using $\Phi_{\mathcal{W}_1}$. Denote these projected words and the corresponding projected tag sets from all resource-rich languages by

$$\mathcal{W}_2 := \left(\Phi_{\mathcal{W}_1}(w^{(i)}) : i = 1 \dots N\right)$$
$$\mathcal{P}_2 := \left(P^{(i)} : i = 1 \dots N\right)$$

4. Perform rank-k_2 CCA on $(\mathcal{W}_2, \mathcal{P}_2)$ to derive a word projection matrix $\Phi_{\mathcal{W}_2}$ and a tag projection matrix $\Phi_{\mathcal{P}_2}$.

5. Set the embedding $\mathbf{e}(w)$ for each word $w \in \mathcal{V} \cup \mathcal{V}'$ as

$$\mathbf{e}(w) = \Phi_{\mathcal{W}_2}(\Phi_{\mathcal{W}_1}(w))$$

Figure 4: Algorithm for deriving word vectors for the (unannotated) test data that use the projected tags in the Bible data.

with the respect to the number of verses, chapters, and books.

Based upon these assumptions, we choose the best translation in a language based on a comparison to a reference Bible, the Modern King James Version (MKJV) in English. We choose the translation for each language that best matches this reference version in terms of chapter and verse numbering.

There are other factors considered if there are more than one candidates satisfying this matching. We focus on the contents of the bible such as the publication time. For instance, 1599 Geneva Bible in English contains old vocabulary with different spelling systems, causing unexpected errors when tagged by POS annotation tools. Also, some of volumes such as Amplified Bible (AMP) contains extraneous comments on verses themselves, causing errors for word alignments.

After the choice of the best volume, we finally select the 10 resource rich languages[4]. The two criteria to select resource rich languages are having i) the matched bible scripts both on the Old and New testament and ii) reliable parts-of-speech annotation tools. If these two requirements are satisfied, we can freely add more languages as resource rich languages in the future research. We use Hunpos tagger for CS, DA, DE, EN, and PT, Treetagger for BG, ES, IT, and NL, and Meltparser for FR.

4.2 Test Data

We use CoNLL parts-of-speech tagged data (selected resource-rich languages), plus Basque (EU), Hungarian (HU) and Turkish (TR)) as our test data. It consists of 5,000-6,000 hand-labeled tokens. The accuracy of each supervised tagger on this data is about 94% on average. Since there is no French tagged CoNLL data, we exclude French on testing but still use it in Training. The accuracy of each supervised tagger on this data is shown in Table 1.

The tag definitions used in CoNLL data are not exactly matched the ones used in the taggers when converted to universal POS tags. For instance in Spanish, we initially follow mapping of Petrov et al. (2011) for CoNLL data. The 'dp' tag for words *sus, su, mi* are mapped to DET but they are mapped to PRON in the bible data because of the Treetagger definitions. Whenever we find this kind of issues, we analyze them and choose the one of mappings for compatibility. For the 'dp' tag, we choose to map PRON.

4.3 Alignments

We perform two kinds of alignments in our data sets; (i) the verse alignment and (ii) the word alignment. When the tagged bible volumes are prepared, we align verses across all resource rich languages. For verse alignments, we pre-process to remove extraneous information such as inline reference (e.g. [REV 4:16]) and HTML tags. These alignments between two languages

[4]Bulgarian (BG), Czech(CS), Danish (DA), English(EN), German (DE), French (FR), Spanish (ES), Italian (IT), Dutch (NL), and Portuguese (PT)

occurred only when volumes have the exact same number of chapters and verses. For instance, Mark must have 16 chapters and the first chapter of the Mark must have 45 verses in our criteria. The correct number of chapters and verses are pre-defined on MKJV volume, and the number of matched verses on each volume is greater than 30,500.

After performing verse alignments, we then perform word alignments. The quality of tags in resource poor languages is highly dependent on the quality of word alignments because parts-of-speech tags will be projected through this alignment path. First, we use GIZA++ for initial one-to-many alignments and we symmetrize by taking their intersection. This ensures that the resulting alignments are of high quality.

4.4 Results

	majority	union	confident
BG	0.8123	0.8167	0.8235
CS	0.8013	0.8094	0.8142
DA	0.8412	0.8497	0.8492
DE	0.8532	0.8611	0.8721
ES	0.8278	0.8345	0.8385
EU	0.8326	0.8413	0.8472
HU	0.7741	0.7789	0.7953
IT	0.8486	0.8445	0.8481
NL	0.7864	0.7876	0.7884
PT	0.8022	0.8081	0.8110
TR	0.6803	0.6739	0.6935
AVG	0.8055	0.8097	0.8165

Table 2: Baseline model CONLL performance depending on criterion for selecting tag projection.

In all experiments, we hold out the tags of the test language. EU, HU and TR used projected tags from 10 resource-rich languages, 9 resource-rich languages are used for the remaining languages. In our first experiment, we consider the state-of-the-art PO-CRF baseline. This model trains a partially observed CRF based on a single projected tag for each token. We experiment with different methods of choosing the projected tags. The results are shown in Table 2. The majority method is to choose the most common tag from the projected tags of the current token. We then experiment with taking the union of all projected tags (i.e. only constraining the lattice based on unanimity of the resource-rich languages). Finally, we considered choosing the high confidence tags, based on our confidence model. The confident tags are defined by a method described in Section 3.2.1 If this ratio is greater than 0.9, we assume that this token has high confidence. As the results indicate, this final method yielded the best tagging performance on the CONLL test data, achieving average accuracy of 82%.

In the remaining experiments we will adopt the confidence-based selection criterion for both the baseline as well as our method.

	PO-CRF	CCA+SVM
BG	0.8450	0.8686
CS	0.8359	0.8442
DA	0.8727	0.8826
DE	0.8862	0.9025
ES	0.8523	0.8816
EU	0.8506	0.8927
HU	0.8461	0.8495
IT	0.8705	0.8911
NL	0.8115	0.8345
PT	0.8346	0.8410
TR	0.7064	0.7389
AVG	0.8375	0.8570

Table 3: Performance on multilingual Bible data

In order to isolate the errors due to projection mismatch versus domain variation, we first test both models on the Bible data itself. To do so, we assume that the tags produced by the test-language's supervised tagger are in fact the ground truth. This experiment allows us to compare to tag projection models using (1) PO-CRF and (2) CCA+SVM. Results are given in Table 3. Unsurprisingly, PO-CRF performs better on the multilingual corpus than on the CONLL data, due to the beneficial constraint of the projected tags. Perhaps interestingly, the CCA+SVM method, which is a simple instance-based classifier using cleverly constructed features, outperforms the sequence labeller, achieving accuracy of nearly 86%[5] .

In our third experiment we use CoNLL test data and compare the PO-CRF models with different settings. See Table 4. This experiment is to show the effects of suffix and Brown cluster features on PO-CRF to relieve the unseen words issue. We also show that the more projecting languages are

[5]Note that some previous researches (Liang et al., 2008; Wisniewski et al., 2014; Moore, 2014) also pointed out that POS tagging does not necessarily require a sequence model for strong performance.

	1 lang (EN) (A)	9/10 langs (W)	9/10 langs (no S/C)	9/10 langs (A)
BG	0.7883	0.7144	0.8094	0.8478
CS	0.6601	0.5589	0.6535	0.7868
DA	0.7820	0.7765	0.8016	0.8227
DE	0.8323	0.6956	0.7589	0.8500
ES	0.7893	0.7608	0.8279	0.8665
EU	0.7764	0.7543	0.8035	0.8661
HU	0.6429	0.6378	0.7834	0.8119
IT	0.8444	0.7588	0.8136	0.8921
NL	0.7887	0.6825	0.7751	0.8214
PT	0.8476	0.7797	0.8464	0.8656
TR	0.6306	0.5727	0.6719	0.7143
AVG	0.7621	0.6993	0.7768	0.8314

Table 4: Accuracy of the PO-CRF models on CoNLL data. A, W, no S/C means: all, word, all but no suffix and cluster features are used, respectively.

included the better the results gets.

For the features, we used word identity, suffixes of up to length 3, Brown cluster and three indicators of (1) capitalization for the first character, (2) containing a hyphen or (3) a digit. Especially, Brown clusters was induced from more than 2 million line documents, making the setting unrealistic for resource-poor language.

With just the word features, the averaged performance is 0.6993 and other indicator features increase the performance to 0.7768. Also note that the suffix and Brown cluster features increase the performance from 0.7768 to 0.8314. As reported, PO-CRF mitigates the adverse effects of the unseen word issues and almost meets the performance in the previous experiment (0.8375) of Täckström et al. (2013) by using these features.

In fourth and final experiment, we used the same features for PO-CRF, with Brown clusters induced on a realistically obtainable sized (3k) corpus for a low resource language. We compare directly to our CCA+SVM model (which does not use Brown clustering features at all). We achieved 0.7983 on PO-CRF with all features and our corresponding model on CCA achieved about 0.8474, shown in Table 5. As reported, our model outperforms the PO-CRF with the realistic settings for resource poor languages.

	PO-CRF 3k Brown	CCA+SVM
BG	0.8318	0.8815
CS	0.7635	0.8232
DA	0.7335	0.8911
DE	0.8296	0.8543
ES	0.8319	0.8713
EU	0.8376	0.8734
HU	0.7817	0.8372
IT	0.8451	0.8474
NL	0.7626	0.8245
PT	0.8768	0.8823
TR	0.6874	0.7354
AVG	0.7983	0.8474

Table 5: Performances on our test data, CoNLL document.

5 Conclusions

We addressed the challenge of POS tagging low-resource languages. Our key idea is to use a massively multilingual corpus. Instead of relying on a single resource-rich language, we leverage the full array of currently available POS taggers. This removes alignment-mismatch noise and identifies a subset of words with highly confident tags. We then use a CCA procedure to induce latent feature representations across domains, incorporating word contexts as well as projected tags. We then train an SVM to predict tags.

Experimentally, we show that this procedure yields accuracy of about 85% for languages with nearly no resources available, beating a state-of-the-art partially observed CRF formulation. In the near future, this technique will enable us to release a suite of POS taggers for hundreds of low-resource languages.

References

Francis R Bach and Michael I Jordan. 2005. A probabilistic interpretation of canonical correlation analysis.

MSo Bartlett. 1936. The square root transformation in analysis of variance. *Supplement to the Journal of the Royal Statistical Society*, pages 68–78.

Koby Crammer and Yoram Singer. 2002. On the learnability and design of output codes for multiclass problems. *Machine Learning*, 47(2-3):201–233.

Dipanjan Das and Slav Petrov. 2011. Unsupervised part-of-speech tagging with bilingual graph-based projections. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 600–609. Association for Computational Linguistics.

John DeNero and Klaus Macherey. 2011. Model-based aligner combination using dual decomposition. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 420–429. Association for Computational Linguistics.

Paramveer S. Dhillon, Jordan Rodu, Dean P. Foster, and Lyle H. Ungar. 2012. Two Step CCA: A new spectral method for estimating vector models of words. In *Proceedings of the 29th International Conference on Machine learning*, ICML'12.

Greg Durrett, Adam Pauls, and Dan Klein. 2012. Syntactic transfer using a bilingual lexicon. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1–11. Association for Computational Linguistics.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. Liblinear: A library for large linear classification. *The Journal of Machine Learning Research*, 9:1871–1874.

Victoria Fossum and Steven Abney. 2005. Automatically inducing a part-of-speech tagger by projecting from multiple source languages across aligned corpora. In *Natural Language Processing–IJCNLP 2005*, pages 862–873. Springer.

Kuzman Ganchev and Dipanjan Das. 2013. Cross-lingual discriminative learning of sequence models with posterior regularization. In *Proceedings of EMNLP*. Association for Computational Linguistics, October.

Kuzman Ganchev, Joao Graça, Jennifer Gillenwater, and Ben Taskar. 2010. Posterior regularization for structured latent variable models. *The Journal of Machine Learning Research*, 11:2001–2049.

Harry R Glahn. 1968. Canonical correlation and its relationship to discriminant analysis and multiple regression. *Journal of the atmospheric sciences*, 25(1):23–31.

Harold Hotelling. 1936. Relations between two sets of variates. *Biometrika*, 28(3-4):321–377.

Sham M Kakade and Dean P Foster. 2007. Multi-view regression via canonical correlation analysis. In *Learning Theory*, pages 82–96. Springer.

Young-Bum Kim and Benjamin Snyder. 2012. Universal grapheme-to-phoneme prediction over latin alphabets. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 332–343. Association for Computational Linguistics.

Young-Bum Kim and Benjamin Snyder. 2013a. Optimal data set selection: An application to grapheme-to-phoneme conversion. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 1196–1205. Association for Computational Linguistics.

Young-Bum Kim and Benjamin Snyder. 2013b. Unsupervised consonant-vowel prediction over hundreds of languages. In *Proceedings of the Association for Computational Linguistics (ACL)*, pages 1527–1536. Association for Computational Linguistics.

Young-Bum Kim, João V Graça, and Benjamin Snyder. 2011. Universal morphological analysis using structured nearest neighbor prediction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 322–332. Association for Computational Linguistics.

Young-Bum Kim, Heemoon Chae, Benjamin Snyder, and Yu-Seop Kim. 2014. Training a korean srl system with rich morphological features. In *Proceedings of the Association for Computational Linguistics (ACL)*, pages 637–642. Association for Computational Linguistics.

Young-Bum Kim, Minwoo Jeong, Karl Stratos, and Ruhi Sarikaya. 2015a. Weakly supervised slot tagging with partially labeled sequences from web search click logs. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 84–92. Association for Computational Linguistics.

Young-Bum Kim, Karl Stratos, Xiaohu Liu, and Ruhi Sarikaya. 2015b. Compact lexicon selection with spectral methods. In *Proceedings of Association for Computational Linguistics (ACL)*, pages 806–811. Association for Computational Linguistics.

Young-Bum Kim, Karl Stratos, and Ruhi Sarikaya. 2015c. Pre-training of hidden-unit crfs. In *Proceedings of the Association for Computational Linguistics (ACL)*, pages 192–198. Association for Computational Linguistics.

Young-Bum Kim, Karl Stratos, Ruhi Sarikaya, and Minwoo Jeong. 2015d. New transfer learning techniques for disparate label sets. In *Proceedings of the Association for Computational Linguistics (ACL)*, pages 473–482. Association for Computational Linguistics.

Shen Li, Joao V Graça, and Ben Taskar. 2012. Wiki-ly supervised part-of-speech tagging. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1389–1398. Proceedings of Association for Computational Linguistics.

Percy Liang, Ben Taskar, and Dan Klein. 2006. Alignment by agreement. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 104–111. Association for Computational Linguistics.

Percy Liang, Hal Daumé III, and Dan Klein. 2008. Structure compilation: trading structure for features. In *Proceedings of the 25th international conference on Machine learning*, pages 592–599. ACM.

Thomas Mayer and Michael Cysouw. 2012. Language comparison through sparse multilingual word alignment. In *Proceedings of the EACL 2012 Joint Workshop of LINGVIS & UNCLH*, pages 54–62. Association for Computational Linguistics.

Ryan McDonald, Slav Petrov, and Keith Hall. 2011. Multi-source transfer of delexicalized dependency parsers. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 62–72. Association for Computational Linguistics.

Robert C Moore. 2014. Fast high-accuracy part-of-speech tagging by independent classifiers. In *Proceedings of COLING*, pages 1165–1176.

Tahira Naseem, Benjamin Snyder, Jacob Eisenstein, and Regina Barzilay. 2009. Multilingual part-of-speech tagging: Two unsupervised approaches. *Journal of Artificial Intelligence Research*, 36(1):341–385.

Slav Petrov, Dipanjan Das, and Ryan McDonald. 2011. A universal part-of-speech tagset. *arXiv preprint arXiv:1104.2086*.

Karl Stratos, Do-kyum Kim, Michael Collins, and Daniel Hsu. 2014. A spectral algorithm for learning class-based n-gram models of natural language. *Proceedings of the Association for Uncertainty in Artificial Intelligence*.

Karl Stratos, Michael Collins, and Daniel Hsu. 2015. Model-based word embeddings from decompositions of count matrices. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 1282–1291. Association for Computational Linguistics, July.

Oscar Täckström, Dipanjan Das, Slav Petrov, Ryan McDonald, and Joakim Nivre. 2013. Token and type constraints for cross-lingual part-of-speech tagging. *Transactions of the Association for Computational Linguistics*, 1:1–12.

Guillaume Wisniewski, Nicolas Pécheux, Souhir Gahbiche-Braham, and François Yvon. 2014. Cross-lingual part-of-speech tagging through ambiguous learning. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1779–1785.

David Yarowsky, Grace Ngai, and Richard Wicentowski. 2001. Inducing multilingual text analysis tools via robust projection across aligned corpora. In *Proceedings of the first international conference on Human language technology research*, pages 1–8. Association for Computational Linguistics.

An Improved Tag Dictionary for Faster Part-of-Speech Tagging

Robert C. Moore
Google Inc.
bobmoore@google.com

Abstract

Ratnaparkhi (1996) introduced a method of inferring a tag dictionary from annotated data to speed up part-of-speech tagging by limiting the set of possible tags for each word. While Ratnaparkhi's tag dictionary makes tagging faster but less accurate, an alternative tag dictionary that we recently proposed (Moore, 2014) makes tagging as fast as with Ratnaparkhi's tag dictionary, but with no decrease in accuracy. In this paper, we show that a very simple semi-supervised variant of Ratnaparkhi's method results in a much tighter tag dictionary than either Ratnaparkhi's or our previous method, with accuracy as high as with our previous tag dictionary but much faster tagging—more than 100,000 tokens per second in Perl.

1 Overview

In this paper, we present a new method of constructing tag dictionaries for part-of-speech (POS) tagging. A tag dictionary is simply a list of words[1] along with a set of possible tags for each word listed, plus one additional set of possible tags for all words not listed. Tag dictionaries are commonly used to speed up POS-tag inference by restricting the tags considered for a particular word to those specified by the dictionary.

Early work on POS tagging generally relied heavily on manually constructed tag dictionaries, sometimes agumented with tag statistics derived from an annotated corpus (Leech et al., 1983; Church, 1988; Cutting et al., 1992). Merialdo (1994) relied only on a tag dictionary extracted from annotated data, but he used the annotated tags from his test data as well as his training data to construct his tag dictionary, so his evaluation was not really fair.[2] Ratnaparkhi (1996) seems to have been the first to use a tag dictionary automatically extracted only from training data.

Ratnaparkhi's method of constructing a tag dictionary substantially speeds up tagging compared to considering every possible tag for every word, but it noticeably degrades accuracy when used with a current state-of-the-art tagging model. We recently presented (Moore, 2014) a new method of constructing a tag dictionary that produces a tagging speed-up comparable to Ratnaparkhi's, but with no decrease in tagging accuracy. In this paper, we show that a very simple semi-supervised variant of Ratnaparkhi's method results in a much tighter tag dictionary than either Ratnaparkhi's or our previous method, with accuracy as high as we previously obtained, while allowing much faster tagging—more than 100,000 tokens per second even in a Perl implementation.

1.1 Tag Dictionaries and Tagging Speed

A typical modern POS tagger applies a statistical model to compute a score for a sequence of tags $t_1, \ldots, t_n$ given a sequence of words $w_1, \ldots, w_n$. The tag sequence assigned the highest score by the model for a given word sequence is selected as the tagging for the word sequence. If $\mathcal{T}$ is the set of possible tags, and there are no restrictions on the form of the model, then the time to find the highest scoring tag sequence is potentially $O(n|\mathcal{T}|^n)$ or worse, which would be intractable.

To make tagging practical, models are normally defined to be factorable in a way that reduces the time complexity to $O(n|\mathcal{T}|^k)$, for some small integer k. For models in which all tagging decisions are independent, or for higher-order mod-

[1] According to the conventions of the field, POS tags are assigned to all tokens in a tokenized text, including punctuation marks and other non-word tokens. In this paper, all of these will be covered by the term *word*.

[2] Merialdo (1994, p. 161) acknowledged this: "In some sense this is an optimal dictionary for this data, since a word will not have all its possible tags (in the language), but only the tags it actually had within the text."

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1303–1308,
Lisbon, Portugal, 17-21 September 2015.

els pruned by fixed-width beam search, $k = 1$, so the time to find the highest scoring tag sequence is $O(n|\mathcal{T}|)$. But this linear dependence on the size of the tag set means that reducing the average number of tags considered per token should further speed up tagging, whatever the underlying model or tagger may be.

1.2 Ratnaparkhi's Method

For each word observed in an annotated training set, Ratnaparkhi's tag dictionary includes all tags observed with that word in the training set, with all possible tags allowed for all other words. Ratnaparkhi reported that using this tag dictionary improved per-tag accuracy from 96.31% to 96.43% on his Penn Treebank (Marcus et al., 1993) Wall Street Journal (WSJ) development set, compared to considering all tags for all words.

With a more accurate model, however, we found (Moore, 2014) that while Ratnaparkhi's tag dictionary decreased the average number of tags per token from 45 to 3.7 on the current standard WSJ development set, it also decreased per-tag accuracy from 97.31% to 97.19%. This loss of accuracy can be explained by the fact that 0.5% of the development set tokens are known words with a tag not seen in the training set, for which our model achieved 44.5% accuracy with all word/tag pairs permitted. With Ratnaparkhi's dictionary, accuracy for these tokens is necessarily 0%.

1.3 Our Previous Method

We previously presented (Moore, 2014) a tag dictionary constructed by using the annotated training set to compute a smoothed probability estimate for any possible tag given any possible word, and for each word in the training set, including in the dictionary the tags having an estimated probability greater than a fixed threshold T. In this approach, the probability $p(t|w)$ of tag t given word w is computed by interpolating a discounted relative frequency estimate of $p(t|w)$ with an estimate of $p(t)$ based on "diversity counts", taking the count of a tag t to be the number of distinct words ever observed with that tag. The distribution $p(t)$ is also used to estimate tag probabilities for unknown words, so the set of possible tags for any word not explicitly listed is $\{t|p(t) > T\}$.

If we think of w followed by t as a word bigram, this is exactly like a bigram language model estimated by the interpolated Kneser-Ney (KN) method described by Chen and Goodman (1999). The way tag diversity counts are used has the desirable property that closed-class tags receive a very low estimated probability of being assigned to a rare or unknown word, even though they occur very often with a small number of frequent words. A single value for discounting the count of all observed word/tag pairs is set to maximize the estimated probability of the reference tagging of the development set. When T was chosen to be the highest threshold that preserves our model's 97.31% per tag WSJ development set accuracy, we obtained an average of 3.5 tags per token.

1.4 Our New Approach

We now present a new method that reduces the average number of tags per token to about 1.5, with no loss of tagging accuracy. We apply a simple variant of Ratnaparkhi's method, with a training set more than 4,000 times larger than the Penn Treebank WSJ training set. Since no such hand-annotated corpus exists, we create the training set automatically by running a version of our tagger on the LDC English Gigaword corpus. We thus describe our approach as a semi-supervised variant of Ratnaparkhi's method. Our method can be viewed as an instance of the well-known technique of self-training (e.g., McClosky et al., 2006), but ours is the first use of self-training we know of for learning inference-time search-space pruning.

We introduce two additional modifications of Ratnaparki's approach. First, with such a large training corpus, we find it unnecessary to keep in the dictionary every tag observed with every word in the automatically-annotated data. So, we estimate a probability distribution over tags for each word in the dictionary according to unsmoothed relative tag frequencies, and include for each word in the dictionary only tags whose probability given the word is greater than a fixed threshold.

Second, since our tokenized version of the English Gigaword corpus contains more than 6 million unique words, we reduce the vocabulary of the dictionary to the approximately 1 million words having 10 or more occurrences in the corpus. We treat all other tokens as instances of unknown words, and we use their combined unsmoothed relative tag frequencies to estimate a tag probability distribution for unknown words. We use the same threshold on this distribution as we do for words explicitly listed in the dictionary, to obtain a set of possible tags for unknown words.

2 Experimental Details

In our experiments, we use the WSJ corpus from Penn Treebank-3, split into the standard training (sections 0–18), development (sections 19–21), and test (sections 22-24) sets for POS tagging.

The tagging model we use has the property that all digits are treated as indistinguishable for all features. We therefore also make all digits indistinguishable in constructing tag dictionaries (by internally replacing all digits by "9"), since it does not seem sensible to give two different dictionary entries based on containing different digits, when the tagging model assigns them the same features.

2.1 The Tagging Model

The model structure, feature set, and learning method we use for POS tagging are essentially the same as those in our earlier work, treating POS tagging as a single-token independent multiclass classification task. Word-class-sequence features obtained by supervised clustering of the annotated training set replace the hidden tag-sequence features frequently used for POS tagging, and additional word-class features obtained by unsupervised clustering of a very large unannotated corpus provide information about words not occurring in the training set. For full details of the feature set, see our previous paper (Moore, 2014).

The model is trained by optimizing the multiclass SVM hinge loss objective (Crammer and Singer, 2001), using stochastic subgradient descent as described by Zhang (2004), with early stopping and averaging. The only difference from our previous training procedure is that we now use a tag dictionary to speed up training, while we previously used tag dictionaries only at test time.

Our training procedure makes multiple passes through the training data considering each training example in turn, comparing the current model score of the correct tag for the example to that of the highest scoring incorrect tag and updating the model if the score of the correct tag does not exceed the score of the highest scoring incorrect tag by a specified margin. In our new version of this procedure, we use the KN-smoothed tag dictionary described in Section 1.3. to speed up finding the highest scoring incorrect tag.

Recall that the KN-smoothed tag dictionary estimates a non-zero probability $p(t|w)$ for every possible word/tag pair, and that the possible tags for a given word are determinted by setting a threshold T on this probability. In each pass through the training set, we use the same probability distribution $p(t|w)$ determined from the statistics of the annotated training data, but we employ an adaptive method to determine what threshold T to use in each pass.

For the first pass through the training set, we set an initial threshold T_0 to the highest value such that for every token in the development set, $p(t|w) \geq T_0$, where t is the correct tag for the token and w is the word for the token. At the end of each training pass i, while evaluating the current model on the development set for early stopping using threshold T_{i-1}, we also find the highest probability threshold T_i such that choosing a lower threshold would not enable any additional correct taggings on the development set using the current model. This threshold will normally be higher than T_0, because we disregard tokens in the development set for which the correct tag would not be selected by the model resulting from the previous pass at any threshold. T_i is then used as the threshold for training pass $i + 1$. Whenever the selected threshold leaves only one tag remaining for a particular training example, we skip that example in training.

On the first pass through the training set, use of this method resulted in consideration of an average of 31.36 tags per token, compared to 45 total possible tags. On the second and all subsequent passes, an average of 10.48 tags were considered per token. This sped up training by a factor of 3.7 compared to considering all tags for all tokens, with no loss of tagging accuracy when a development-set-optimized KN-smoothed tag dictionary is also used at test time.

2.2 Tagging the Gigaword Corpus

To construct our new tag dictionary, we need an automatically-tagged corpus several orders of magnitude larger than the hand-tagged WSJ training set. To obtain this corpus we ran a POS tagger on the LDC English Gigaword Fifth Edition[3] corpus, which consists of more than 4 billion words of English text from seven newswire sources. We first removed all SGML mark-up, and performed sentence-breaking and tokenization using the Stanford CoreNLP toolkit (Manning et al, 2014). This produced 4,471,025,373 tokens of

[3] https://catalog.ldc.upenn.edu/LDC2011T07

Tag Dictionary	Accuracy	Tags/Token	Unambig	Tokens/Sec
Pruned KN-smoothed	97.31%	3.48	45.3%	69k
Unpruned semi-supervised	97.31%	1.97	51.7%	82k
Pruned semi-supervised	97.31%	1.51	66.8%	103k

Table 1: WSJ development set token accuracy and tagging speed for different tag dictionaries

6,616,812 unique words. We tagged this corpus using the model described in Section 2.1 and a KN-smoothed tag dictionary as described in Section 1.3, with a threshold $T = 0.0005$. The tagger we used is based on the fastest of the methods described in our previous work (Moore, 2014, Section 3.1). Tagging took about 26 hours using a single-threaded implementation in Perl on a Linux workstation equipped with Intel Xeon X5550 2.67 GHz processors.

2.3 Extracting the Tag Dictionary

We extracted a Ratnaparkhi-like tag dictionary for the 957,819 words with 10 or more occurrences in our corpus. Tokens of all other words in the corpus were treated as unknown word tokens and used to define a set of 24 tags[4] to be used for words not explicitly listed in the dictionary. To allow pruning the dictionary as described in Section 1.4, for each word (including the unknown word), we computed a probability distribution $p(t|w)$ using unsmoothed relative frequencies. As noted above, we treated all digits as indistinguishable in constructing and applying the dictionary.

3 Experimental Results

Tagging the WSJ development set with an unpruned semi-supervised tag dictionary obtained from the automatic tagging of the English Gigaword corpus produced the same tagging accuracy as allowing all tags for all tokens or using the pruned KN-smoothed tag dictionary used in tagging the Gigaword corpus. Additional experiments showed that we could prune this dictionary with a threshold on $p(t|w)$ as high as $T = 0.0024$ without decreasing development set accuracy. In addition to applying this threshold to the tag probabilities for all listed words, we also applied it to the tag probabilities for unknown words, leaving 13 possible tags[5] for those.

Tagging the WSJ development set with these two dictionaries is compared in Table 1 to tagging with our previous pruned KN-smoothed dictionary. The second column shows the accuracy per tag, which is 97.31% for all three dictionaries. The third column shows the mean number of tags per token allowed by each dictionary. The fourth column shows the percentage of tokens with only one tag allowed, which is significant since the tagger need not apply the model for such tokens—it can simply output the single possible tag.

The last column shows the tagging speed in tokens per second for each of the three tag dictionaries, using the fast tagging method we previously described (Moore, 2014), in a single-threaded implementation in Perl on a Linux workstation equipped with Intel Xeon X5550 2.67 GHz processors. Speed is rounded to the nearest 1,000 tokens per second, because we measured times to a precision of only about one part in one hundred. For the pruned KN-smoothed dictionary, we previously reported a speed of 49,000 tokens per second under similar conditions. Our current faster speed of 69,000 tokens per second is due to an improved low-level implementation for computing the model scores for permitted tags, and a slightly faster version of Perl (v5.18.2).

The most restrictive tag dictionary, the pruned semi-supervised dictionary, allows only 1.51 tags per token, and our implementation runs at 103,000 tokens per second on the WSJ development set. For our final experiments, we tested our tagger with this dictionary on the standard Penn Treebank WSJ test set and on the Penn Treebank-3 parsed Brown corpus subset, as an out-of-domain evaluation. For comparison, we tested our previous tagger and the fast version (english-left3words-distsim) of the Stanford tagger (Toutanova et al., 2003; Manning, 2011) recommended for practical use on the Stanford tagger website, which we found to be by far the fastest of the six publicly available taggers tested in our previous work (Moore, 2014). The results of these tests are shown in Table 2.[6]

[4] CC, CD, DT, FW, IN, JJ, JJR, JJS, MD, NN, NNP, NNPS, NNS, PRP, RB, RBR, RP, UH, VB, VBD, VBG, VBN, VBP, and VBZ

[5] CD, FW, JJ, NN, NNP, NNPS, NNS, RB, VB, VBD, VBG, VBN, and VBZ

[6] In Table 2, "OOV" has the standard meaning of a token

Tagger	WSJ Tokens/Sec	All WSJ Accuracy	OOV WSJ Accuracy	Brown Tokens/Sec	All Brown Accuracy	OOV Brown Accuracy
This work	102k	97.36%	91.09%	96k	96.55%	89.25%
Our previous	51k/54k/69k	97.34%	90.98%	40k/43k/56k	96.54%	89.36%
Stanford fast	80k	96.87%	89.69%	50k	95.53%	87.38%

Table 2: WSJ test set and Brown corpus tagging speeds and token accuracies

For our previous tagger, we give three speeds: the speed we reported earlier, a speed for a duplicate of the earlier experiment using the faster version of Perl that we use here, and a third measurement including both the faster version of Perl and our improved low-level tagger implementation.

With the pruned semi-supervised dictionary, our new tagger has slightly higher all-token accuracy than our previous tagger on both the WSJ test set and Brown corpus set, and it is much more accurate than the fast Stanford tagger. The accuracy on the standard WSJ test set is 97.36%, one of the highest ever reported. The new tagger is also much faster than either of the other taggers, achieving a speed of more than 100,000 tokens per second on the WSJ test set, and almost 100,000 tokens per second on the out-of-domain Brown corpus data.

4 Conclusions

Our method of constructing a tag dictionary is technically very simple, but remarkably effective. It reduces the mean number of possible tags per token by 57% and increases the number of unambiguous tokens by by 47%, compared to the previous state of the art (Moore, 2014) for a tag dictionary that does not degrade tagging accuracy. When combined with our previous work on fast high-accuracy POS tagging, this tag dictionary produces by far the fastest POS tagger reported with anything close to comparable accuracy.

References

Stanley F. Chen and Joshua T. Goodman. 1999. An empirical study of smoothing techniques for language modeling. *Computer Speech and Language*, 13(4):359–393.

Kenneth Ward Church. 1988. A stochastic parts program and noun phrase parser for unrestricted text. In *Proceedings of the Second Conference on Applied Natural Language Processing of the Association for Computational Linguistics*, February 9–12, Austin, Texas, USA, 136–143.

Koby Crammer and Yoram Singer. 2001. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, 2:265–292.

Doug Cutting, Julian Kupiec, Jan Pedersen, and Penelope Siburn. 1992. A practical part-of-speech tagger. In *Proceedings of the Third Conference on Applied Natural Language Processing of the Association for Computational Linguistics*, March 31–April 3, Trento, Italy, 133–140.

Geoffrey Leech, Roger Garside, and Eric Atwell. 1983. The automatic tagging of the LOB corpus. *ICAME Journal: International Computer Archive of Modern and Medieval English Journal*, 7:13–33.

Christopher D. Manning. 2011. Part-of-speech tagging from 97% to 100%: Is it time for some linguistics? In Alexander Gelbukh (ed.), *Computational Linguistics and Intelligent Text Processing, 12th International Conference, CICLing 2011, Proceedings, Part I. Lecture Notes in Computer Science 6608*, Springer, 171–189.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, June 23–24, Baltimore, Maryland, USA, 55–60.

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

David McClosky, Eugene Charniak, and Mark Johnson. 2006. Effective Self-Training for

of a word not occurring in the annotated training set. Many of these tokens do match words in our large semi-supervised tag dictionary, however.

Parsing. In *Proceedings of the 2006 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, June 4–9, New York, New York, USA, 152-159.

Bernard Merialdo. 1994. Tagging English text with a probabalistic model. *Computational Linguistics*, 20(2):155–171.

Robert C. Moore. 2014. Fast high-accuracy part-of-speech tagging by independent classifiers. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, August 23–29, Dublin, Ireland, 1165–1176.

Adwait Ratnaparkhi. 1996. A maximum entropy model for part-of-speech tagging. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, May 17–18, Philadelphia, Pennsylvania, USA, 133–142.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, May 27–June 1, Edmonton, Alberta, Canada, 173–180.

Tong Zhang. 2004. Solving large scale linear prediction problems using stochastic gradient descent algorithms. In *Proceedings of the 21st International Conference on Machine Learning*, July 4–8, Banff, Alberta, Canada, 919–926.

Improving Arabic Diacritization through Syntactic Analysis

Anas Shahrour, Salam Khalifa and Nizar Habash
Computational Approaches to Modeling Language Lab
New York University Abu Dhabi
United Arab Emirates
{anas.shahrour,salamkhalifa,nizar.habash}@nyu.edu

Abstract

We present an approach to Arabic automatic diacritization that integrates syntactic analysis with morphological tagging through improving the prediction of case and state features. Our best system increases the accuracy of word diacritization by 2.5% absolute on all words, and 5.2% absolute on nominals over a state-of-the-art baseline. Similar increases are shown on the full morphological analysis choice.

1 Introduction

Modern Standard Arabic (MSA) orthography generally omits diacritical marks which encode lexical as well as syntactic (case) information. The task of Arabic automatic diacritization is about the automatic restoration of the missing diacritics. Diacritization improvement in Arabic has important implications for downstream processing for Arabic natural language processing, e.g. speech recognition (Ananthakrishnan et al., 2005; Biadsy et al., 2009), speech synthesis (Elshafei et al., 2002), and machine translation (Diab et al., 2007; Zbib et al., 2010).

Previous efforts on diacritization utilized morphological tagging techniques to disambiguate word forms. Habash et al. (2007a) observe that while such techniques work relatively well on lexical diacritics (located on word stems), they are much worse for syntactic case diacritics (typically word final). They suggest that syntactic analysis may help with automatic diacrtization, but stop short of testing the idea, and instead demonstrate that complex linguistic features and rules are needed to model complex Arabic case using gold syntactic analyses. In this paper, we develop an approach for improving the quality of automatic Arabic diacritization through the use of automatic syntactic analysis. Our approach combines handwritten rules for case assignment and agreement with machine learning of case and state adjustment on the output of a state-of-the-art morphological tagger. Our best system increases the accuracy of word diacrtization by 2.5% absolute overall, and 5.2% absolute on nominals over a state-of-the-art baseline.

2 Linguistic Background

Arabic automatic processing, and specifically diacritization is hard for a number of reasons.

First, Arabic words are morphologically rich. The morphological analyzer we use represents Arabic words with 15 features (Pasha et al., 2014).[1] We focus on case and state in this paper. In our data set, *case* has five values: nominative (n), accusative (a), genitive (g), undefined (u) and not applicable (na). Cases n, a and g are usually expressed with an overt morpheme. Case u is used to mark words without an overt morpheme expression of case (e.g., invariable nouns such as شَكوَى *šakwaý* 'complaint'), or those not assigned a case in the manual annotations. Most of the missing assignments are for foreign proper nouns, which often do not receive case markers. However, this is not done consistently in the training data we use. Case na is used for non-nominals. *State* is a nominal feature that has four values: definite (d), indefinite (i), construct (c) and not applicable (na). State generally reflects the definiteness in nominals (d vs i) and whether a nominal is the head of genitive construction (aka *Idafa*) (c). State na is used for non-nominals.[2] For the most part, case and state realize as a single word-final morpheme, e.g., the suffix اً **Aã** in كِتَابًا *kitAbAã*[3] is a morpheme indicating the word is (cas:a, stt:i).

Second, undiacritized Arabic words are highly ambiguous: in our training data, words had an average of 12.8 analyses per word, most of which are associated with different diacritizations. Some diacritization differences reflect different analysis

[1] Lemma (lex), Part-of-Speech (pos), Gender (gen), Number (num), Case (cas), State (stt), Person (per), Aspect (asp), Voice (vox), Mood (mod), four proclitics (prc$n$), and one enclitic (enc0).

[2] For a detailed discussion of Arabic case and state, see (Smrž, 2007; Habash et al., 2007a).

[3] Arabic transliteration is presented in the Habash-Soudi-Buckwalter scheme (Habash et al., 2007b).

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1309–1315,
Lisbon, Portugal, 17-21 September 2015.

lemmas; while others are due to morpho-syntactic differences. For example, the undiacritized version of the word we used above (كتابا *ktAbA*) has two other diacritizations and analyses: كُتَّابًا *kut∼AbAã* (cas:a stt:i) 'writers' (different lemma) and كِتَابَا *kitAbA* (cas:n stt:c num:d) 'two books of [...]' (different features).

Third, Arabic has complex case/state assignment and agreement patterns that interact with the sentence's syntax. For example, a noun may get its case by being subject of a verb and its state by being the head of an idafa construction; while adjectives modifying this noun agree with it in its case, their state is determined by the state of the last element in the idafa chain the noun heads.

For more information on Arabic orthography, morphology, and syntax, see Habash (2010).

3 Related Work

Much work has been done on Arabic diacritization (Vergyri and Kirchhoff, 2004; Nelken and Shieber, 2005; Zitouni et al., 2006; Habash and Rambow, 2007; Alghamdi and Muzafar, 2007; Rashwan et al., 2009; Bebah et al., 2014; Hadj Ameur et al., 2015; Abandah et al., 2015; Bouamor et al., 2015). We refer the reader to the extensive literature review by Abandah et al. (2015), and focus on systems we compare with.

Most of the previous approaches cited above utilize different sequence modeling techniques that use varying degrees of knowledge from shallow letter and word forms to deeper morphological information; none to our knowledge make use of syntax. Habash and Rambow (2007) approach diacritization as a part of the morphological disambiguation problem, where they select the optimal full morphological tag for Arabic in context and use it to select from a list of possible analyses produced by a morphological analyzer. They use independent taggers for all features; and language models for lemmas and diacritized surface forms. Their work is part of the state-of-the-art Arabic morphological tagger MADAMIRA (Pasha et al., 2014). Our paper is most closely related to Habash and Rambow (2007) and Pasha et al. (2014). We extend their work using additional syntactic features to improve morphological disambiguation accuracy. We demonstrate improvements in terms of both full morphological analysis choice (lemmatization, tokenization, all features) as well as word diacritization.

Most recently, Abandah et al. (2015) presented a recurrent neural network approach to diacritize full sentences with impressive results. We do not compare to their effort here but we note that they use an order of magnitude more diacritized data than we do, and they focus on diacritization only as opposed to full morphological analysis.

In related work on modeling Arabic case and syntax, Habash et al. (2007a) compared rule-based and machine learning approaches to capture the complexity of Arabic case assignment and agreement. They demonstrated their results on gold syntactic analyses showing that given good syntactic representations, case prediction can be done at a high degree of accuracy. Alkuhlani et al. (2013) later extended this work to cover all morphological features, including state. Additionally, Marton et al. (2013) demonstrated that in the context of syntactic dependency parsing, case is the best feature to use *in gold settings* and is the worst feature to use *in predicted settings*. In this paper we use automatic (i.e. not gold) syntactic features to improve case prediction, which improves morphological analysis and word diacrtization.

4 Approach

Motivation We are motivated by an error analysis we conducted of 1,200 words of the MADAMIRA system output. We found a large number of surprising *syntactically impossible* case errors such as genitive nouns following verbs or construct nouns followed by non-genitives. We explain these errors by MADAMIRA's contextual models being limited to a small window of neighboring words and with no modeling of syntax, which leads to a much worse performance on case and diacritization compared to lemmas and POS (almost 10% absolute drop from 96% to 86%).

Proposed Solution Our approach is to provide better prediction of case and state using models with access to additional information, in particular syntactic analysis and rules. The predicted case and state values are then used to re-tag the MADAMIRA output by selecting the best match in its ranked morphological analyses. We limit our retagging to nominals. Since what we are learning to predict is how to correct MADAMIRA's baseline choice (as opposed to a generative model of case-state), we also re-apply the model on its output to fix primarily propagated agreement errors in a manner similar to Habash et al. (2007a)'s agreement classifier.[4]

[4]We optimized for a re-application limit, which we found invariably to be +1 time or +2 times in our *DevTest* experi-

5 Experimental Results

We present next our experimental results and compare five case-state prediction techniques. The results for these techniques are compared to our state-of-the-art baseline system, which has been compared to a number of other approaches.

5.1 Experimental Setup

Data We used the Penn Arabic Treebank (PATB, parts 1, 2 and 3) (Maamouri et al., 2004; Maamouri et al., 2006; Maamouri et al., 2009) split into *Train*, *Dev* and (blind) *Test* along the recommendations of Diab et al. (2013) which were also used in the baseline system. The morphological feature representations we use are derived from the PATB analyses following the approach used in the MADA and later MADAMIRA systems (Habash and Rambow, 2005; Habash and Rambow, 2007; Pasha et al., 2014). We further divide *Dev* into two parts with equal number of sentences: *DevTrain* (30K words) for training our case-state classifiers, and *DevTest* (33K words) for development testing. The *Test* set has 63K words.[5]

Evaluation Metrics We report our accuracy in terms of two metrics: a. **Diac**, the percentage of correctly fully diacrtized words; and b. **All**, the percentage of words for which a full morphological analysis (lemma, POS, all inflectional and clitic features, and diacritization) is correctly predicted. We report the results on all words (**All Words**) as well as on nominals[6] with no u case in the gold (henceforth, **Nominals**). We do not report on case and state prediction accuracy, nor on the character-level diacritization.

Morphological Analysis, Baseline and Topline For our baseline, we use the morphological analysis and disambiguation tool MADAMIRA (Pasha et al., 2014), which produces a contextually ranked list of analyses for each word. We computed an oracular topline using the PATB gold case and state values in the retagging process.

ments. We do not report more on this due to space limitations. Because of the need to have the same of number of parse tree tokens for reapplication in the models using syntax, we constrain the retagging to maintain the same clitic signature (number of clitics) in the MADAMIRA baseline top analysis.

[5]To address the concern that we are using more "training" data than our baseline, we compared the performance of MADAMIRA's release (baseline) to a version that was trained on Train + *DevTrain*. The small increase in training data made no significant difference from the baseline system in terms of our metrics.

[6]Nominals consists of nouns (including noun_quant and noun_num), adjectives, proper nouns, adverbs, and pronouns.

	All Words		**Nominals**	
System	**Diac**	**All**	**Diac**	**All**
Oracle Topline	96.2	94.2	98.0	95.7
Baseline	87.4	85.0	81.9	78.2
Morphology Rules	88.0	85.5	83.1	79.4
Morphology Classifier	88.4	85.9	83.7	80.1
Syntax Rules	87.4	84.6	**86.4**	82.2
Syntax Classifier	89.1	86.6	85.2	81.4
Syntax Rules+Classifier	**89.7**	**87.1**	**86.4**	**82.5**

Table 1: Results on *DevTest*.

Syntactic Analysis For syntactic features, we trained an Arabic dependency parser using MaltParser (Nivre et al., 2007) on the Columbia Arabic Treebank (CATiB) version of the PATB (Habash and Roth, 2009). The *Train* data followed the same splits mentioned above. The Nivre "eager" algorithm was used in all experiments. The CATiB dependency tree has six simple POS tags and eight relations (Habash and Roth, 2009; Habash et al., 2009). The PATB tokenization as well as the CATiB POS tags were produced by the baseline system MADAMIRA and used as input to the parser. We also used the well performing yet simple CATiBex expansion of the CATiB tags as implemented in the publicly available parsing pipeline from Marton et al. (2013). Our parser's performance on the PATB *Dev* set is comparable to Marton et al. (2013): 84.2% labeled attachment, 86.6% unlabeled attachment, and 93.6% label accuracy.

Machine Learning Technique Given the small size of *DevTrain*, we opted to train unlexicalized models that we expect to capture morpho-syntactic abstractions. We tried a number of machine learning techniques and settled on using the J48 Decision tree classifier with its default settings in WEKA (Hall et al., 2009; Quinlan, 1993) for all of the classification experiments in this paper.

5.2 Case and State Classification Techniques

We detail and report on five case-state classification techniques we experimented with. All results on the *DevTest* are presented in Table 1.

Morphology Rules We created a simple manual word-morphology based classifier that handled the most salient case errors seen in our pilot study (Section 4) and whose correction has a high precision. The scope of the rules was limited to word bigrams and included three conditions: (i) post-

verbal genitive nouns are changed to the first non-genitive analysis available from MADAMIRA; (ii) post-construct state non-genitive nouns should become genitive; and (iii) adjectives agreeing with the nouns they follow in gender, number, and definiteness, but not in case should match the nouns' case. The collective improvement of all the rules applied in the order presented above adds up to 0.6% absolute on All Word Diac, and 1.2% absolute on Nominal Diac.

Morphology-based Classifier We trained a classifier to predict a correction of the baseline case and state of a word using the *DevTrain* data set. For features, we included all the non-lexical morphological features (all features mentioned in *footnote 1* except for lemma). Clitic feature values were binarized to indicate if a clitic is present or not. We used *Nil* values for all out-of-vocabulary words in MADAMIRA. BOS and EOS placeholders were used for sentence boundaries. We excluded all *DevTrain* words whose predicted POS switches from nominal to non-nominal or vice versa, but kept them as part of other words' context. This minimizes noise and sparsity in the training data, especially given its small size and the rarity of such examples. We experimented with adding features from neighboring words within a window size of +/- 2 words. The best performing setup was with window size +/- 1. This classifier makes around 0.5% absolute gain over the simple word morphology rules.

Syntax Rules Inspired by Habash et al. (2007a)'s simple set of rules for determining case on gold dependency trees, we re-implemented these rules to work with our different dependency representation and extended them to include state assignment in a manner similar to Alkuhlani et al. (2013).[7] These rules improve over the baseline by 4.5% absolute in Nominal Diac accuracy, but produce no gains in All Words setting. An investigation of the error patterns reveals the main reason to be the rules' inability to predict the problematic u case.

Syntax-based Classifier We trained a classifier to predict a correction of the case and state of a word on a parsed version of the *DevTrain* data set. Since the syntactic parse separates most clitics in the PATB tokenization, we align the tree tokens with the word morphology before extracting classifier features. The features we used are the token's morphological features (same as those used for words in the morphology-based classifier), the parent and grandparent's features, the relations between token and parent, and between parent and grandparent, and the features of the neighboring +/- n tokens. The tokenized clitics are only used as context features (tree and surface). We tested all combinations of values of n as 0, 1, 2, and 3 and of including parent and grandparent features. The best performing setting was with including the parent and grandparent as well as a window of +/- 1 tokens. The syntax-based classifier improves over the morphology-based classifier but it trails behind syntax rules in terms of Nominal Diac. Its All Word Diac accuracy is the highest so far.

Combination of Syntax Rules and Classifier We combine the last two approaches to exploit their complementary performance by including the syntax rule predictions as features in the syntax-based classifier. The resulting system is our best performer achieving *DevTest* accuracy improvements of 2.3% absolute (All Word Diac), and 4.5% absolute (Nominal Diac).

5.3 Blind Test Results

The results of applying the best approach and settings on our blind *Test* set are presented in Table 2. While the baseline on *Test* is slightly higher than *DevTest*, the performance on all metrics are comparable. The increase in Diac accuracy on All Words is 2.5% absolute and on Nominals is 5.2% absolute. The corresponding relative reductions in error to the oracle toplines are 30% and 34%. Similar increases are shown on the full morphological analysis choice.

	All Words		**Nominals**	
System	**Diac**	**All**	**Diac**	**All**
Oracle Topline	96.3	94.4	97.9	95.6
Baseline	88.1	86.0	82.4	79.4
Syntax Rules+Classifier	**90.6**	**88.5**	**87.6**	**84.5**

Table 2: Results on the blind *Test* set.

[7] For nominal case assignment, our rules are: (i) default case is a; (ii) children of $root$ are n; (iii) object of preposition and idafa children are assigned g; (iv) predicates not headed by verbs are assigned n; and (v) subjects and topics not headed (or grand-headed) by a class of words called *Ân∼a and her sisters* are assigned n. After case assignment, we apply two case agreement rules: (i) for all nominals modifying case-assigned nominals (i.e., with tree relation mod), copy the case of the modified nominal; and (ii) for all nominals conjoined to case-assigned nominals (i.e., with tree chain $nominal\text{-}conjunction\text{-}nominal$), copy the case of the heading nominal. For state, we assign d to words with the definite article proclitic, and c to words heading an idafa construction. All other nominals are assigned state i.

(a) تعليقي الأساسي أن الأمن في لبنان جَيِّدٌ tçlyqy AlÂsAsy Ân AlÂmn fy lbnAn jay∼idũ

My main comment is that security in Lebanon is good

Word	Dependency	Gold	Baseline	Best System
جيد jyd	PRD of Ân 'that'	jay∼idũ [NOM]	jay∼idī [GEN]	jay∼idũ [NOM]

(b) كان جان غانم يَدَهُ اليُمنَى kAn jAn γAnm ydh Alymný

Jean Ghanem was his right hand

Word	Dependency	Gold	Baseline	Best System
يده ydh	PRD of kAn 'was'	yadahu [ACC]	yadihi [GEN]	yadahu [ACC]
اليمنى Alymný	MOD of yd 'hand'	Alyum.naý [U]	Alyamaniy∼i [GEN]	Alyum.naý [U]

(c) سيعزز العَلَاقَاتِ الِاقتِصَادِيَّةَ مع أوروبا syçzz AlçlAqAt AlAqtSAdyħ mç Âwrwbā

It will reinforce the economic relations with Europe

Word	Dependency	Gold	Baseline	Best System
العلاقات AlçlAqAt	OBJ of yçzz 'reinforce'	AlçalAqAti [ACC]	AlçalAqAti [GEN]	AlçalAqAti [ACC]
الاقتصاديّة AlAqtSAdyħ	MOD of AlçlAqAt 'relations'	AlAiqtiSAdiy∼aħa [ACC]	AlAiqtiSAdiy∼aħi [GEN]	AlAiqtiSAdiy∼aħa [ACC]

(d) أكد وزير الاقتصاد باسل فليحان إثر لقائه أمس رَئِيسَ الوزراء رفيق الحريري أن اتفاق ...

Âkd wzyr AlAqtSAd bAsl flyHAn Ăθr lqAŷh Âms rŷys AlwzrA' rfyq AlHryry Ân AtfAq ...

Following his meeting with the Prime Minister Rafiq Alhariri,
the Minister of Economy Basil Flaihan has confirmed that the agreement ...

Word	Dependency	Gold	Baseline	Best System
رئيس rŷys	SBJ of Âkd 'confirmed'	raŷiysa [ACC]	raŷiysa [ACC]	raŷiysu [NOM]

Figure 1: Examples of corrections from our best performing system.

5.4 Error Analysis and Examples

We manually investigated the types of errors in the first 100 errors in the *DevTest* in our best system's output. In about a quarter of the cases (23%), all of which proper nouns, the gold reference was not fully diacritized, making it impossible to evaluate. In an additional 7%, typographical errors in the input including missing sentence breaks led to bad parses which are the likely cause of error. The rest of the errors are system failures: 31% are connected with syntactic tree errors (although a third of these are due to agreement-propagated errors over correct trees); 28% are due to other morphological analysis issues (half are out-of-vocabulary and 4% are no-analysis cases); and 11% are other case selection errors unrelated to the above-mentioned issues.

Figure 1 shows four examples from the *DevTest* where the analysis of our best system for the underlined words is different from baseline. Examples (a), (b), and (c) are cases where our best system's analysis matched the gold. The dependency relation also matched the gold and was the likely cause of correction. In example (d), our best system incorrectly changed the correct baseline analysis in agreement with the wrong dependency relation provided by the parser, which is the likely cause of error.

6 Conclusion and Future Work

We have demonstrated the value of using automatic syntactic analysis as part of the task of automatic diacritization and morphological tagging of Arabic. Our best solution is a hybrid approach that combines statistical parsing and manual syntactic rules as part of a machine learning model for correcting case and state features.

In the future, we plan to investigate the development of joint morphological disambiguation and syntactic parsing models. We will also work on improving the quality of Arabic parsing which is behind many of the errors according to our error analysis. Other possible directions include using more sophisticated machine learning techniques and richer lexical features. We also plan to host a demo and make our system available through the website of the Computational Approaches to Modeling Language (CAMeL) Lab: `www.camel-lab.com`.

Acknowledgments

We would like to thank Arfath Pasha for his helpful support overall, and especially for retraining the special version of MADAMIRA we discuss in *footnote 5*.

References

Gheith A Abandah, Alex Graves, Balkees Al-Shagoor, Alaa Arabiyat, Fuad Jamour, and Majid Al-Taee. 2015. Automatic diacritization of Arabic text using recurrent neural networks. *International Journal on Document Analysis and Recognition (IJDAR)*, 18(2):183–197.

Mansour Alghamdi and Zeeshan Muzafar. 2007. KACST Arabic diacritizer. In *First International Symposium on Computers and Arabic Language*, pages 25–28.

Sarah Alkuhlani, Nizar Habash, and Ryan Roth. 2013. Automatic morphological enrichment of a morphologically underspecified treebank. In *HLT-NAACL*, pages 460–470.

S. Ananthakrishnan, S. Narayanan, and S. Bangalore. 2005. Automatic diacritization of Arabic transcripts for ASR. In *Proceedings of ICON*, Kanpur, India, December.

Mohamed Bebah, Chennoufi Amine, Mazroui Azzeddine, and Lakhouaja Abdelhak. 2014. Hybrid approaches for automatic vowelization of Arabic texts. *arXiv preprint arXiv:1410.2646*.

Fadi Biadsy, Nizar Habash, and Julia Hirschberg. 2009. Improving the Arabic Pronunciation Dictionary for Phone and Word Recognition with Linguistically-Based Pronunciation Rules. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 397–405, Boulder, Colorado.

Houda Bouamor, Wajdi Zaghouani, Mona Diab, Ossama Obeid, Kemal Oflazer, Mahmoud Ghoneim, and Abdelati Hawwari. 2015. A pilot study on Arabic multi-genre corpus diacritization. In *Proceedings of the Second Workshop on Arabic Natural Language Processing*, pages 80–88, Beijing, China, July. Association for Computational Linguistics.

Mona Diab, Mahmoud Ghoneim, and Nizar Habash. 2007. Arabic Diacritization in the Context of Statistical Machine Translation. In *Proceedings of Machine Translation Summit (MT-Summit)*, Copenhagen, Denmark.

Mona Diab, Nizar Habash, Owen Rambow, and Ryan Roth. 2013. LDC Arabic treebanks and associated corpora: Data divisions manual. *arXiv preprint arXiv:1309.5652*.

Moustafa Elshafei, Husni Al-Muhtaseb, and Mansour Al-Ghamdi. 2002. Techniques for high quality Arabic speech synthesis. *Information sciences*, 140(3):255–267.

Nizar Habash and Owen Rambow. 2005. Arabic Tokenization, Part-of-Speech Tagging and Morphological Disambiguation in One Fell Swoop. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 573–580, Ann Arbor, Michigan.

Nizar Habash and Owen Rambow. 2007. Arabic Diacritization through Full Morphological Tagging. In *Proceedings of the 8th Meeting of the North American Chapter of the Association for Computational Linguistics/Human Language Technologies Conference (HLT-NAACL07)*.

Nizar Habash and Ryan M Roth. 2009. CATiB: The Columbia Arabic Treebank. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 221–224. Association for Computational Linguistics.

Nizar Habash, Ryan Gabbard, Owen Rambow, Seth Kulick, and Mitchell P Marcus. 2007a. Determining case in Arabic: Learning complex linguistic behavior requires complex linguistic features. In *EMNLP-CoNLL*, pages 1084–1092.

Nizar Habash, Abdelhadi Soudi, and Tim Buckwalter. 2007b. On Arabic Transliteration. In A. van den Bosch and A. Soudi, editors, *Arabic Computational Morphology: Knowledge-based and Empirical Methods*. Springer.

Nizar Habash, Reem Faraj, and Ryan Roth. 2009. Syntactic Annotation in the Columbia Arabic Treebank. In *Proceedings of MEDAR International Conference on Arabic Language Resources and Tools*, Cairo, Egypt.

Nizar Habash. 2010. *Introduction to Arabic Natural Language Processing*. Morgan & Claypool Publishers.

Mohamed Seghir Hadj Ameur, Youcef Moulahoum, and Ahmed Guessoum. 2015. Restoration of Arabic diacritics using a multilevel statistical model. In *Computer Science and Its Applications*, volume 456 of *IFIP Advances in Information and Communication Technology*, pages 181–192. Springer International Publishing.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. 2009. The WEKA data mining software: an update. *ACM SIGKDD explorations newsletter*, 11(1):10–18.

Mohamed Maamouri, Ann Bies, Tim Buckwalter, and Wigdan Mekki. 2004. The Penn Arabic Treebank: Building a Large-Scale Annotated Arabic Corpus. In *NEMLAR Conference on Arabic Language Resources and Tools*, pages 102–109, Cairo, Egypt.

Mohamed Maamouri, Ann Bies, and Seth Kulick. 2006. Diacritization: A challenge to Arabic treebank annotation and parsing. In *Proceedings of the Conference of the Machine Translation SIG of the British Computer Society*.

Mohamed Maamouri, Ann Bies, and Seth Kulick. 2009. Creating a Methodology for Large-Scale Correction of Treebank Annotation: The Case of the Arabic Treebank. In *Proceedings of MEDAR International Conference on Arabic Language Resources and Tools*, Cairo, Egypt.

Yuval Marton, Nizar Habash, and Owen Rambow. 2013. Dependency parsing of modern standard Arabic with lexical and inflectional features. *Computational Linguistics*, 39(1):161–194.

Rani Nelken and Stuart Shieber. 2005. Arabic Diacritization Using Weighted Finite-State Transducers. In *Proceedings of the Workshop on Computational Approaches to Semitic Languages at 43rd Meeting of the Association for Computational Linguistics (ACL'05)*, pages 79–86, Ann Arbor, Michigan.

Joakim Nivre, Johan Hall, Jens Nilsson, Atanas Chanev, Gülsen Eryigit, Sandra Kübler, Svetoslav Marinov, and Erwin Marsi. 2007. MaltParser: a language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(02):95–135.

Arfath Pasha, Mohamed Al-Badrashiny, Ahmed El Kholy, Ramy Eskander, Mona Diab, Nizar Habash, Manoj Pooleery, Owen Rambow, and Ryan Roth. 2014. MADAMIRA: A Fast, Comprehensive Tool for Morphological Analysis and Disambiguation of Arabic. In *In Proceedings of LREC*, Reykjavik, Iceland.

Ross Quinlan. 1993. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers, San Mateo, CA.

Mohsen Rashwan, Mohammad Al-Badrashiny, Mohamed Attia, and S Abdou. 2009. A hybrid system for automatic Arabic diacritization. In *The 2nd International Conference on Arabic Language Resources and Tools*. Citeseer.

Otakar Smrž. 2007. *Functional Arabic Morphology. Formal System and Implementation*. Ph.D. thesis, Charles University in Prague, Prague, Czech Republic.

Dimitra Vergyri and Katrin Kirchhoff. 2004. Automatic Diacritization of Arabic for Acoustic Modeling in Speech Recognition. In Ali Farghaly and Karine Megerdoomian, editors, *COLING 2004 Workshop on Computational Approaches to Arabic Script-based Languages*, pages 66–73, Geneva, Switzerland.

Rabih Zbib, Spyros Matsoukas, Richard Schwartz, and John Makhoul. 2010. Decision trees for lexical smoothing in statistical machine translation. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, pages 428–437, Uppsala, Sweden, July. Association for Computational Linguistics.

Imed Zitouni, Jeffrey S. Sorensen, and Ruhi Sarikaya. 2006. Maximum entropy based restoration of Arabic diacritics. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 577–584, Sydney, Australia, July. Association for Computational Linguistics.

Combining Discrete and Continuous Features for Deterministic Transition-based Dependency Parsing

Meishan Zhang and **Yue Zhang**
Singapore University of Technology and Design
{meishan_zhang, yue_zhang}@sutd.edu.sg

Abstract

We investigate a combination of a traditional linear sparse feature model and a multi-layer neural network model for deterministic transition-based dependency parsing, by integrating the sparse features into the neural model. Correlations are drawn between the hybrid model and previous work on integrating word embedding features into a discrete linear model. By analyzing the results of various parsers on web-domain parsing, we show that the integrated model is a better way to combine traditional and embedding features compared with previous methods.

1 Introduction

Transition-based parsing algorithms construct output syntax trees using a sequence of shift-reduce actions. They are attractive in computational efficiency, allowing linear time decoding with deterministic (Nivre, 2008) or beam-search (Zhang and Clark, 2008) algorithms. Using rich non-local features, transition-based parsers achieve state-of-the-art accuracies for dependency parsing (Zhang and Nivre, 2011; Zhang and Nivre, 2012; Bohnet and Nivre, 2012; Choi and McCallum, 2013; Zhang et al., 2014).

Deterministic transition-based parsers works by making a sequence of greedy local decisions (Nivre et al., 2004; Honnibal et al., 2013; Goldberg et al., 2014; Gómez-Rodríguez and Fernández-González, 2015). They are attractive by very fast speeds. Traditionally, a linear model has been used for the local action classifier. Recently, Chen and Manning (2014) use a neural network (*NN*) to replace linear models, and report improved accuracies.

A contrast between a neural network model and a linear model is shown in Figure 1 (a) and (b).

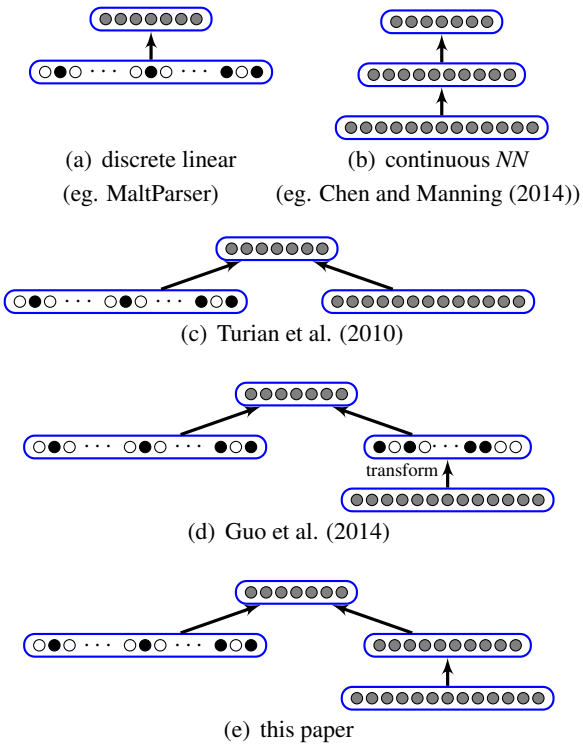


Figure 1: Five deterministic transition-based parsers with discrete and continuous features.

A neural network model takes continuous vector representations of words as inputs, which can be pre-trained using large amounts of unlabeled data, thus containing more information. In addition, using an extra hidden layer, a neural network is capable of learning non-linear relations between automatic features, achieving feature combinations automatically.

Discrete manual features and continuous features complement each other. A natural question that arises from the contrast is whether traditional discrete features and continuous neural features can be integrated for better accuracies. We study this problem by constructing the neural network shown in Figure 1 (e), which incorporates the discrete input layer of the linear model (Figure 1 (a)) into the *NN* model (Figure 1 (b)) by conjoining

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1316–1321,
Lisbon, Portugal, 17-21 September 2015.

it with the hidden layer. This architecture is connected with previous work on incorporating word embeddings into a linear model.

In particular, Turian et al. (2010) incorporate word embeddings as real-valued features into a CRF model. The architecture is shown in Figure 1(c), which can be regarded as Figure 1(e) without the hidden layer. Guo et al. (2014) find that the accuracies of Turian et al can be enhanced by discretizing the embedding features before combining them with the traditional features. They use simple binarization and clustering to this end, finding that the latter works better. The architecture is shown in Figure 1(d). In contrast, Figure 1(e) directly combines discrete and continuous features, replacing the hard-coded transformation function of Guo et al. (2014) with a hidden layer, which can be tuned by supervised training.[1]

We correlate and compare all the five systems in Figure 1 empirically, using the SANCL 2012 data (Petrov and McDonald, 2012) and the standard Penn Treebank data. Results show that the method of this paper gives higher accuracies than the other methods. In addition, the method of Guo et al. (2014) gives slightly better accuracies compared to the method of Turian et al. (2010) for parsing task, consistent with Guo et al's observation on named entity recognition (NER). We make our C++ code publicly available under GPL at `https://github.com/SUTDNLP/NNTransitionParser`.

2 Parser

We take Chen and Manning (2014), which uses the arc-standard transition system (Nivre, 2008). Given an POS-tagged input sentence, it builds a projective output y by performing a sequence of state transition actions using greedy search. Chen and Manning (2014) can be viewed as a neutral alternative of MaltParser (Nivre, 2008).

Although not giving state-of-the-art accuracies, deterministic parsing is attractive for its high parsing speed (1000+ sentences per second). Our incorporation of discrete features does not harm the overall speed significantly. In addition, deterministic parsers use standard neural classifiers, which allows isolated study of feature influences.

[1]Yet another alternative structure is to directly combine the two types of inputs, and replacing the input layer of (b) using them. Wang and Manning (2013) compared this architecture with (c) using a CRF network, finding that the latter works better for NER and chunking.

3 Models

Following Chen and Manning (2014), training of all the models using a cross-entropy loss objective with a L2-regularization, and mini-batched AdaGrad (Duchi et al., 2011). We unify below the five deterministic parsing models in Figure 1.

3.1 Baseline linear (*L*)

We build a baseline linear model using logistic regression (Figure 1(a)). Given a parsing state x, a vector of discrete features $\Phi_d(x)$ is extracted according to the *arc-standard* feature templates of Ma et al. (2014a), which is based on the *arc-eager* templates of Zhang and Nivre (2011). The score of an action a is defined by

$$\text{Score}(a) = \sigma\big(\Phi_d(x) \cdot \overrightarrow{\theta}_{d,a}\big),$$

where σ represents the sigmoid activation function, $\overrightarrow{\theta}_d$ is the set of model parameters, denoting the feature weights with respect to actions, a can be SHIFT, LEFT(l) and RIGHT(l).

3.2 Baseline Neural (*NN*)

We take the Neural model of Chen and Manning (2014) as another baseline (Figure 1(b)). Given a parsing state x, the words are first mapped into continuous vectors by using a set of pre-trained word embeddings. Denote the mapping as $\Phi_e(x)$. In addition, denote the hidden layer as Φ_h, and the ith node in the hidden as $\Phi_{h,i}$ ($0 \leq i \leq |\Phi_h|$). The hidden layer is defined as

$$\Phi_{h,i} = \big(\Phi_e(x) \cdot \overrightarrow{\theta}_{h,i}\big)^3,$$

where $\overrightarrow{\theta}_h$ is the set of parameters between the input and hidden layers. The score of an action a is defined as

$$\text{Score}(a) = \sigma\big(\Phi_h(x) \cdot \overrightarrow{\theta}_{c,a}\big),$$

where $\overrightarrow{\theta}_{c,a}$ is the set of parameters between the hidden and output layers. We use the *arc-standard* features Φ_e as Chen and Manning (2014), which is also based on the *arc-eager* templates of Zhang and Nivre (2011), similar to those of the baseline model *L*.

3.3 Linear model with real-valued embeddings (*Turian*)

We apply the method of Turian et al. (2010), combining real-valued embeddings with discrete features in the linear baseline (Figure 1(c)). Given a

state x, the score of an action a is defined as

$$\text{Score}(a) = \sigma\Big(\big(\Phi_d(x) \oplus \Phi_e(x)\big) \cdot (\overrightarrow{\theta}_{d,a} \oplus \overrightarrow{\theta}_{c,a}) \Big),$$

where $\oplus$ is the vector concatenation operator.

3.4 Linear model with transformed embeddings (*Guo*)

We apply the method of Guo et al. (2014), combining embeddings into the linear baseline by first transforming into discrete values. Given a state x, the score of an action is defined as

$$\text{Score}(a) = \sigma\Big(\big(\Phi_d(x) \oplus d(\Phi_e(x))\big) \cdot (\overrightarrow{\theta}_{d,a} \oplus \overrightarrow{\theta}_{c,a}) \Big),$$

where d is a transformation function from real-value to binary features. We use clustering of embeddings for d as it gives better performances according to Guo et al. (2014). Following Guo et al. (2014), we use compounded clusters learnt by K-means algorithm of different granularities.

3.5 Directly combining linear and neural features (*This*)

We directly combine linear and neural features (Figure 1(e)). Given a state x, the score of an action is defined as

$$\text{Score}(a) = \sigma\Big(\big(\Phi_d(x) \oplus \Phi_h(x)\big) \cdot (\overrightarrow{\theta}_{d,a} \oplus \overrightarrow{\theta}_{c,a}) \Big),$$

where Φ_h is the same as the *NN* baseline. Note that like d in Guo, Φ_h is also a function that transforms embeddings Φ_e. The main difference is that it can be tuned in supervised training.

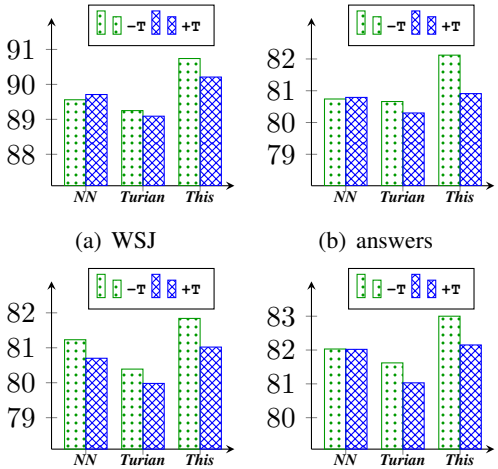
4 Web Domain Experiments

4.1 Setting

We perform experiments on the SANCL 2012 web data (Petrov and McDonald, 2012), using the Wall Street Journal (WSJ) training corpus to train the models and the WSJ development corpus to tune parameters. We clean the web domain texts following the method of Ma et al. (2014b). Automatic POS tags are produced by using a CRF model trained on the WSJ training corpus. The POS tags are assigned automatically on the training corpus by ten-fold jackknifing. Table 1 shows the corpus details.

Domain	#Sent	#Word	TA
WSJ-train	30,060	731,678	97.03
WSJ-dev	1,336	32,092	96.88
WSJ-test	1,640	35,590	97.51
answers	1,744	28,823	91.93
newsgroups	1,195	20,651	93.75
reviews	1,906	28,086	92.66

Table 1: Corpus statistics of our experiments, where TA denotes POS tagging accuracy.



(a) WSJ (b) answers

(c) newsgroups (d) reviews

Figure 2: Dev results on fine-tuning (UAS).

Following Chen and Manning (2014), we use the pre-trained word embedding released by Collobert et al. (2011), and set $h = 200$ for the hidden layer size, $\lambda = 10^{-8}$ for L2 regularization, and $\alpha = 0.01$ for the initial learning rate of Adagrad.

4.2 Development Results

Fine-tuning of embeddings. Chen and Manning (2014) fine-tune word embeddings in supervised training, consistent with Socher et al. (2013). Intuitively, fine-tuning embeddings allows in-vocabulary words to join the parameter space, thereby giving better fitting to in-domain data. However, it also forfeits the benefit of large-scale pre-training, because out-of-vocabulary (OOV) words do not have their embeddings fine-tuned. In this sense, the method of Chen and Manning resembles a traditional supervised sparse linear model, which can be weak on OOV.

On the other hand, the semi-supervised learning methods such as Turian et al. (2010) and Guo et al. (2014), do not fine-tune the word embeddings. Embeddings are taken as inputs rather than model

Model	WSJ				answers				newsgroups				reviews			
	UAS	LAS	OOV	OOE	UAS	LAS	OOV	OOE	UAS	LAS	OOV	OOE	UAS	LAS	OOV	OOE
L	88.19	86.16	83.72	—	79.30	74.24	68.43	—	82.55	79.06	69.07	—	80.77	76.16	72.20	—
NN	89.81	87.83	84.94	84.94	79.27	74.30	69.18	**69.18**	83.71	80.35	69.60	**69.60**	81.63	77.22	72.04	**72.04**
Turian	89.17	87.21	84.13	91.35	79.57	74.57	69.60	54.21	82.89	79.65	68.48	52.63	81.33	77.04	72.30	55.03
Guo	89.33	87.21	83.82	90.83	79.32	74.22	67.36	51.76	82.75	79.31	68.18	55.06	81.87	77.25	73.03	56.80
This	**90.61**	**88.68**	**88.00**	**93.77**	**80.08**	**75.18**	**69.97**	54.21	**84.64**	**81.35**	**69.66**	53.44	**82.53**	**78.15**	**73.39**	57.20
ZPar-local	88.95	86.90	84.63	—	78.98	73.81	68.15	—	82.43	79.01	67.30	—	81.21	76.45	70.38	—
C&M(2014)	89.56	87.55	79.15	79.15	79.82	74.63	67.78	67.78	83.39	79.72	67.95	67.95	81.60	76.91	68.83	68.83

Table 2: Main results on SANCL. All systems are deterministic.

parameters. Therefore, such methods can expect better cross-domain accuracies.

We empirically compare the models *NN*, *Turian* and *This* by fine-tuning (+T) and not fine-tuning (-T) word embeddings, and the results are shown in Figure 2. As expected, the baseline NN model gives better accuracies on WSJ with fine-tuning, but worse cross-domain accuracies. Interestingly, our combined model gives consistently better accuracies with fine-tuning. We attribute this to the use of sparse discrete features, which allows the model to benefit from large-scale pre-trained embeddings without sacrificing in-domain performance. The observation on *Turian* is similar. For the final experiments, we apply fine-tuning on the *NN* model, but not to the *Turian* and *This*. Note also tat for all experiments, the POS and label embedding features of Chen and Manning (2014) are fine-tuned, consistent with their original method.

Dropout rate. We test the effect of dropout (Hinton et al., 2012) during training, using a default ratio of 0.5 according to Chen and Manning (2014). In our experiments, we find that the dense *NN* model and our combined model achieve better performances by using dropout, but the other models do not benefit from dropout.

4.3 Final Results

The final results across web domains are shown in Table 2. Our logistic regression linear parser and re-implementation of Chen and Manning (2014) give comparable accuracies to the perceptron ZPar[2] and Stanford NN Parser[3], respectively.

It can be seen from the table that both *Turian* and *Guo*[4] outperform *L* by incorporating embedding features. *Guo* gives overall higher improvements, consistent with the observation of Guo et al. (2014) on NER. Our methods gives significantly[5] better results compared with *Turian* and *Guo*, thanks to the extra hidden layer.

Our OOV performance is higher than *NN*, because the embeddings of OOV words are not tuned, and hence the model can handle them effectively. Interestingly, *NN* gives higher accuracies on web domain out-of-embedding-vocabulary (OOE) words, out of which 54% are in-vocabulary.

Note that the accuracies of our parsers are lower than the best systems in the SANCL shared task, which use ensemble models. Our parser enjoys the fast speed of deterministic parsers, and in particular the baseline *NN* parser (Chen and Manning, 2014).

5 WSJ Experiments

For comparison with related work, we conduct experiments on Penn Treebank corpus also. We use the WSJ sections 2-21 for training, section 22 for development and section 23 for testing. WSJ constituent trees are converted to dependency trees using Penn2Malt[6]. We use auto POS tags consistent with previous work. The ZPar POS-tagger is used to assign POS tags. Ten-fold jackknifing is performed on the training data to assign POS automatically. For this set of experiments, the parser hyper-parameters are taken directly from the best settings in the Web Domain experiments.

The results are shown in Table 3, together with some state-of-the-art deterministic parsers. Comparing the *L*, *NN* and *This* models, the observations are consistent with the web domain.

[2]https://sourceforge.net/projects/zpar/, version 0.7.

[3]http://nlp.stanford.edu/software/nndep.shtml

[4]We compound six clusters of granularities 500, 1000, 1500, 2000, 2500, 3000.

[5]The p-values are below 0.01 using pairwise t-test.

[6]http://stp.lingfil.uu.se/ nivre/research/Penn2Malt.html

System	UAS	LAS
L	89.36	88.33
NN	91.15	90.04
This	**91.80**	**90.68**
ZPar-local	89.94	88.92
Ma et al. (2014a)	90.38	–
Chen and Manning (2014)	91.17	89.99
Honnibal et al. (2013)	91.30	90.00
Ma et al. (2014a)*	91.32	–

Table 3: Main results on WSJ. All systems are deterministic.

Our combined parser gives accuracies competitive to state-of-the-art deterministic parsers in the literature. In particular, the method of Chen and Manning (2014) is the same as our *NN* baseline. Note that Zhou et al. (2015) reports a UAS of 91.47% by this parser, which is higher than the results we obtained. The main results include the use of different batch size during, while Zhou et al. (2015) used a batch size of 100,000, we used a batch size of 10,000 in all experiments. Honnibal et al. (2013) applies dynamic oracle to the deterministic transition-based parsing, giving a UAS of 91.30%. Ma et al. (2014a) is similar to ZPar local, except that they use the *arc-standard* transitions, while ZPar-local is based on *arc-eager* transitions. Ma et al. (2014a)* uses a special method to process punctuations, leading to about 1% UAS improvements over the vanilla system.

Recently, Dyer et al. (2015) proposed a deterministic transition-based parser using LSTM, which gives a UAS of 93.1% on Stanford conversion of the Penn Treebank. Their work shows that more sophisticated neural network structures with long term memories can significantly improve the accuracy over local classifiers. Their work is orthogonal to ours.

6 Related Work

As discussed in the introduction, our work is related to previous work on integrating word embeddings into discrete models (Turian et al., 2010; Yu et al., 2013; Guo et al., 2014). Along this line, there has also been work that uses a neural network to automatically vectorize the structures of a sentence, and then taking the resulting vector as features in a linear NLP model (Socher et al., 2012; Tang et al., 2014; Yu et al., 2015). Our results show that the use of a hidden neural layer gives superior results compared with both direct integration and integration via a hard-coded transformation function (e.g binarization or clustering).

There has been recent work integrating continuous and discrete features for the task of POS tagging (Ma et al., 2014b; Tsuboi, 2014). Both models have essentially the same structure as our model. In contrast to their work, we systematically compare various ways to integrate discrete and continuous features, for the dependency parsing task. Our model is also different from Ma et al. (2014b) in the hidden layer. While they use a form of restricted Boltzmann machine to pre-train the embeddings and hidden layer from large-scale ngrams, we fully rely on supervised learning to train complex feature combinations.

Wang and Manning (2013) consider integrating embeddings and discrete features into a neural CRF. They show that combined neural and discrete features work better without a hidden layer (i.e. Turian et al. (2010)). They argue that nonlinear structures do not work well with high dimensional features. We find that using a hidden layer specifically for embedding features gives better results compared with using no hidden layers.

7 Conclusion

We studied the combination of discrete and continuous features for deterministic transition-based dependency parsing, comparing several methods to incorporate word embeddings and traditional sparse features in the same model. Experiments on both in-domain and cross-domain parsing show that directly adding sparse features into a neural network gives higher accuracies compared with all previous methods to incorporate word embeddings into a traditional sparse linear model.

Acknowledgments

We thank the anonymous reviewers for their constructive comments, which helped to improve the paper. This work is supported by the Singapore Ministry of Education (MOE) AcRF Tier 2 grant T2MOE201301 and SRG ISTD 2012 038 from Singapore University of Technology and Design.

References

Bernd Bohnet and Joakim Nivre. 2012. A transition-based system for joint part-of-speech tagging and

labeled non-projective dependency parsing. In *EMNLP-CONLL*, pages 1455–1465.

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

Jinho D. Choi and Andrew McCallum. 2013. Transition-based dependency parsing with selectional branching. In *ACL*, pages 1052–1062.

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

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *ACL*, pages 334–343.

Yoav Goldberg, Francesco Sartorio, and Giorgio Satta. 2014. A tabular method for dynamic oracles in transition-based parsing. *Transactions of the Association for Computational Linguistics*, 2:119–130.

Carlos Gómez-Rodríguez and Daniel Fernández-González. 2015. An efficient dynamic oracle for unrestricted non-projective parsing. In *ACL-IJCNLP*, pages 256–261.

Jiang Guo, Wanxiang Che, Haifeng Wang, and Ting Liu. 2014. Revisiting embedding features for simple semi-supervised learning. In *Proceedings of the 2014 Conference on EMNLP*, pages 110–120.

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

Matthew Honnibal, Yoav Goldberg, and Mark Johnson. 2013. A non-monotonic arc-eager transition system for dependency parsing. In *Proceedings of the 17th CONLL*, pages 163–172.

Ji Ma, Yue Zhang, and Jingbo Zhu. 2014a. Punctuation processing for projective dependency parsing. In *ACL (Volume 2: Short Papers)*, pages 791–796.

Ji Ma, Yue Zhang, and Jingbo Zhu. 2014b. Tagging the web: Building a robust web tagger with neural network. In *ACL*, pages 144–154.

Joakim Nivre, Johan Hall, and Jens Nilsson. 2004. Memory-based dependency parsing. In *CoNLL*.

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

Slav Petrov and Ryan McDonald. 2012. Overview of the 2012 shared task on parsing the web. In *Notes of the First Workshop on Syntactic Analysis of Non-Canonical Language (SANCL)*.

Richard Socher, Brody Huval, Christopher D Manning, and Andrew Y Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on EMNLP-CONLL*, pages 1201–1211.

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

Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. 2014. Learning sentiment-specific word embedding for twitter sentiment classification. In *ACL*, pages 1555–1565.

Yuta Tsuboi. 2014. Neural networks leverage corpus-wide information for part-of-speech tagging. In *Proceedings of the 2014 EMNLP*, pages 938–950.

Joseph Turian, Lev-Arie Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the 48th ACL*, pages 384–394.

Mengqiu Wang and Christopher D. Manning. 2013. Effect of non-linear deep architecture in sequence labeling. In *IJCNLP*, pages 1285–1291.

Mo Yu, Tiejun Zhao, Daxiang Dong, Hao Tian, and Dianhai Yu. 2013. Compound embedding features for semi-supervised learning. In *NAACL*, pages 563–568.

Mo Yu, Matthew R Gormley, and Mark Dredze. 2015. Combining word embeddings and feature embeddings for fine-grained relation extraction. In *NAACL*.

Yue Zhang and Stephen Clark. 2008. A tale of two parsers: Investigating and combining graph-based and transition-based dependency parsing. In *EMNLP*, pages 562–571.

Yue Zhang and Joakim Nivre. 2011. Transition-based dependency parsing with rich non-local features. In *Proceedings of the 49th ACL*, pages 188–193.

Yue Zhang and Joakim Nivre. 2012. Analyzing the effect of global learning and beam-search on transition-based dependency parsing. In *COLING 2012: Posters*, pages 1391–1400.

Yuan Zhang, Tao Lei, Regina Barzilay, Tommi Jaakkola, and Amir Globerson. 2014. Steps to excellence: Simple inference with refined scoring of dependency trees. In *ACL*, pages 197–207.

Hao Zhou, Yue Zhang, Shujian Huang, and Jiajun Chen. 2015. A neural probabilistic structured-prediction model for transition-based dependency parsing. In *ACL*, pages 1213–1222.

Efficient Inner-to-outer Greedy Algorithm for Higher-order Labeled Dependency Parsing

Xuezhe Ma and **Eduard Hovy**
Language Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA 15213, USA
xuezhem@cs.cmu.edu, ehovy@andrew.cmu.edu

Abstract

Many NLP systems use dependency parsers as critical components. Jonit learning parsers usually achieve better parsing accuracies than two-stage methods. However, classical joint parsing algorithms significantly increase computational complexity, which makes joint learning impractical. In this paper, we proposed an efficient dependency parsing algorithm that is capable of capturing multiple edge-label features, while maintaining low computational complexity. We evaluate our parser on 14 different languages. Our parser consistently obtains more accurate results than three baseline systems and three popular, off-the-shelf parsers.

1 Introduction

Natural language processing (NLP) systems, like machine translation (Xie et al., 2011), resource-low languages processing (McDonald et al., 2013; Ma and Xia, 2014), word sense disambiguation (Fauceglia et al., 2015) , and entity coreference resolution (Durrett and Klein, 2013), are becoming more sophisticated, in part because of utilizing syntacitc knowledges such as dependency parsing trees.

Dependency parsers predict dependency structures and dependency type labels on each edge. However, most graph-based dependency parsing algorithms only produce unlabeled dependency trees, particularly when higher-order factorizations are used (Koo and Collins, 2010; Ma and Zhao, 2012b; Martins et al., 2013; Ma and Zhao, 2012a). A two-stage method (McDonald, 2006) is often used because the complexity of some joint learning models is unacceptably high. On the other hand, joint learning models can benefit from edge-label information that has proven to be important to provide more accurate tree structures and labels (Nivre and Scholz, 2004).

Previous studies explored the trade-off between computational costs and parsing performance. Some work (McDonald, 2006; Carreras, 2007) simplified labeled information to only single label features. Other work (Johansson and Nugues, 2008; Bohnet, 2010) used richer label features but increased systems' complexities significantly, while achieving better parsing accuracy. Yet, there are no previous work addressing the problem of good balance between parsing accuracy and computational costs for joint parsing models.

In this paper, we propose a new dependency parsing algorithm that can utilize edge-label information of more than one edge, while simultaneously maintaining low computational complexity. The component needed to solve this dilemma is an inner-to-outer greedy approximation to avoid an exhaustive search. The contributions of this work are (i) showing the effectiveness of edge-label information on both UAS and LAS. (ii) proposing a joint learning parsing model which achieves both effectiveness and efficience. (iii) giving empirical evaluations of this parser on different treebanks over 14 languages.

2 Joint Parsing Algorithm

2.1 Basic Notations

In the following, x represents a generic input sentence, and y represents a generic dependency tree. Formally, for a sentence x, dependency parsing is the task of finding the dependency tree y with the highest-score for x:

$$y^*(x) = \underset{y \in \mathcal{Y}(x)}{\operatorname{argmax}} \operatorname{Score}(x, y). \quad (1)$$

Here $\mathcal{Y}(x)$ denotes the set of possible dependency trees for sentence x.

In this paper, we adopt the second-order sibling factorization (Eisner, 1996; McDonald and

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1322–1328,
Lisbon, Portugal, 17-21 September 2015.

Pereira, 2006), in which each sibling part consists of a tuple of indices (h, m, c) where (h, m) and (h, c) are a pair of adjacent edges to the same side of the head h. By adding labele information to this factorization, $\mathrm{Score}(x, y)$ can be rewritten as:

$$\begin{aligned}\mathrm{Score}(x, y) &= \sum_{(h,m,c,l_1,l_2)\in y} \mathrm{S_{sib}}(h, m, c, l_1, l_2) \\ &= \sum_{(h,m,c,l_1,l_2)\in y} \lambda^T \mathbf{f}(h, m, c, l_1, l_2)\end{aligned} \quad (2)$$

where $\mathrm{S_{sib}}(h, m, c, l_1, l_2)$ is the score function for the sibling part (h, m, c) with l_1 and l_2 being the labels of edge (h, m) and (h, c), respectively. $\mathbf{f}$ are feature functions and λ is the parameters of parsing model.

2.2 Exact Search Parsing

The unlabeled sibling parser introduces three types of dynamic-programming structures: *complete* spans $C_{(s,t)}$, which consist of the headword s and its descendents on one side with the endpoint t, *incomplete* spans $I_{(s,t)}$, which consist of the dependency (s, t) and the region between the head s and the modifier t, and *sibling* spans $S_{(s,t)}$, which represent the region between successive modifiers s and t of some head. To capture label information, we have to extend each incomplete span $I_{s,t}$ to $I_{s,t,l}$ to store the label of dependency edge from s to t. The reason is that there is an edge shared by two adjacent sibling parts (e.g. (h, m, c) and (h, c, c') share the edge (h, c)). So the incomplete span $I_{(s,t)}$ does not only depend on the label of dependency (s, t), but also the label of the dependency (s, r) for each split point r. The dynamic-programming procedure for new incomplete spans[1] are

$$I_{(s,t,l)} = \max_{s<r\le t} S_{(r,t)} + \max_{l'\in L} I_{(s,r,l')} + \mathrm{S_{sib}}(s, r, t, l', l) \quad (3)$$

where L is set of all edge labels. Then we have,

$$I_{(s,t)} = \max_{l\in L} I_{(s,t,l)} \quad (4)$$

$$l^*_{(s,t)} = \operatorname*{argmax}_{l\in L} I_{(s,t,l)} \quad (5)$$

The graphical specification of the this parsing algorithm is provided in Figure 1 (a). The computational complexity of the exactly searching algorithm is $O(|L|^2 n^3)$ time and $O(|L|n^2)$ space. In practice, $|L|$ is probably large. For English, the number of edge labels in Stanford Basic Dependencies (De Marneffe et al., 2006) is 45, and the number in the treebank of CoNLL-2008 shared task is 70. So it is impractical to perform an exhaustive search for parsing, and more efficient approximating algorithms are needed.

[1] Symmetric right-headed versions are elided for brevity

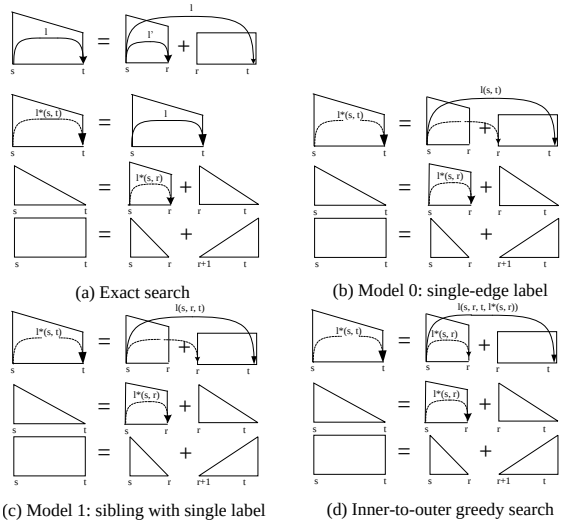


Figure 1: The dynamic-programming structures and derivation of four parsing algorithms. $I_{(s,t,l)}$ and $I_{(s,t)}$ are depicted as trapezoids with solid and dashed lines, respectively. $C_{(s,t)}$ are depicted as triangles and $S_{(s,t)}$ are depicted as boxes. Symmetric right-headed versions are elided for brevity.

2.3 Two Intermediate Models

In this section, we describe two intuitive simplifications of the labeled parsing model presented above. For the two simplified parsing models, efficient algorithms are available.

2.3.1 Model 0: Single-edge Label

In this parsing model, labeled features are restricted to a single edge. Specifically,

$$\mathrm{S_{sib}}(h, m, c, l_1, l_2) = \mathrm{S_{sib}}(h, c, l_2).$$

Then the dynamic-programming derivation for each incomplete span becomes

$$\begin{aligned}I_{(s,t)} &= \max_{s<r\le t} \left\{ I_{(s,r)} + S_{(r,t)} + \max_{l\in L} \mathrm{S_{sib}}(s, t, l) \right\} \\ &= \max_{s<r\le t} \left\{ I_{(s,r)} + S_{(r,t)} + \mathrm{S_{sib}}(s, t, l(s, t)) \right\}\end{aligned}$$

where $l(s, t) = \operatorname*{argmax}_{l\in L} \mathrm{S_{sib}}(s, t, l)$.

In this case, therefore, we do not have to extend incomplete spans. The computational cost to calculate $l(s, t)$ is $O(|L|n^2)$ time, so the computational complexity of this algorithm is $O(n^3 + |L|n^2)$ time and $O(n^2)$ space.

2.3.2 Model 1: Sibling with Single Label

As remarked in McDonald (2006), Model 0 can be slightly enriched to include single label features associated with a sibling part. Formally,

$$\mathrm{S}_{\mathrm{sib}}(h, m, c, l_1, l_2) = \mathrm{S}_{\mathrm{sib}}(h, m, c, l_2).$$

Now, the dynamic-programming derivation is

$$I_{(s,t)} = \max_{s<r\leq t} \left\{ I_{(s,r)} + S_{(r,t)} + \mathrm{S}_{\mathrm{sib}}(s, r, t, l(s, r, t)) \right\}$$

where $l(s, r, t) = \operatorname*{argmax}_{l \in L} \mathrm{S}_{\mathrm{sib}}(s, r, t, l)$.

The additional algorithm to calculate the best edge label $l(s, r, t)$ takes $O(|L|n^3)$ time. Therefore, this algorithm requires $O(|L|n^3)$ time and $O(n^2)$ space[2]. Figure 1 (b) and Figure 1 (c) provide the graphical specifications for Model 0 and Model 1, respectively.

2.4 Inner-to-outer Greedy Search

Though the two intermediate parsing models, model 0 and model 1, encode edge-label information and have efficient parsing algorithms, the labeled features they are able to capture are relatively limited due to restricting their labeled feature functions to a single label. Our experimental results show that utilizing these edge-label information yields a slight improvement of parsing accuracy (see Section 3 for details). In this section, we describe our new labeled parsing model that can exploit labeled features involving two edge-labels in a sibling part. To achieve efficient search, we adopt an method characterized by inferring labels of outer parts from the labels of inner ones.

Formally, consider the maximization problem in Eq 3. It can be treated as a two-layer maximization: first fixes a split point r and maximizes over all edge-label l, then maximizes over all possible split points. Our approach approximates the maximization in the first layer:

$$\begin{aligned} & \max_{l' \in L} I_{(s,r,l')} + \mathrm{S}_{\mathrm{sib}}(s, r, t, l', l) \\ \approx \quad & I_{(s,r)} + \mathrm{S}_{\mathrm{sib}}(s, r, t, l^*_{(s,r)}, l) \end{aligned} \tag{6}$$

Then the dynamic-programming derivation for each incomplete span is

$$I_{(s,t)} = \max_{s<r\leq t} I_{(s,r)} + S_{(r,t)} + \max_{l \in L} \mathrm{S}_{\mathrm{sib}}(s, r, t, l^*_{(s,r)}, l) \tag{7}$$

To compute $I_{(s,t)}$, we need to calculate

$$l(s, r, t, l^*_{(s,r)}) = \operatorname*{argmax}_{l \in L} \mathrm{S}_{\mathrm{sib}}(s, r, t, l^*_{(s,r)}, l), \tag{8}$$

which is similar to the calculation of $l(s, r, t)$ in Model 1. The only difference between them is $l^*_{(s,r)}$ that can be calculated in previous derivations. Thus, their computation costs are almost the same.

The procedure of our algorithm to derivate incomplete spans can be regarded as two steps. At the first step, the algorithm goes through all possible split points (Eq 7). Then at the second step, at each split point r, it calculate the label $l(s, r, t, l^*_{(s,r)})$ (Eq 8) based on the sibling part (s, r, t) and the label $l^*_{(s,r)}$ which is the "best" label for dependency edge (s, r) based on incomplete span $I_{(s,r)}$. The key insight of this algorithm is the inner-to-outer dynamic-programming structure: inner modifiers (r) of a head (s) and their "best" labels ($l^*_{(s,r)}$) are generated before outer ones (t). Thus, using already computed "best" labels of inner dependency edges makes us get rid of maximizing over two labels, l' and l. Moreover, we do not have to extend each incomplete span by the augmentation with a "label" index. This makes the space complexity remains $O(n^2)$, which is important in practice. The graphical specification is provided in Figure 1 (d).

3 Experiments

3.1 Setup

We conduct our experiments on 14 languages, including the English treebank from CoNLL-2008 shared task (Surdeanu et al., 2008) and all 13 treebanks from CoNLL-2006 shared task (Buchholz and Marsi, 2006). We train our parser using The k-best version of the Margin Infused Relaxed Algorithm (MIRA) (Crammer and Singer, 2003; Crammer et al., 2006; McDonald, 2006). In our experiments, we set $k = 1$ and fix the number of iteration to 10, instead of tuning these parameters on development sets. Following previous work, all experiments are evaluated on the metrics of unlabeled attachment score (UAS) and Labeled attachment score (LAS), using the official scorer[3] of CoNLL-2006 shared task.

3.2 Non-Projective Parsing

The parsing algorithms described in the paper fall into the category of *projective* dependency parsers, which exclude crossing dependency edges. Since the treebanks from CoNLL shared tasks contain non-projective edges, we use the "mountain-

[2] We do not have to store $l(s, r, t)$, as each $l(s, r, t)$ will be calculated exactly once.

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

	Two-stage		Model 0		Model 1		Our Model		Best in CoNLL		
	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	System
ar	78.52	64.69	79.38	66.91	78.72	66.67	**79.60**	**67.09**	79.34	66.91	MD06
bg	91.98	86.75	92.34	87.47	92.12	87.41	**92.68**	**87.79**	92.04	87.57	MD06
zh	91.25	86.50	91.81	87.53	92.27	88.23	92.58	88.51	**93.18**	**89.96**	RD06
cs	87.32	76.56	87.36	78.42	87.72	78.90	**88.01**	79.31	87.30	**80.18**	MD06
da	90.96	84.63	91.24	85.47	91.50	85.79	**91.44**	**85.55**	90.58	84.79	MD06
nl	83.79	78.81	84.25	80.49	84.27	80.41	**84.45**	**80.31**	83.57	79.19	MD06
en	91.92	88.09	92.10	88.96	92.19	89.18	**92.45**	89.43	92.38	**90.13**	JN08
de	90.52	87.46	90.52	87.12	90.40	87.30	**90.79**	**87.74**	90.38	87.34	MD06
ja	93.14	90.95	93.32	91.29	93.52	91.80	**93.54**	**91.80**	93.10	91.65	NV06
pt	**91.60**	86.05	91.04	86.46	91.02	87.30	91.54	**87.68**	91.22	87.60	NV06
sl	83.03	70.80	83.23	72.88	83.93	73.38	**84.39**	**73.74**	83.17	73.44	MD06
es	85.61	80.95	86.05	82.83	86.42	**83.59**	**86.44**	83.29	86.05	82.25	MD06
sv	89.07	81.88	89.74	82.77	89.82	83.13	**89.94**	83.09	89.50	**84.58**	NV06
tr	75.02	57.78	**75.40**	60.25	74.75	59.73	75.32	60.39	**75.82**	**65.68**	NV06
av	87.41	80.14	87.70	81.35	87.76	81.63	**88.08**	81.84	87.69	**82.23**	– –

Table 1: UAS and LAS of non-projective versions of our parsing algorithms on 14 treebanks from CoNLL shared tasks, together with three baseline systems and the best systems for each language reported in CoNLL shared tasks. MD06 is McDonald et al. (2006), RD06 is Riedel et al. (2006), JN08 is Johansson and Nugues (2008), and NV06 is Nivre et al. (2006) Bold indicates the best result for a language. Red values represent statistically significant improvements over two-stage baseline system on the corresponding metrics with $p < 0.01$, using McNemar's test. Blue values indicate statistically significant improvements with $p < 0.05$.

climbing" non-projective parsing algorithm proposed in McDonald and Pereira (2006). This approximating algorithm first searches the highest scoring projective parse tree and then it rearranges edges in the tree until the rearrangements do not increase the score for the tree anymore [4].

3.3 Results and Comparison

Table 1 illustrates the parsing results our parser with non-projective parsing algorithm, together with three baseline systems—the two-stage system (McDonald, 2006) and the two intermediate models, Model 0 and Model 1—and the best systems reported in CoNLL shared tasks for each language. Our parser achieves better parsing performance on both UAS and LAS than all the three baseline systems for 12 languages. The two exceptions are Portuguese and Turkish, on which our parser achieves better LAS and comparable UAS.

Comparing with the best systems from CoNLL, our parser achieves better performance on both UAS and LAS for 9 languages. Moreover, the average UAS of our parser over the 14 languages is better than that of the best systems in CoNLL. It should be noted that the best results for 14 languages in CoNLL are not from one single system, but different systems that achieved best results for different languages. The system of McDonald et al. (2006) achieved the best average parsing performance over 13 languages (excluding English) in CoNLL-2006 shared tasks. Its average UAS and LAS are 87.03% and 80.83%, respectively, while our average UAS and LAS excluding English are 87.79% and 81.29%. So our parser shows significant improvement over the single best system reported in CoNLL-2006 shared task.

[4] Additional care is required in the non-projective approximation since a change of one edge could result in a label change for multiple edges

System	UAS	LAS
MaltParser	89.3	86.9
MSTParser	90.7	87.6
DNNParser	91.8	89.6
This Paper	**92.4**	**89.9**

Table 2: Parsing performance on PTB. The results for MaltParser, MSTParser and DNNParser are from table 5 of Chen and Manning (2014).

3.4 Experiments on PTB

To make a thorough empirical comparison with previous studies, we also evaluate our system on the English Penn Treebanks (Marcus et al., 1993) with Stanford Basic Dependencies (De Marneffe et al., 2006). We compare our parser with three off-the-shelf parsers: MaltParser (Nivre and Scholz, 2004; Zhang and Clark, 2008; Zhang and Nivre, 2011), MSTParser (McDonald et al., 2005), and the parser using Neural Networks

Label	Description	F1 (UAS) TST	Ours	diff	F1 (LAS) TST	Ours	diff
MNR	Adverbial of manner	54.01	66.10	12.10	52.23	62.15	9.92
OPRD	Predicative complement of raising/control verb	86.61	96.92	10.31	86.61	89.89	3.28
APPO	Apposition	77.40	82.62	5.22	72.74	77.03	4.29
ADV	General adverbial	73.33	77.65	4.32	69.86	73.14	3.28
AMOD	Modifier of adjective or adverbial	75.49	79.58	4.09	72.60	76.91	4.30
TMP	Temporal adverbial or nominal modifier	73.47	76.76	3.29	64.50	68.59	4.09
DIR	Adverbial of direction	62.42	65.02	2.60	62.42	64.61	2.19
LOC	Locative adverbial or nominal modifier	75.78	78.35	2.57	63.11	65.83	2.72
OBJ	Object	91.69	94.08	2.39	90.62	93.23	2.61

Table 3: Top 10 dependency labels on which our algorithm achieves most improvements on the F1 score of UAS, together with the corresponding improvements of LAS. "TST" indicates the two-stage system. The first column is the label name in the treebank. The second column is the label's description from Surdeanu et al. (2008).

(DNNParser) (Chen and Manning, 2014). The results are listed in Table 2. Clearly, our parser is superior in terms of both UAS and LAS.

3.5 Analysis

To better understand the performance of our parser, we analyze the distribution of our parser's UAS and LAS over different dependency labels on the English CoNLL treebank, compared with the ones of the two-stage model. Table 3 lists the top 10 dependency labels on which our algorithm achieves most improvements on the F1 score of UAS, together with the corresponding improvements of LAS.

From Table 3 we can see among the 10 labels, there are 5 labels — "MNR", "ADV", "TMP", "DIR", "LOC" — which are a specific kind of adverbials. This illustrates that our parser performs well on the recognition of different kinds of adverbials. Moreover, the label "OPRD" and "OBJ" indicate dependency relations between verbs and their modifiers, too. In addition, our parser also significantly improves the accuracy of appositional relations ("APPO").

4 Conclusion

We proposed a new dependency parsing algorithm which can jointly learn dependency structures and edge labels. Our parser is able to use multiple edge-label features, while maintaining low computational complexity. Experimental results on 14 languages show that our parser significantly improves the accuracy of both dependency structures (UAS) and edge labels (LAS), over three baseline systems and three off-the-shelf parsers. This demonstrates that jointly learning dependency structures and edge labels can benefit both performance of tree structures and labeling accuracy. Moreover, our parser outperforms the best systems of different languages reported in CoNLL shared task for 9 languages.

In future, we are interested in extending our parser to higher-order factorization by increasing horizontal context (e.g., from siblings to "tri-siblings") and vertical context (e.g., from siblings to "grand-siblings") and validating its effectiveness via a wide range of NLP applications.

Acknowledgements

This research was supported in part by DARPA grant FA8750-12-2-0342 funded under the DEFT program. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of DARPA.

References

Bernd Bohnet. 2010. Very high accuracy and fast dependency parsing is not a contradiction. In *Proceedings of Coling-2010*, pages 89–97, Beijing, China, August.

Sabine Buchholz and Erwin Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proceeding of CoNLL-2006*, pages 149–164, New York, NY.

Xavier Carreras. 2007. Experiments with a higher-order projective dependency parser. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CONLL*, pages 957–961.

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

Koby Crammer and Yoram Singer. 2003. Ultraconservative online algorithms for multiclass problems. *Journal of Machine Learining Research*, 3:951–991.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006. Online passive-aggressive algorithms. *Jornal of Machine Learning Research*, 7:551–585.

Marie-Catherine De Marneffe, Bill MacCartney, Christopher D Manning, et al. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC-2006*, pages 449–454.

Greg Durrett and Dan Klein. 2013. Easy victories and uphill battles in coreference resolution. In *Proceedings of EMNLP-2013*, pages 1971–1982, Seattle, Washington, USA, October.

Jason Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of COLING-1996*, pages 340–345.

Nicolas R Fauceglia, Yiu-Chang Lin, Xuezhe Ma, and Eduard Hovy. 2015. Word sense disambiguation via propstore and ontonotes for event mention detection. In *Proceedings of the The 3rd Workshop on EVENTS: Definition, Detection, Coreference, and Representation*, pages 11–15, Denver, Colorado, June.

Richard Johansson and Pierre Nugues. 2008. Dependency-based syntactic-semantic analysis with propbank and nombank. In *Proceedings of the CoNLL-2008*, pages 183–187.

Terry Koo and Michael Collins. 2010. Efficient third-order dependency parsers. In *Proceedings of ACL-2010*, pages 1–11, Uppsala, Sweden, July.

Xuezhe Ma and Fei Xia. 2014. Unsupervised dependency parsing with transferring distribution via parallel guidance and entropy regularization. In *Proceedings of ACL-2014*, pages 1337–1348, Baltimore, Maryland, June.

Xuezhe Ma and Hai Zhao. 2012a. Fourth-order dependency parsing. In *Proceedings of COLING 2012: Posters*, pages 785–796, Mumbai, India, December.

Xuezhe Ma and Hai Zhao. 2012b. Probabilistic models for high-order projective dependency parsing. *Technical Report, arXiv:1502.04174*.

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

Andre Martins, Miguel Almeida, and Noah A. Smith. 2013. Turning on the turbo: Fast third-order non-projective turbo parsers. In *Proceedings of ACL-2013 (Volume 2: Short Papers)*, pages 617–622, Sofia, Bulgaria, August.

Ryan McDonald and Fernando Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proceedings of EACL-2006*, pages 81–88, Trento, Italy, April.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajic. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of HLT/EMNLP-2005*, pages 523–530, Vancouver, Canada, October.

Ryan McDonald, Kevin Lerman, and Fernando Pereira. 2006. Multilingual dependency analysis with a two-stage discriminative parser. In *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL-X)*, pages 216–220, New York City, June.

Ryan McDonald, Joakim Nivre, Yvonne Quirmbach-Brundage, Yoav Goldberg, Dipanjan Das, Kuzman Ganchev, Keith Hall, Slav Petrov, Hao Zhang, Oscar Täckström, Claudia Bedini, Núria Bertomeu Castelló, and Jungmee Lee. 2013. Universal dependency annotation for multilingual parsing. In *Proceedings of ACL-2013*, pages 92–97, Sofia, Bulgaria, August.

Ryan McDonald. 2006. *Discriminative learning spanning tree algorithm for dependency parsing*. Ph.D. thesis, University of Pennsylvania.

Joakim Nivre and Mario Scholz. 2004. Deterministic dependency parsing of English text. In *Proceedings of COLING-2004*, pages 64–70, Geneva, Switzerland, August 23-27.

Joakim Nivre, Johan Hall, Jens Nilsson, Gülşen Eryiğit, and Svetoslav Marinov. 2006. Labeled pseudo-projective dependency parsing with support vector machines. In *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL-X)*, pages 221–225, New York City, June.

Sebastian Riedel, Ruket Çakıcı, and Ivan Meza-Ruiz. 2006. Multi-lingual dependency parsing with incremental integer linear programming. In *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL-X)*, pages 226–230, New York City, June.

Mihai Surdeanu, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. The conll-2008 shared task on joint parsing of syntactic and semantic dependencies. In *Proceedings of CoNLL-2008*, pages 159–177.

Jun Xie, Haitao Mi, and Qun Liu. 2011. A novel dependency-to-string model for statistical machine translation. In *Proceedings of EMNLP-2011*, pages 216–226, Edinburgh, Scotland, UK., July.

Yue Zhang and Stephen Clark. 2008. A tale of two parsers: investigating and combining graph-based and transition-based dependency parsing using beam search. In *Proceedings of EMNLP*, pages 562–571.

Yue Zhang and Joakim Nivre. 2011. Transition-based dependency parsing with rich non-local features. In *Proceeding of ACL 2011*, pages 188–193, Portland, Oregon, June.

Online Updating of Word Representations for Part-of-Speech Tagging

Wenpeng Yin
LMU Munich
wenpeng@cis.lmu.de

Tobias Schnabel
Cornell University
tbs49@cornell.edu

Hinrich Schütze
LMU Munich
inquiries@cislmu.org

Abstract

We propose *online unsupervised domain adaptation (DA)*, which is performed *incrementally* as data comes in and is applicable when batch DA is not possible. In a part-of-speech (POS) tagging evaluation, we find that online unsupervised DA performs as well as batch DA.

1 Introduction

Unsupervised domain adaptation is a scenario that practitioners often face when having to build robust NLP systems. They have labeled data in the source domain, but wish to improve performance in the target domain by making use of *unlabeled data* alone. Most work on unsupervised domain adaptation in NLP uses *batch learning*: It assumes that a large corpus of unlabeled data of the target domain is available before testing. However, batch learning is not possible in many real-world scenarios where incoming data from a new target domain must be processed immediately. More importantly, in many real-world scenarios the data does not come with neat domain labels and it may not be immediately obvious that an input stream is suddenly delivering data from a new domain.

Consider an NLP system that analyzes emails at an enterprise. There is a constant stream of incoming emails and it changes over time – without any clear indication that the models in use should be adapted to the new data distribution. Because the system needs to work in real-time, it is also desirable to do any adaptation of the system *online*, without the need of stopping the system, changing it and restarting it as is done in batch mode.

In this paper, we propose *online unsupervised domain adaptation* as an extension to traditional unsupervised DA. In online unsupervised DA, domain adaptation is performed incrementally as data comes in. Specifically, we adopt a form of *representation learning*. In our experiments, the incremental updating will be performed for representations of words. Each time a word is encountered in the stream of data at test time, its representation is updated.

To the best of our knowledge, the work reported here is the first study of online unsupervised DA. More specifically, we evaluate online unsupervised DA for the task of POS tagging. We compare POS tagging results for three distinct approaches: static (the baseline), batch learning and online unsupervised DA. Our results show that online unsupervised DA is comparable in performance to batch learning while requiring no retraining or prior data in the target domain.

2 Experimental setup

Tagger. We reimplemented the FLORS tagger (Schnabel and Schütze, 2014), a fast and simple tagger that performs well in DA. It treats POS tagging as a window-based (as opposed to sequence classification), multilabel classification problem. FLORS is ideally suited for online unsupervised DA because its representation of words includes distributional vectors – these vectors can be easily updated in both batch learning and online unsupervised DA. More specifically, a word's representation in FLORS consists of four feature vectors: one each for its suffix, its shape and its left and right distributional neighbors. Suffix and shape features are standard features used in the literature; our use of them is exactly as described by Schnabel and Schütze (2014).

Distributional features. The i^{th} entry x_i of the left distributional vector of w is the weighted number of times the *indicator word* c_i occurs immediately to the left of w:

$$x_i = \text{tf}\left(\text{freq}\left(\text{bigram}(c_i, w)\right)\right)$$

where c_i is the word with frequency rank i in the corpus, $\text{freq}\left(\text{bigram}(c_i, w)\right)$ is the number of occurrences of the bigram "$c_i\ w$" and we weight non-

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1329–1334,
Lisbon, Portugal, 17-21 September 2015.

	newsgroups		reviews		weblogs		answers		emails		wsj	
	ALL	OOV	ALL	OOV	ALL	OOV	ALL	OOV	ALL	OOV	ALL	OOV
TnT	88.66	54.73	90.40	56.75	93.33	74.17	88.55	48.32	88.14	58.09	95.75	88.30
Stanford	89.11	56.02	91.43	58.66	94.15	77.13	88.92	49.30	88.68	58.42	**96.83**	90.25
SVMTool	89.14	53.82	91.30	54.20	94.21	76.44	88.96	47.25	88.64	56.37	96.63	87.96
C&P	89.51	57.23	91.58	59.67	94.41	78.46	89.08	48.46	88.74	58.62	96.78	88.65
S&S	90.86	66.42	92.95	75.29	94.71	83.64	90.30	62.16	89.44	62.61	96.59	90.37
S&S (reimpl.)	90.68	65.52	93.00	75.50	94.64	82.91	90.18	61.98	89.53	62.46	96.60	89.70
BATCH	**90.87**	**71.18**	**93.07**	**79.03**	**94.86**	**86.53**	**90.70**	**65.29**	89.84	65.44	96.63	**91.86**
ONLINE	90.85	71.00	**93.07**	**79.03**	**94.86**	**86.53**	90.68	65.16	**89.85**	**65.48**	96.62	91.69

Table 1: BATCH and ONLINE accuracies are comparable and state-of-the-art. Best number in each column is bold.

zero frequencies logarithmically: $\mathrm{tf}(x) = 1 + \log(x)$. The right distributional vector is defined analogously. We restrict the set of indicator words to the $n = 500$ most frequent words. To avoid zero vectors, we add an entry x_{n+1} to each vector that counts omitted contexts:

$$x_{501} = \mathrm{tf}(\textstyle\sum_{j:j>n} \mathrm{freq}\,(\mathrm{bigram}(c_j, w)))$$

Let $f(w)$ be the concatentation of the two distributional and suffix and shape vectors of word w. Then FLORS represents token v_i as follows:

$$f(v_{i-2}) \oplus f(v_{i-1}) \oplus f(v_i) \oplus f(v_{i+1}) \oplus f(v_{i+2})$$

where $\oplus$ is vector concatenation. FLORS then tags token v_i based on this representation.

FLORS assumes that the association between distributional features and labels does not change fundamentally when going from source to target. This is in contrast to other work, notably Blitzer et al. (2006), that carefully selects "stable" distributional features and discards "unstable" distributional features. The hypothesis underlying FLORS is that basic distributional POS properties are relatively stable across domains – in contrast to semantic and other more complex tasks. The high performance of FLORS (Schnabel and Schütze, 2014) suggests this hypothesis is true.

Data. *Test set.* We evaluate on the development sets of six different TDs: five SANCL (Petrov and McDonald, 2012) domains – newsgroups, weblogs, reviews, answers, emails – and sections 22-23 of WSJ for in-domain testing.

We use two *training sets* of different sizes. In condition *l:big* (big labeled data set), we train FLORS on sections 2-21 of Wall Street Journal (WSJ). Condition *l:small* uses 10% of l:big.

Data for word representations. We also vary the size of the datasets that are used to compute the word representations before the FLORS model is trained on the training set. In condition *u:big*, we compute distributional vectors on the joint corpus of all labeled and unlabeled text of source and target domains (except for the test sets). We also include 100,000 WSJ sentences from 1988 and 500,000 sentences from Gigaword (Parker, 2009). In condition *u:0*, only labeled training data is used.

Methods. We implemented the following modification compared to the setup in (Schnabel and Schütze, 2014): distributional vectors are kept in memory as count vectors. This allows us to increase the counts during online tagging.

We run experiments with three versions of FLORS: STATIC, BATCH and ONLINE. All three methods compute word representations on "data for word representations" (described above) before the model is trained on one of the two "training sets" (described above).

STATIC. Word representations are not changed during testing.

BATCH. Before testing, we update count vectors by $\mathrm{freq}\,(\mathrm{bigram}(c_i, w))$ += $\mathrm{freq}^*\,(\mathrm{bigram}(c_i, w))$, where $\mathrm{freq}^*(\cdot)$ denotes the number of occurrences of the bigram "c_i w" in the entire test set.

ONLINE. Before tagging a test sentence, both left and right distributional vectors are updated via $\mathrm{freq}\,(\mathrm{bigram}(c_i, w))$ += 1 for each appearance of bigram "c_i w" in the sentence. Then the sentence is tagged using the updated word representations. As tagging progresses, the distributional representations become increasingly specific to the target domain (TD), converging to the representations that BATCH uses at the end of the tagging process.

In all three modes, suffix and shape features are always fully specified, for both known and unknown words.

3 Experimental results

Table 1 compares performance on SANCL for a number of baselines and four versions of FLORS: S&S, Schnabel and Schütze (2014)'s version of FLORS, "S&S (reimpl.)", our reimplementation of that version, and BATCH and ONLINE, the two versions of FLORS we use in this paper. Compar-

			u:0				u:big			
			ALL	KN	SHFT	OOV	ALL	KN	SHFT	OOV
newsgroups	l:small	STATIC	87.02	90.87	71.12	57.16	89.02	91.48	81.53	58.30
		ONLINE	87.99	90.87	76.10	65.64	**89.84**	**92.38**	82.58	**67.09**
		BATCH	**88.28**	**91.08**	**77.01**	**66.37**	89.82	92.37	**82.65**	67.03
	l:big	STATIC	89.69	93.00	82.65	57.82	89.93	92.41	84.94	58.97
		ONLINE	90.51	**93.13**	82.51	67.57	90.85	**93.04**	84.94	71.00
		BATCH	**90.69**	93.12	**83.24**	**69.43**	**90.87**	93.03	**85.20**	**71.18**
reviews	l:small	STATIC	89.08	91.96	66.55	65.90	91.45	92.47	80.11	70.81
		ONLINE	89.67	92.14	**70.14**	69.67	**92.11**	**93.62**	81.46	**78.42**
		BATCH	**89.79**	**92.23**	69.86	**71.27**	92.10	93.60	**81.51**	**78.42**
	l:big	STATIC	91.96	93.94	82.30	67.97	92.42	93.53	84.65	69.97
		ONLINE	92.33	94.03	**83.59**	72.50	**93.07**	**94.36**	**85.71**	**79.03**
		BATCH	**92.42**	**94.09**	83.53	**73.35**	**93.07**	**94.36**	**85.71**	**79.03**
weblogs	l:small	STATIC	91.58	94.29	79.95	72.74	93.42	94.77	89.80	77.42
		ONLINE	92.51	94.52	81.76	80.46	**94.21**	95.40	**91.08**	**84.03**
		BATCH	**92.68**	**94.60**	**82.34**	**81.20**	94.20	**95.42**	91.03	83.87
	l:big	STATIC	93.45	95.64	**90.15**	72.68	94.09	95.54	91.90	76.94
		ONLINE	94.18	95.82	89.80	80.35	**94.86**	95.81	**92.60**	**86.53**
		BATCH	**94.34**	**95.85**	90.03	**81.84**	**94.86**	**95.82**	**92.60**	**86.53**
answers	l:small	STATIC	86.93	90.89	66.51	53.43	88.98	91.09	77.63	57.36
		ONLINE	87.48	**91.18**	68.07	56.47	**89.71**	92.42	78.11	**64.21**
		BATCH	**87.56**	91.11	**68.25**	**58.44**	**89.71**	**92.43**	**78.23**	64.09
	l:big	STATIC	89.54	92.76	78.65	56.22	90.06	92.18	80.70	58.25
		ONLINE	89.98	92.97	**79.07**	59.77	90.68	93.21	81.48	65.16
		BATCH	**90.14**	**93.10**	79.01	**60.72**	**90.70**	**93.22**	**81.54**	**65.29**
emails	l:small	STATIC	85.43	90.85	57.85	51.65	87.76	90.35	70.86	56.76
		ONLINE	86.30	91.26	60.56	55.83	88.45	92.31	71.67	61.57
		BATCH	**86.42**	**91.31**	**61.03**	**56.32**	**88.46**	**92.32**	**71.71**	**61.65**
	l:big	STATIC	88.31	92.98	71.38	52.71	89.21	91.74	73.80	58.99
		ONLINE	88.86	93.08	**72.38**	57.78	**89.85**	**93.30**	**75.32**	**65.48**
		BATCH	**88.96**	**93.11**	72.28	**58.85**	89.84	**93.30**	75.27	65.44
wsj	l:small	STATIC	94.64	95.44	83.38	82.72	95.73	95.88	**90.36**	87.87
		ONLINE	**94.86**	**95.53**	85.37	85.22	**95.80**	96.21	89.89	**89.70**
		BATCH	94.80	95.46	**85.51**	**85.38**	**95.80**	**96.22**	89.89	**89.70**
	l:big	STATIC	96.44	**96.85**	92.75	85.38	96.56	96.72	93.35	88.04
		ONLINE	**96.50**	**96.85**	**93.55**	86.38	96.62	**96.89**	93.35	91.69
		BATCH	96.47	96.82	93.48	**86.54**	**96.63**	**96.89**	**93.42**	**91.86**

Table 2: ONLINE / BATCH accuracies are generally better than STATIC (see bold numbers) and improve with both more training data and more unlabeled data.

ing lines "S&S" and "S&S (reimpl.)" in the table, we see that our reimplementation of FLORS is comparable to S&S's. For the rest of this paper, our setup for BATCH and ONLINE differs from S&S's in three respects. (i) We use Gigaword as additional unlabeled data. (ii) When we train a FLORS model, then the corpora that the word representations are derived from do not include the test set. The set of corpora used by S&S for this purpose includes the test set. We make this change because application data may not be available at training time in DA. (iii) The word representations used when the FLORS model is trained are derived from all six SANCL domains. This simplifies the experimental setup as we only need to train a single model, not one per domain. Table 1 shows that our setup with these three changes (lines BATCH and ONLINE) has state-of-the-art performance on SANCL for domain adaptation (bold numbers).

Table 2 investigates the effect of sizes of labeled and unlabeled data on performance of ONLINE and BATCH. We report accuracy for all (ALL) tokens, for tokens occurring in both l:big and l:small (KN), tokens occurring in neither l:big nor l:small (OOV) and tokens ocurring in l:big, but not in l:small (SHFT).[1] Except for some minor variations in a few cases, both using more labeled data and using more unlabeled data improves tagging accuracy for both ONLINE and BATCH. ONLINE and BATCH are generally better or as good as STATIC (in bold), always on ALL and OOV, and with a few exceptions also on KN and SHFT.

ONLINE performance is comparable to BATCH performance: it is slightly worse than BATCH on u:0 (largest ALL difference is .29) and at most .02 different from BATCH for ALL on u:big. We ex-

[1] We cannot give the standard, single OOV evaluation number here since OOVs are different in different conditions, hence the breakdown into three measures.

		unknowns				unseens				known words			
		u:0		u:big		u:0		u:big		u:0		u:big	
		err	std	err	std	err	std	err	std	err	std	err	std
l:small	STATIC	.3670†	.00085	.2104	.00081	.1659†	.00076	.1084	.00056	.1309†	.00056	.0801	.00042
	ONLINE	.3094	.00160	.2102*	.00093	.1467	.00120	.1086*	.00074	.1186	.00095	.0802*	.00048
	BATCH	.3050†	.00143	.2101	.00083	.1646†	.00145	.1076	.00060	.1251†	.00103	.0801	.00040
l:big	STATIC	.1451†	.00114	.1042	.00100	.0732	.00052	.0690	.00042	.0534	.00027	.0503	.00025
	ONLINE	.1404	.00125	.1037*	.00098	.0727	.00051	.0689*	.00051	.0529	.00031	.0502*	.00031
	BATCH	.1382†	.00140	.1033	.00112	.0723	.00065	.0680	.00062	.0528	.00033	.0502	.00031

Table 3: Error rates (err) and standard deviations (std) for tagging. † (resp. $*$): significantly different from ONLINE error rate above&below (resp. from "u:0" error rate to the left).

plain below why ONLINE is sometimes (slightly) *better* than BATCH, e.g., for ALL and condition l:small/u:big.

3.1 Time course of tagging accuracy

The ONLINE model introduced in this paper has a property that is unique compared to most other work in statistical NLP: its *predictions change* as it tags text because its *representations change.*

To study this time course of changes, we need a large application domain because subtle changes will be too variable in the small test sets of the SANCL TDs. The only labeled domain that is big enough is the WSJ corpus. We therefore reverse the standard setup and train the model on the dev sets of the five SANCL domains (*l:big*) or on the first 5000 labeled words of reviews (*l:small*). In this reversed setup, *u:big* uses the five unlabeled SANCL data sets and Gigaword as before. Since variance of performance is important, we run on 100 randomly selected 50% samples of WSJ and report average and standard deviation of tagging error over these 100 trials.

The results in Table 3[2] show that error rates are only slightly worse for ONLINE than for BATCH or the same. In fact, l:small/u:0 known error rate (.1186) is *lower* for ONLINE than for BATCH (similar to what we observed in Table 2). This will be discussed at the end of this section.

Table 3 includes results for "unseens" as well as unknowns because Schnabel and Schütze (2014) show that unseens cause at least as many errors as unknowns. We define *unseens* as words with a tag that did not occur in training; we compute unseen error rates on *all occurrences* of unseens, i.e., occurrences with both seen and unseen tags are included. As Table 3 shows, the error rate for unknowns is greater than for unseens which is in turn greater than the error rate on known words.

[2]Significance test: test of equal proportion, $p < .05$

Examining the single conditions, we can see that ONLINE fares better than STATIC in 10 out of 12 cases and only slightly worse for l:small/u:big (unseens, known words: .1086 vs .1084, .0802 vs .0801). In four conditions it is significantly better with improvements ranging from .005 (.1404 vs .1451: l:big/u:0, unknown words) to $>$.06 (.3094 vs .3670: l:small/u:0, unknown words).

The differences between ONLINE and STATIC in the other eight conditions are negligible. For the six u:big conditions, this is not surprising: the Gigaword corpus consists of news, so the large unlabeled data set is in reality the same domain as WSJ. Thus, if large unlabeled data sets are available that are similar to the TD, then one might as well use STATIC tagging since the extra work required for ONLINE/BATCH is unlikely to pay off.

Using more labeled data (comparing l:small and l:big) always considerably decreases error rates. We did not test for significance here because the differences are so large. By the same token, using more unlabeled data (comparing u:0 and u:big) also consistently decreases error rates. The differences are large and significant for ONLINE tagging in all six cases (indicated by $*$ in the table).

There is no large difference in variability ONLINE vs. BATCH (see columns "std"). Thus, given that it has equal variability and higher performance, ONLINE is preferable to BATCH since it assumes no dataset available prior to the start of tagging.

Figure 1 shows the time course of tagging accuracy.[3] BATCH and STATIC have constant error rates since they do not change representations during tagging. ONLINE error decreases for unknown words – approaching the error rate of BATCH – as

[3]In response to a reviewer question, the initial (leftmost) errors of ONLINE and STATIC are *not* identical; e.g., ONLINE has a better chance of correctly tagging the very first occurrence of an unknown word because that very first occurrence has a meaningful (as opposed to random) distributed representation.

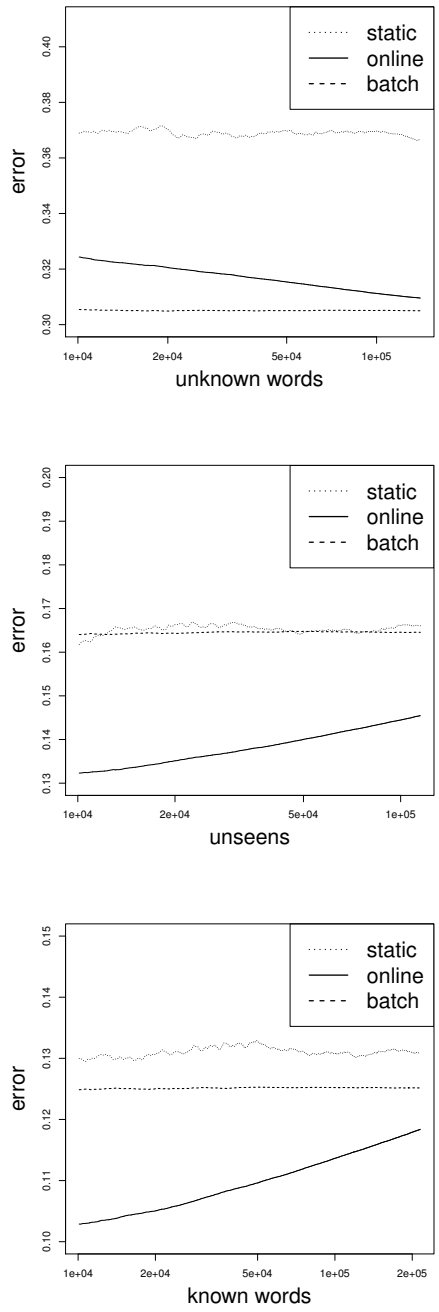


Figure 1: Error rates for unknown words, words with unseen tags and known words for l:small/u:0. The x axis represents the number of tokens of the respective type (e.g., number of tokens of unknown words).

more and more is learned with each additional occurrence of an unknown word (top).

Interestingly, the error of ONLINE *increases* for unseens and known words (middle&bottom panels) (even though it is always below the error rate of BATCH). The reason is that the BATCH update swamps the original training data for l:small/u:0 because the WSJ test set is bigger by a large factor than the training set. ONLINE fares better here because in the beginning of tagging the updates of the distributional representations consist of small increments. We noticed this in Table 2 too: there, ONLINE outperformed BATCH in some cases on KN for l:small/u:big. In future work, we plan to investigate how to weight distributional counts from the target data relative to that from the (labeled und unlabeled) source data.

4 Related work

Online learning usually refers to *supervised* learning algorithms that update the model each time after processing a few training examples. Many supervised learning algorithms are online or have online versions. Active learning (Lewis and Gale, 1994; Tong and Koller, 2001; Laws et al., 2011) is another supervised learning framework that processes training examples – usually obtained interactively – in small batches (Bordes et al., 2005).

All of this work on *supervised online learning* is not directly relevant to this paper since we address the problem of *unsupervised DA*. Unlike online supervised learners, we keep the statistical model unchanged during DA and adopt a representation learning approach: each unlabeled context of a word is used to update its representation.

There is much work on unsupervised DA for POS tagging, including work using constraint-based methods (Subramanya et al., 2010; Rush et al., 2012), instance weighting (Choi and Palmer, 2012), self-training (Huang et al., 2009; Huang and Yates, 2010), and co-training (Kübler and Baucom, 2011). All of this work uses batch learning. For space reasons, we do not discuss supervised DA (e.g., Daumé III and Marcu (2006)).

5 Conclusion

We introduced online updating of word representations, a new domain adaptation method for cases where target domain data are read from a stream and BATCH processing is not possible. We showed that online unsupervised DA performs as well as batch learning. It also significantly lowers error rates compared to STATIC (i.e., no domain adaptation). Our implementation of FLORS is available at `cistern.cis.lmu.de/flors`

Acknowledgments. This work was supported by a Baidu scholarship awarded to Wenpeng Yin and by Deutsche Forschungsgemeinschaft (grant DFG SCHU 2246/10-1 FADeBaC).

References

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *EMNLP*, pages 120–128.

Antoine Bordes, Seyda Ertekin, Jason Weston, and Léon Bottou. 2005. Fast kernel classifiers with online and active learning. *The Journal of Machine Learning Research*, 6:1579–1619.

Jinho D. Choi and Martha Palmer. 2012. Fast and robust part-of-speech tagging using dynamic model selection. In *ACL: Short Papers*, pages 363–367.

Hal Daumé III and Daniel Marcu. 2006. Domain adaptation for statistical classifiers. *Journal of Artificial Intelligence Research*, 26:101–126.

Fei Huang and Alexander Yates. 2010. Exploring representation-learning approaches to domain adaptation. In *DANLP*, pages 23–30.

Zhongqiang Huang, Vladimir Eidelman, and Mary Harper. 2009. Improving a simple bigram HMM part-of-speech tagger by latent annotation and self-training. In *NAACL-HLT: Short Papers*, pages 213–216.

Sandra Kübler and Eric Baucom. 2011. Fast domain adaptation for part of speech tagging for dialogues. In *RANLP*, pages 41–48.

Florian Laws, Christian Scheible, and Hinrich Schütze. 2011. Active learning with Amazon Mechanical Turk. In *Conference on Empirical Methods in Natural Language Processing*, pages 1546–1556.

David D. Lewis and William A. Gale. 1994. A sequential algorithm for training text classifiers. In *ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 3–12.

Robert Parker. 2009. *English gigaword fourth edition*. Linguistic Data Consortium.

Slav Petrov and Ryan McDonald. 2012. Overview of the 2012 shared task on parsing the web. In *Notes of the First Workshop on Syntactic Analysis of Non-Canonical Language (SANCL)*, volume 59.

Alexander M. Rush, Roi Reichart, Michael Collins, and Amir Globerson. 2012. Improved parsing and POS tagging using inter-sentence consistency constraints. In *EMNLP-CoNLL*, pages 1434–1444.

Tobias Schnabel and Hinrich Schütze. 2014. FLORS: Fast and simple domain adaptation for part-of-speech tagging. *TACL*, 2:15–26.

Amarnag Subramanya, Slav Petrov, and Fernando Pereira. 2010. Efficient graph-based semi-supervised learning of structured tagging models. In *EMNLP*, pages 167–176.

Simon Tong and Daphne Koller. 2001. Support vector machine active learning with applications to text classification. *J. Mach. Learn. Res.*, 2:45–66.

Empty Category Detection using Path Features and Distributed Case Frames

Shunsuke Takeno[†], **Masaaki Nagata**[‡], **Kazuhide Yamamoto**[†]
[†]Nagaoka University of Technology,
1603-1 Kamitomioka, Nagaoka, Niigata, 940-2188 Japan
{takeno, yamamoto}@jnlp.org
[‡]NTT Communication Science Laboratories, NTT Corporation,
2-4 Hikaridai, Seika-cho, Soraku-gun, Kyoto, 619-0237 Japan
nagata.masaaki@labs.ntt.co.jp

Abstract

We describe an approach for machine learning-based empty category detection that is based on the phrase structure analysis of Japanese. The problem is formalized as tree node classification, and we find that the path feature, the sequence of node labels from the current node to the root, is highly effective. We also find that the set of dot products between the word embeddings for a verb and those for case particles can be used as a substitution for case frames. Experiments show that the proposed method outperforms the previous state-of the art method by 68.6% to 73.2% in terms of F-measure.

1 Introduction

Empty categories are phonetically null elements that are used for representing dropped pronouns ("pro" or "small pro"), controlled elements ("PRO" or "big pro") and traces of movement ("T" or "trace"), such as WH-questions and relative clauses. They are important for pro-drop languages such as Japanese, in particular, for the machine translation from pro-drop languages to non-pro-drop languages such as English. Chung and Gildea (2010) reported their recover of empty categories improved the accuracy of machine translation both in Korean and in Chinese. Kudo et al. (2014) showed that generating zero subjects in Japanese improved the accuracy of preordering-based translation.

State-of-the-art statistical syntactic parsers had typically ignored empty categories. Although Penn Treebank (Marcus et al., 1993) has annotations on PRO and trace, they provide only labeled bracketing. Johnson (2002) proposed a statistical pattern-matching algorithm for post-processing the results of syntactic parsing based on minimal unlexicalized tree fragments from empty node to its antecedent. Dienes and Dubey (2003) proposed a machine learning-based "trace tagger" as a preprocess of parsing. Campbell (2004) proposed a rule-based post-processing method based on linguistically motivated rules. Gabbard et al. (2006) replaced the rules with machine learning-based classifiers. Schmid (2006) and Cai et al. (2011) integrated empty category detection with the syntactic parsing.

Empty category detection for pro (dropped pronouns or zero pronoun) has begun to receive attention as the Chinese Penn Treebank (Xue et al., 2005) has annotations for pro as well as PRO and trace. Xue and Yang (2013) formalized the problem as classifying each pair of the location of empty category and its head word in the dependency structure. Wang et al. (2015) proposed a joint embedding of empty categories and their contexts on dependency structure. Xiang et al. (2013) formalized the problem as classifying each IP node (roughly corresponds to S and SBAR in Penn Treebank) in the phrase structure.

In this paper, we propose a novel method for empty category detection for Japanese that uses conjunction features on phrase structure and word embeddings. We use the Keyaki Treebank (Butler et al., 2012), which is a recent development. As it has annotations for pro and trace, we show our method has substantial improvements over the state-of-the-art machine learning-based method (Xiang et al., 2013) for Chinese empty category detection as well as linguistically-motivated manually written rule-based method similar to (Campbell, 2004).

2 Baseline systems

The Keyaki Treebank annotates the phrase structure with functional information for Japanese sentences following a scheme adapted from the Annotation manual for the Penn Historical Corpora and

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1335–1340,
Lisbon, Portugal, 17-21 September 2015.

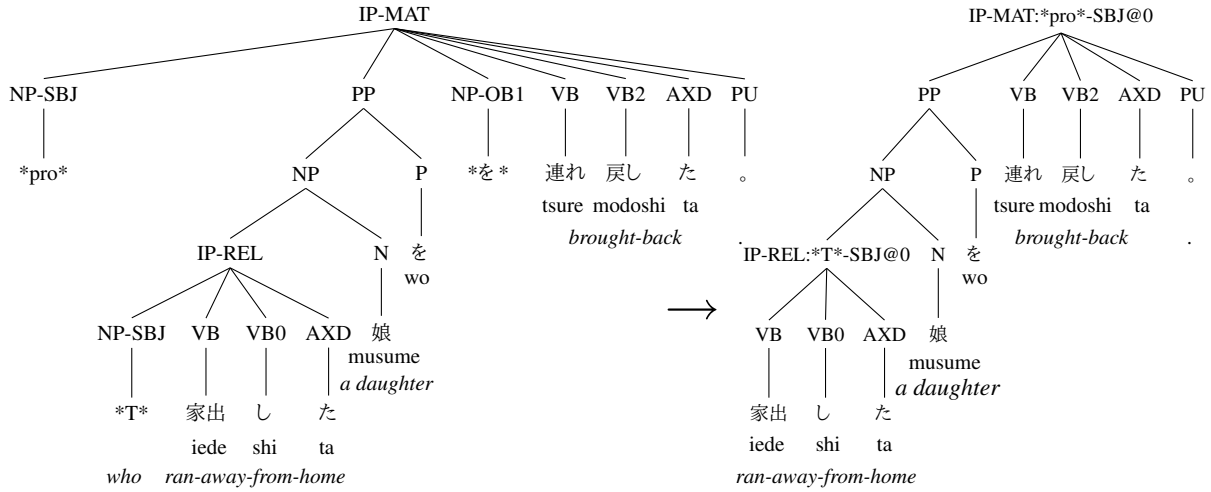


Figure 1: An annotation example of 家出した娘を 連れ戻した。*(*pro* brought back a daughter who ran away from home.)* in Keyaki Treebank. (The left tree is the original tree and the right tree is a converted tree based on Xiang et al.'s (2013) formalism)

the PCEEC (Santorini, 2010). There are some major changes: the VP level of structure is typically absent, function is marked on all clausal nodes (such as IP-REL and CP-THT) and all NPs that are clause level constituents (such as NP-SBJ). Disambiguation tags are also used for clarifying the functions of its immediately preceding node, such as NP-OBJ *を *(wo) for PP, however, we removed them in our experiment.

Keyaki Treebank has annotation for trace markers of relative clauses (*T*) and dropped pronouns (*pro*), however, it deliberately has no annotation for control dependencies (PRO) (Butler et al., 2015). It has also fine grained empty categories of *pro* such as *speaker* and *hearer*, but we unified them into *pro* in our experiment.

HARUNIWA(Fang et al., 2014) is a Japanese phrase structure parser trained on the treebank. It has a rule-based post-processor for adding empty categories, which is similar to (Campbell, 2004). We call it RULE in later sections and use it as one of two baselines.

We also use Xiang et al's (2013) model as another baseline. It formulates empty category detection as the classification of IP nodes. For example, in Figure 1, empty nodes in the left tree are removed and encoded as additional labels with its position information to IP nodes in the right tree. As we can uniquely decode them from the extended IP labels, the problem is to predict the labels for the input tree that has no empty nodes.

Let $T = t_1 t_2 \cdots t_n$ be the sequence of nodes produced by the post-order traversal from root node, and e_i be the empty category tag associated with t_i. The probability model of (Xiang et al., 2013) is formulated as MaxEnt model:

$$P(e_1^n|T) = \prod_{i=1}^{n} P(e_i|e_1^{i-1}, T) = \prod_{i=1}^{n} \frac{\exp(\boldsymbol{\theta} \cdot \boldsymbol{\phi}(e_i, e_1^{i-1}, T))}{Z(e_1^{i-1}, T)} \quad (1)$$

where $\boldsymbol{\phi}$ is a feature vector, $\boldsymbol{\theta}$ is a weight vector to $\boldsymbol{\phi}$ and Z is normalization factor:

$$Z(e_1^{i-1}, T) = \sum_{e \in \mathcal{E}} \exp(\boldsymbol{\theta} \cdot \boldsymbol{\phi}(e, e_1^{i-1}, T))$$

where $\mathcal{E}$ represents the set of all empty category types to be detected.

Xiang et al. (2013) grouped their features into four types: tree label features, lexical features, empty category features and conjunction features as shown in Table 1. As the features for (Xiang et al., 2013) were developed for Chinese Penn Treebank, we modify their features for Keyaki Treebank: First, the traversal order is changed from post-order (bottom-up) to pre-order (top-down). As PROs are implicit in Keyaki Treebank, the decisions on IPs in lower levels depend on those on higher levels in the tree. Second, empty category features are extracted from ancestor IP nodes, not from descendant IP nodes, in accordance with the first change.

Table 2 shows the accuracies of Japanese empty category detection, using the original and our modification of the (Xiang et al., 2013) with ablation test. We find that the conjunction features

Tree label features	
1	current node label
2	parent node label
3	grand-parent node label
4	left-most child label or POS tag
5	right-most child label or POS tag
6	label or POS tag of the head child
7	the number of child nodes
8	one level CFG rule
9	left-sibling label or POS tag (up to two siblings)
10	right-sibling label or POS tag (up to two siblings)
Lexical features	
11	left-most word under the current node
12	right-most word under the current node
13	word immediately left to the span of the current node
14	word immediately right to the span of the current node
15	head word of the current node
16	head word of the parent node
17	is the current node head child of its parent? (binary)
Empty category features	
18	predicted empty categories of the left sibling
19*	the set of detected empty categories of ancestor nodes
Conjunction features	
20	current node label with parent node label
21*	current node label with features computed from ancestor nodes
22	current node label with features computed from left-sibling nodes
23	current node label with lexical features

Table 1: List of features of Xiang et al.'s (2013). (* indicates the features we changed for the Keyaki Treebank)

Features	F(Gold)	Δ
original (Xiang et al., 2013)	68.2	−0.40
modified (Xiang et al., 2013)	68.6	-
− Tree label	68.6	−0.00
− Empty category	68.3	−0.30
− Lexicon	68.6	−0.00
− Conjunction	58.5	−10.1

Table 2: Ablation result of (Xiang et al., 2013)

are highly effective compared to the three other features. This observation leads to the model proposed in the next section.

3 Proposed model

In the proposed model, we use combinations of path features and three other features, namely head word feature, child feature and empty category feature. Path feature (PATH) is a sequence of non-terminal labels from the current node to the ancestor nodes up to either the root node or the nearest CP node. For example, in Figure 1, if the current node is IP-REL, four paths are extracted; IP-REL, IP-REL → NP, IP-REL → NP → PP and IP-REL → NP → PP → IP-MAT.

Head word feature (HEAD) is the surface form of the lexical head of the current node. Child feature (CHILD) is the set of labels for the children of the current node. The label is augmented with the surface form of the rightmost terminal node if it is a function word. In the example of Figure 1, if the current node is IP-MAT, HEAD is 連れ (tsure) and CHILD includes: PP-を (wo), VB, VB2, AXD-た (ta) and PU-。. Empty category feature (EC) is a set of empty categories detected in the ancestor IP nodes. For example in Figure 1, if the current node is IP-REL, EC is *pro*.

We then combine the PATH with others. If the current node is the IP-MAT node in right-half of Figure 1, the combination of PATH and HEAD is:IP-MAT× 連れ (tsure) and the combinations of PATH and CHILD are: IP-MAT×PP-を (wo), IP-MAT×VB, IP-MAT×VB2, IP-MAT×AXD-た (ta) and IP-MAT×PU-。.

3.1 Using Word Embedding to approximate Case Frame Lexicon

A case frame lexicon would be obviously useful for empty category detection because it provides information on the type of argument the verb in question takes. The problem is that case frame lexicon is not usually readily available. We propose a novel method to approximate case frame lexicon for languages with explicit case marking such as Japanese using word embeddings. According to (Pennington et al., 2014), they designed their embedding model GloVe so that the dot product of two word embeddings approximates the logarithm of their co-occurrence counts. Using this characteristic, we can easily make a feature that approximate the case frame of a verb. Given a set of word embeddings for case particles $q_1, q_2, \cdots, q_N \in Q$, the distributed case frame feature (DCF) for a verb w_i is defined as:

$$\tilde{v_i} = w_i \cdot (q_1, q_2, \cdots, q_N) \quad (2)$$

$$v_i = \frac{\tilde{v_i}}{||\tilde{v_i}||} \quad (3)$$

In our experiment, we used a set of high frequency case particles が (ga), は (ha), も (mo), の (no), を (wo), に (ni), へ (he) and から (kara) as Q.

4 Experiment

4.1 Dataset

We divided the Keyaki Treebank into training, development and test sets. As of May 8, 2015, there

		ALL	development/test		
			transcript	blog	newswire
#pro	SBJ	13343	598	187	346
	OB1	1568	43	1	27
	OB2	59	2	0	0
#T	ADT	97	0	0	8
	LOC	164	9	7	7
	OB1	755	0	11	35
	OB2	15	0	1	2
	SBJ	3788	5	40	266
	TMP	53	0	0	3
	MSR	14	0	1	4
	TPC	10	0	0	0
	char/sent.	32.0	32.1	69.5	96.5
	#sent.	22649	591	109	303
	#IP node	46684	1129	544	1841

Table 3: Statistics of Keyaki Treebank

are 22,639 sentences in Keyaki Treebank. We used 1,000 sentences as the development set, 1,003 sentences as the test set. They were taken from the files blog_KNB.psd (blog), spoken_CIAIR.psd (transcript), newswire_MAINICHI-1995.psd (newswire) to balance the domain. The remaining 20,646 sentences are used for training. Further statistics are shown in Table 3.

We used GloVe as word embedding, Wikipedia articles in Japanese as of January 18, 2015, are used for training, which amounted to 660 million words and 23.4 million sentences. By using the development set, we set the dimension of word embedding and the window size for co-occurrence counts as 200 and 10, respectively.

4.2 Result and Discussion

We tested in two conditions: gold parse and system parse. In gold parse condition, we used the trees of Keyaki Treebank without empty categories as input to the systems. In system parse condition, we used the output of the Berkeley Parser model of HARUNIWA before rule-based empty category detection[1]. We evaluated them using the word-position-level identification metrics described in (Xiang et al., 2013). It projects the predicted empty category tags to the surface level. An empty node is regarded as correctly predicted surface position in the sentence, type (T or pro) and function (SBJ, OB1 and so on) are matched with the reference.

To evaluate the effectiveness of the proposed distributed case frame (DCF), we used an existing case frame lexicon (Kawahara and Kurohashi, 2006) and tested three different ways of encoding the case frame information: BIN encodes each case as binary features. SET encodes each combination of required cases as a binary feature. DIST is a vector of co-occurrence counts for each case particle, which can be thought of an unsmoothed version of our DCF.

Table 4 shows the accuracies of various empty category detection methods, for both gold parse and system parse. In the gold parse condition, the two baselines, the rule-based method (RULE) and the modified (Xiang et al., 2013) method, achieved the F-measure of 62.6% and 68.6% respectively.

We also implemented the third baseline based on (Johnson, 2002). Minimal unlexicalized tree fragments from empty node to its antecedent were extracted as pattern rules based on corpus statistics. For *pro*, which has no antecedent, we used the statistics from empty node to the root. Although the precision of the method is high, the recall is very low, which results in the F-measure of 38.1%.

Among the proposed models, the combination of path feature and child feature (PATH $\times$ CHILD) even outperformed the baselines. It reached 73.2% with all features. As for the result of system-parse condition, the F-measure dropped considerably from 73.2% to 54.7% mostly due to the parsing errors on the IP nodes and its function.

We find that there are no significant differences among the different encodings of the case frame lexicon, and the improvement brought by the proposed distributed case frame is comparable to the existing case frame lexicon.

Table 5 shows the ablation result of the proposed model. It indicates conjunction between PATH and CHILD feature is most effective.

	F(Gold)	Δ	F(System)	Δ
Proposed	72.1	-	53.9	-
$-$CHILD	47.4	-24.7	33.7	-20.2
$-$EC	70.8	-1.3	52.4	-1.5
$-$HEAD	70.0	-2.1	51.6	-2.3

Table 5: Ablation result of PATH $\times$ (CHILD + EC + HEAD) model

[1]There are two models available in HARUNIWA, namely the BitPar model (Schmid, 2004) and Berkeley Parser binary branching model (Petrov and Klein, 2007). The output of the later is first flattened, then added disambiguation tags and empty categories using tsurgeon script (Levy and Andrew, 2006).

Models		Gold parse			System parse			
	CF	P	R	F	P	R	F	#nonZ
RULE	-	54.4	73.7	62.6	57.4	50.5	53.7	-
modified (Xiang et al., 2013)	-	76.8	62.0	68.6	57.4	46.9	51.6	321k
modified (Johnson, 2002)	-	81.3	25.0	38.1	66.8	18.1	28.6	-
PATH × CHILD	-	71.0	67.7	69.3	55.9	49.1	52.3	108k
PATH × (CHILD + HEAD + EC)	-	74.8	69.5	72.1	56.2	51.8	53.9	123k
PATH × (CHILD + HEAD + EC)	+DCF	78.0	68.9	73.2	59.7	50.5	54.7	124k
PATH × (CHILD + HEAD + EC)	+BIN	77.1	70.2	73.5	58.8	51.6	55.0	124k
PATH × (CHILD + HEAD + EC)	+SET	77.5	70.0	73.6	58.5	51.4	54.7	126k
PATH × (CHILD + HEAD + EC)	+DIST	77.5	68.3	72.6	60.4	50.6	55.1	124k

Table 4: Result of our models with baselines. #nonZ means the amount of non-zero weight of model

5 Conclusion

In this paper, we proposed a novel model for empty category detection in Japanese using path features and the distributed case frames. Although it achieved fairly high accuracy for the gold parse, there is much room for improvement when applied to the output of a syntactic parser. Since the accuracy of the empty category detection implemented as a post-process highly depends on that of the underlying parser, we want to explore models that can solve them jointly, such as the lattice parsing approach of (Cai et al., 2011). We would like to report the results in the future version of this paper.

References

Alastair Butler, Tomoko Hotta, Ruriko Otomo, Kei Yoshimoto, Zhen Zhou, and Hong Zhu. 2012. Keyaki Treebank : phrase structure with functional information for Japanese. In *Proceedings of Text Annotation Workshop*.

Alastair Butler, Shota Hhiyama, and Kei Yoshimoto. 2015. Coindexed null elements for a Japanese parsed corpus. In *Proceedings of the 21th Annual Meeting of the Association for Natural Language Processing*, pages 708–711.

Shu Cai, David Chiang, and Yoav Goldberg. 2011. Language-Independent Parsing with Empty Elements. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers*, volume 2, pages 212–216.

Richard Campbell. 2004. Using linguistic principles to recover empty categories. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 645–652.

Tagyoung Chung and Daniel Gildea. 2010. Effects of Empty Categories on Machine Translation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 636–645.

Péter Dienes and Amit Dubey. 2003. Deep Syntactic Processing by Combining Shallow Methods. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 431–438.

Tsaiwei Fang, Alastair Butler, and Kei Yoshimoto. 2014. Parsing Japanese with a PCFG treebank grammar. In *Proceedings of The Twentieth Meeting of the Association for Natural Language Processing*, volume C, pages 432–435.

Ryan Gabbard, M Marcus, and Seth Kulick. 2006. Fully Parsing the Penn Treebank. In *Proceedings of the main conference on Human Language Technology of the North American Chapter of the Association of Computational Linguistics*, pages 184–191.

Mark Johnson. 2002. A simple pattern-matching algorithm for recovering empty nodes and their antecedents. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 136–143.

Daisuke Kawahara and Sadao Kurohashi. 2006. Case frame compilation from the web using high-performance computing. In *Proceedings of the 5th International Conference on Language Resources and Evaluation*, pages 1344–1347.

Taku Kudo, Hiroshi Ichikawa, and Hideto Kazawa. 2014. A joint inference of deep case analysis and zero subject generation for Japanese-to-English statistical machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 557–562.

Roger Levy and Galen Andrew. 2006. Tregex and Tsurgeon: tools for querying and manipulating tree data structures. In *Proceedings of 5th International Conference on Language Resources and Evaluation*, pages 2231–2234.

Mitchell P Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a Large Annotated

Corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

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

Slav Petrov and Dan Klein. 2007. Improved Inferencing for Unlexicalized Parsing. In *Proceedings of HLT-NAACL 2007 - Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 404–411.

Santorini. 2010. Annotation Manual for the Penn Historical Corpora and the PCEEC (Release 2).

Helmut Schmid. 2004. Efficient parsing of highly ambiguous context-free grammars with bit vectors. In *Proceedings of the 20th International Conference on Computational Linguistics*, pages 162–168.

Helmut Schmid. 2006. Trace Prediction and Recovery with Unlexicalized PCFGs and Slash Features. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 177–184.

Bing Xiang, Xiaoqiang Luo, and Bowen Zhou. 2013. Enlisting the ghost: Modeling empty categories for machine translation. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 822–831.

Nianwen Xue and Yaqin Yang. 2013. Dependency-based empty category detection via phrase structure trees. In *Proceedings of HLT-NAACL 2013, Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings of the Main Conference*, pages 1051–1060.

Naiwen Xue, Fei Xia, Fu-Dong Chiou, and Marta Palmer. 2005. The Penn Chinese TreeBank: Phrase structure annotation of a large corpus. *Natural Language Engineering*, 11(2):207–238.

Foreebank: Syntactic Analysis of Customer Support Forums

Rasoul Kaljahi[1], **Jennifer Foster**[1] **Johann Roturier**[2]
Corentin Ribeyre[3], **Teresa Lynn**[1], **Joseph Le Roux**[4]
[1]ADAPT Centre, School of Computing, Dublin City University, Ireland
{rkaljahi, jfoster, tlynn}@computing.dcu.ie
[2]Symantec Research Labs, Dublin, Ireland
johann_roturier@symantec.com
[3]Alpage, INRIA, Univ Paris Diderot, Sorbonne Paris Cité, France
corentin.ribeyre@inria.fr
[4]Université Paris Nord, France
joseph.le.roux@gmail.com

Abstract

We present a new treebank of English and French technical forum content which has been annotated for grammatical errors and phrase structure. This double annotation allows us to empirically measure the effect of errors on parsing performance. While it is slightly easier to parse the corrected versions of the forum sentences, the errors are not the main factor in making this kind of text hard to parse.

1 Introduction

The last five years has seen a considerable amount of research carried out on web and social media text parsing, with new treebanks being created (Foster et al., 2011; Seddah et al., 2012; Mott et al., 2012; Kong et al., 2014), and new parsing systems developed (Petrov and McDonald, 2012; Kong et al., 2014). In this paper we explore a particular source of user-generated text, namely, posts from technical support forums, which are a popular means for customers to resolve their queries about a product. An accurate parser for this kind of text can be used to inform forum-level question-answering, machine translation and quality estimation of machine translation.

We create a 2000-sentence treebank called *Foreebank* which contains sentences from the Symantec Norton English and French technical support forums.[1] The phrase structure of the sentences is annotated *and* any grammatical errors are marked in the trees. Marking the grammatical errors allows us to precisely measure the amount of grammatical noise in this kind of text, and joint error and syntactic annotation allows us to determine its effect on parsing.

Foster (2010) explored the effect of spelling errors on parsing performance of conversational forum text. We extend this study to include grammatical errors, focusing on more technical content. Foster et al. (2008) explored the effect of artificially generated grammatical errors on Wall Street Journal parsing. We concentrate on forum text rather than newspaper text, and, crucially, examine the effect of *real* grammatical errors. We find that the level of grammatical noise is lower than expected, with capitalisation and punctuation errors being the most frequent. While correcting all the errors does result in a performance increase of 1.5% for English and 0.8% for French, the major challenge in parsing these sentences seems not to be "bad language" (Eisenstein, 2013) per se.

The main contribution of the paper is the Foreebank data set itself[2] but we also carry out preliminary parsing experiments evaluating the accuracy of a PCFG-LA parser on Foreebank, examining the effect of grammatical errors on parsing and experimenting with different training sets.

2 Related Work

Other treebanks of English web text include the English Web Treebank (aka Google Web Treebank) (Mott et al., 2012), the small treebank of tweets and football discussion forum posts described in Foster et al. (2011) and the tweet dependency bank described in Kong et al. (2014). The English Web Treebank is a corpus of over 250K words, selected from blogs, newsgroups, emails, local business reviews and Yahoo! answers. It adapts the Penn Treebank (Marcus et al., 1994) and Switchboard (Taylor, 1996) annotation guidelines to address the phenomena specific

[1]http://community.norton.com

[2]www.computing.dcu.ie/mt/confidentmt.html

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1341–1347,
Lisbon, Portugal, 17-21 September 2015.

to this type of text. The annotation of the 1000-sentence treebank described in Foster et al. (2011) is based on the Penn Treebank, whereas the annotation of the treebank described in Kong et al. (2014) is dependency-based. The *French Social Media Bank* developed by Seddah et al. (2012) is a treebank of 1,700 French sentences from various type of social media including Facebook, Twitter and discussion forums (video game and medical). An extended version of the FTB-UC annotation guidelines (Candito and Crabbé, 2009) is employed during annotation and subcorpora containing particularly noisy utterances are identified.

The main difference between Foreebank and other web/social media treebanks is that grammatical errors in the Foreebank sentences are marked and corrected as part of the annotation process. Error annotation not only provides more insight into this type of text but it also enables us to directly measure the effect of these errors on parsing accuracy and leaves open the possibility of performing joint parsing and error detection by directly learning the error annotation during parser training.

A learner corpus (Granger, 2008) contains utterances produced by language learners and serves as a resource for second language acquisition, computational linguistic and computer-aided language learning research. Examples include the International Corpus of Learner English (Granger, 1993), the Cambridge Learner Corpus (Nicholls, 1999; Yannakoudakis et al., 2011), the NUS Corpus of Learner English (Dahlmeier et al., 2013) and the German Falko corpus (Lüdeling, 2008; Rehbein et al., 2012). We can compare Foreebank to a learner corpus since both contain utterances that are potentially ungrammatical and because in a learner corpus the errors are often annotated, as they are in Foreebank. In the last five years, there have been several shared tasks in grammatical error correction including the Helping Our Own (HOO) shared tasks of 2011 and 2012 (Dale and Kilgariff, 2011; Dale et al., 2012), and the CoNLL 2013 and 2014 shared tasks (Ng et al., 2013; Ng et al., 2014). With the exception of HOO 2011, all shared tasks involve error-annotated sentences from learner corpora. Annotation schemes vary but most involve marking the span of an error, classifying the error according to some taxonomy designed with L2 utterances in mind, and sometimes providing the correction or "target hypothesis" (Hirschmann et al., 2007).

Regarding syntactic annotation of learner data, Dickinson and Ragheb (2009) propose a dependency annotation scheme based on the CHILDES scheme (Sagae et al., 2007) developed for first language learners. They treat the developing language of learners as an interlanguage, as suggested by Díaz-Negrillo et al. (2010), and annotate it as is. They use two POS tags and two dependency labels for error cases: one for the surface form and one for the intended form. Rosén and De Smedt (2010) criticise the approach of Dickinson and Ragheb (2009) involving "annotating language text as is" arguing that interpretation of the language is required at all annotation levels. They use NorGram, a Norwegian Lexical-Functional Grammar, to annotate a learner corpus with constituency structure, functional structure and semantic structure, in order to provide a means to search for contexts in which learner errors occur. Nagata et al. (2011) describe an English learner corpus which has been manually annotated with shallow syntax, introducing two new POS tags and two new chunk labels for errors.

3 Building the Foreebank

The Foreebank treebank contains 1000 English sentences and 1000 French sentences. The English sentences come from the Symantec Norton technical support user forum. Half of the French sentences come from the French Norton forum and the other half are human translations of sentences from the English forum. Four annotators were involved in the annotation process. Their main task was to correct automatically parsed phrase structure trees using an annotation tool developed for this project.[3] The English annotators were guided by the Penn Treebank bracketing guidelines and a Foreebank-adapted version of the English Web Treebank bracketing guidelines. The French annotators used the French treebank (FTB) (Abeillé et al., 2003) guidelines, following the SPMRL strategy for multiword expressions (Seddah et al., 2013; Candito and Crabbé, 2009). The two primary annotators, one for French and one for English, annotated all the data for their language. The two secondary annotators annotated a 100-sentence subset. Inter-annotator agreement was calculated by measuring the Parseval

[3]The Stanford parser (Klein and Manning, 2003) was used to parse the English data and the Berkeley parser (Petrov et al., 2006) was used for the French sentences.

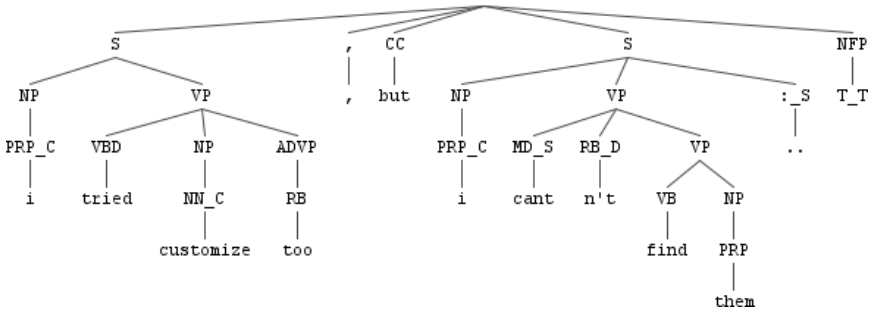


Figure 1: The Foreebank annotation of *i tried customize too , but i cant find them .. T_T* corrected as *I tried Customize too , but I ca n't find them ... T_T*

Suffix	Explanation	Example		FB_{en}		FB_{fr}	
		English	French	#	%	#	%
_D	Deleted token	It fixed [*the*] problem.	Cela a résolu [*le*] problème.	170	1.10	56	0.29
_X	Extraneous tokens	It fixed *the* my problem.	Cela a résolu *le* mon problème.	35	0.23	17	0.09
_W	Wrong form error	It *fix* the problem.	Cela *résoudre* le problème.	69	0.45	43	0.22
_S	Misspelled token	It fixed my *prbolem*.	Cela a résolu mon *prbolème*.	81	0.53	117	0.60
_C	Capitalisation error	*it* fixed my problem.	*cela* a résolu mon problème.	161	1.00	194	1.00
_B	Broken token	It *fix ed* my problem.	Cela a *réso lu* mon problème.	2	0.01	12	0.06
_I	Innovative initialism	I have problem *w/* this software.	J'ai un problème *av.* ce logiciel.	1	0.01	7	0.04
_M	Merged sentences	It fixed the problem Thank you.	Cela a résolu le problème Merci.	3	0.30	29	2.90

Table 1: Foreebank Error Suffixes. The last two columns refer to their frequency.

F1 of trees produced by the secondary annotators against those produced by the primary annotators. For English this was 88 and for French it was 86.7.

Prior to correcting a parse tree produced by the automatic parser, the annotators are asked to correct any errors they find in the sentence.[4] The corrected text is entered in a field of the annotation tool. As part of the syntactic annotation process, errors are marked by appending an error suffix to the preterminals of the affected words in the tree. The error suffixes used in Foreebank are listed in Table 1 and an example tree from Foreebank is shown in Figure 1. There are three kinds of substitution error suffixes: `C` for marking problems with capitalisation, `S` for marking spelling errors and `W` for marking the wrong form of a word which encompasses inflection errors (*they* instead of *them*), real-word spelling errors (*test* instead of *text*) and lexical choice errors (*desk* instead of *chair*). The POS tag of the corrected form is used in the tree instead of the POS tag of the incorrect form.[5] Although this annotation scheme contains fewer error types than the taxonomies used for learner corpora, its granularity increases when the error suffixes are interpreted in the syntactic context in which they occur. For example, we can distinguish a missing determiner (`DT_D`) from a missing preposition (`IN_D`).

The "sentences" that the annotators see are the result of passing the forum text through an automatic sentence splitter (NLTK[6]) and tokeniser (in-house). This is another important difference between Foreebank and the English Web Treebank (EWT). In the EWT, sentence boundary detection and tokenisation has been carried out manually before annotation. Both approaches are valid but ours was chosen in order to stay closer to the more realistic scenario of less than perfect automatic preprocessing tools. This means that annotators have a special class of errors that result from noisy sentence splitting and tokenisation that must be marked during annotation.

There are two types of sentence splitting errors: merged sentences such as (1) in which a sentence boundary was not detected before the word *When* due to the use of a comma instead of a full stop, and split sentences such as (2).

(1) `7. Combofix will start, **When** it is scanning don't move the mouse cursor inside the box,`

(2) `The questions to <CompanyName>:`

[4] Minimal correction is encouraged to prevent annotators from rewriting the sentence in their preferred writing style. Instead they are instructed to just focus on fixing the errors.

[5] An alternative would have been to use one POS tag for the erroneous form and one for the corrected form, either combined a la Nagata et al. (2011) or separate a la Dickinson and Ragheb (2009).

[6] `http://www.nltk.org/`

Merged sentences are gathered under one root node with the error suffix `M` (e.g. `S_M`) , and split sentences are annotated as if they are standalone.

Tokenisation problems can also be categorised as merged (3) or split (4 and 5). Merged tokens are treated as a combination of a spelling error (*whenI* instead of *when*) and a deleted token (*I*). When the split is morphological as in (4), they are tagged with the POS tag of the whole intended token, along with the error suffix `B` (for "broken"). So in (4), the POS tag of *anti* would be annotated as `NN_B` and the POS tag of *vir* as `NN_B`. When there is no such clean morphological break (as in (5)), the first token is treated as a spelling error and the second as an extraneous token.

(3) **whenI** tried to use ...

(4) he should buy **anti vir** programs

(5) **i t** keeps causing <*ProductName*> to lock up ...

4 Analysing the Foreebank

Table 2 presents the average and the maximum sentence length in Foreebank, and, for comparison, WSJ and FTB. It also gives the out-of-vocabulary (OOV) rate of these data sets with respect to the WSJ and FTB. The Foreebank sentences are shorter on average than the WSJ and FTB sentences. The table also shows that the OOV rate of Foreebank with respect to WSJ/FTB is high: 33.3% for English and 39.1% for French. These numbers can be compared to the OOV rate of the WSJ test set with respect to its training set which is 13.2% and the FTB which is 20.6%. The higher OOV rate for the French Foreebank compared to the English is most likely due to the larger size of the WSJ compared to the FTB. The OOV rate of the English Foreebank is more than 2.5 times as large as that of the WSJ test set, while the OOV rate of the French Foreebank is less than 2 times as large as that of the FTB test set. This suggests that a bigger performance drop due to unknown words should be expected in parsing the English Foreebank sentences than the French.

The last four columns in Table 1 display the absolute and relative frequency of each error suffix. In sum, it seems that capitalisation is the major error type in Foreebank especially in the French data. Deleted tokens are also a major source of problem on the English side. Most of the capitalisation errors involve proper nouns (e.g. product names) and most of the deleted tokens are cases

	FB$_{en}$	WSJ	FB$_{fr}$	FTB
Avg. sentence length	15.4	23.8	19.6	28.4
Max. sentence length	89	141	86	260
OOV rate (%)	31.6	-	33.6	-

Table 2: Characteristics of the English (FB$_{en}$) and French (FB$_{fr}$) Foreebank compared with those of the WSJ and FTB.

English	FB$_{en}$ All	WSJ Test	**French**	FB$_{fr}$ All	FTB Test
WSJ$_{all}$	77.0	-	FTB$_{all}$	76.3	-
WSJ$_{train}$	75.4	89.6	FTB$_{train}$	76.0	81.3

Table 3: Foreebank and WSJ/FTB test set results.

of missing punctuation. Overall, the errors occur on only a small fraction of the tokens in both data sets. We also calculate the edit distance between each Foreebank sentence and its correction by summing the number of error suffixes and dividing by the maximum of the original and corrected sentence lengths. The average edit distance for the English section of Foreebank is 0.04 and for the French section is 0.03. Despite the existence of some near-to-incomprehensible sentences, the overall error level is very low.

5 Parsing the Foreebank

We first evaluate newswire-trained parsers on Foreebank, using our in-house PCFG-LA parser with the max-rule parsing algorithm (Petrov and Klein, 2007) and 6 split-merge cycles. The English model is trained on the entire WSJ and the French model on the entire FTB. For comparison, we parse the WSJ/FTB and so we additionally use models trained only on the training sections. We remove the error suffixes and any D-suffixed nodes (representing deleted words) from the gold Foreebank trees before evaluation. The results are shown in Table 3. As expected, we see a significant drop for both languages when we move from in-domain data to Foreebank. Compared to parsing the English side of Foreebank, the performance drop for French is relatively smaller: the former drops 14.2 points from 89.6 F_1 points to 75.4 and the latter 5.3 points from 81.3 to 76. This suggests that, either the French parsing model is better generalisable to the forum text, or alternatively, that the FTB test set is more distant from its training set than the WSJ one. The second hypoth-

esis is more likely because 1) it is on par with the OOV rate observed in Section 4, and 2) the performance of the English and French parsers are close on Foreebank but further apart on the newswire test sets. The effect of using the entire WSJ and FTB instead of only their training sections is also worth noting. While adding the WSJ development and test sets (about 5,500 sentences, a 14% increase) improves the F_1 of English parsing by 1.6 points, the 2,500 FTB development and test sentences (a 25% increase) have little effect on the French parsing, suggesting that either these new sentences are still not enough or do not bring additional information to the parsing model.

Since the annotators correct the errors made by the forum users, we are able to parse the corrected versions of the Foreebank sentences and examine how accurately they are parsed compared to the original sentences. We use the WSJ_{all} and FTB_{all} parsing models described above. Correcting the user errors before parsing leads to an improved parsing F_1 of 78.6 for the English sentences, an increase of 1.6 points (2%). A smaller impact is observed on the French sentences where the edited sentences receive an F_1 of 77.1 (an increase of 0.8 points). Referring to the distribution of error suffixes in Table 1, this suggests that the inserted and deleted tokens may have a larger effect on parser error than the substituted tokens, as their number is higher for English. Many substitution errors are capitalisation errors, typically involving a confusion between proper and common nouns, which tends not to affect the surrounding tree.

The simplest method to improve the accuracy of parsing Foreebank is to use it as supplementary training data. We do this using a 5-fold cross validation, in which Foreebank is randomly split into five parts, with each part used for the evaluation of the parsers trained on WSJ/FTB plus the other four parts. The results are shown in Table 4. Combining the larger treebank and Foreebank improves the F_1 by 2.6 points for English and 3.2 for French. Considering that Foreebank is orders of magnitude smaller than the WSJ/FTB, these gains are encouraging. We try to overcome the small size of Foreebank by 1) using the EWT as training data, and 2) increasing the weight of Foreebank by training on multiple copies of it. The EWT is not a substitute for the WSJ but it does provide a modest improvement when used in conjunction with Foreebank and WSJ. The replication of Foreebank trees has mixed results, providing a 0.5 point improvement for English and none for French.

English		**French**	
Training Set	F_1	Training Set	F_1
WSJ_{all}	77.0	FTB_{all}	76.3
WSJ_{all}+FB_{en}	79.6	FTB_{all}+FB_{fr}	**79.5**
WSJ_{all}+5FB_{en}	80.1	FTB_{all}+5FB_{fr}	**79.5**
EWT	75.0	-	-
EWT+FB_{en}	79.0	-	-
WSJ_{all}+FB_{en}+EWT	**80.3**	-	-
FB_{en}	71.1	FB_{fr}	72.4
FB_{en_suf}	70.2	FB_{fr_suf}	71.8

Table 4: Training on Foreebank/WSJ/EWT/FTB and testing on Foreebank

In all experiments up to now, we have excluded the error suffixes from the Foreebank trees (during training and testing). We next try to directly learn trees containing the error suffixes (except for deleted tokens). That is, we use the original Foreebank trees containing the error suffixes for training and evaluate against Foreebank trees containing the error suffixes. The second last row of Table 4 shows the 5-fold CV results when the version of Foreebank without the error suffixes is used for training and the last row the results when the error suffixes are included. Including the suffixes decreases the accuracy, most likely due to the increased data sparsity caused by the suffixed tags.

6 Conclusion

We have introduced a treebank of technical forum sentences for English and French, based on an annotation strategy adapted to suit user-generated text in a realistic NLP setting. By marking the errors on the trees, we studied their prevalence as well as their impact on parsing and found that despite their low frequency, they do negatively affect parser performance, while not being the most important factor. Our next steps include learning error suffixes during a prior tagging phase and experimenting with the French Social Media Bank.

Acknowledgments

This research has been supported by the Irish Research Council Enterprise Partnership Scheme (EPSPG/2011/102) and the Science Foundation Ireland (Grant 12/CE/I2267) as part of CNGL (www.cngl.ie) at Dublin City University.

References

Anne Abeillé, Lionel Clément, and François Toussenel. 2003. Building a Treebank for French. In *Treebanks: Building and Using Syntactically Annotated Corpora*, pages 165–187. Kluwer Academic Publishers.

Marie Candito and Benoît Crabbé. 2009. Improving Generative Statistical Parsing with Semi-supervised Word Clustering. In *Proceedings of IWPT*, pages 138–141.

Daniel Dahlmeier, Hwee Tou Ng, and Siew Mei Wu. 2013. Building a large annotated corpus of learner English: The NUS Corpus of Learner English. In *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 22–31.

Robert Dale and Adam Kilgariff. 2011. Helping Our Own: The HOO 2011 Pilot Shared Task. In *Proceedings of the 13th European Workshop on Natural Language Generation (ENLG)*, pages 242–249.

Robert Dale, Ilya Anisimoff, and George Narroway. 2012. HOO 2012: A Report on the Preposition and Determiner Error Correction Shared Task. In *Proceedings oftThe 7th Workshop on the Innovative Use of NLP for Building Educational Applications*, pages 54–62.

Ana Díaz-Negrillo, Detmar Meurers, Salvador Valera, and Holger Wunsch. 2010. Towards interlanguage pos annotation for effective learner corpora in sla and flt. In *Language Forum*, volume 36, pages 139–154.

Markus Dickinson and Marwa Ragheb. 2009. Dependency annotation for learner corpora. In *Proceedings of the Eighth Workshop on Treebanks and Linguistic Theories (TLT-8)*, pages 59–70.

Jacob Eisenstein. 2013. What to do about bad language on the internet. In *Proceedings of NAACL*, pages 359–369.

Jennifer Foster, Joachim Wagner, and Josef Van Genabith. 2008. Adapting a WSJ-trained parser to grammatically noisy text. In *Proceedings of ACL: Short Papers*, pages 221–224.

Jennifer Foster, Özlem Çetinoğlu, Joachim Wagner, Joseph Le Roux, Joakim Nivre, Deirdre Hogan, and Josef van Genabith. 2011. From News to Comment: Benchmarks and Resources for Parsing the Language of Web 2.0. In *Proceedings of IJCNLP*, pages 893–901.

Jennifer Foster. 2010. cba to check the spelling investigating parser performance on discussion forum posts. In *Proceedings of NAACL*, pages 381–384.

Sylviane Granger. 1993. International corpus of learner english. In J. Aarts, P. de Haan, and N.Oostdijk, editors, *English Language Corpora: Design, Analysis and Exploitation*, pages 57–71. Rodopi, Amsterdam.

Sylviane Granger. 2008. Learner corpora. In Anke Lüdeling and M. Kytö, editors, *Corpus Linguistics. An International Handbook*, pages 259–275. Berlin:Mouton de Gruyter.

Hagen Hirschmann, Seanna Doolittle, and Anke Lüdeling. 2007. Syntactic annotation of non-canonical linguistic structures. In *Proceedings of Corpus Linguistics*.

Dan Klein and Christopher D. Manning. 2003. Accurate Unlexicalized Parsing. In *Proceedings of ACL*, pages 423–430.

Lingpeng Kong, Nathan Schneider, Swabha Swayamdipta, Archna Bhatia, Chris Dyer, and Noah A. Smith. 2014. A dependency parser for tweets. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1001–10012.

Anke. Lüdeling. 2008. Mehrdeutigkeiten und kategorisierung: Probleme bei der annotation von lernerkorpora. In M. Walter and P. Grommes, editors, *Fortgeschrittene Lernervarietten*, pages 119–140. Niemeyer, Tbingen.

Mitchell Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. 1994. The Penn Treebank: Annotating Predicate Argument Structure. In *Proceedings of the 1994 ARPA Speech and Natural Language Workshop*, pages 114–119.

Justin Mott, Ann Bies, John Laury, and Colin Warner. 2012. Bracketing Webtext: An Addendum to Penn Treebank II Guidelines. Technical report, Linguistic Data Consortium.

Ryo Nagata, Edward Whittaker, and Vera Sheinman. 2011. Creating a manually error-tagged and shallow-parsed learner corpus. In *Proceedings of ACL-HLT*, pages 1210–1219.

Hwee Tou Ng, Siew Mei Wu, Yuanbin Wu, Christian Hadiwinoto, and Joel Tetreault. 2013. The CoNLL-2013 Shared Task on Grammatical Error Correction. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–12.

Hwee Tou Ng, Siew Mei Wu, Ted Briscoe, Christian Hadiwinoto, Raymond Hendy Susanto, and Christopher Bryant. 2014. The CoNLL-2014 Shared Task on Grammatical Error Correction. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–14.

Diane Nicholls. 1999. The cambridge learner corpus – error coding and analysis. In *Summer Workshop on Learner Corpora*, Tokyo, Japan.

Slav Petrov and Dan Klein. 2007. Improved Inference for Unlexicalized Parsing. In *Proceedings of HLT-NAACL*, pages 404–411.

Slav Petrov and Ryan McDonald. 2012. Overview of the 2012 shared task on parsing the web. *Notes of the First Workshop on Syntactic Analysis of Non-Canonical Language (SANCL)*, 59.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning Accurate, Compact and Interpretable Tree Annotation. In *Proceedings of COLING-ACL*.

Ines Rehbein, Hagen Hirschmann, Anke Lüdeling, and Marc Reznicek. 2012. Better tags give better trees or do they? *Linguistic Issues in Language Technology*, 7(10).

Victoria Rosén and Koenraad De Smedt. 2010. Syntactic annotation of learner corpora. *Systematisk, variert, men ikke tilfeldig*, pages 120–132.

Kenji Sagae, Eric Davis, Alon Lavie, Brian MacWhinney, and Shuly Wintner. 2007. High-accuracy annotation and parsing of childes transcripts. In *Proceedings of the Workshop on Cognitive Aspects of Computational Language Acquisition*, pages 25–32.

Djamé Seddah, Benoit Sagot, Marie Candito, Virginie Mouilleron, and Vanessa Combet. 2012. The French Social Media Bank: a Treebank of Noisy User Generated Content. In *Proceedings of COLING*, pages 2441–2458.

Djamé Seddah, Reut Tsarfaty, Sandra Kübler, Marie Candito, Jinho D. Choi, Richárd Farkas, Jennifer Foster, Iakes Goenaga, Koldo Gojenola Galletebeitia, Yoav Goldberg, Spence Green, Nizar Habash, Marco Kuhlmann, Wolfgang Maier, Joakim Nivre, Adam Przepiórkowski, Ryan Roth, Wolfgang Seeker, Yannick Versley, Veronika Vincze, Marcin Woliński, Alina Wróblewska, and Eric Villemonte de la Clergerie. 2013. Overview of the SPMRL 2013 shared task: A cross-framework evaluation of parsing morphologically rich languages. In *Proceedings of the Fourth Workshop on Statistical Parsing of Morphologically-Rich Languages*, pages 146–182.

Ann Taylor. 1996. Bracketing Switchboard: An Addendum to the Treebank II Guidelines. Technical report.

Helen Yannakoudakis, Ted Briscoe, and Ben Medlock. 2011. A New Dataset and Method for Automatically Grading ESOL Texts. In *Proceedings of ACL*, pages 180–189.

Semi-supervised Dependency Parsing using Bilexical Contextual Features from Auto-Parsed Data

Eliyahu Kiperwasser
Computer Science Department
Bar-Ilan University
Ramat-Gan, Israel
elikip@gmail.com

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

Abstract

We present a semi-supervised approach to improve dependency parsing accuracy by using bilexical statistics derived from auto-parsed data. The method is based on estimating the attachment potential of head-modifier words, by taking into account not only the head and modifier words themselves, but also the words surrounding the head and the modifier. When integrating the learned statistics as features in a graph-based parsing model, we observe nice improvements in accuracy when parsing various English datasets.

1 Introduction

We are concerned with semi-supervised dependency parsing, namely how to leverage large amounts of unannotated data, in addition to annotated Treebank data, to improve dependency parsing accuracy. Our method (Section 2) is based on parsing large amounts of unannotated text using a baseline parser, extracting word-interaction statistics from the automatically parsed corpus, and using these statistics as the basis of additional parser features. The automatically-parsed data is used to acquire statistics about lexical interactions, which are too sparse to estimate well from any realistically-sized Treebank. Specifically, we attempt to infer a function $assoc(head, modifer)$ measuring the "goodness" of head-modifier relations ("how good is an arc in which *black* is a modifier of *jump*"). A similar approach was taken by Chen et al. (2009) and Van Noord et al. (2007). We depart from their work by extending the scoring to include a wider lexical context. That is, given the sentence fragment in Figure 1, we score the (incorrect) dependency arc (*black*, *jump*) based on the triplets (*the black fox*, *will jump over*). Learning a function between word triplets raises an extreme data sparsity issue, which we deal with by decomposing the interaction between triplets to a sum of interactions between word pairs. The decomposition we use is inspired by recent work in word-embeddings and dense vector representations (Mikolov et al., 2013a; Mnih and Kavukcuoglu, 2013). Indeed, we initially hoped to leverage the generalization abilities associated with vector-based representations. However, we find that in our setup, reverting to direct count-based statistics achieve roughly the same results (Section 3).

Our derived features improve the accuracy of a first-order dependency parser by 0.75 UAS points (absolute) when evaluated on the in-domain WSJ test-set, obtaining a final accuracy of 92.32 UAS for a first-order parser. When comparing to the strong baseline of using Brown-clusters based features (Koo et al., 2008), we find that our triplets-based method outperform them by over 0.27 UAS points. This is in contrast to previous works (e.g. (Bansal et al., 2014)) in which improvements over using Brown-clusters features were achieved only by adding to the cluster-based features, not by replacing them. As expected, combining both our features and the brown-cluster features result in some additional gains.

2 Our Approach

Our departure point is a graph-based parsing model (McDonald et al., 2005):

$$parse(x) = \underset{y \in \mathcal{Y}(x)}{\operatorname{argmax}}\, score(x, y)$$

$$score(x, y) = w \cdot \Phi(x, y) = \sum_{part \in y} w \cdot \phi(x, part)$$

Given a sentence x we look for the highest-scoring parse tree y in the space $\mathcal{Y}(x)$ of valid dependency trees over x. The score of a tree is determined by a linear model parameterized by a weights vector w, and a feature function $\Phi(x, y)$. To make the search

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1348–1353,
Lisbon, Portugal, 17-21 September 2015.

Features in $\phi^{ij}_{lex}(x,y)$
$\texttt{bin}(S^{ij}(x,y))$
$\texttt{bin}(S^{ij}(x,y)) \circ \texttt{dist}(x,y)$
$\texttt{bin}(S^{ij}(x,y)) \circ \texttt{pos}(x) \circ \texttt{pos}(y)$
$\texttt{bin}(S^{ij}(x,y)) \circ \texttt{pos}(x) \circ \texttt{pos}(y) \circ \texttt{dist}(x,y)$

Table 1: All features are binary indicators. x and y are token indices. S^{ij} is estimated from auto-parsed corpora as described in the text. The values of $S(\cdot,\cdot)$ are in the range $(0,1)$, which is split by $\texttt{bin}$ into 10 equally-spaced intervals. $\texttt{dist}$ is the signed and binned sentence-distance between x and y. $\texttt{pos}(x)$ is the part of speech of token x. $\circ$ indicates a concatenation of features.

tractable, the feature function is decomposed into local feature functions over tree-parts $\phi(x, part)$. The features in ϕ are standard graph-based dependency parsing features, capturing mostly structural information from the parse tree.

We extend the scoring function by adding an additional term capturing the strength of *lexical* association between head word h and modifier word m in each dependency arc:

$$score(x,y) = \sum_{part \in y} w \cdot \phi(x, part) + \sum_{(h,m) \in y} assoc(h,m)$$

The association function $assoc$ is also modeled as a linear model $assoc(h,m) = w_{lex} \cdot \phi_{lex}(h,m)$. While the weights w_{lex} are trained jointly with w based on supervised training data, the features in ϕ_{lex} do not look at h and m directly, but instead rely on a quantity $S(h,m)$ that reflects the "goodness" of the arc (h,m). The quantity $S(h,m)$ ranges between 0 and 1, and is estimated based on large quantities of auto-parsed data. Given a value for $S(h,m)$, ϕ_{lex} is composed of indicator functions indicating the binned ranges of $S(h,m)$, possibly conjoined with information such as the binned surface distance between the tokens h and m and their parts of speech. The complete specification of ϕ_{lex} we use is shown in Table 1 (the meaning of the ij indices will be discussed in Section 2.2).

2.1 Estimating $S(h,m)$

One way of estimating $S(h,m)$, which was also used in (Chen et al., 2009), is using rank statistics. Let D be the list of observed (h,m) pairs sorted by their frequencies, and let $rank(h,m)$ be the index of the pair (h,m) in D. We now set:

$$S_{\text{RANK}}(h,m) = \frac{rank(h,m)}{|D|}$$

While effective, this approach has two related shortcomings: first, it requires storing counts for all the pairs (h,m) appearing in the auto-parsed data, resulting in memory requirement that scales quadratically with the vocabulary size. Second, even with very large auto-parsed corpora many plausible head-modifier pairs are likely to be unobserved.

An alternative way of estimating $S(h,m)$ that does not require storing all the observed pairs and that has a potential of generalizing beyond the seen examples is using a log-bilinear embedding model similar to the skip-gram model presented by Mikolov et al. (2013b) to embed word pairs such that compatible pairs receive high scores. The model assigns two disjoint sets of d-dimensional continuous latent vectors, u and v, where $u_h \in R^d$ is an embedding of a head word h, and $v_m \in R^d$ is an embedding of a modifier word m. The embedding is done by trying to optimize the following corpus-wide objective that is maximizing the dot product of the vectors of observed (h,m) pairs and minimizing the dot product of vectors of random h and m pairs. Formally:

$$\sum_{h,m \in C} \left[ln\left(\sigma\left(u_h \cdot v_m\right)\right) - \sum_{i=1}^{k} \mathbb{E}_{m_i \sim P_m} ln\left(\sigma\left(u_h \cdot v_{m_i}\right)\right) \right]$$

where $\sigma(x) = 1/(1+e^{-x})$, and k is the number of negative samples, drawn from the corpus-based Unigram distribution P_m. For further details, see (Mikolov et al., 2013b; Goldberg and Levy, 2014). We then take:[1]

$$S_{\text{EMBED}}(h,m) = \sigma(u_h \cdot v_m)$$

In contrast to the counts based method, this model is able to estimate the strength of a pair of words even if the pair did not appear in the corpus due to sparsity.

Finally, Levy and Goldberg (2014b) show that the skip-grams with negative-sampling model described above achieves its optimal solution when $u_h \cdot v_m = PMI(h,m) - \log\ k$. This gives rise to another natural way of estimating S:

[1] The embedding we derive are very similar to the ones described in (Levy and Goldberg, 2014a; Bansal et al., 2014), and which were used by Bansal et al.(2014) for deriving semi-supervised parsing features. An important difference from these previous work is that after training, they keep only one set of vectors (u or v) and ignore the other, basing the features on the derived vector representations. In contrast, we keep both sets of vectors and are interested in the association measure induced by the dot product $u_h \cdot v_m$.

Figure 1: Illustration of the bilexical information including context. When scoring the (incorrect) arc between h^0 and m^0, we take into account also the surrounding words h^{-1}, h^{+1}, m^{-1} and m^{+1}.

$$S_{\text{PMI}}(h,m) = \sigma(PMI(h,m)) = \frac{p(h,m)}{p(h,m)+p(h)p(m)}$$

where $p(h,m)$, $p(h)$ and $p(m)$ are unsmoothed maximum-likelihood estimates based on the auto-parsed corpus.

Like S_{RANK} and unlike S_{EMBED}, S_{PMI} requires storing statistics for all observed word pairs, and is not able of generalizing beyond (h,m) pairs seen in the auto-parsed data. However, as we see in Section 3, this method performing similarly in practice, suggesting that the generalization capabilities of the embedding-based method do not benefit the parsing task.

2.2 Adding additional context

Estimating the association between a pair of words is effective. However, we would like to go a step further and take into account also the context in which these words occur. Specifically, our complete model attempts to estimate the association between word trigrams centered around the head and the modifier words.

A naive solution that defines each trigram as its own vocabulary item will increase the vocabulary size by two orders of magnitude and result in severe data sparsity. An alternative solution would be to associate each word in the triplet (h^{-1}, h^0, h^{+1}) with its own unique vocabulary item, and likewise for modifier words. In the embeddings-based model, this results in 6 vector sets $u^{-1}, u^0, u^{+1}, v^{-1}, v^0, v^{+1}$, where v^{-1}_{dog}, for example, represents the word "dog" when it appears to the left of the modifier word, and u^{+1}_{walk} the word "walk" when it appears to the right of the head word.[2] This amounts to only a 3-fold increase in the required vocabulary size. We then model the strength of association between $h^{-1}h^0h^{+1}$ and $m^{-1}m^0m^{+1}$ as a weighted sum of pairwise interactions:[3]

$$assoc(h^{-1}h^0h^{+1}, m^{-1}m^0m^{+1}) = \sum_{i=-1}^{1}\sum_{j=-1}^{1} \alpha_{ij}\ assoc_{ij}(h^i, m^j)$$

As before, the pairwise association measure $assoc_{ij}(h^i, m^j)$ is modeled as a linear model:

$$assoc_{ij}(h^i, m^j) = w^{ij}_{lex} \cdot \phi^{ij}_{lex}(h^i, m^j)$$

Where ϕ^{ij}_{lex} is again defined in terms of a goodness function $S^{ij}(x,y)$. For example, $S^{-1,+1}(the, over)$ corresponds to the goodness of a head-modifier arc where the word to the left of the head word is "the" and the word to the right of the modifier word is "over". $S^{ij}(h^i, m^j)$, the goodness of the arc induced by the pair (h^i, m^j), can be estimated by either S_{RANK}, S_{EMBED} or S_{PMI} as before. For example, in the embeddings model we set $S^{ij}(x,y) = \sigma(u^i_x \cdot v^j_y)$.

We update the bilexical features to include context as explained above. Instead of learning α and w_{lex} coefficients separately, we absorb the α_{ij} terms into w_{lex}, learning both at the same time:

$$\begin{aligned} &assoc(h^{-1}h^0h^{+1}, m^{-1}m^0m^{+1}) = \\ &\sum_{i\in\{-1,0,1\}}\sum_{j\in\{-1,0,1\}} \alpha_{ij} \cdot assoc_{ij}(h^i, m^j) = \\ &\sum_{i\in\{-1,0,1\}}\sum_{j\in\{-1,0,1\}} \alpha_{ij} \cdot w^{ij}_{lex} \cdot \phi^{ij}_{lex}(h^i, m^j) = \\ &\sum_{i\in\{-1,0,1\}}\sum_{j\in\{-1,0,1\}} w'^{ij}_{lex} \cdot \phi^{ij}_{lex}(h^i, m^j) \end{aligned}$$

Finally, the parser selects a dependency tree which maximizes the following:

$$\begin{aligned} w' \cdot \Phi(x,y) = &\sum_{part\in y} w \cdot \phi(x, part) \\ &+ \sum_{(h,m)\in y}\sum_{i=-1}^{1}\sum_{j=-1}^{1} w^{ij}_{lex} \cdot \phi^{ij}_{lex}(h^i, m^j) \end{aligned}$$

[2] Sentences are padded by special sentence-boundary symbols.

[3] In the word embeddings literature, it is common to represent a word triplet as the sum of the individual component vectors, resulting in $u_{x,y,z} \cdot v_{a,b,c} = (u^{-1}_x + u^0_y + u^{+1}_z) \cdot (v^{-1}_a + v^0_b + v^{+1}_c)$. Expanding the terms will result in a very similar formulation to our proposal, but we allow the extra flexibility of associating a different strength α_{ij} with each pairwise term.

	Dev	Test	Brown	Answers	Blogs	Email	News	Reviews
Baseline	91.97	91.57	86.86	81.58	84.88	79.75	82.59	83.22
Base + Brown	92.16	92.05	87.16	81.96	85.46	80.27	83.02	83.56
Base + HM(S_{RANK})	92.16	91.74	87.01	82.06	85.36	80.29	82.72	83.62
Base + HM(S_{EMBED})	92.29	91.98	87.04	81.97	85.34	79.93	82.76	83.33
Base + HM(S_{PMI})	92.35	92.00	87.14	82.20	85.65	80.34	82.83	83.81
Base + TRIP(S_{RANK})	92.23	91.91	87.02	82.31	85.59	**80.50**	83.30	83.79
Base + TRIP(S_{EMBED})	92.38	92.27	87.15	82.34	85.71	80.41	83.21	83.68
Base + TRIP(S_{PMI})	**92.61**	**92.32**	**87.29**	**82.58**	**85.88**	80.43	**83.57**	**84.18**
Base + Brown + TRIP(S_{RANK})	92.43	92.33	87.23	82.60	85.75	80.57	83.48	84.00
Base + Brown + TRIP(S_{EMBED})	92.51	**92.45**	87.33	82.42	86.06	80.44	83.40	83.87
Base + Brown + TRIP(S_{PMI})	**92.70**	92.40	**87.42**	**82.74**	**86.08**	**80.72**	**83.78**	**84.26**

Table 2: Parsing accuracies (UAS, excluding puctuation) of the different models on various corpora. All models are trained on the PTB training set. Dev and Test are sections 22 and 23. Brown is the Brown portion of the PTB. The other columns correspond to the test portions of the Google Web Treebanks. Automatic POS-tags are assigned in all cases. HM indicates using $assoc(h, m)$ and TRIP using $assoc(h^{-1}h^0h^1, m^{-1}m^0m^1)$. + BROWN indicate using features based on Brown clustering.

3 Experiments and Results

Data Our experiments are based on the Penn Treebank (PTB) (Marcus et al., 1993) as well as the Google Web Treebanks (LDC2012T13), covering both in-domain and out-of-domain scenarios. We use the Stanford-dependencies representation (de Marneffe and Manning, 2008). All the constituent-trees are converted to Stanford-dependencies based on the settings of Version 1.0 of the Universal Treebank (McDonald et al., 2013).[4] These are based on the Stanford Dependencies converter but use some non-default flags, and change some of the dependency labels. All of the models are trained on section 2-21 of the WSJ portion of the PTB. For in-domain data, we evaluate on sections 22 (Dev) and 23 (Test). All of the parameter tuning were performed on the Dev set, and we report test-set numbers only for the "most interesting" configurations. For out-of-domain data, we use the Brown portion of the PTB (Brown), as well as the test-sets of different domains available in the Google Web Treebank: Answers, Blogs, Emails, Reviews and Newsgroups.

All trees have automatically assigned part-of-speech tags, assigned by the TurboTagger POS-tagger.[5] The train-set POS-tags were derived in a 10-fold jacknifing, and the different test datasets receive tags from a tagger trained on sections 2-21.

For auto-parsed data, we parse the text of the BLLIP corpus (Charniak, 2000) using our baseline parser. This is the same corpus used for deriving Brown clusters for use as features in (Koo et al., 2008). We use the clusters provided by Terry Koo[6]. Parsing accuracy is measured by unlabeled attachment score (UAS) excluding punctuations.

Implementation Details We focus on first-order parsers, as they are the most practical graph-based parsers in terms of running time in realistic parsing scenarios. Our base model is a re-implementation of a first-order projective Graph-based parser (McDonald et al., 2005), which we extend to support the semi-supervised ϕ_{lex} features. The parser is trained for 10 iterations of online-training with passive-aggressive updates (Crammer et al., 2006). For the Brown-cluster features, we use the feature templates described by (Koo et al., 2008; Bansal et al., 2014).

The embedding vectors are trained using the freely available `word2vecf` software[7], by conjoining each word with its relative position (-1, 0 or 1) and treating the head words as "words" and the modifier words to be "contexts". The words are embedded into 300-dimensional vectors. All code and vectors will be available at the first author's website.

Results The results are shown in Table 2. The second block (HM) compares the baseline parser to a parser including the $assoc(h, m)$ lexical component, using various ways of computing $s(h, m)$. We observe a clear improvement above the baseline from using the lexical component across all domains. The different estimation methods perform very similar to each other.

In the third block (TRIP) we switch to the triplet-based lexical association. With S_{RANK},

[4] `https://github.com/ryanmcd/uni-dep-tb/raw/master/universal_treebanks_v1.0.tar.gz`

[5] `http://www.ark.cs.cmu.edu/TurboParser/`

[6] `http://people.csail.mit.edu/maestro/papers/bllip-clusters.gz`

[7] `http://www.bitbucket.org/yoavgo/word2vecf`

there is very little advantage over looking at just pairs. However, with S_{EMBED} or S_{PMI} the improvement of using the triplet-based method over using just the head-modifier pairs is clear. The counting-based PMI method performs on par with the Embedding based approximation of it.

The second line of the first block (Base+Brown) represents the current state-of-the-art in semi-supervised training of graph-based parsing: using Brown-cluster derived features (Koo et al., 2008; Bansal et al., 2014). The Brown-derived features provide similar (sometimes larger) gains to using our HM method, and substantially smaller gains than our TRIP method. To the best of our knowledge, we are the first to show a semi-supervised method that significantly outperforms the use of Brown-clusters without using Brown-clusters as a component.

As expected, combining our features and the Brown-based features provide an additional improvement, as can be seen in the last block of Table 2 (Base+Brown+TRIP).

4 Related Work

Semi-supervised approaches to dependency parsing can be roughly categorized into two groups: those that use unannotated data and those that use automatically-parsed data. Our proposed method falls in the second group.

Among the words that use unannotated data, the dominant approach is to derive either word clusters (Koo et al., 2008) or word vectors (Chen and Manning, 2014) based on unparsed data, and use these as additional features for a supervised parsing model. While the word representations used in such methods are not specifically designed for the parsing task, they do provide useful features for parsing, and in particular the method of (Koo et al., 2008), relying on features derived using the Brown-clustering algorithm, provides very competitive state-of-the-art results. To the best of our knowledge, we are the first to show a substantial improvement over using Brown-clustering derived features without using Brown-cluster features as a component.

Among the words that use auto-parsed data, a dominant approach is self-training (McClosky et al., 2006), in which a parser A (possibly an ensemble) is used to parse large amounts of data, and a parser B is then trained over the union of the gold data and the auto-parsed data produced by parser A. In the context of dependency-parsing, successful uses of self-training require parser A to be stronger than parser B (Petrov et al., 2010) or use a selection criteria for training only on high-quality parses produced by parser A (Sagae and Tsujii, 2007; Weiss et al., 2015). In contrast, our work uses the same parser (modulo the feature-set) for producing the auto-parsed data and for training the final model, and does not employ a high-quality parse selection criteria when creating the auto-parsed corpus. It is possible that high-quality parse selection can improve our proposed method even further.

Works that derive features from auto-parsed data include (Sagae and Gordon, 2009; Bansal et al., 2014). Such works assign a representation (either cluster or vector) for individual word in the vocabulary based on their syntactic behavior. In contrast, our learned features are designed to capture *interactions* between words. As discussed in sections (1) and (2), most similar to ours is the work of (Chen et al., 2009; Van Noord, 2007). We extend their approach to take into account not only direct word-word interactions but also the lexical surroundings in which these interactions occur.

Another recent approach that takes into account various syntactic interactions was recently introduced by Chen et al. (2014), who propose to learn to embed complex features that are being used in a graph-based parser based on other features they co-occur with in auto-parsed data. Similar to our approach, the embedded features are then used as additional features in a conventional graph-based model. The approaches are to a large extent complementary, and could be combined.

Finally, our work adds additional features to a graph-based parser which is based on a linear-model. Recently, progress in dependency parsing has been made by introducing non-linear, neural-network based models (Pei et al., 2015; Chen and Manning, 2014; Weiss et al., 2015; Dyer et al., 2015; Zhou et al., 2015). Adapting our approach to work with such models is an interesting research direction.

5 Conclusions

We presented a semi-supervised method for dependency parsing and demonstrated its effectiveness on a first-order graph-based parser. Taking into account not only the (head,modifier) word-pair but also their immediate surrounding words add a clear benefit to parsing accuracy.

References

Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2014. Tailoring continuous word representations for dependency parsing. In *Proc. of ACL*.

Eugene Charniak. 2000. Bllip 1987-89 wsj corpus release 1. In *Linguistic Data Consortium*, Philadelphia.

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

Wenliang Chen, Jun'ichi Kazama, Kiyotaka Uchimoto, and Kentaro Torisawa. 2009. Improving dependency parsing with subtrees from auto-parsed data. In *EMNLP*, pages 570–579.

Wenliang Chen, Yue Zhang, and Min Zhang. 2014. Feature embedding for dependency parsing. In *Proc. of COLING*.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006. Online passive-aggressive algorithms. *The Journal of Machine Learning Research*, 7:551–585.

Marie-Catherine de Marneffe and Christopher D. Manning. 2008. The stanford typed dependencies representation. In *Coling 2008: Proceedings of the workshop on Cross-Framework and Cross-Domain Parser Evaluation*, CrossParser '08, pages 1–8, Stroudsburg, PA, USA. Association for Computational Linguistics.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proc. of ACL*.

Yoav Goldberg and Omer Levy. 2014. word2vec explained: deriving Mikolov et al.'s negative-sampling word-embedding method. *arXiv preprint arXiv:1402.3722*.

Terry Koo, Xavier Carreras, and Michael Collins. 2008. Simple semi-supervised dependency parsing. In *Proc. of ACL*, pages 595–603.

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

Omer Levy and Yoav Goldberg. 2014b. Neural word embeddings as implicit matrix factorization. In *Proc. of NIPS*, pages 2177–2185.

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

David McClosky, Eugene Charniak, and Mark Johnson. 2006. Effective self-training for parsing. In *Proc. of NAACL*, pages 152–159.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *Proc. of ACL*.

Ryan T McDonald, Joakim Nivre, Yvonne Quirmbach-Brundage, Yoav Goldberg, Dipanjan Das, Kuzman Ganchev, Keith B Hall, Slav Petrov, Hao Zhang, Oscar Täckström, et al. 2013. Universal dependency annotation for multilingual parsing. In *ACL (2)*, pages 92–97.

Tomas Mikolov, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

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

Andriy Mnih and Koray Kavukcuoglu. 2013. Learning word embeddings efficiently with noise-contrastive estimation. In *NIPS*.

Wenzhe Pei, Tao Ge, and Baobao Chang. 2015. An effective neural network model for graph-based dependency parsing. In *Proc. of ACL*.

Slav Petrov, Pi-Chuan Chang, Michael Ringgaard, and Hiyan Alshawi. 2010. Uptraining for accurate deterministic question parsing. In *Proc. of EMNLP-2010*, October.

Kenji Sagae and Andrew S. Gordon. 2009. Clustering words by syntactic similarity improves dependency parsing of predicate-argument structures. In *IWPT*, pages 192–201.

Kenji Sagae and Jun'ichi Tsujii. 2007. Dependency parsing and domain adaptation with LR models and parser ensembles. In *Proc. of CoNLL-2007 Shared Task*.

Gertjan Van Noord. 2007. Using self-trained bilexical preferences to improve disambiguation accuracy. In *Proceedings of the 10th International Conference on Parsing Technologies*, pages 1–10. Association for Computational Linguistics.

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

Hao Zhou, Yue Zhang, and Jiajun Chen. 2015. A neural probabilistic structured-prediction model for transition-based dependency parsing. In *Proc. of ACL*.

Improved Transition-Based Parsing and Tagging with Neural Networks

Chris Alberti **David Weiss** **Greg Coppola** **Slav Petrov**
Google Inc
New York, NY
{chrisalberti,djweiss,gcoppola,slav}@google.com

Abstract

We extend and improve upon recent work in structured training for neural network transition-based dependency parsing. We do this by experimenting with novel features, additional transition systems and by testing on a wider array of languages. In particular, we introduce set-valued features to encode the predicted morphological properties and part-of-speech confusion sets of the words being parsed. We also investigate the use of joint parsing and part-of-speech tagging in the neural paradigm. Finally, we conduct a multi-lingual evaluation that demonstrates the robustness of the overall structured neural approach, as well as the benefits of the extensions proposed in this work. Our research further demonstrates the breadth of the applicability of neural network methods to dependency parsing, as well as the ease with which new features can be added to neural parsing models.

1 Introduction

Transition-based parsers (Nivre, 2008) are extremely popular because of their high accuracy and speed. Inspired by the greedy neural network transition-based parser of Chen and Manning (2014), Weiss et al. (2015) and Zhou et al. (2015) concurrently developed structured neural network parsers that use beam search and achieve state-of-the-art accuracies for English dependency parsing.[1] While very successful, these parsers have made use only of a small fraction of the rich options provided inside the transition-based framework: for example, all of these parsers use virtually identical atomic features and the *arc-standard* transition system.

In this paper we extend this line of work and introduce two new types of features that significantly improve parsing performance: (1) a set-valued (i.e., bag-of-words style) feature for each word's morphological attributes, and (2) a weighted set-valued feature for each word's k-best POS tags. These features can be integrated naturally as atomic inputs to the embedding layer of the network and the model can learn arbitrary conjunctions with all other features through the hidden layers. In contrast, integrating such features into a model with discrete features requires non-trivial manual tweaking. For example, Bohnet and Nivre (2012) had to carefully discretize the real-valued POS tag score in order to combine it with the other discrete binary features in their system. Additionally, we also experiment with different transition systems, most notably the *integrated* parsing and part-of-speech (POS) tagging system of Bohnet and Nivre (2012) and also the *swap* system of Nivre (2009).

We evaluate our parser on the CoNLL '09 shared task dependency treebanks, as well as on two English setups, achieving the best published numbers in many cases.

2 Model

In this section, we review the baseline model, and then introduce the features (which are novel) and the transition systems (taken from existing work) that we propose as extensions. We measure the impact of each proposed change on the development sets of the multi-lingual CoNLL '09 shared task treebanks (Hajič et al., 2009). For details on our experimental setup, see Section 3.

2.1 Baseline Model

Our baseline model is the structured neural network transition-based parser with beam search of Weiss et al. (2015). We use a feed-forward network with embedding, hidden and softmax layers. The input consists of a sequence of matrices

[1] There is of course a much longer tradition of neural network dependency parsing models, going back at least to Titov and Henderson (2007).

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1354–1359,
Lisbon, Portugal, 17-21 September 2015.

	Ca	Ch	Cz	En	Ge	Ja	Sp
Pipelined							
baseline	87.67	79.10	81.26	88.34	86.79	93.26	87.31
+morph	88.77	"	**84.50**	"	87.26	93.31	88.86
+morph +ktags	88.75	79.42	84.45	**88.62**	87.13	93.35	89.40
Integrated Tagging & Parsing							
+morph	88.93	79.71	84.41	88.57	87.07	93.32	89.35
+morph +ktags	**89.23**	**80.03**	84.27	88.55	**87.88**	**93.50**	**89.76**

Table 1: Ablation study on CoNLL'09 dev set. All scores in this table are LAS with beam 32. The first three rows use a pipeline of tagging and then parsing, while the last two rows use integrated parsing and tagging. Chinese and English have no morphology features provided in the dataset, so we omit morphology for those languages.

extracted deterministically from a transition-based parse configuration (consisting of a stack and a buffer). Each matrix $\mathbf{X}^g$, corresponds to a feature *group* g (one of *words*, *tags*, or *labels*), and has dimension $F^g \times V^g$. Here, X^g_{fv} is 1 if the f'th feature takes on value v for group g, i.e. each row $\mathbf{X}^g$ is a one-hot vector. These features are embedded and then concatenated to form the embedding layer, which in turn is input to the first hidden layer. The concatenated embedding layer can then be written as follows:

$$\mathbf{h}_0 = [\mathbf{X}^g\mathbf{E}^g \mid g \in \{\text{word}, \text{tag}, \text{label}\}] \quad (1)$$

where $\mathbf{E}^g$ is a (learned) $V^g \times D^g$ embedding matrix for group g, and D^g is the embedding dimension for group g. Beyond the embedding layer, there are two non-linear hidden layers (with non-linearity introduced using a rectified linear activation function), and a softmax layer that outputs class probabilities for each possible decision.

Training proceeds in two stages: We first train the network as a classifier by extracting decisions from gold derivations of the training set, as in Chen and Manning (2014). We then train a structured perceptron using the output of all network activations as features, as in Weiss et al. (2015). We use structured training and beam search during inference in all experiments. We train our models only on the treebank training set and do not use tri-training or other semi-supervised learning approaches (aside from using pre-trained word embeddings).

2.2 New Features

Prior work using neural networks for dependency parsing has not ventured beyond the use of one-hot feature activations for each feature type-location pair. In this work, we experiment with set-valued features, in which a set (or *bag*) of features for a given location fire at once, and are embedded into the same embedding space. Note that for both of the features we introduce, we extract features from the same 20 tokens as used in the *tags* and *words* features from Weiss et al. (2015), i.e. various locations on the stack and input buffer.

	Ca	Ch	Cz	En	Ge	Ja	Sp
CRF ($k = 1$)	98.60	93.19	98.25	97.55	96.73	97.47	98.02
Linear ($k = 4$)	98.75	93.71	98.48	97.70	97.27	97.75	98.33
Neural ($k = 4$)	**99.09**	**94.62**	**99.37**	**97.85**	**97.77**	**98.01**	**98.97**
BN'12 ($k = 3$)	-	93.06	99.32	97.77	97.63	-	-

Table 2: POS tagging results on the CoNLL '09 test set for integrated POS tagging and parsing. We compare the accuracy of our baseline CRF tagger, 'Linear' (our re-implementation of Bohnet and Nivre (2012, BN'12)), 'Neural' (the neural parser presented in this work), and results reported by BN'12.

Morphology. It is well known that morphological information is very important for parsing morphologically rich languages (see for example Bohnet et al. (2013)). We incorporate morphological information into our model using a set-valued feature function. We define the feature group *morph* as the matrix $\mathbf{X}^{morph}$ such that, for $1 \le f \le F^{morph}$, $1 \le v \le F^{morph}$,

$$\mathbf{X}^{morph}_{f,v} = \begin{cases} 1/N_f, & \text{token } f \text{ has attribute } v \\ 0, & \text{otherwise} \end{cases}, \quad (2)$$

where N_f is the number of morphological features active on the token indexed by f. In other words, we embed a bag of features into a shared embedding space by *averaging* the individual feature embeddings.

***k*-best Tags.** The non-linear network models of Weiss et al. (2015) and Chen and Manning (2014) embed the 1-best tag, according to a first-stage tagger, for a select set of tokens for any configuration. Inspired by the work of Bohnet and Nivre (2012), we embed the *set* of top tags according to a first-stage tagger. Specifically, we define the feature group *ktags* as the matrix $\mathbf{X}^{ktags}$ such that, for $1 \le f \le F^{ktags}$, $1 \le v \le V^{ktags}$,

$$\mathbf{X}^{ktags}_{f,v} = \begin{cases} P(POS = v \mid f), & v \in \text{top } k \text{ tags for } f \\ 0, & \text{otherwise} \end{cases}, \quad (3)$$

where $P(POS = v \mid f)$ is the marginal probability that the token indexed by f has the tag indexed by v, according to the first-stage tagger.

Method	B	Catalan UAS	Catalan LAS	Chinese UAS	Chinese LAS	Czech UAS	Czech LAS	English UAS	English LAS	German UAS	German LAS	Japanese UAS	Japanese LAS	Spanish UAS	Spanish LAS
Best Shared Task Result	-	-	87.86	-	79.17	-	80.38	-	89.88	-	87.48	-	92.57	-	87.64
Pipelined															
Zhang and McDonald (2014)	-	91.41	87.91	82.87	78.57	86.62	80.59	92.69	90.01	89.88	87.38	92.82	91.87	90.82	87.34
Lei et al. (2014)	-	91.33	87.22	81.67	76.71	88.76	81.77	92.75	90.00	90.81	87.81	**94.04**	91.84	91.16	87.38
This work linear	32	90.81	87.74	81.62	77.62	85.61	76.50	91.86	89.42	89.28	86.79	92.56	91.90	90.02	86.92
This work neural	32	92.31	89.17	83.34	79.50	88.35	83.50	92.37	90.21	90.12	87.79	93.99	**93.10**	91.71	88.68
Integrated Tagging & Parsing															
Bohnet and Nivre (2012)	40	92.02	88.97	81.18	77.00	88.07	82.70	92.06	89.54	90.43	88.23	93.67	92.63	91.43	88.54
Bohnet and Nivre (2012)+G+C	80	**92.44**	**89.60**	82.52	78.51	**88.82**	**83.73**	**92.87**	**90.60**	**91.37**	**89.38**	93.52	92.63	92.24	**89.60**
This work linear	32	91.02	87.98	82.26	78.32	85.73	78.37	91.57	88.83	88.80	86.38	93.28	92.38	90.24	87.09
This work neural	32	92.21	89.15	**83.57**	**79.90**	88.45	83.57	92.70	90.56	90.58	88.20	93.85	92.97	**92.26**	89.33

Table 3: Final CoNLL '09 test set results. The results not from this work were solicited from the respective authors.

Results. The contributions of our new features for pipelined arc-standard parsing are shown in Table 1. Morphology features (+*morph*) contributed a labeled accuracy score (LAS) gain of 2.9% in Czech, 1.5% in Spanish, and 0.9% in Catalan. Adding the k-best tag feature (+*morph* +*ktags*) provides modest gains (and modest losses), peaking at 0.54% LAS for Spanish. This feature proves more beneficial in the integrated transition system, discussed in the next section. We note the ease with which we can obtain these gains in a multilayer embedding framework, without the need for any hand-tuning.

2.3 Integrating Parsing and Tagging

While past work on neural network transition-based parsing has focused exclusively on the *arc-standard* transition system, it is known that better results can often be obtained with more sophisticated transition systems that have a larger set of possible actions. The *integrated arc-standard* transition system of Bohnet and Nivre (2012) allows the parser to participate in tagging decisions, rather than being forced to treat the tagger's tags as given, as in the arc-standard system. It does this by replacing the *shift* action in the arc-standard system with an action $shift_p$, which, aside from shifting the top token on the buffer also assigns it one of the k best POS tags from a first-stage tagger. We also experiment with the *swap* action of Nivre (2009), which allows reordering of the tokens in the input sequence. This transition system is able to produce non-projective parse trees, which is important for some languages.

Results. The effect of using the integrated transition system is quantified in the bottom part of Table 1. The use of both 1) +*morph* +*kbest* features and 2) integrated parsing and tagging achieves the best score for 5 out of 7 languages tested. The use of integrated parsing and tagging provides, for example, a 0.8% LAS gain in German.

3 Experiments

In this section we provide final test set results for our baseline and full models on three standard setups from the literature: *CoNLL '09*, *English WSJ* and *English Treebank Union*.

3.1 General Setup

To train with predicted POS tags, we use a CRF-based POS tagger to generate 5-fold jack-knifed POS tags on the training set and predicted tags on the dev, test and tune sets; our tagger gets comparable accuracy to the Stanford POS tagger (Toutanova et al., 2003) with 97.44% on the WSJ test set. The candidate tags allowed by the integrated transition system on every $shift_p$ action are chosen by taking the top 4 tags for a token according to the CRF tagger, sorted by posterior probability, with no minimum posterior probability for a tag to be selected. We report unlabeled attachment score (UAS) and labeled attachment score (LAS). Whether punctuation is included in the evaluation is specified in each subsection.

We use 1024 units in all hidden layers, a choice made based on the development set. We found network sizes to be of critical importance for the accuracy of our models. For example, LAS improvements can be as high as 0.98% in *CoNLL'09* German when increasing the size of the two hidden layers from 200 to 1024. We use $B = 16$ or $B = 32$ based on the development set performance per language. For ease of experimentation, we deviate from Bohnet and Nivre (2012) and use a single unstructured beam, rather than separate beams for POS tag and parse differences.

We train our neural networks on the standard training sets only, except for initializing with word

Method	B	UAS	LAS
Graph-based pipelined			
Bohnet (2010)	-	92.88	90.71
Martins et al. (2013)	-	92.89	90.55
Zhang and McDonald (2014)	-	93.22	91.02
Transition-based pipelined			
Zhang and Nivre (2011)	32	93.00	90.95
Bohnet and Kuhn (2012)	80	93.27	91.19
Chen and Manning (2014)	1	91.80	89.60
Dyer et al. (2015)	1	93.20	90.90
Weiss et al. (2015), supervised	8	93.99	92.05
Weiss et al. (2015), semi-sup.	8	**94.26**	**92.41**
Transition-based integrated			
Bohnet and Nivre (2012)	80	93.33	91.22
This work, supervised	32	**94.23**	**92.36**

Table 4: WSJ test set results on Stanford dependencies. Both the best supervised and semi-supervised results are bolded.

embeddings generated by word2vec and using cluster features in our POS tagger. Unlike Weiss et al. (2015) we train our model only on the treebank training set and do not use tri-training, which can likely further improve the results.

3.2 CoNLL '09

Our multilingual evaluation follows the setup of the CoNLL '09 shared task[2] (Hajič et al., 2009). As standard, we use the supplied predicted morphological features from the shared task data; however, we predict k-best tags with our own POS tagger since k-best tags are not part of the given data. We follow standard practice and include all punctuation in the evaluation. We used the (integrated) arc-standard transition system for all languages except for Czech where we added a swap transition, obtaining a 0.4% absolute improvement in UAS and LAS over just using arc-standard.

Results. In Table 3, we compare our models to the winners of the CoNLL '09 shared task, Gesmundo et al. (2009), Bohnet (2009), Che et al. (2009), Ren et al. (2009), as well as to more recent results on the same datasets. It is worth pointing out that Gesmundo et al. (2009) is itself a neural net parser. Our models achieve higher labeled accuracy than the winning systems in the shared task in all languages. Additionally, our pipelined neural network parser always outperforms its linear counterpart, an in-house reimplementation of the system of Zhang and Nivre (2011), as well as the more recent and highly accurate parsers of Zhang and McDonald (2014) and Lei et al. (2014). For the integrated models our neural network parser again outperforms its linear counterpart (Bohnet and Nivre, 2012), however, in some cases the addition of graph-based and cluster features (Bohnet and Nivre, 2012)+G+C can lead to even better results. The improvements in POS tagging (Table 2) range from 0.3% for English to 1.4% absolute for Chinese and are always higher for the neural network models compared to the linear models.

[2]http://ufal.mff.cuni.cz/conll2009-st/results/results.php

	News		Web		QTB	
Method	UAS	LAS	UAS	LAS	UAS	LAS
Bohnet (2010)	93.29	91.38	88.22	85.22	94.01	91.49
Martins et al. (2013)	93.10	91.13	88.23	85.04	94.21	91.54
Zhang et al. (2014)	93.32	91.48	88.65	85.59	93.37	90.69
Weiss et al. (2015)	93.91	92.25	89.29	86.44	94.17	92.06
This work (B=16)	**94.10**	**92.55**	**89.55**	**86.85**	**94.74**	**93.04**

Table 5: Final English Treebank Union test set results.

3.3 English WSJ

We experiment on English using the Wall Street Journal (WSJ) part of the Penn Treebank (Marcus et al., 1993), with standard train/test splits. We convert the constituency trees to Stanford style dependencies (De Marneffe et al., 2006) using version 3.3.0 of the converter. We use predicted POS tags and exclude punctuation from the evaluation, as is standard for English.

Results. The results shown in Table 4, we find that our full model surpasses, to our knowledge, all previously reported supervised parsing models for the Stanford dependency conversions. It surpasses its linear analog, the work of Bohnet and Nivre (2012) on Stanford Dependencies UAS by 0.9% UAS and by 1.14% LAS. It also outperforms the pipeline neural net model of Weiss et al. (2015) by a considerable margin and matches the semi-supervised variant of Weiss et al. (2015).

3.4 English Treebank Union

Turning to cross-domain results, and the "Treebank Union" datasets, we use an identical setup to the one described in Weiss et al. (2015). This setup includes the WSJ with Stanford Dependencies, the OntoNotes corpus version 5 (Hovy et al., 2006), the English Web Treebank (Petrov and McDonald, 2012), and the updated and corrected Question Treebank (Judge et al., 2006). We train on the union of each corpora's training set and test on each domain separately.

Results. The results of this evaluation are shown in Table 5. As for the WSJ we find that the integrated transition system combined with our novel

features performs better than previous work and in particular the model of Weiss et al. (2015), which serves as the starting point for this work. The improvements on the out-of-domain Web and Question corpora are particularly promising.

4 Conclusions

Weiss et al. (2015) presented a parser that advanced the state of the art for English Stanford dependency parsing. In this paper we showed that this parser can be significantly improved by introducing novel set features for morphology and POS tag ambiguities, which are added with almost no feature engineering effort. The resulting parser is already competitive in the multi-lingual setting of the CoNLL'09 shared task, but can be further improved by utilizing an integrated POS tagging and parsing transition system. We find that for all settings the dense neural network model produces higher POS tagging and parsing accuracy gains than its sparse linear counterpart.

Acknowledgements. We thank Bernd Bohnet for running his parser on additional data sets, and Emily Pitler for helpful comments.

References

Bernd Bohnet and Jonas Kuhn. 2012. The best of both worlds: a graph-based completion model for transition-based parsers. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 77–87.

Bernd Bohnet and Joakim Nivre. 2012. A transition-based system for joint part-of-speech tagging and labeled non-projective dependency parsing. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1455–1465.

Bernd Bohnet, Joakim Nivre, Igor Boguslavsky, Richárd Farkas, Filip Ginter, and Jan Hajič. 2013. Joint morphological and syntactic analysis for richly inflected languages. *Transactions of the Association for Computational Linguistics*, 1:415–428.

Bernd Bohnet. 2009. Efficient parsing of syntactic and semantic dependency structures. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning: Shared Task*, pages 67–72.

Bernd Bohnet. 2010. Top accuracy and fast dependency parsing is not a contradiction. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 89–97.

Wanxiang Che, Zhenghua Li, Yongqiang Li, Yuhang Guo, Bing Qin, and Ting Liu. 2009. Multilingual dependency-based syntactic and semantic parsing. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning: Shared Task*, pages 49–54.

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

Marie-Catherine De Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of Fifth International Conference on Language Resources and Evaluation*, pages 449–454.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*, pages 334–343.

Andrea Gesmundo, James Henderson, Paola Merlo, and Ivan Titov. 2009. A latent variable model of synchronous syntactic-semantic parsing for multiple languages. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning: Shared Task*, pages 37–42.

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, Pavel Straňák, Mihai Surdeanu, Nianwen Xue, and Yi Zhang. 2009. The conll-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–18.

Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. Ontonotes: The 90% solution. In *Proceedings of the Human Language Technology Conference of the NAACL, Short Papers*, pages 57–60.

John Judge, Aoife Cahill, and Josef van Genabith. 2006. Questionbank: Creating a corpus of parse-annotated questions. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 497–504.

Tao Lei, Yu Xin, Yuan Zhang, Regina Barzilay, and Tommi Jaakkola. 2014. Low-rank tensors for scoring dependency structures. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 1381–1391.

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

Andre Martins, Miguel Almeida, and Noah A. Smith. 2013. Turning on the turbo: Fast third-order non-projective turbo parsers. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 617–622.

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

Joakim Nivre. 2009. Non-projective dependency parsing in expected linear time. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 351–359.

Slav Petrov and Ryan McDonald. 2012. Overview of the 2012 shared task on parsing the web. Notes of the First Workshop on Syntactic Analysis of Non-Canonical Language (SANCL).

Han Ren, Donghong Ji, Jing Wan, and Mingyao Zhang. 2009. Parsing syntactic and semantic dependencies for multiple languages with a pipeline approach. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning: Shared Task*, pages 97–102.

Ivan Titov and James Henderson. 2007. Constituent parsing with incremental sigmoid belief networks. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 632–639.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 173–180.

David Weiss, Chris Alberti, Michael Collins, and Slav Petrov. 2015. Structured training for neural network transition-based parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*, pages 323–333.

Hao Zhang and Ryan McDonald. 2014. Enforcing structural diversity in cube-pruned dependency parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 656–661.

Yue Zhang and Joakim Nivre. 2011. Transition-based dependency parsing with rich non-local features. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 188–193.

Hao Zhou, Yue Zhang, and Jiajun Chen. 2015. A neural probabilistic structured-prediction model for transition-based dependency parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*, pages 1213–1222.

Syntactic Parse Fusion

Do Kook Choe
Brown University
Providence, RI
dc65@cs.brown.edu

David McClosky
IBM Research
Yorktown Heights, NY
dmcclosky@us.ibm.com

Eugene Charniak
Brown University
Providence, RI
ec@cs.brown.edu

Abstract

Model combination techniques have consistently shown state-of-the-art performance across multiple tasks, including syntactic parsing. However, they dramatically increase runtime and can be difficult to employ in practice. We demonstrate that applying constituency model combination techniques to n-best lists instead of n different parsers results in significant parsing accuracy improvements. Parses are weighted by their probabilities and combined using an adapted version of Sagae and Lavie (2006). These accuracy gains come with marginal computational costs and are obtained on top of existing parsing techniques such as discriminative reranking and self-training, resulting in state-of-the-art accuracy: 92.6% on WSJ section 23. On out-of-domain corpora, accuracy is improved by 0.4% on average. We empirically confirm that six well-known n-best parsers benefit from the proposed methods across six domains.

1 Introduction

Researchers have proposed many algorithms to combine parses from multiple parsers into one final parse (Henderson and Brill, 1999; Zeman and Žabokrtskỳ, 2005; Sagae and Lavie, 2006; Nowson and Dale, 2007; Fossum and Knight, 2009; Petrov, 2010; Johnson and Ural, 2010; Huang et al., 2010; McDonald and Nivre, 2011; Shindo et al., 2012; Narayan and Cohen, 2015). These new parses are substantially better than the originals: Zhang et al. (2009) combine outputs from multiple n-best parsers and achieve an F_1 of 92.6% on the WSJ test set, a 0.5% improvement over their best n-best parser. Model combination approaches tend to fall into the following categories: *hybridization*, where multiple parses are combined into a single parse; *switching*, which picks a single parse according to some criteria (usually a form of voting); *grammar merging* where grammars are combined before or during parsing; and *stacking*, where one parser sends its prediction to another at runtime. All of these have at least one of the caveats that (1) overall computation is increased and runtime is determined by the slowest parser and (2) using multiple parsers increases the system complexity, making it more difficult to deploy in practice. In this paper, we describe a simple hybridization extension ("fusion") which obtains much of hybridization's benefits while using only a single n-best parser and minimal extra computation. Our method treats each parse in a single parser's n-best list as a parse from n separate parsers. We then adapt parse combination methods by Henderson and Brill (1999), Sagae and Lavie (2006), and Fossum and Knight (2009) to fuse the constituents from the n parses into a single tree. We empirically show that six n-best parsers benefit from parse fusion across six domains, obtaining state-of-the-art results. These improvements are complementary to other techniques such as reranking and self-training. Our best system obtains an F_1 of 92.6% on WSJ section 23, a score previously obtained only by combining the outputs from multiple parsers. A reference implementation is available as part of BLLIP Parser at `http://github.com/BLLIP/bllip-parser/`

2 Fusion

Henderson and Brill (1999) propose a method to combine trees from m parsers in three steps: populate a chart with constituents along with the number of times they appear in the trees; remove any constituent with count less than $m/2$ from the chart; and finally create a final tree with all the remaining constituents. Intuitively their method

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1360–1366,
Lisbon, Portugal, 17-21 September 2015.

constructs a tree with constituents from the majority of the trees, which boosts precision significantly. Henderson and Brill (1999) show that this process is guaranteed to produce a valid tree. Sagae and Lavie (2006) generalize this work by reparsing the chart populated with constituents whose counts are above a certain threshold. By adjusting the threshold on development data, their generalized method balances precision and recall. Fossum and Knight (2009) further extend this line of work by using n-best lists from multiple parsers and combining productions in addition to constituents. Their model assigns sums of joint probabilities of constituents and parsers to constituents. Surprisingly, exploiting n-best trees does not lead to large improvement over combining 1-best trees in their experiments.

Our extension takes the n-best trees from a parser as if they are 1-best parses from n parsers, then follows Sagae and Lavie (2006). Parses are weighted by the estimated probabilities from the parser. Given n trees and their weights, the model computes a constituent's weight by summing weights of all trees containing that constituent. Concretely, the weight of a constituent spanning from ith word to jth word with label ℓ is

$$c_\ell(i \to j) = \sum_{k=1}^{n} W(k) C_\ell^k(i \to j) \qquad (1)$$

where $W(k)$ is the weight of kth tree and $C_\ell^k(i \to j)$ is one if a constituent with label ℓ spanning from i to j is in kth tree, zero otherwise. After populating the chart with constituents and their weights, it throws out constituents with weights below a set threshold t. Using the threshold $t = 0.5$ emulates the method of Henderson and Brill (1999) in that it constructs the tree with the constituents in the majority of the trees. The CYK parsing algorithm is applied to the chart to produce the final tree.

Note that populating the chart is linear in the number of words and the chart contains substantially fewer constituents than charts in well-known parsers, making this a fast procedure.

2.1 Score distribution over trees

We assume that n-best parsers provide trees along with some kind of scores (often probabilities or log probabilities). Given these scores, a natural way to obtain weights is to normalize the probabilities. However, parsers do not always provide accurate estimates of parse quality. We may obtain better performance from parse fusion by altering this distribution and passing scores through a nonlinear function, $f(\cdot)$. The kth parse is weighted:

$$W(k) = \frac{f(\text{SCORE}(k))}{\sum_{i=1}^{n} f(\text{SCORE}(i))} \qquad (2)$$

where $\text{SCORE}(i)$ is the score of ith tree.[1] We explore the family of functions $f(x) = x^\beta$ which can smooth or sharpen the score distributions. This includes a tunable parameter, $\beta \in \mathbb{R}_0^+$:

$$W(k) = \frac{\text{SCORE}(k)^\beta}{\sum_{i=1}^{n} \text{SCORE}(i)^\beta} \qquad (3)$$

Employing $\beta < 1$ flattens the score distribution over n-best trees and helps over-confident parsers. On the other hand, having $\beta > 1$ skews the distribution toward parses with higher scores and helps under-confident parsers. Note that setting $\beta = 0$ weights all parses equally and results in majority voting at the constituent level. We leave developing other nonlinear functions for fusion as future work.

3 Experiments

Corpora: Parse fusion is evaluated on British National Corpus (BNC), Brown, GENIA, Question Bank (QB), Switchboard (SB) and Wall Street Journal (WSJ) (Foster and van Genabith, 2008; Francis and Kučera, 1989; Kim et al., 2003; Judge et al., 2006; Godfrey et al., 1992; Marcus et al., 1993). WSJ is used to evaluate in-domain parsing, the remaining five are used for out-of-domain. For divisions, we use tune and test splits from Bacchiani et al. (2006) for Brown, McClosky's test PMIDs[2] for GENIA, Stanford's test splits[3] for QuestionBank, and articles 4000–4153 for Switchboard.

Parsers: The methods are applied to six widely used n-best parsers: Charniak (2000), Stanford (Klein and Manning, 2003), BLLIP (Charniak and Johnson, 2005), Self-trained BLLIP (McClosky et al., 2006)[4], Berkeley (Petrov et al., 2006), and Stanford RNN (Socher et al., 2013). The list of parsers and their accuracies on the WSJ test set is reported in Table 1. We convert to Stanford

[1] For parsers that return log probabilities, we turn these into probabilities first.

[2] `http://nlp.stanford.edu/~mcclosky/biomedical.html`

[3] `http://nlp.stanford.edu/data/QuestionBank-Stanford.shtml`

[4] Using the '`WSJ+Gigaword-v2`' BLLIP model.

Parser	F_1	UAS	LAS
Stanford	85.4	90.0	87.3
Stanford RNN[5]	89.6	92.9	90.4
Berkeley	90.0	93.5	91.2
Charniak	89.7	93.2	90.8
BLLIP	91.5	94.4	92.0
Self-trained BLLIP	92.2	94.7	92.2

Table 1: Six parsers along with their 1-best F_1 scores, unlabeled attachment scores (UAS) and labeled attachment scores (LAS) on WSJ section 23.

Dependencies (basic dependencies, version 3.3.0) and provide dependency metrics (UAS, LAS) as well.

Supervised parsers are trained on the WSJ training set (sections 2–21) and use section 22 or 24 for development. Self-trained BLLIP is self-trained using two million sentences from Gigaword and Stanford RNN uses word embeddings trained from larger corpora.

Parameter tuning: There are three parameters for our fusion process: the size of the n-best list ($2 < n \leq 50$), the smoothing exponent from Section 2.1 ($\beta \in [0.5, 1.5]$ with 0.1 increments), and the minimum threshold for constituents ($t \in [0.2, 0.7]$ with 0.01 increments). We use grid search to tune these parameters for two separate scenarios. When parsing WSJ (in-domain), we tune parameters on WSJ section 24. For the remaining corpora (out-of-domain), we use the tuning section from Brown. Each parser is tuned separately, resulting in 12 different tuning scenarios. In practice, though, in-domain and out-of-domain tuning regimes tend to pick similar settings within a parser. Across parsers, settings are also fairly similar (n is usually 30 or 40, t is usually between 0.45 and 0.5). While the smoothing exponent varies from 0.5 to 1.3, setting $\beta = 1$ does not significantly hurt accuracy for most parsers.

To study the effects of these parameters, Figure 1 shows three slices of the tuning surface for BLLIP parser on WSJ section 24 around the optimal settings ($n = 30, \beta = 1.1, t = 0.47$). In each graph, one of the parameters is varied while the other is held constant. Increasing n-best size improves accuracy until about $n = 30$ where there seems to be sufficient diversity. For BLLIP, the smoothing exponent (β) is best set around 1.0, with accuracy falling off if the value deviates too much. Finally, the threshold parameter is empirically optimized a little below $t = 0.5$ (the value suggested by Henderson and Brill (1999)). Since score values are normalized, this means that constituents need roughly half the "score mass" in order to be included in the chart. Varying the threshold changes the precision/recall balance since a high threshold adds only the most confident constituents to the chart (Sagae and Lavie, 2006).

[5]Socher et al. (2013) report an F_1 of 90.4%, but this is the result of using an ensemble of two RNNs (p.c.). We use a single RNN in this work.

Parser	WSJ	Brown
BLLIP	90.6	85.7
+ Fusion	91.0	86.0
+ Majority voting ($\beta = 0$)	89.1	83.8
+ Rank-based weighting	89.3	84.1

Table 2: F_1 of a baseline parser, fusion, and baselines on development sections of corpora (WSJ section 24 and Brown tune).

Baselines: Table 2 gives the accuracy of fusion and baselines for BLLIP on the development corpora. Majority voting sets $n = 50, \beta = 0, t = 0.5$ giving all parses equal weight and results in constituent-level majority voting. We explore a rank-based weighting which ignores parse probabilities and weight parses only using the rank: $W_{\text{rank}}(k) = 1/(2^k)$. These show that accurate parse-level scores are critical for good performance.

Final evaluation: Table 3 gives our final results for all parsers across all domains. Results in blue are significant at $p < 0.01$ using a randomized permutation test. Fusion generally improves F_1 for in-domain and out-of-domain parsing by a significant margin. For the self-trained BLLIP parser, in-domain F_1 increases by 0.4% and out-of-domain F_1 increases by 0.4% on average. Berkeley parser obtains the smallest gains from fusion since Berkeley's n-best lists are ordered by factors other than probabilities. As a result, the probabilities from Berkeley can mislead the fusion process.

We also compare against model combination using our reimplementation of Sagae and Lavie (2006). For these results, all six parsers were given equal weight. The threshold was set to 0.42 to optimize model combination F_1 on development data (similar to Setting 2 for constituency parsing in Sagae and Lavie (2006)). Model combination

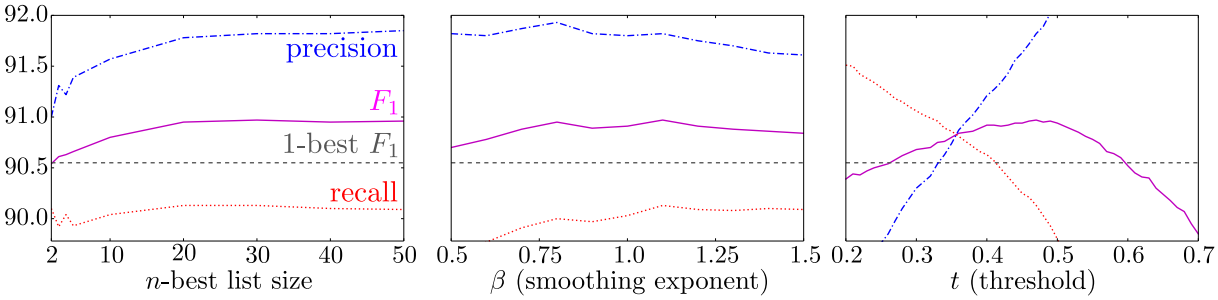


Figure 1: Tuning parameters independently for BLLIP and their impact on F_1 for WSJ section 24 (solid purple line). For each graph, non-tuned parameters were set at the optimal configuration for BLLIP ($n = 30$, $\beta = 1.1$, $t = 0.47$). The dashed grey line represents the 1-best baseline at 90.6% F_1.

Parser	BNC	Brown	GENIA	SB	QB	WSJ
Stanford	78.4 / 79.6	80.7 / 81.6	73.1 / 73.9	67.0 / 67.9	78.6 / 80.0	85.4 / 86.2
Stanford RNN	82.0 / 82.3	84.0 / 84.3	76.0 / 76.2	70.7 / 71.2	82.9 / 83.6	89.6 / 89.7
Berkeley	82.3 / 82.9	84.6 / 84.6	76.4 / 76.6	74.5 / 75.1	86.5 / 85.9	90.0 / 90.3
Charniak	82.5 / 83.0	83.9 / 84.6	74.8 / 75.7	76.8 / 77.6	85.6 / 86.3	89.7 / 90.1
BLLIP	84.1 / 84.7	85.8 / 86.0	76.7 / 77.1	79.2 / 79.5	88.1 / 88.9	91.5 / 91.7
Self-trained BLLIP	85.2 / 85.8	87.4 / 87.7	77.8 / 78.2	80.9 / 81.7	89.5 / 89.5	92.2 / 92.6
Model combination	86.6	87.7	79.4	80.9	89.3	92.5

Table 3: Evaluation of the constituency fusion method on six parsers across six domains. x/y indicates the F_1 from the baseline parser (x) and the baseline parser with fusion (y) respectively. Blue indicates a statistically significant difference between fusion and its baseline parser ($p < 0.01$).

performs better than fusion on BNC and GENIA, but surprisingly fusion outperforms model combination on three of the six domains (not usually not by a significant margin). With further tuning (e.g., specific weights for each constituent-parser pair), the benefits from model combination should increase.

Multilingual evaluation: We evaluate fusion with the Berkeley parser on Arabic (Maamouri et al., 2004; Green and Manning, 2010), French (Abeillé et al., 2003), and German (Brants et al., 2002) from the SPMRL 2014 shared task (Seddah et al., 2014) but did not observe any improvement. We suspect this has to do with the same ranking issues seen in the Berkeley Parser's English results. On the other hand, fusion helps the parser of Narayan and Cohen (2015) on the German NEGRA treebank (Skut et al., 1997) to improve from 80.9% to 82.4%.

Runtime: As discussed in Section 2, fusion's runtime overhead is minimal. Reranking parsers (e.g., BLLIP and Stanford RNN) already need to perform n-best decoding as input for the reranker. Using a somewhat optimized implementation fusion in C++, the overhead over BLLIP parser is less than 1%.

Discussion: Why does fusion help? It is possible that a parser's n-list and its scores act as a weak approximation to the full parse forest. As a result, fusion seems to provide part of the benefits seen in forest reranking (Huang, 2008).

Results from Fossum and Knight (2009) imply that fusion and model combination might not be complementary. Both n-best lists and additional parsers provide syntactic diversity. While additional parsers provide greater diversity, n-best lists from common parsers are varied enough to provide improvements for parse hybridization.

We analyzed how often fusion produces completely novel trees. For BLLIP on WSJ section 24, this only happens about 11% of the time. Fusion picks the 1-best tree 72% of the time. This means that for the remaining 17%, fusion picks an existing parse from the rest of the n-list, acting similar to a reranker. When fusion creates unique trees, they are significantly better than the original 1-best trees (for the 11% subset of WSJ 24, F_1 scores are 85.5% with fusion and 84.1% without, $p < 0.003$). This contrasts with McClosky et al. (2012) where novel predic-

tions from model combination (stacking) were worse than baseline performance. The difference is that novel predictions with fusion better incorporate model confidence whereas when stacking, a novel prediction is less trusted than those produced by one or both of the base parsers.

Preliminary extensions: Here, we summarize two extensions to fusion which have yet to show benefits. The first extension explores applying fusion to dependency parsing. We explored two ways to apply fusion when starting from constituency parses: (1) fuse constituents and then convert them to dependencies and (2) convert to dependencies then fuse the dependencies as in Sagae and Lavie (2006). Approach (1) does not provide any benefit (LAS drops between 0.5% and 2.4%). This may result from fusion's artifacts including unusual unary chains or nodes with a large number of children — it is possible that adjusting unary handling and the precision/recall tradeoff may reduce these issues. Approach (2) provided only modest benefits compared to those from constituency parsing fusion. The largest LAS increase for (2) is 0.6% for the Stanford Parser, though for Berkeley and Self-trained BLLIP, dependency fusion results in small losses (-0.1% LAS). Two possible reasons are that the dependency baseline is higher than its constituency counterpart and some dependency graphs from the n-best list are duplicates which lowers diversity and may need special handling, but this remains an open question.

While fusion helps on top of a self-trained parser, we also explored whether a fused parser can self-train (McClosky et al., 2006). To test this, we (1) parsed two million sentences with BLLIP (trained on WSJ), (2) fused those parses, (3) added the fused parses to the gold training set, and (4) retrained the parser on the expanded training. The resulting model did not perform better than a self-trained parsing model that didn't use fusion.

4 Conclusions

We presented a simple extension to parse hybridization which adapts model combination techniques to operate over a single parser's n-best list instead of across multiple parsers. By weighting each parse by its probability from the n-best parser, we are able to better capture the confidence at the constituent level. Our best configuration obtains state-of-the-art accuracy on WSJ with an F_1 of 92.6%. This is similar to the accuracy obtained from actual model combination techniques but at a fraction of the computational cost. Additionally, improvements are not limited to a single parser or domain. Fusion improves parser accuracy for six n-best parsers both in-domain and out-of-domain.

Future work includes applying fusion to n-best dependency parsers and additional (parser, language) pairs. We also intend to explore how to better apply fusion to converted dependencies from constituency parsers. Lastly, it would be interesting to adapt fusion to other structured prediction tasks where n-best lists are available.

Acknowledgements

We are grateful to Shay Cohen and Shashi Narayan who gave us early access to their parser's German results. We would also like to thank Mohit Bansal, Yoav Goldberg, Siddharth Patwardhan, and Kapil Thadani for helpful discussions and our anonymous reviewers for their insightful comments and suggestions.

References

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

Michiel Bacchiani, Michael Riley, Brian Roark, and Richard Sproat. 2006. Map adaptation of stochastic grammars. *Computer speech & language*, 20(1):41–68.

Sabine Brants, Stefanie Dipper, Silvia Hansen, Wolfgang Lezius, and George Smith. 2002. The tiger treebank. In *Proceedings of the workshop on treebanks and linguistic theories*, volume 168.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 173–180, Ann Arbor, Michigan, June. Association for Computational Linguistics.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*, pages 132–139. Association for Computational Linguistics.

Victoria Fossum and Kevin Knight. 2009. Combining constituent parsers. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, pages 253–256, Boulder, Colorado, June. Association for Computational Linguistics.

Jennifer Foster and Josef van Genabith. 2008. Parser evaluation and the BNC: Evaluating 4 constituency parsers with 3 metrics. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*. European Language Resources Association (ELRA).

Winthrop Nelson Francis and Henry Kučera. 1989. *Manual of information to accompany a standard corpus of present-day edited American English, for use with digital computers*. Brown University, Department of Linguistics.

John J. Godfrey, Edward C. Holliman, and Jane McDaniel. 1992. Switchboard: Telephone speech corpus for research and development. In *Acoustics, Speech, and Signal Processing, 1992. ICASSP-92., 1992 IEEE International Conference on*, volume 1, pages 517–520. IEEE.

Spence Green and Christopher D Manning. 2010. Better arabic parsing: Baselines, evaluations, and analysis. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 394–402. Association for Computational Linguistics.

John C. Henderson and Eric Brill. 1999. Exploiting diversity in natural language processing: Combining parsers. In *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 187–194, College Park, MD, June. Association for Computational Linguistics.

Zhongqiang Huang, Mary Harper, and Slav Petrov. 2010. Self-training with products of latent variable grammars. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 12–22. Association for Computational Linguistics.

Liang Huang. 2008. Forest reranking: Discriminative parsing with non-local features. In *Proceedings of ACL-08: HLT*, pages 586–594, Columbus, Ohio, June. Association for Computational Linguistics.

Mark Johnson and Ahmet Engin Ural. 2010. Reranking the berkeley and brown parsers. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 665–668. Association for Computational Linguistics.

John Judge, Aoife Cahill, and Josef van Genabith. 2006. Questionbank: Creating a corpus of parse-annotated questions. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 497–504, Sydney, Australia, July. Association for Computational Linguistics.

J-D Kim, Tomoko Ohta, Yuka Tateisi, and Junichi Tsujii. 2003. Genia corpus—a semantically annotated corpus for bio-textmining. *Bioinformatics*, 19(suppl 1):i180–i182.

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

Mohamed Maamouri, Ann Bies, Tim Buckwalter, and Wigdan Mekki. 2004. The penn arabic treebank: Building a large-scale annotated arabic corpus. In *NEMLAR conference on Arabic language resources and tools*, pages 102–109.

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

David McClosky, Eugene Charniak, and Mark Johnson. 2006. Effective self-training for parsing. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 152–159, New York City, USA, June. Association for Computational Linguistics.

David McClosky, Sebastian Riedel, Mihai Surdeanu, Andrew McCallum, and Christopher D. Manning. 2012. Combining joint models for biomedical event extraction. *BMC Bioinformatics*, 13(Suppl 11):S9.

Ryan McDonald and Joakim Nivre. 2011. Analyzing and integrating dependency parsers. *Computational Linguistics*, 37(1):197–230.

Shashi Narayan and Shay B. Cohen. 2015. Diversity in spectral learning for natural language parsing. In *Proceedings of EMNLP*.

Scott Nowson and Robert Dale. 2007. Charting democracy across parsers. In *Proceedings of the Australasian Language Technology Workshop*, pages 75–82.

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

Slav Petrov. 2010. Products of random latent variable grammars. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 19–27. Association for Computational Linguistics.

Kenji Sagae and Alon Lavie. 2006. Parser combination by reparsing. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, pages 129–132, New York City, USA, June. Association for Computational Linguistics.

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

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

Wojciech Skut, Brigitte Krenn, Thorsten Brants, and Hans Uszkoreit. 1997. An annotation scheme for free word order languages. In *Proceedings of the fifth conference on Applied natural language processing*, pages 88–95. Association for Computational Linguistics.

Richard Socher, John Bauer, Christopher D. Manning, and Ng Andrew Y. 2013. Parsing with compositional vector grammars. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 455–465, Sofia, Bulgaria, August. Association for Computational Linguistics.

Daniel Zeman and Zdeněk Žabokrtskỳ. 2005. Improving parsing accuracy by combining diverse dependency parsers. In *Proceedings of the Ninth International Workshop on Parsing Technology*, pages 171–178. Association for Computational Linguistics.

Hui Zhang, Min Zhang, Chew Lim Tan, and Haizhou Li. 2009. K-best combination of syntactic parsers. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1552–1560, Singapore, August. Association for Computational Linguistics.

Not All Contexts Are Created Equal: Better Word Representations with Variable Attention

Wang Ling **Lin Chu-Cheng** **Yulia Tsvetkov** **Silvio Amir**
Ramón Fernandez Astudillo **Chris Dyer** **Alan W Black** **Isabel Trancoso**

L^{2}F Spoken Systems Lab, INESC-ID, Lisbon, Portugal
Language Technologies Institute, Carnegie Mellon University, Pittsburgh, PA, USA
Instituto Superior Técnico, Lisbon, Portugal
{lingwang,chuchenl,ytsvetko,cdyer,awb}@cs.cmu.edu
{ramon.astudillo,samir,isabel.trancoso}@inesc-id.pt

Abstract

We introduce an extension to the bag-of-words model for learning words representations that take into account both syntactic and semantic properties within language. This is done by employing an attention model that finds within the contextual words, the words that are relevant for each prediction. The general intuition of our model is that some words are only relevant for predicting local context (e.g. function words), while other words are more suited for determining global context, such as the topic of the document. Experiments performed on both semantically and syntactically oriented tasks show gains using our model over the existing bag of words model. Furthermore, compared to other more sophisticated models, our model scales better as we increase the size of the context of the model.

1 Introduction

Learning word representations using raw text data have been shown to improve many NLP tasks, such as part-of-speech tagging (Collobert et al., 2011), dependency parsing (Chen and Manning, 2014; Kong et al., 2014) and machine translation (Liu et al., 2014; Kalchbrenner and Blunsom, 2013; Devlin et al., 2014; Sutskever et al., 2014). These embeddings are generally learnt by defining an objective function, which predicts words conditioned on the context surrounding those words. Once trained, these can be used as features (Turian et al., 2010), as initializations of other neural networks (Hinton and Salakhutdinov, 2012; Erhan et al., 2010; Guo et al., 2014).

The continuous bag-of-words (Mikolov et al., 2013) is one of the many models that learns word representations from raw textual data. While these models are adequate for learning semantic features, one of the problems of this model is the lack of sensitivity for word order, which limits their ability of learn syntactically motivated embeddings (Ling et al., 2015a; Bansal et al., 2014). While models have been proposed to address this problem, the complexity of these models ("Structured skip-n-gram" and "CWindow") grows linearly as size of the window of words considered increases, as a new set of parameters is created for each relative position. On the other hand, the continuous bag-of-words model requires no additional parameters as it builds the context representation by summing over the embeddings in the window and its performance is an order of magnitude higher than of other models.

In this work, we propose an extension to the continuous bag-of-words model, which adds an attention model that considers contextual words differently depending on the word type and its relative position to the predicted word (distance to the left/right). The main intuition behind our model is that the prediction of a word is only dependent on certain words within the context. For instance, in the sentence *We won the game! Nicely played!*, the prediction of the word *played*, depends on both the syntactic relation from *nicely*, which narrows down the list of candidates to verbs, and on the semantic relation from *game*, which further narrows down the list of candidates to verbs related to games. On the other hand, the words *we* and *the* add very little to this particular prediction. On the other hand, the word *the* is important for predicting the word *game*, since it is generally followed by nouns. Thus, we observe that the same

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1367–1372,
Lisbon, Portugal, 17-21 September 2015.

word can be informative in some contexts and not in others. In this case, distance is a key factor, as the word the is informative to predict the immediate neighboring words, but not distance ones.

2 Attention-Based Continuous Bag-of-words

2.1 Continuous Bag-Of-Words (CBOW)

The work in (Mikolov et al., 2013) is frequently used to learn word embeddings. It defines projection matrix $\mathbf{W} \in \Re^{d \times |V|}$ where d is the embedding dimension with the vocabulary V. These parameters are optimized by by maximizing the likelihood that words are predicted from their context. Two models were defined, the skip-gram model and the continuous bag-of-words model. In this work, we focus on the continuous bag-of-words model. The CBOW model predicts the center word w_0 given a representation of the surrounding words $\mathbf{w}_{-b}, \ldots, \mathbf{w}_{-1}, \mathbf{w}_1, \mathbf{w}_b$, where b is a hyperparameter defining the window of context words. The context vector is obtained by averaging the embeddings of each word $\mathbf{c} = \frac{1}{2b} \sum_{i \in [-b,b]-\{0\}} \mathbf{w}_i$ and the prediction of the center word $\mathbf{w}_0$ is obtained by performing a softmax over all the vocabulary V. More formally, define the output matrix $\mathbf{O} \in \Re^{|V| \times d_w}$, which maps the context vector $\mathbf{c}$ into a $|V|$-dimensional vector representing the predicted word, and maximizes the following probability:

$$p(\mathbf{v}_0 \mid \mathbf{w}_{[-b,b]-\{0\}}) = \frac{\exp \mathbf{v}_0^\top \mathbf{O}\mathbf{c}}{\sum_{\mathbf{v} \in V} \exp \mathbf{v}^\top \mathbf{O}\mathbf{c}} \quad (1)$$

where $\mathbf{O}\mathbf{c}$ corresponds to the projection of the context vector $\mathbf{c}$ onto the vocabulary V and $\mathbf{v}$ is a one-hot representation. For larger vocabularies it is inefficient to compute the normalizer $\sum_{\mathbf{v} \in V} \exp \mathbf{v}^\top \mathbf{O}\mathbf{c}$. Solutions for problem are using the hierarchical softmax objective function (Mikolov et al., 2013) or resorting to negative sampling to approximate the normalizer (Goldberg and Levy, 2014).

The continuous bag-of-words model differs from other proposed models in the sense that its complexity does not rise substantially as we increase the window b, since it only requires two extra additions to compute $\mathbf{c}$, which correspond to d_w operations each. On the other hand, the skip-n-gram model requires two extra predictions corresponding to $d_w \times V$ operations each, which is an order of magnitude more expensive even when subsampling V. However, the drawback the bag-of-words model is that it does not learn embeddings that are prone for learning syntactically oriented tasks, mainly due to lack of sensitivity to word order, since the context is defined by a sum of surrounding words. Extensions are proposed in (Ling et al., 2015a), where the sum if replaced by the concatenation of the word embeddings in the order these occur. However, this model does not scale well as b increases as it requires $V \times d_w$ more parameters for each new word in the window.

Finally, setting a good value for b is difficult as larger values may introduce a degenerative behavior in the model, as more effort is spent predicting words that are conditioned on unrelated words, while smaller values of b may lead to cases where the window size is not large enough include words that are semantically related. For syntactic tasks, it has been shown that increasing the window size can adversely impact in the quality of the embeddings (Bansal et al., 2014; Lin et al., 2015).

2.2 CBOW with Attention

We present a solution to these problems while maintaining the efficiency underlying the bag-of-words model, and allowing it to consider contextual words within the window in a non-uniform way. We first rewrite the context window c as:

$$\mathbf{c} = \sum_{i \in [-b,b]-\{0\}} a_i(w_i)\mathbf{w}_i \quad (2)$$

where we replace the average of the word embeddings with a weighted sum of the individual word embeddings within the context. That is, each word is w_i at relative position i is attributed an attention level representing how much the attention model believes this it is important to look at in order to predict the center word. The attention $a_i(w)$ given to word $w \in V$ at the relative position i is computed as:

$$a_i(w) = \frac{\exp k_{w,i} + s_i}{\sum_{j \in [-b,b]-\{0\}} \exp k_{w,j} + s_j} \quad (3)$$

where $\mathbf{K} \in \Re^{|V| \times 2b}$ (with elements $k_{i,j}$) is a set of parameters that which determines the importance of each word type in each (relative) position, $\mathbf{s} \in \Re^{2b}$ is a bias, which is conditioned only on the relative position. As this is essentially a softmax over context words, the default bag-of-words model can be seen as a special case of this model

where all parameters $\mathbf{K}$ and $\mathbf{s}$ are fixed at zero. Computing the attention of all words in the input requires $2b$ operations, as it simply requires retrieving one value from the lookup matrix $\mathbf{K}$ for each word and one value from the bias $\mathbf{s}$ for each word in the window. Considering that these models must be trainable on billions of tokens, efficiency is paramount. Although more sophisticated attentional models are certainly imaginable (Bahdanau et al., 2014), ours is a good balance of computational efficiency and modeling expressivity.

2.3 Parameter Learning

Gradients of the loss function with respect to the parameters $(\mathbf{W}, \mathbf{O}, \mathbf{K}, \mathbf{s})$ are computed with backpropagation, and parameters are updated after each training instance using a fixed learning rate.

3 Experiments

3.1 Word Vectors

We used a subsample from an English Wikipedia dump[1] containing 10 million documents, containing a total of 530 million tokens. We built word embeddings using the original CBOW and our proposed attentional model on this dataset.

In both cases, word vectors were constructed using window size $b = 20$, which enables us to capture longer-range dependencies between words. We set the embedding size $d_w = 50$ and used a negative sampling rate of 10. Finally, the vocabulary was reduced to words with more than 40 occurrences. In terms of computational speed, the original bag-of-words implementation was able to compute approximately 220k words per second, while our model computes approximately 100k words per second. The slowdown is tied to the fact that we are computing the gradients, the attention model parameters, as well as the word embeddings. On the other hand, the skip-n-gram model process words at only 10k words per second, as it must predict every word in the window b.

Figure 1 illustrates the attention model for the prediction of the word *south* in the sentence *antartica has little rainfall with the south pole making it a continental desert*. Darker cell indicate higher attention values from $a(i, w)$. We can observe that function words (has, the and a) tend to be attributed very low attentions, as these are generally less predictive power. On the other hand,

[1]Collected in September of 2014

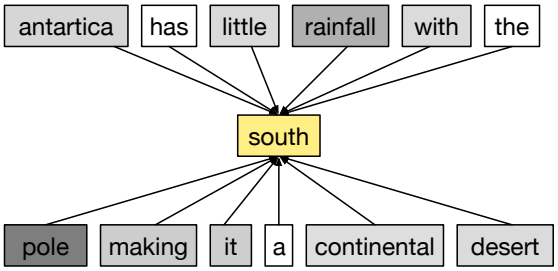


Figure 1: Illustration of the inferred attention parameters for a sentence from our training data when predicting the word *south*; darker cells indicate higher weights.

content words, such as antartica, rainfall, continental and desert are attributed higher weights as these words provide hints that the predicted word is likely to be related to these words. Finally, the word *pole* is assigned the highest attention as it close to the predicted word, and there is a very likely chance that *south* will precede *pole*.

3.2 Syntax Evaluation

For syntax, we evaluate our embeddings in the domain of part-of-speech tagging in both supervised (Ling et al., 2015b) and unsupervised tasks (Lin et al., 2015). This later task is newly proposed, but we argue that success in it is a compelling demonstration of separation of words into syntactically coherent clusters.

Part-of-speech induction. The work in (Lin et al., 2015) attempts to infer POS tags with a standard bigram hmm, which uses word embeddings to infer POS tags without supervision. We use the same dataset, obtained from the ConLL 2007 shared task (Nivre et al., 2007) Scoring is performed using the V-measure (Rosenberg and Hirschberg, 2007), which is used to predict syntactic classes at the word level. It has been shown in (Lin et al., 2015) that word embeddings learnt from structured skip-ngrams tend to work better at this task, mainly because it is less sensitive to larger window sizes. These results are consistent with our observations found in Table 1, in rows "Skip-ngram" and "SSkip-ngram". We can observe that our attention based CBOW model (row "CBOW Attention") improves over these results for both tasks and also the original CBOW model (row "CBOW").

	POS Induction	POS Tagging	Sentiment Analysis
CBOW	50.40	97.03	71.99
Skip-ngram	33.86	97.19	**72.10**
SSkip-ngram	47.64	**97.40**	69.96
CBOW Attention	**54.00**	97.39	71.39

Table 1: Results for unsupervised POS induction, supervised POS tagging and Sentiment Analysis (one per column) using different types of embeddings (one per row).

Part-of-speech tagging. The evaluation is performed on the English PTB, with the standard train (Sections 0-18), dev (Sections 19-21) and test (Sections 22-24) splits. The model is trained with the Bidirectional LSTM model presented in (Ling et al., 2015b) using the same hyper-parameters. Results on the POS accuracy on the test set are reported on Table 1. We can observe our model can obtain similar results compared to the structured skip-ngram model on this task, while training the model is significantly faster. The gap between the usage of different embeddings is not as large as in POS induction, as this is a supervised task, where pre-training generally leads to smaller improvements.

3.3 Semantic Evaluation

To evaluate the quality of our vectors in terms of semantics, we use the sentiment analysis task **(Senti)** (Socher et al., 2013), which is a binary classification task for movie reviews. We simply use the mean of the word vectors of words in a sentence, and use them as features in an ℓ_2-regularized logistic regression classifier. We use the standard training/dev/test split and report accuracy on the test set in table 1.

We can see that in this task, our models do not perform as well as the CBOW and Skipngram model, which hints that our model is learning embeddings that learn more towards syntax. This is expected as it is generally uncommon for embeddings to outperform existing models on both syntactic and semantic tasks simultaneously, as embeddings tend to be either more semantically or syntactically oriented. It is clear that the skipngram model learns embeddings that are more semantically oriented as it performs badly on all syntactic tasks. The structured skip-ngram model on the other hand performs badly on the syntactic tasks, but we observe a large drop on this semantically oriented task. Our attention-based model, on the other hand, out performs all other models on syntax-based tasks, while maintaining a competitive score on semantic tasks. This is an encouraging result that shows that it is possible to learn representations that can perform well on both semantic and syntactic tasks.

4 Related Work

Many methods have been proposed for learning word representations. Earlier work learns embeddings using a recurrent language model (Collobert et al., 2011), while several simpler and more lightweight adaptations have been proposed (Huang et al., 2012; Mikolov et al., 2013). While most of the learnt vectors are semantically oriented, work has been done in order to extend the model to learn syntactically oriented embeddings (Ling et al., 2015a). Attention models are common in vision related tasks (Tang et al., 2014), where models learn to pay attention to certain parts of a image in order to make accurate predictions. This idea has been recently introduced in many NLP tasks, such as machine translation (Bahdanau et al., 2014). In the area of word representation learning, no prior work that uses attention models exists to our knowledge.

5 Conclusions

In this work, we presented an extension to the CBOW model by introducing an attention model to select relevant words within the context to make more accurate predictions. As consequence, the model learns representations that are both syntactic and semantically motivated that do not degrade with large window sizes, compared to the original CBOW and skip-ngram models. Efficiency is maintained by learning a position-based attention model, which can compute the attention of surrounding words with a relatively small number of operations. Finally, we show improvements on syntactically oriented tasks, without degrading results significantly on semantically oriented tasks.

Acknowledgements

The PhD thesis of Wang Ling is supported by FCT grant SFRH/BD/51157/2010. This research was supported in part by the U.S. Army Research Laboratory, the U.S. Army Research Office under contract/grant number W911NF-10-1-0533 and NSF IIS-1054319 and FCT through the pluri-anual contract UID/CEC/50021/2013 and grant number SFRH/BPD/68428/2010.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2014. Tailoring continuous word representations for dependency parsing. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

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

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

Jacob Devlin, Rabih Zbib, Zhongqiang Huang, Thomas Lamar, Richard Schwartz, and John Makhoul. 2014. Fast and robust neural network joint models for statistical machine translation. In *52nd Annual Meeting of the Association for Computational Linguistics, Baltimore, MD, USA, June*.

Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. 2010. Why does unsupervised pre-training help deep learning? *The Journal of Machine Learning Research*, 11:625–660.

Yoav Goldberg and Omer Levy. 2014. word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method. *arXiv preprint arXiv:1402.3722*.

Jiang Guo, Wanxiang Che, Haifeng Wang, and Ting Liu. 2014. Revisiting embedding features for simple semi-supervised learning. In *Proceedings of EMNLP*.

Geoffrey E Hinton and Ruslan Salakhutdinov. 2012. A better way to pretrain deep boltzmann machines. In *Advances in Neural Information Processing Systems*, pages 2447–2455.

Eric H Huang, Richard Socher, Christopher D Manning, and Andrew Y Ng. 2012. Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 873–882. Association for Computational Linguistics.

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

Lingpeng Kong, Nathan Schneider, Swabha Swayamdipta, Archna Bhatia, Chris Dyer, and Noah A. Smith. 2014. A dependency parser for tweets. In *Proc. of EMNLP*, pages 1001–1012, Doha, Qatar, October.

Chu-Cheng Lin, Waleed Ammar, Chris Dyer, and Lori Levin. 2015. Unsupervised POS induction with word embeddings. In *Proceedings of NAACL*.

Wang Ling, Chris Dyer, Alan Black, and Isabel Trancoso. 2015a. Two/too simple adaptations of word2vec for syntax problems. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics.

Wang Ling, Tiago Luíis, Luíis Marujo, Rámon Fernandez Astudillo, Silvio Amir, Chris Dyer, Alan W Black, and Isabel Trancoso. 2015b. Finding function in form: Compositional character models for open vocabulary word representation. EMNLP.

Shujie Liu, Nan Yang, Mu Li, and Ming Zhou. 2014. A recursive recurrent neural network for statistical machine translation. In *Proceedings of ACL*, pages 1491–1500.

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

Joakim Nivre, Johan Hall, Sandra Kübler, Ryan McDonald, Jens Nilsson, Sebastian Riedel, and Deniz Yuret. 2007. The CoNLL 2007 shared task on dependency parsing. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 915–932, Prague, Czech Republic, June. Association for Computational Linguistics.

Andrew Rosenberg and Julia Hirschberg. 2007. V-measure: A conditional entropy-based external cluster evaluation measure. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning(EMNLP-CoNLL)*, pages 410–420.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng,

and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proc. of EMNLP*.

Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112.

Yichuan Tang, Nitish Srivastava, and Ruslan R Salakhutdinov. 2014. Learning generative models with visual attention. In *Advances in Neural Information Processing Systems*, pages 1808–1816.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 384–394, Stroudsburg, PA, USA. Association for Computational Linguistics.

An Improved Non-monotonic Transition System for Dependency Parsing

Matthew Honnibal
spaCy.io
Berlin, Germany
matt@spacy.io

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

Abstract

Transition-based dependency parsers usually use transition systems that monotonically extend partial parse states until they identify a complete parse tree. Honnibal et al. (2013) showed that greedy one-best parsing accuracy can be improved by adding additional non-monotonic transitions that permit the parser to "repair" earlier parsing mistakes by "over-writing" earlier parsing decisions. This increases the size of the set of complete parse trees that each partial parse state can derive, enabling such a parser to escape the "garden paths" that can trap monotonic greedy transition-based dependency parsers.

We describe a new set of non-monotonic transitions that permits a partial parse state to derive a larger set of completed parse trees than previous work, which allows our parser to escape from a larger set of garden paths. A parser with our new non-monotonic transition system has 91.85% directed attachment accuracy, an improvement of 0.6% over a comparable parser using the standard monotonic arc-eager transitions.

1 Introduction

Recent work from Dyer et al. (2015) and Weiss et al. (2015) show that neural network models can improve greedy transition-based parsers dramatically, even beyond the 20% error reduction reported by Chen and Manning (2014). Improvements on beam-search parsing are much more limited, due to the difficulty of applying neural networks to structured prediction.

We suggest that the lack of a ready search solution may present the next barrier to further improvements in accuracy. Some degree of search flexibility seems inherently necessary, no matter how powerful the local model becomes, as even the human sentence processor can be 'garden pathed' by local structural ambiguities.

We take inspiration from Frazier and Rayner (1982) and other psycholinguists and propose repair actions as a light-weight alternative to beam-search. In a transition-based dependency parser, transitions map parse states to parse states, ultimately producing completed parse trees. This process is non-deterministic, since usually more than one transition can apply to a parse state. This means that each partial parse state can be associated with a set of complete parse trees (i.e., the complete parses that can be produced by applying sequences of transitions to the partial parse state). In general adding additional transitions (monotonic or non-monotonic) increases the number of complete parse trees that any given partial parse state can derive.

We explore adding non-monotonic parsing transitions to a greedy arc-eager dependency parser in this paper, in order to permit the parser to recover from attachment errors made early in the parsing process. These additional non-monotonic transitions permit the parser to modify what would have been irrevocable parsing decisions in the monotonic arc-eager system when later information justifies this action. Thus one effect of adding the non-monotonic parsing transitions is to effectively delay the location in the input where the parser must ultimately commit to a particular attachment.

Our transition-system builds on the work of Honnibal et al. (2013) and Nivre and Fernandez-Gonzalez (2014), who each present modifications

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1373–1378,
Lisbon, Portugal, 17-21 September 2015.

to the arc-eager transition system that introduce some non-monotonic behaviour, resulting in small improvements in accuracy. However, these systems only apply non-monotonic transitions to a relatively small number of configurations, so they can only have a small impact on parse accuracy.

We introduce a non-monotonic transition system that combines ideas from these two approaches, and allows substantially more repair capability (and hence search flexibility). We observe a 0.6% improvement in accuracy on the OntoNotes corpus, which is an error reduction of 6.25% over a competitive baseline. A parser using our transition system is guaranteed to run in linear time, and the modifications to the algorithm have no negative impact on run-time in our implementation.

Very recently there has been considerable success in applying neural network models to predict which transition to apply in greedy one-best transition-based parsing. In their preprints, both Dyer et al. (2015) and Weiss et al. (2015) report error reductions of around 20-30% for greedy one-best parsing, and much more modest improvements for transition-based parsers with beam search. Because the neural network approaches improve the local model that predicts which transition to apply next, while this paper suggests changes to the transition system itself, it is reasonable to expect that the improvements reported here are largely orthogonal to those obtained using the neural network techniques. In future work we would like to explore integrating such neural network models of transition prediction with the extended transition system proposed here.

2 Improved non-monotonic transition system

Our transition-system is based on the tree-constrained arc-eager system of Nivre and Fernandez-Gonzalez (2014), which extends the classic arc-eager system (Nivre, 2003) with a new non-monotonic operation that they call "Unshift". They introduce the Unshift action to repair configurations where the buffer is exhausted and the stack contains multiple words that are without incoming arcs (i.e. without governors). The original arc-eager configuration outputs partial parses in this situation.

Nivre and Fernandez-Gonzalez restrict their Unshift action, such that it can only be applied when the buffer is exhausted and the word on top of the stack has no incoming arc. In this configuration, the Unshift action is the only action that can be applied. The use of the new action is therefore entirely deterministic, and they do not need to produce example configurations for the Unshift action during training. They train their model with what Goldberg and Nivre (2012) term a 'static oracle', which can only label configurations that are consistent with the gold-standard parse.

We take the Nivre and Fernandez-Gonzalez (2014) Unshift operation, and import it into the non-monotonic parsing model of Honnibal et al. (2013), which uses a dynamic oracle to determine the gold-standard actions for configurations produced by the parser. This training strategy is critical to the success of a non-monotonic transition system. The model cannot learn to recover from previous errors if the training data cannot contain configurations that result from incorrect actions.

Honnibal et al. (2013) allow the parser to correct prior misclassifications between the Shift and Right-Arc actions. Both of these actions push the first word of the buffer onto the stack, but the Right-Arc action also adds an arc. After the Right-Arc is applied, the top two words of the stack are connected.

In the original arc-eager system, the presence or absence of this arc determines which of the two pop moves, Reduce or Left-Arc, is valid. If the arc is present, then Left-Arc is excluded; if it is absent, the Reduce action is excluded. Honnibal et al. (2013) argue that these deterministic constraints are unmotivated when the parser is trained using a dynamic, instead of static, oracle. Instead of a constraint, they suggest that consistency be achieved by refining the logic of the actions, so that they have a broader applicability. Instead of preventing the Left-Arc from applying when the word on top of the stack has an incoming arc, they update the definition of the Left-Arc so that it first deletes the existing arc if necessary. A corresponding change is made to the Reduce action: if the model predicts Reduce when the word on top of the stack has no incoming arc, the 'missing' arc is inserted. The arc is labelled by noting the best-scoring Right-Arc label on each Shift action, so that the label can be assigned during non-monotonic Reduce.

We show that the Nivre and Fernandez-Gonzalez Unshift operation serves as a far superior non-monotonic Reduce action than the one

Notation			
$(\sigma, \beta, \mathbf{A}, \mathbf{S})$ is a configuration, where $\sigma \vert s$ is a stack of word indices with topmost element s $b \vert \beta$ is a buffer of word indices with first element b $\mathbf{A}$ is a vector of head indices $\mathbf{A}(i) = j$ denotes an arc $w_j \rightarrow w_i$ $\mathbf{S}$ is a bit-vector used to prevent Shift/Unshift cycles			
Initial	$([\,], [1...n], \mathbf{A}(1) = 1)$		
Terminal	$([i], [\,], \mathbf{A})$		
Shift	$(\sigma, b \vert \beta, \mathbf{A}, \mathbf{S}(b) = 0)$	$\Rightarrow$	$(\sigma \vert b, \beta, \mathbf{A}, \mathbf{S}(b) = 1)$
Right-Arc	$(\sigma \vert s, b \vert \beta, \mathbf{A}, \mathbf{S})$	$\Rightarrow$	$(\sigma \vert s \vert b, \beta, \mathbf{A}(b) = s, \mathbf{S})$
Reduce	$(\sigma \vert s, \beta, \mathbf{A}(s) \neq 0, \mathbf{S})$	$\Rightarrow$	$(\sigma, \beta, \mathbf{A}, \mathbf{S})$
Unshift	$(\sigma \vert s, \beta, \mathbf{A}(s) = 0, \mathbf{S})$	$\Rightarrow$	$(\sigma, s \vert \beta, \mathbf{A}, \mathbf{S})$
Left-Arc	$(\sigma \vert s, b \vert \beta, \mathbf{A}, \mathbf{S})$	$\Rightarrow$	$(\sigma, s \vert \beta, \mathbf{A}(s) = b, \mathbf{S}$

Table 1: Our non-monotonic transition system, which integrates the Unshift action of Nivre and Fernandez-Gonzalez (2014) into the model of Honnibal et al. (2013).

Honnibal et al. use in their system, and that the resulting transition system improves parse accuracy by considerably more than either the Honnibal et al or Nivre et al systems do.

2.1 Definition of Transition System

The hybrid transition system is defined in Table 1. Arcs are stored in a vector, $\mathbf{A}$, where the entry $\mathbf{A}(i) = j$ denotes an arc $w_j \rightarrow w_i$. Words are pushed from the buffer β onto the stack σ, using either the Shift or the Right-Arc actions.

If a word was pushed with the Shift action, it will not have an incoming arc. The new Unshift action will then be valid, at any point at which the word is on top of the stack — even after many actions have been performed.

The Unshift action pops the top word of the stack, s, and places it at the start of the buffer. Parsing then proceeds as normal. To prevent cycles, the Shift action checks and sets a bit in the new boolean vector $\mathbf{S}$. The Shift action is invalid if $\mathbf{S}(b) = 1$, for a word b at the front of the buffer. This bit will be set if the word was previously Shifted, and then Unshifted.

At worst, each word can be pushed and popped from the stack twice, so parsing is guaranteed to terminate after a maximum of $4n$ transitions for a sentence of length n.

The terminal condition is reached when the buffer is exhausted and exactly one word remains on the stack. This word will be deemed the root of the sentence. No 'dummy' root token is necessary, removing the need to choose whether the token is placed at the beginning or end of the buffer (Ballesteros and Nivre, 2013).

Note that if the two words each seem like the governor of the sentence, such that the parser deems all incoming arcs to these words unlikely, the transition system is guaranteed to arrive at a configuration where these two words are adjacent to each other. The model can then predict an arc between them, initiated by either word.

2.2 Dynamic Training Oracle

Goldberg and Nivre (2013) describe three questions that need to be answered in order to implement their training algorithm.

Exploration Policy: *When do we follow an incorrect transition, and which one do we follow?*
We always follow the predicted transition, i.e. their two hyper-parameters are set $k = 1$ and $p = 1.0$.

Optimality: *What constitutes an optimal transition in configurations from which the gold tree is not reachable?*
We follow Honnibal et al. (2013) in defining a transition as optimal if it:

1. Renders no additional arcs unreachable using the *monotonic* arc-eager transitions; and
2. Renders no additional arcs unreachable using the *non-monotonic* transitions.

Said another way, we mark a transition as optimal if it leads to an analysis with as few errors as possible, and in cases of ties, uses as few non-monotonic transitions as possible.

For example, given the input string *I saw Jack*, consider a configuration where *saw* is on the stack, *Jack* is at the front of the buffer, and *I* is attached to *saw*. The gold arcs are *saw* → *I* and *saw* → *Jack*. In the monotonic system, the Shift action would make the gold arc *saw* → *Jack* newly unreachable. In our system, this arc is still reachable after Shift, via the Unshift action, but we consider the Shift move non-optimal, so that the non-monotonic actions are reserved as "repair" operations.

Oracle: *Given a definition of optimality, how do we calculate the set of optimal transitions in a given configuration?*

Goldberg and Nivre (2013) show that with the monotonic arc-eager actions, the following arcs are reachable from an arbitrary configuration:

1. Arcs $\{w_i \rightarrow w_j : i \in \sigma,\ j \in \beta\}$ — i.e. all arcs from stack words to buffer words;
2. Arcs $\{w_i \rightarrow w_j : i \in \beta,\ j \in \sigma,\ \mathbf{A}(j) = 0\}$ — i.e. all arcs from buffer words to headless stack words;
3. Arcs $\{w_i \rightarrow w_j : i \in \beta,\ j \in \beta\}$ — i.e. all arcs between words in buffer.

Our non-monotonic actions additionally allow the following arcs to be reached:

4. Arcs $\{w_i \rightarrow w_j : i \in \beta,\ j \in \sigma, \mathbf{A}(j) \neq 0\}$ (LeftArc can now "clobber" existing heads)
5. Arcs $\{w_i \rightarrow w_j$ or $w_j \rightarrow w_i : i, j \in \sigma, i < j, \mathbf{A}(j) = 0\}$ — i.e. if a word i is on the stack, it can reach an arc to or from a word j ahead of it on the stack if that word does not have a head set.

In practice, we therefore only need to add two rules to determine the set of optimal transitions:

1. If σ_0 has a head, and its true head is in the buffer, the Reduce action is now non-optimal.
2. If σ_0 does not have a head, and its true head is in the stack, the LeftArc action is now non-optimal.

The oracle calculation is simple because the system preserves the *arc decomposition* property that Goldberg and Nivre (2013) prove for the arc eager system: if two arcs of a projective tree are individually reachable from a configuration, a projective tree that includes both arcs is also reachable. To see that this property is preserved in our system, consider that an arc $h \rightarrow d$ between two stack words is only unreachable if $h < d$ and $A(d) \neq 0$. But a projective tree with arc $h \rightarrow d$ cannot also have an arc $x \rightarrow y$ such that $h < x < d < y$. So there can be no other arc part of the same projective tree as $h \rightarrow d$ that would require d to be assigned to some other head.

3 Training Procedure

We follow Honnibal et al. (2013) in using the dynamic oracle-based search-and-learn training strategy introduced by Goldberg and Nivre (2012). A dynamic oracle is a function that labels configurations with gold-standard actions. Importantly, a dynamic oracle can label arbitrary configurations, while a so-called 'static' oracle can only assign labels to configurations that are part of gold-standard derivations.

We employ the dynamic oracle in an online learning strategy, similar to imitation-based learning, where the examples are configurations produced by following the current model's predictions. The configurations are labelled by the dynamic oracle, which determines which of the available actions excludes the fewest gold-standard arcs.

Often, multiple actions will be labelled as gold-standard for a given configuration. This implies either spurious ambiguity (the same analysis reachable via different derivations) or previous errors, such that the best parse reachable by different actions are equally bad. When this occurs, we base the perceptron update on the highest-scoring gold-standard label.

3.1 Single class for Unshift/Reduce

The Unshift and Reduce actions are applicable to a disjoint set of configurations. If the word on top of the stack already has an incoming arc, the Reduce move is valid; otherwise, the Unshift move is valid. For the purpose of training and prediction, we therefore model these actions as a single class, which we interpret based on the configuration. This allows us to learn the Unshift action more effectively, as it is allowed to share a representation with the Reduce move. In preliminary development, we found that assigning a distinct class to the Unshift action was not effective. We plan to evaluate this option more rigorously in future work.

4 Experiments

We implemented a greedy transition-based parser, and used rich contextual features following Zhang and Nivre (2011). We extended the feature set to include Brown cluster features, using the cluster prefix trick described by Koo and Collins (2010). Brown clusters are a standard way to improve the cross-domain performance of supervised linear models. The use of Brown cluster features accounts for the 0.7% improvement in accuracy observed between our baseline parser and the Goldberg and Nivre (2012) result shown in Table 2. The two models are otherwise the same.

Part-of-speech tags were predicted using a greedy averaged perceptron model that achieved 97.2% accuracy on the evaluation data. Most previous work uses a n-way jack-knifing to train the stacked tagger/parser model. For convenience, we instead train the tagger at the same time as the parser, as both allow online learning. We find this makes no difference to accuracy.

Our parsers are trained and evaluated on the same data used by Tetreault et al. (2015) in their recent 'bake-off' of leading dependency parsing models. Specifically, we use the OntoNotes corpus converted into dependencies using the ClearNLP 3.1 converter, with the train / dev / test split of the CoNLL 2012 shared task.

5 Results

We implemented three previous versions of the arc-eager transition system, in order to evaluate the effect of our proposed transition-system on parser accuracy. The four systems differ only in their transition system — they are otherwise identical. All use identical features, and all are trained with the dynamic oracle.

Orig. Arc Eager (Nivre, 2003): the original arc-eager system, which constrains the Reduce and Left-Arc actions to ensure monotonicity; **Prev. Non-Monotic** (Honnibal et al., 2013): relaxes the monotonicity constraints, allowing Left-Arc to "clobber" existing arcs, and inserting missing arcs on Reduce with a simple heuristic; **Tree Constrained** (Nivre and Fernandez-Gonzalez, 2014): adds an Unshift action to the arc-eager system, that is only employed when the buffer is exhausted; **This work**: merges the Unshift action into our previous non-monotonic transition system.

Transition System	Search	UAS	LAS
Orig. Arc Eager	Greedy	91.25	89.40
Tree Constrained	Greedy	91.40	89.50
Prev. Non-Monotonic	Greedy	91.36	89.52
This work	Greedy	91.85	89.91
Chen and Manning (2014)	Greedy	89.59	87.63
Goldberg and Nivre (2012)	Greedy	90.54	88.75
Choi and Mccallum (2013)	Branch	92.26	90.84
Zhang and Nivre (2011)	Beam_{32}	92.24	90.50
Bohnet (2010)	Graph	92.50	90.70

Table 2: Our non-monotonic transition system improves accuracy by 0.6% unlabelled attachment score, for a final score of 91.85 on the OntoNotes corpus.

Table 2 shows the unlabelled and labelled attachment scores of the parsers on the evaluation data. The two previous non-monotonic systems, Prev. Non-monotonic and Tree Constrained, were slightly more accurate than the Orig. Arc Eager system. Our new transition-system had a much bigger impact, improving UAS by 0.6% and LAS by 0.51%. To put the scores in context, we have also included figures reported in a recent survey of the current state-of-the-art (Tetreault et al., 2015). Our parser out-performs existing greedy parsers, and is much more efficient than non-greedy parsers.

6 Conclusions and Future Work

This paper integrates innovations from Honnibal et al. (2013) and Nivre and Fernandez-Gonzalez (2014) to produce a novel non-monotonic set of transitions for transition-based dependency parsing. Doing this required us to use the dynamic oracle of Goldberg and Nivre (2012) during training in order to produce configurations that exercise the non-monotonic transitions. We show that this combination of innovations results in a parser with 91.85% directed accuracy, which is an improvement of 0.6% directed accuracy over an equivalent arc-standard parser. Interestingly, the Honnibal et al and Nivre et al innovations applied on their own only produce improvements of 0.11% and 0.15% respectively, so it seems that these improvements taken together do interact synergistically.

Because our innovation largely affects the search space of a greedy one-best parser, it is likely to be independent of the recent improvements in parsing accuracy that come from using neural networks to predict the best next parsing transition. In future work we plan to combine such neural network models with a version of our parser that incorporates a much larger set of non-monotonic parsing transitions.

References

Miguel Ballesteros and Joakim Nivre. 2013. Going to the roots of dependency parsing. *Computational Linguistics. 39:1.*

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

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of the 53nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13–24. Association for Computational Linguistics, Beijing, China.

Lyn Frazier and Keith Rayner. 1982. Making and correcting errors during sentence comprehension: Eye movements in the analysis of structurally ambiguous sentences. *Cognitive Psychology*, 14(2):178–210.

Yoav Goldberg and Joakim Nivre. 2012. A dynamic oracle for arc-eager dependency parsing. In *Proceedings of the 24th International Conference on Computational Linguistics (Coling 2012)*. Association for Computational Linguistics, Mumbai, India.

Yoav Goldberg and Joakim Nivre. 2013. Training deterministic parsers with non-deterministic oracles. *TACL*, 1:403–414.

Matthew Honnibal, Yoav Goldberg, and Mark Johnson. 2013. A non-monotonic arc-eager transition system for dependency parsing. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 163–172. Association for Computational Linguistics, Sofia, Bulgaria.

Terry Koo and Michael Collins. 2010. Efficient third-order dependency parsers. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1–11.

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

Joakim Nivre and Daniel Fernandez-Gonzalez. 2014. Arc-eager parsing with the tree constraint. *Computational Linguistics*, 40(2):259–267.

Joel Tetreault, Jin ho Choi, and Amanda Stent. 2015. It depends: Dependency parser comparison using a web-based evaluation tool. In *Proceedings of the 53nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13–24. Association for Computational Linguistics, Beijing, China.

David Weiss, Chris Alberti, Michael Collins, and Slav Petrov. 2015. Structured training for neural network transition-based parsing. In *Proceedings of the 53nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13–24. Association for Computational Linguistics, Beijing, China.

Yue Zhang and Joakim Nivre. 2011. Transition-based dependency parsing with rich non-local features. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 188–193. Association for Computational Linguistics, Portland, Oregon, USA.

Improving Statistical Machine Translation with a Multilingual Paraphrase Database

Ramtin Mehdizadeh Seraj, Maryam Siahbani, Anoop Sarkar
School of Computing Science
Simon Fraser University
Burnaby BC. Canada
rmehdiza,msiahban,anoop@cs.sfu.ca

Abstract

The multilingual Paraphrase Database (PPDB) is a freely available automatically created resource of paraphrases in multiple languages. In statistical machine translation, paraphrases can be used to provide translation for out-of-vocabulary (OOV) phrases. In this paper, we show that a graph propagation approach that uses PPDB paraphrases can be used to improve overall translation quality. We provide an extensive comparison with previous work and show that our PPDB-based method improves the BLEU score by up to 1.79 percent points. We show that our approach improves on the state of the art in three different settings: when faced with limited amount of parallel training data; a domain shift between training and test data; and handling a morphologically complex source language. Our PPDB-based method outperforms the use of distributional profiles from monolingual source data.

1 Introduction

Translation coverage is a major concern in statistical machine translation (SMT) which relies on large amounts of parallel, sentence-aligned text. In (Callison-Burch et al., 2006), even with a training data size of 10 million word tokens, source vocabulary coverage in unseen data does not go above 90%. The problem is worse with multi-word OOV phrases. Copying OOVs to the output is the most common solution. However, even noisy translations of OOVs can improve reordering and language model scores (Zhang et al., 2012). Transliteration is useful but not a panacea for the OOV problem (Irvine and Callison-Burch, 2014b). We find and remove the named entities, dates, etc. in the source and focus on the use of paraphrases to help translate the remaining OOVs. In Sec. 5.2 we show that handling such OOVs correctly does improve translation scores.

In this paper, we build on the following research: *Bilingual lexicon induction* is the task of learning translations of words from monolingual data in source and target languages (Schafer and Yarowsky, 2002; Koehn and Knight, 2002; Haghighi et al., 2008). The *distributional profile* (DP) approach uses context vectors to link words as potential paraphrases to translation candidates (Rapp, 1995; Koehn and Knight, 2002; Haghighi et al., 2008; Garera et al., 2009). DPs have been used in SMT to assign translation candidates to OOVs (Marton et al., 2009; Daumé and Jagarlamudi, 2011; Irvine et al., 2013; Irvine and Callison-Burch, 2014a). Graph-based semi-supervised methods extend this approach and propagate translation candidates across a graph with phrasal nodes connected via weighted paraphrase relationships (Razmara et al., 2013; Saluja et al., 2014; Zhao et al., 2015). Saluja et al. (2014) extend paraphrases for SMT from the words to phrases, which we also do in this work. *Bilingual pivoting* uses parallel data instead of context vectors for paraphrase extraction (Mann and Yarowsky, 2001; Schafer and Yarowsky, 2002; Bannard and Callison-Burch, 2005; Callison-Burch et al., 2006; Zhao et al., 2008; Callison-Burch, 2008). Ganitkevitch and Callison-Burch (2014) published a large-scale multilingual Paraphrase Database (PPDB) `http://paraphrase.org` which includes lexical, phrasal, and syntactic paraphrases (available for 22 languages with up to 170 million paraphrases each).

To our knowledge, this paper is the first comprehensive study of the use of PPDB for statistical machine translation model training. Our framework has three stages: 1) a novel graph construction approach for PPDB paraphrases linked

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1379–1390,
Lisbon, Portugal, 17-21 September 2015.

with phrases from parallel training data. 2) Graph propagation that uses PPDB paraphrases. 3) An SMT model that incorporates new translation candidates. Sec. 3 explains these three stages in detail.

Using PPDB has several advantages: 1) Resources such as PPDB can be built and used for many different tasks including but not limited to SMT. 2) PPDB contains many features that are useful to rank the strength of a paraphrase connection and with more information than distributional profiles. 3) Paraphrases in PPDB are often better than paraphrases extracted from monolingual or comparable corpora because a large-scale multilingual paraphrase database such as PPDB can pivot through a large amount of data in many different languages. It is not limited to using the source language data for finding paraphrases which distinguishes it from previous uses of paraphrases for SMT.

PPDB is a natural resource for paraphrases. However, PPDB was not built with the specific application to SMT in mind. Other applications such as text-to-text generation have used PPDB (Ganitkevitch et al., 2011) but SMT brings along a specific set of concerns when using paraphrases: translation candidates should be transferred suitably across paraphrases. There are many cases, e.g. when faced with different word senses where transfer of a translation is not appropriate. Our proposed methods of using PPDB use graph propagation to transfer translation candidates in a way that is sensitive to SMT concerns.

In our experiments (Sec. 5) we compare our approach with the state-of-the-art in three different settings in SMT: 1) when faced with limited amount of parallel training data; 2) a domain shift between training and test data; and 3) handling a morphologically complex source language. In each case, we show that our PPDB-based approach outperforms the distributional profile approach.

2 Paraphrase Extraction

Our goal is to produce translations for OOV phrases by exploiting paraphrases from the multilingual PPDB (Ganitkevitch and Callison-Burch, 2014) by using graph propagation. Since our approach relies on phrase-level paraphrases we compare with the current state of the art approaches that use monolingual data and distributional profiles to construct paraphrases and use graph propagation (Razmara et al., 2013; Saluja et al., 2014).

2.1 Paraphrases from Distributional Profiles

A *distributional profile* (DP) of a word or phrase was first proposed in (Rapp, 1995) for SMT. Given a word f, its distributional profile is:

$$DP(f) = \{\langle A(f, w_i)\rangle \mid w_i \in V\}$$

V is the vocabulary and the surrounding words w_i are taken from a monolingual corpus using a fixed window size. We use a window size of 4 words based on the experiments in (Razmara et al., 2013). DPs need an association measure $A(\cdot, \cdot)$ to compute distances between potential paraphrases. A comparison of different association measures appears in (Marton et al., 2009; Razmara et al., 2013; Saluja et al., 2014) and our preliminary experiments validated the choice of the same association measure as in these papers, namely *Pointwise Mutual Information* (Lin, 1998) (PMI). For each potential context word w_i:

$$A(f, w_i) = log_2 \frac{P(f, w_i)}{P(f)P(w_i)} \quad (1)$$

To evaluate the similarity between two phrases we use cosine similarity. The cosine coefficient of two phrases f_1 and f_2 is:

$$S(f_1, f_2) = cos(DP(f_1), DP(f_2)) = \frac{\sum_{w_i \in V} A(f_1, w_i)A(f_2, w_i)}{\sqrt{\sum_{w_i \in V} A(f_1, w_i)^2}\sqrt{\sum_{w_i \in V} A(f_2, w_i)^2}} \quad (2)$$

where V is the vocabulary. Note that in Eqn. (2) w_i's are the words that appear in the context of f_1 or f_2, otherwise the PMI values would be zero.

Considering all possible candidate paraphrases is very expensive. Thus, we use the heuristic applied in previous works (Marton et al., 2009; Razmara et al., 2013; Saluja et al., 2014) to reduce the search space. For each phrase we keep candidate paraphrases which appear in one of the surrounding context (e.g. *Left__Right*) among all occurrences of the phrase.

2.2 Paraphrases from bilingual pivoting

Bilingual pivoting uses parallel corpora between the source language, F, and a pivot language T. If two phrases, f_1 and f_2, in a same language are paraphrases, then they share a translation in other languages with $p(f_1|f_2)$ as a paraphrase score:

$$S(f_1, f_2) = p(f_1|f_2) = \sum_t p(f_1|t)p(t|f_2) \quad (3)$$

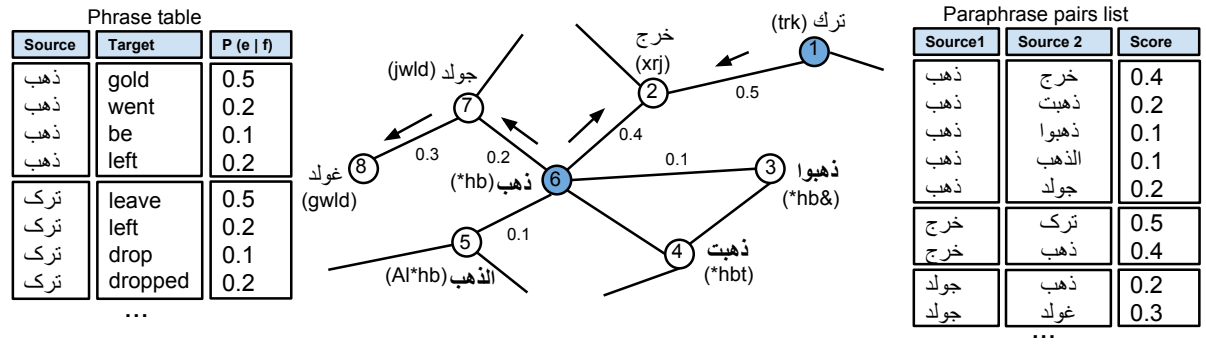


Figure 2: A small sample of the real graph constructed from the Arabic PPDB for Arabic to English translation. Filled nodes (1 and 6) are phrases from the SMT phrase table (unfilled nodes are not). Edge weights are set using a log-linear combination of scores from PPDB. Phrase #6 has different senses ('gold' or 'left'); and it has a paraphrase in phrase #7 for the 'gold' sense and a paraphrase in phrase #2 for the 'left' sense. After propagation, phrase #2 receives translation candidates from phrase #6 and phrase #1 reducing the probability of translation from unrelated senses (like the 'gold' sense). Phrase #8 is a misspelling of phrase #7 and is also captured as a paraphrase. Phrase #6 propagates translation candidates to phrase #8 through phrase #7. Morphological variants of phrase #6 (shown in bold) also receive translation candidates through graph propagation giving translation candidates for morphologically rich OOVs.

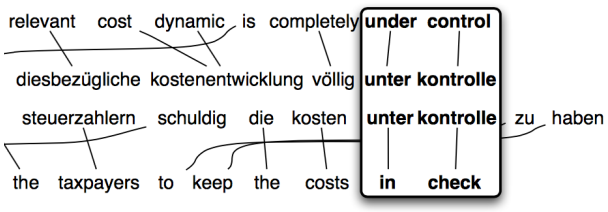


Figure 1: English paraphrases extracted by pivoting over German shared translation (Bannard and Callison-Burch, 2005).

where t is a phrase in language T. $p(f_1|t)$ and $p(t|f_2)$ are taken from the phrase table extracted from parallel data for languages F and T. In Fig. 1 from (Bannard and Callison-Burch, 2005) we see that paraphrase pairs like (*in check*, *under control*) can be extracted by pivoting over the German phrase *unter kontrolle*.

The multilingual Paraphrase Database (PPDB) (Ganitkevitch and Callison-Burch, 2014) is a published resource for paraphrases extracted using bilingual pivoting. It leverages syntactic information and other resources to filters and scores each paraphrase pair using a large set of features. These features can be used by a log linear model to score paraphrases (Zhao et al., 2008). We used a linear combination of these features using the equation in Sec. 3 of (Ganitkevitch and Callison-Burch, 2014) to score paraphrase pairs. PPDB version 1 is broken into different levels of coverage. The smaller sizes contain only better-scoring, high-precision paraphrases, while larger sizes aim for high coverage.

Algorithm 1 PPDB Graph Propagation for SMT

```
PhrTable = PhraseTableGeneration();
ParaDB = ParaphraseExtraction();                    (Sec. 2)
InitGraph = GraphConstruct(PhrTable, ParaDB);       (Sec. 3.1)
PropGraph = GraphPropagation(InitGraph);            (Sec. 3.2)
for phrase ∈ {OOVs} do
    newTrans = TranslationFinder(PropGraph, phrase);
    Augment(PhrTable, newTrans);                    (Sec. 3.3)
TuneMT(PhrTable);
```

3 Methodology

After paraphrase extraction we have paraphrase pairs, (f_1, f_2) and a score $S(f_1, f_2)$ we can induce new translation rules for OOV phrases using the steps in Algo. (1): 1) A graph of source phrases is constructed as in (Razmara et al., 2013); 2) translations are propagated as labels through the graph as explained in Fig. 2; and 3) new translation rules obtained from graph-propagation are integrated with the original phrase table.

3.1 Graph Construction

We construct a graph $G(V, E, W)$ over all source phrases in the paraphrase database and the source language phrases from the SMT phrase table extracted from the available parallel data. V corresponds to the set of vertices (source phrases), E is the set of edges between phrases and W is weight of each using the score function S defined in Sec. 2. V has two types of nodes: seed (labeled) nodes, V_s, from the SMT phrase table, and regular nodes, V_r. Note that in this step OOVs are part of these regular nodes, and we try to find translation in the propagation step for all of these regular nodes. In graph construction and propagation,

we do not know which phrasal nodes correspond to OOVs in the dev and test set. Fig. 2 shows a small slice of the actual graph used in one of our experiments; This graph is constructed using the paraphrase database on the right side of the figure. Filled nodes have a distribution over translations (the possible "labels" for that node). In our setting, we consider the translation e to be the "label" and so we propagate the labeling distribution $p(e|f)$ which is taken from the feature function for the SMT log-linear model that is taken from the SMT phrase table and we propagate this distribution to unlabeled nodes in the graph.

3.2 Graph Propagation

Considering the translation candidates of known phrases in the SMT phrase table as the "labels" we apply a soft label propagation algorithm in order to assign translation candidates to "unlabeled" nodes in the graph, which include our OOV phrases. As described by the example in Fig. 2 we wish two outcomes: 1) transfer of translations (or "labels") to unlabeled nodes (OOV phrases) from labeled nodes, and 2) smoothing the label distribution at each node. We use the Modified Adsorption (MAD) algorithm (Talukdar and Crammer, 2009) for graph propagation. Suppose we have m different possible labels plus one *dummy label*, a soft label $\hat{Y} \in \Delta^{m+1}$ is a $m+1$ dimension probability vector. The dummy label is used when there is low confidence on correct labels. Based on MAD, we want to find soft label vectors for each node by optimizing the objective function below:

$$\begin{aligned}\min_{\hat{Y}} \mu_1 \sum_{v \in V_s} P_{1,v} ||Y_v - \hat{Y}||_2^2 + \\ \mu_2 \sum_{v \in V, u \in N(v)} P_{2,v} W_{v,u} ||\hat{Y}_v - \hat{Y}_u||_2^2 + \\ \mu_3 \sum_{v \in V} P_{3,v} ||\hat{Y}_v - R_v||_2^2\end{aligned} \quad (4)$$

In this objective function, μ_i and $P_{i,v}$ are hyper-parameters ($\forall v : \Sigma_i P_{i,v} = 1$). $R_v \in \Delta^{m+1}$ is our prior belief about labeling. First component of the function tries to minimize the difference of new distribution to the original distribution for the seed nodes. The second component insures that nearby neighbours have similar distributions, and the final component is to make sure that the distribution does not stray from a prior distribution. At the end of propagation, we wish to find a label distribution for our OOV phrases. We describe in Sec. 4.2.2 the reasons for choosing MAD over other graph propagation algorithms. The MAD graph propagation generalizes the approach used in (Razmara et al., 2013). The Structured Label Propagation algorithm (SLP) was used in (Saluja et al., 2014; Zhao et al., 2015) which uses a graph structure on the target side phrases as well. However, we have found that in our diverse experimental settings (see Sec. 5) MAD had two properties we needed compared to SLP: one was the use of graph random walks which allowed us to control translation candidates and MAD also has the ability to penalize nodes with a large number of edges (also see Sec. 4.2.2).

3.3 Phrase Table Integration

After propagation, for each potential OOV phrase we have a list of possible translations with corresponding probabilities. A potential OOV is any phrase which does not appear in training, but could appear in unseen data. We do not look at the dev or test data to produce the augmented phrase table. The original phrase table is now augmented with new entries providing translation candidates for potential OOVs; Last column in Table 2 shows how many entries have been added to the phrase table for each experimental settings. A new feature is added to the standard SMT log-linear discriminative model and introduced into the phrase table. This new feature is set to either 1.0 for the phrase table entries that already existed; or ℓ_i which is the log probability (from graph propagation) for the translation candidate i for potential OOVs. In case the dummy label exists with high probability or the label distribution is uniform, an identity rule is added to the phrase table (copy over source to target).

4 Analysis of the Framework

4.1 Propagation of poor translations

Automatic paraphrase extraction generates many possible paraphrase candidates and many of them are likely to be false positives for finding translation candidates for OOVs. Distributional profiles rely on context information which is not sufficient to derive accurate paraphrases for many phrases and this results in many low quality paraphrase candidates. Bilingual pivoting uses word alignments which can also introduce errors depending on the size and quality of the bilingual data used. Alignment errors also introduce poor translations.

Size	Nodes	Edges	Max Neigh.	Ave Neigh.
S	23K	31K	32	1.38
M	41K	69K	33	1.69
L	74K	199K	67	2.69
XL	103K	548K	330	5.33
XXL	122K	2073K	1231	16.968
XXXL	125K	7558K	5255	60.27

Table 1: Statistics of the graph constructed using the English lexical PPDB. We have built similar graphs for French and Arabic.

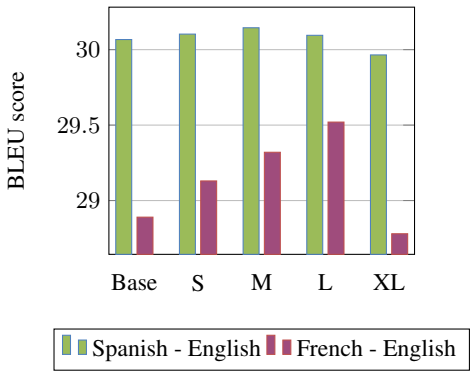


Figure 3: Effect of PPDB size on improving BLEU score for Spanish and French

In graph propagation, these errors may be propagated and result in poor translations for OOVs.

We could address this issue by aggressively pruning the potential paraphrase candidates to improve the precision. However, this results in a dramatic drop in coverage and many OOV phrases do not obtain any translation candidates. We use a combination of the following three steps to augment our graph propagation framework.

4.1.1 Graph pruning and PPDB sizes

Pruning the graph avoids error propagation by removing unreliable edges. Pruning removes edges with an edge weight lower than a minimum threshold or by limiting the number of neighbours to the top-K edges (Talukdar, 2009). PPDB has different sizes with different levels of accuracy and coverage. We can do graph pruning simply by choosing to use different sizes of PPDB. As we can see in Fig. 3 results vary from language to language depending on the pruning used. For instance, the L size results in the best score for French-English. We choose the best size of PPDB for each language based on a separate held-out set and independently from each of the SMT-based tasks in our experimental results. Our conclusion from our experiments with the different sizes of PPDB is that removing phrases (or nodes in our graph) is not desirable. However, removing unreliable edges is useful. As seen in Table 1, increasing the size of PPDB leads to a rapid increase in nodes followed by a larger number of edges in the very large PPDB sizes.

4.1.2 Pruning the translation candidates

Another solution to the error propagation issue is to propagate all translation candidates but when providing translations to OOVs in the final phrase table to eliminate all but the top L translations for each phrase (which is the usual *ttable* limit in phrase-based SMT (Koehn et al., 2003)). Based on a development set, separate from the test sets we used, we found that the best value of L was 10.

4.1.3 External Resources for Filtering

Applying more informative filters can be also used to improve paraphrase quality. This can be done through additional features for paraphrase pairs. For example, edit distance can be used to capture misspelled paraphrases. We use a Named Entity Recognizer to exclude names, numbers and dates from the paraphrase candidates. Even after removing these tokens, 3.32% of tokens of test set are still OOVs . In addition, we use a list of stop words to remove nodes which have too many connections. These two filters improve our results (more in Sec. 5).

4.2 Path sensitivity

Graph propagation has been used in many NLP tasks like POS tagging, parsing, etc. but propagating translations in a graph as labels is much more challenging. Due to huge number of possible labels (translations) and many low quality edges, it is very likely that many wrong translations are rapidly propagated in few steps. Razmara et al. (2013) show that unlabeled nodes inside the graph, called *bridge nodes*, are useful for the transfer of translations when there is no other connection between an OOV phrase and a node with known translation candidates. However, they show that using the full graph with long paths of bridge nodes hurts performance. Thus the propagation has to be constrained using *path sensitivity*. Fig. 4 shows this issue in a part of an English para-

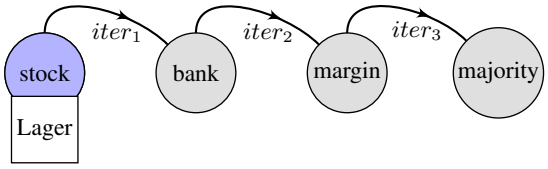


Figure 4: Sensitivity issue in graph propagation for translations. "Lager" is a translation candidate for "stock", which is transferred to "majority" after 3 iterations.

phrase graph. After three iterations, German translation "Lager" reaches "majority" which is totally irrelevant as a translation candidate. Transfer of translation candidates should prefer close neighbours and only with a very low probability to other nodes in the graph.

4.2.1 Pre-structuring the graph

Razmara et al. (2013) avoid a fully connected graph structure. They pre-structure the graph into bipartite graphs (only connections between phrases with known translation and OOV phrases) and tripartite graphs (connections can also go from a known phrasal node to an OOV phrasal node through one node that is a paraphrase of both but does not have translations, i.e. it is an unlabeled node). In these pre-structured graphs there are no connections between nodes of the same type (known, OOV or unlabeled). We apply this method in our low resource setting experiments (Sec. 5.3) to compare our bipartite and tripartite results to Razmara et al. (2013). In the rest of the experiments we use the tripartite approach since it outperforms the bipartite approach.

4.2.2 Graph random walks

Our goal is to limit the number of hops in the propagation of translation candidates preferring closely connected and highly probable edge weights. Optimization for the Modified Adsorption (MAD) objective function in Sec. 3.2 can be viewed as a controlled random walk (Talukdar et al., 2008; Talukdar and Crammer, 2009). This is formalized as three actions: *inject*, *continue* and *abandon* with corresponding pre-defined probabilities P_{inj}, P_{cont} and P_{abnd} respectively as in (Talukdar and Crammer, 2009). A random walk through the graph will transfer labels from one node to another node, and probabilities P_{cont} and P_{abnd} control exploration of the graph. By reducing the values of P_{cont} and increasing P_{abnd} we can control the label propagation process to optimize the quality of translations for OOV phrases. Again, this is done on a held-out development set and not on the test data. The optimal values in our experiments for these probabilities are $P_{inj} = 0.9, P_{cont} = 0.001, P_{abnd} = 0.01$.

4.2.3 Early stopping of propagation

In Modified Adsorption (MAD) (see Sec. 3.2) nodes in the graph that are closely linked will tend to similar label distributions as the number of iterations increase (even when the path lengths increase). In our setting, smoothing the label distribution helps in the first few iterations, but is harmful as the number of iterations increase due to the factors shown in Fig. 4. We use *early stopping* which limits the number of iterations. We varied the number of iterations from 1 to 10 on a held-out dev set and found that 5 iterations was optimal.

5 Evaluation

We first show the effect of OOVs on translation quality, then evaluate our approach in three different SMT settings: low resource SMT, domain shift, and morphologically complex languages. In each case, we compare results of using paraphrases extracted by Distributional Profile (DP) and PPDB in an end-to-end SMT system.
Important: no subset of the test data sentences are used in the bilingual corpora for paraphrase extraction process.

5.1 Experimental Setup

We use CDEC[1] (Dyer et al., 2010) as an end-to-end SMT pipeline with its standard features[2]. `fast_align` (Dyer et al., 2013) is used for word alignment, and weights are tuned by minimizing BLEU loss on the dev set using MIRA (Crammer and Singer, 2003). This setup is used for most of our experiments: oracle (Sec. 5.2), domain adaptation (Sec. 5.4) and morphologically complex languages (Sec. 5.5). But as we wish to fairly compare our approach with Razmara et al. (2013) on low resource setting, we follow their setup in Sec. 5.3: Moses (Koehn et al., 2007) as SMT pipeline, GIZA++ (Och and Ney, 2003) for word alignment and MERT (Och, 2003) for tuning. We add our own feature to the SMT log-linear model as described in Sec. 3.3.

[1] http://www.cdec-decoder.org

[2] EgivenFCoherent, SampleCountF, CountEF, MaxLexFgivenE, MaxLexEgivenF, IsSingletonF, IsSingletonEF

Experiments	OOV type/token	Rules added
Case 1	1830 / 2163	7.0K
Case 2 - Med.	2294 / 4190	7.8K
Case 2 - Sci.	5272 / 14121	10.4K
Case 3	1543 / 1895	8.1K

Table 2: Statistics of settings in Sec. 5. Last column shows how many rules added in the phrase table integration step.

KenLM (Heafield, 2011) is used to train a 5-gram language model on English Gigaword (V5: LDC2011T07). For scalable graph propagation we use the Junto framework[3]. We use maximum phrase length 10. For our experiments we use the Hadoop distributed computing framework executed on a cluster with 12 nodes (each node has 8 cores and 16GB of RAM). Each graph propagation iteration takes about 3 minutes.

For French, we apply a simple heuristic to detect named entities: words that are capitalized in the original dev/test set that do not appear at the beginning of a sentence are named entities. Based on eyeballing the results, this works very well in our data. For Arabic, AQMAR is used to exclude named-entities (Mohit et al., 2012). For each of the experimental settings below we show the OOV statistics in Table 2.

5.2 Impact of OOVs: Oracle experiment

This oracle experiment shows that translation of OOVs beyond named entities, dates, etc. is potentially very useful in improving output translation. We trained a SMT system on 10K French-English sentences from the Europarl corpus(v7) (Koehn, 2005). WMT 2011 and WMT 2012 are used as dev and test data respectively. Table 4 shows the results in terms of BLEU on dev and test. The first row is baseline which simply copies OOVs to output. The second and third rows show the result of augmenting phrase-table by adding translations for single-word OOVs and phrases containing OOVs. The last row shows the oracle result where dev and test sentences exist inside the training data and all the OOVs are known (Fully observers cannot avoid model and search errors).

5.3 Case 1: Limited Parallel Data

In this experiment we use a setup similar to (Razmara et al., 2013). To have fair comparison, we use 10K French-English parallel sentences, randomly chosen from Europarl to train translation system, as reported in (Razmara et al., 2013). ACL/WMT 2005[4] is used for dev and test data. We re-implement their paraphrase extraction method (DP) to extract paraphrases from French side of Europarl (2M sentences). We use unigram nodes to construct graphs for both DP and PPDB. In bipartite graphs, each node is connected to at most 20 nodes. For tripartite graphs, each node is connected to 15 labeled and 5 unlabeled nodes.

Fr-En	Dev	Test
Baseline	27.90	28.08
+ Lexical OOV	28.10	28.31
+ Phrasal OOV	28.50	28.85
Fully observed	46.88	49.21

Table 4: The impact of translating OOVs.

For intrinsic evaluation, we use Mean-Reciprocal-Rank (MRR) and Recall. MRR is the mean of reciprocal rank of the candidate list compared to the gold list (Eqn. 5). Recall shows percentage of gold list covered by the candidate list (Eqn. 6). Gold translations for OOVs are given by concatenating the test data to training and running a word aligner.

$$\text{MRR} = \frac{1}{|O|}\sum_{i=1}^{|O|}\frac{1}{rank_i} \text{ for } O = \{\text{OOVs}\} \quad (5)$$

$$\text{Recall} = \frac{|\{\text{gold list}\} \cap \{\text{candidate list}\}|}{|\{\text{gold list}\}|} \quad (6)$$

Table 5 compares DP and PPDB in terms of BLEU, MRR and Recall. It indicates that PPDB (large size) outperforms DP in both intrinsic and extrinsic evaluation measures. Although tripartite graph did not improve the results for DP, it results in statistically significantly better BLEU score for PPDB in comparison to DP (evaluated by MultEval (Clark et al., 2011)). Thus we use tripartite graph in the rest of experiments. The last row in the table shows the result of combining DP and PPDB by multiplying the normalized scores of both paraphrase lists.

This setting is included for three reasons: 1) we exploit the small data size to explore different choices in our approach such as, e.g. choosing bipartite versus tripartite graph structures; 2)

[3]Junto : https://github.com/parthatalukdar/junto

[4]http://www.statmt.org/wpt05/mt-shared-task/

OOV	PPDB NNs	DP NNs	Reference sentence	PPDB output	DP output
procédés	processus	méthodes outils matériaux	... an agreement on **procedures** in itself is a good thing ...	... an agreement on **the procedure** is a good ...	... an agreement on **products** is a good ...
quantique	quantiques	-	... allowed us to achieve **quantum** degeneracy ...	... allowed **quantum** degeneracy ...	... **quantique** allowed degeneracy ...
mlzm	mlzmA	ADTr	... voted 97-0 last week for a non-**binding** resolution ...	... voted 97 last week on **not binding** resolution ...	... voted 97 last week on **having** resolution ...

Table 3: Examples comparing DP versus PPDB outputs on the test sets. NNs refer to nearest neighbours in the graph for OOV phrase. Each row respectively corresponds to experimental settings (cases 1 to 3).

System	MRR	Recall	BLEU
baseline	-	-	28.89
DP-bipartite	5.34	11.90	29.27
DP-tripartite	5.34	11.95	29.27
PPDB_{fr} (L)-bipartite	**12.05**	22.08	29.46
PPDB_{fr} (L)-tripartite	10.22	**22.87**	**29.52**
Combined-tripartite	-	-	29.28

Table 5: Results of PPDB and DP techniques.

to show how well our PPDB approach does compared to the DP approach in terms of MRR and recall; and 3) to show applicability of our approach for a low-resource language. However we used French instead of a language which is truly resource-poor due to the lack of available paraphrases for a true resource poor language, e.g. Malagasy.

5.4 Case 2: Domain Adaptation

Domain adaptation is another case that suffers from massive number of OOVs. We compare our approach with Marginal Matching (Irvine et al., 2013), a state of the art approach in SMT domain adaptation. We use their setup and data and compare our results to their reported results (Irvine et al., 2013). 250K lines of Hansard parliamentary proceeding are used for training MT. Dev and test sets are available for two different domains: Medical and Science domains. For medical domain random subset of EMEA corpus (Tiedemann, 2009) and for the science domain a corpus of scientific articles (Carpuat et al., 2012) has been used. Unigram paraphrases using DP are extracted from French side of Europarl.

Table 6 compares the results in terms of BLEU score. In both medical and science domains, graph-propagation approach using PPDB (large) performs significantly better than DP ($p < 0.02$), and has comparable results to Marginal Matching.

Systems	Science	Medical
baseline	22.20	25.32
DP-tripartite	22.76	25.81
PPDB_{fr} (L)-tripartite	22.97	**27.11**
Marginal Matching	**23.62**	26.97

Table 6: BLEU scores for domain adaptation.

Systems	BLEU
baseline	29.59
DP-tripartite	30.08
$\text{PPDB}_{\text{arabic}}$ (L)-tripartite	**31.12**

Table 7: BLEU score results for Arabic-English.

Marginal Matching performs better in science domain but graph-propagation approach with PPDB outperforms it in medical domain getting a $+1.79$ BLEU score improvement over the baseline.

5.5 Case 3: Morphologically Rich Languages

Both Distribution Profiling and Bilingual Pivoting propose morphological variants of a word as paraphrase pairs. Even more so in PPDB due to pivoting over English. We choose Arabic-English task for this experiment. We train the SMT system on 685K sentence pairs (randomly selected from LDC2007T08 and LDC2008T09) and use NIST OpenMT 2012 for dev and test data. Arabic side of 1M sentences of LDC2007T08 and LDC2008T09 is used to extract unigram paraphrases for DP. Table 7 shows that PPDB (large; with phrases) resulted in $+1.53$ BLEU score improvement over DP which only slightly improved over baseline.

6 Related Work

Sentence level paraphrasing has been used for generating alternative reference translations (Madnani et al., 2007; Kauchak and Barzilay, 2006), or augmenting the training data with sentential para-

phrases (Bond et al., 2008; Nakov, 2008; Mirkin et al., 2009). Phrase level paraphrasing was done using crowdsourcing (Resnik et al., 2010) or by using paraphrases in lattice decoding (Onishi et al., 2010; Du et al., 2010).

Daumé and Jagarlamudi (2011) apply a generative model to domain adaptation based on canonical correlation analysis Haghighi et al. (2008). However, they use artificially created monolingual corpora very related to the same domain as test data. Irvine and Callison-Burch (2014a) generate a large, noisy phrase table by composing unigram translations which are obtained by a supervised method (Irvine and Callison-Burch, 2013). Comparable monolingual data is used to re-score and filter the phrase table. Zhang and Zong (2013) use a large manually generated lexicon for domain adaptation. In contrast to these methods, our method is unsupervised.

Alexandrescu and Kirchhoff (2009) use a graph-based semi-supervised model determine similarities between sentences, then use it to rerank the n-best translation hypothesis. Liu et al. (2012) extend this model to derive some features to be used during decoding. These approaches are orthogonal to our approach. Saluja et al. (2014) use Structured Label Propagation (Liu et al., 2012) in two parallel graphs constructed on source and target paraphrases. In their case the graph construction is extremely expensive. Leveraging a morphological analyzer, they reach significant improvement on Arabic. We can not directly compare our results to (Saluja et al., 2014) because they exploit several external resources such as a morphological analyzer and also had different sizes of training and test. In experiments (Sec. 5) we obtained comparable BLEU score improvement on Arabic-English by using bilingual pivoting only on source phrases. (Saluja et al., 2014) also use methods similar to (Habash, 2008) that expand the phrase table with spelling and morphological variants of OOVs in test data. We do not use the dev/test data to augment the phrase table.

Using comparable corpora to extract parallel sentences and phrases (Munteanu and Marcu, 2006; Smith et al., 2010; Tamura et al., 2012) are orthogonal to the approach we discuss here.

Bilingual and multilingual word and phrase representation using neural networks have been applied to machine translation (Zou et al., 2013; Mikolov et al., 2013a; Zhang et al., 2014). However, most of these methods focus on frequent words or an available bilingual phrase table (Zou et al., 2013; Zhang et al., 2014; Gao et al., 2014). Mikolov et al. (2013a) learn a global linear projection from source to target using representation of frequent words on both sides. This model can be used to generate translations for new words, but a large amounts of bilingual data is required to create such a model. (Mikolov et al., 2013b) also uses bilingual data to project new translation rules. Zhao et al. (2015) extend Mikolov's model to learn one local linear projection for each phrase. Their model reaches comparable results to Saluja et al. (2014) while works faster. Alkhouli et al. (2014) use neural network phrase representation for paraphrasing OOVs and find translation for them using a phrase-table created from limited parallel data. Our experimental settings is different from the approaches in (Alkhouli et al., 2014; Mikolov et al., 2013a; Mikolov et al., 2013b).

7 Conclusion and Future work

In future work, we would like to include translations for infrequent phrases which are not OOVs. We would like to explore new propagation methods that can directly use confidence estimates and control propagation based on label sparsity. We also would like to expand this work for morphologically rich languages by exploiting other resources like morphological analyzer and campare our approach to the current state of art approaches which are using these types of resources. In conclusion, we have shown significant improvements to the quality of statistical machine translation in three different cases: low resource SMT, domain shift, and morphologically complex languages. Through the use of semi-supervised graph propagation, a large scale multilingual paraphrase database can be used to improve the quality of statistical machine translation.

Acknowledgments

The authors would like to thank Chris Callison-Burch and Juri Ganitkevitch for providing us the latest version of PPDB, the anonymous reviewers for their comments. The research was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC RGPIN 262313 and RGPAS 446348) to the last author.

References

Andrei Alexandrescu and Katrin Kirchhoff. 2009. Graph-based learning for statistical machine translation. In *NAACL 2009*.

Tamer Alkhouli, Andreas Guta, and Hermann Ney. 2014. Vector space models for phrase-based machine translation. In *EMNLP 2014: Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*.

Colin Bannard and Chris Callison-Burch. 2005. Paraphrasing with bilingual parallel corpora. In *ACL 2005*.

Francis Bond, Eric Nichols, Darren Scott Appling, and Michael Paul. 2008. Improving statistical machine translation by paraphrasing the training data. In *IWSLT 2008*.

Chris Callison-Burch, Philipp Koehn, and Miles Osborne. 2006. Improved statistical machine translation using paraphrases. In *NAACL 2006*.

Chris Callison-Burch. 2008. Syntactic constraints on paraphrases extracted from parallel corpora. In *EMNLP 2008*.

Marine Carpuat, H Daumé III, Alexander Fraser, Chris Quirk, Fabienne Braune, Ann Clifton, Ann Irvine, Jagadeesh Jagarlamudi, John Morgan, Majid Razmara, Aleš Tamchyna, Katharine Henry, and Rachel Rudinger. 2012. Domain adaptation in machine translation: Final report. In *2012 Johns Hopkins Summer Workshop*.

Jonathan H. Clark, Chris Dyer, Alon Lavie, and Noah A. Smith. 2011. Better hypothesis testing for statistical machine translation: controlling for optimizer instability. In *ACL 2011*.

Koby Crammer and Yoram Singer. 2003. Ultraconservative online algorithms for multiclass problems. *The Journal of Machine Learning Research*.

Hal Daumé, III and Jagadeesh Jagarlamudi. 2011. Domain adaptation for machine translation by mining unseen words. In *ACL 2011*.

Jinhua Du, Jie Jiang, and Andy Way. 2010. Facilitating translation using source language paraphrase lattices. In *EMNLP 2010*.

Chris Dyer, Adam Lopez, Juri Ganitkevitch, Johnathan Weese, Ferhan Ture, Phil Blunsom, Hendra Setiawan, Vladimir Eidelman, and Philip Resnik. 2010. cdec: A decoder, alignment, and learning framework for finite-state and context-free translation models. In *ACL 2010*.

Chris Dyer, Victor Chahuneau, and Noah A Smith. 2013. A Simple, Fast, and Effective Reparameterization of IBM Model 2. In *NAACL HLT 2013*.

Juri Ganitkevitch and Chris Callison-Burch. 2014. The multilingual paraphrase database. In *LREC 2014*.

Juri Ganitkevitch, Chris Callison-Burch, Courtney Napoles, and Benjamin Van Durme. 2011. Learning sentential paraphrases from bilingual parallel corpora for text-to-text generation. In *NAACL HLT 2011*.

Jianfeng Gao, Xiaodong He, Wen-tau Yih, and Li Deng. 2014. Learning continuous phrase representations for translation modeling. In *ACL 2014*.

Nikesh Garera, Chris Callison-Burch, and David Yarowsky. 2009. Improving translation lexicon induction from monolingual corpora via dependency contexts and part-of-speech equivalences. In *CoNLL 2009*.

Nizar Habash. 2008. Four Techniques for Online Handling of Out-of-Vocabulary Words in Arabic-English Statistical Machine Translation. In *ACL 2008*.

Aria Haghighi, Percy Liang, Taylor Berg-Kirkpatrick, and Dan Klein. 2008. Learning bilingual lexicons from monolingual corpora. In *ACL 2008*.

Kenneth Heafield. 2011. KenLM: faster and smaller language model queries. In *WMT 2011*.

Ann Irvine and Chris Callison-Burch. 2013. Supervised bilingual lexicon induction with multiple monolingual signals. In *NAACL 2013*.

Ann Irvine and Chris Callison-Burch. 2014a. Hallucinating phrase translations for low resource MT. *CoNLL-2014*.

Ann Irvine and Chris Callison-Burch. 2014b. Using comparable corpora to adapt mt models to new domains. *ACL 2014*.

Ann Irvine, Chris Quirk, and Hal Daumé III. 2013. Monolingual marginal matching for translation model adaptation. In *EMNLP 2013*.

David Kauchak and Regina Barzilay. 2006. Paraphrasing for automatic evaluation. In *NAACL 2008*.

Philipp Koehn and Kevin Knight. 2002. Learning a translation lexicon from monolingual corpora. In *ACL 2002 workshop on unsupervised lexical acquisition*.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *NAACL 2003*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, and et al. 2007. Moses: open source toolkit for statistical machine translation. In *ACL 2007*.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *MT summit 2005*, volume 5.

Dekang Lin. 1998. Automatic retrieval and clustering of similar words. In *ACL 1998*.

Shujie Liu, Chi-Ho Li, Mu Li, and Ming Zhou. 2012. Learning translation consensus with structured label propagation. In *ACL 2012*.

Nitin Madnani, Necip Fazil Ayan, Philip Resnik, and Bonnie J Dorr. 2007. Using paraphrases for parameter tuning in statistical machine translation. In *WMT 2007*.

Gideon S. Mann and David Yarowsky. 2001. Multipath translation lexicon induction via bridge languages. In *NAACL 2001*.

Yuval Marton, Chris Callison-Burch, and Philip Resnik. 2009. Improved statistical machine translation using monolingually-derived paraphrases. In *EMNLP 2009*.

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

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

Shachar Mirkin, Lucia Specia, Nicola Cancedda, Ido Dagan, Marc Dymetman, and Idan Szpektor. 2009. Source-language entailment modeling for translating unknown terms. In *ACL-IJCNLP 2009*.

Behrang Mohit, Nathan Schneider, Rishav Bhowmick, Kemal Oflazer, and Noah A Smith. 2012. Recall-oriented learning of named entities in arabic wikipedia. In *EACL 2012*, pages 162–173.

Dragos Stefan Munteanu and Daniel Marcu. 2006. Extracting parallel sub-sentential fragments from non-parallel corpora. In *ACL 2006*.

Preslav Nakov. 2008. Improved statistical machine translation using monolingual paraphrases. In *ECAI 2008: 18th European Conference on Artificial Intelligence*. IOS Press.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Comput. Linguist.*

Franz Josef Och. 2003. Minimum error rate training for statistical machine translation. In *ACL 2003*.

Takashi Onishi, Masao Utiyama, and Eiichiro Sumita. 2010. Paraphrase lattice for statistical machine translation. In *ACL 2010*.

Reinhard Rapp. 1995. Identifying word translations in non-parallel texts. In *ACL 1995*.

Majid Razmara, Maryam Siahbani, Reza Haffari, and Anoop Sarkar. 2013. Graph propagation for paraphrasing out-of-vocabulary words in statistical machine translation. In *ACL 2013*.

Philip Resnik, Olivia Buzek, Chang Hu, Yakov Kronrod, Alex Quinn, and Benjamin B Bederson. 2010. Improving translation via targeted paraphrasing. In *EMNLP 2010*.

Avneesh Saluja, Hany Hassan, Kristina Toutanova, and Chris Quirk. 2014. Graph-based semi-supervised learning of translation models from monolingual data. In *ACL 2014*.

Charles Schafer and David Yarowsky. 2002. Inducing translation lexicons via diverse similarity measures and bridge languages. In *CoNLL 2002*.

Jason R. Smith, Chris Quirk, and Kristina Toutanova. 2010. Extracting parallel sentences from comparable corpora using document level alignment. In *NAACL 2010*.

Partha Pratim Talukdar and Koby Crammer. 2009. New Regularized Algorithms for Transductive Learning. In *European Conference on Machine Learning*.

Partha Pratim Talukdar, Joseph Reisinger, Marius Paşca, Deepak Ravichandran, Rahul Bhagat, and Fernando Pereira. 2008. Weakly-supervised acquisition of labeled class instances using graph random walks. In *EMNLP 2008*.

Partha Pratim Talukdar. 2009. Topics in graph construction for semi-supervised learning. Technical Report MS-CIS-09-13, University of Pennsylvania, Dept of Computer and Info. Sci.

Akihiro Tamura, Taro Watanabe, and Eiichiro Sumita. 2012. Bilingual lexicon extraction from comparable corpora using label propagation. In *EMNLP-CoNLL 2012*.

Jörg Tiedemann. 2009. News from OPUS-A collection of multilingual parallel corpora with tools and interfaces. In *Recent advances in natural language processing*.

Jiajun Zhang and Chengqing Zong. 2013. Learning a phrase-based translation model from monolingual data with application to domain adaptation. In *ACL 2013*.

Jiajun Zhang, Feifei Zhai, and Chengqing Zong. 2012. Handling unknown words in statistical machine translation from a new perspective. In *Natural Language Processing and Chinese Computing*. Springer.

Jiajun Zhang, Shujie Liu, Mu Li, Ming Zhou, and Chengqing Zong. 2014. Bilingually-constrained phrase embeddings for machine translation. In *ACL 2014*.

Shiqi Zhao, Haifeng Wang, Ting Liu, and Sheng Li. 2008. Pivot approach for extracting paraphrase patterns from bilingual corpora. In *ACL 2008*.

Kai Zhao, Hany Hassan, and Michael Auli. 2015. Learning translation models from monolingual continuous representations. In *NAACL 2015*.

Will Zou, Richard Socher, Daniel Cer, and Christopher Manning. 2013. Bilingual word embeddings for phrase-based machine translation. In *EMNLP 2013*.

Learning Semantic Representations for Nonterminals in Hierarchical Phrase-Based Translation

Xing Wang, Deyi Xiong* and **Min Zhang**
Soochow University, Suzhou, China 215006
xingwsuda@gmail.com, {dyxiong, minzhang}@suda.edu.cn

Abstract

In hierarchical phrase-based translation, coarse-grained nonterminal Xs may generate inappropriate translations due to the lack of sufficient information for phrasal substitution. In this paper we propose a framework to refine nonterminals in hierarchical translation rules with real-valued semantic representations. The semantic representations are learned via a weighted mean value and a minimum distance method using phrase vector representations obtained from large scale monolingual corpus. Based on the learned semantic vectors, we build a semantic nonterminal refinement model to measure semantic similarities between phrasal substitutions and nonterminal Xs in translation rules. Experiment results on Chinese-English translation show that the proposed model significantly improves translation quality on NIST test sets.

1 Introduction

Hierarchical phrase-based translation (Chiang, 2007) explores formal synchronous context free grammar (SCFG) rules for translation. Two types of nonterminal symbols are used in translation rules: nonterminal X in ordinary SCFG rules and nonterminal S in glue rules that are specially introduced to concatenate nonterminal Xs in a monotonic manner. The same generic symbol X for all ordinary nonterminals makes it difficult to distinguish and select proper translation rules.

In order to address this issue, researchers either use syntactic labels to annotate nonterminal Xs (Zollmann and Venugopal, 2006; Zollmann and Vogel, 2011; Li et al., 2012; Hanneman and Lavie, 2013), or employ syntactic information from parse trees to refine nonterminals with real-valued vectors (Venugopal et al., 2009; Huang et al., 2013). In addition to syntactic knowledge, semantic structures are also leveraged to refine nonterminals (Gao and Vogel, 2011). All these efforts focus on incorporating linguistic knowledge into hierarchical translation rules.

Unfortunately, syntactic or semantic parsers for many languages are not accessible due to the lack of labeled training data. In contrast, a large amount of unlabeled data are easily available. Therefore, can we mine syntactic or semantic properties for nonterminals from unlabeled data? Or can we exploit these data to refine nonterminals for SMT?

Learning semantic representations for terminals (words, multi-word phrases or sentences) from unlabeled data has achieved substantial progress in recent years (Mitchell and Lapata, 2008; Turian et al., 2010; Socher et al., 2010; Mikolov et al., 2013c; Blunsom et al., 2014). These representations have been used successfully in various NLP tasks. However, there is no attempt to learn semantic representations for nonterminals from unlabeled data. In this paper we propose a framework to learn semantic representations for nonterminal Xs in translation rules. Our framework is established on the basis of real-valued vector representations learned for multi-word phrases, which are substituted with nonterminal Xs during hierarchical rule extraction. We propose a weighted mean value and a minimum distance method to obtain nonterminal representations from representations of their phrasal substitutions. We further build a semantic nonterminal refinement model with semantic representations of nonterminals to compute similarities between phrasal substitutions and nonterminals. In doing so, we want to enhance phrasal substitution and translation rule selection during decoding.

The big challenge here is that thousands of tar-

*Corresponding author

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1391–1400,
Lisbon, Portugal, 17-21 September 2015.

get phrasal substitutions will be generated for one single nonterminal during decoding. Computing vector representations for all these phrases will be very time-consuming. We therefore introduce two different methods to handle it. In the first method, we project representations of source phrases onto their target counterparts linearly/nonlinearly via a neural network. These projected vectors are used as approximations to real target representations to compute semantic similarities. In the second method, we decode sentences in two passes. The first pass collects target phrase candidates from n-best translations of sentences generated by the baseline. The second pass calculates vector representations of these collected target phrases and then computes similarities between them and target-side nonterminals.

Our contributions are two-fold. First, we learn semantic representations for nonterminals from their phrasal substitutions with two different methods. This is the first time, to the best of our knowledge, to induce semantic representations for nonterminals from unlabeled data in the context of SMT. Second, we successfully address the issue of time-consuming target-side phrase-nonterminal similarity computation mentioned above. We incorporate both source-/target-side semantic nonterminal refinement model and their combination based on learned nonterminal representations into translation system. Experiment results show that our method can achieve an improvement of 1.16 BLEU points over the baseline system on NIST MT evaluation test sets.

The rest of this paper is organized as follows. Section 2 briefly reviews related work. Section 3 presents our approach of learning semantic vectors for nonterminals, followed by Section 4 describing the details of our semantic nonterminal refinement model. Section 5 introduces the integration of the proposed model into SMT. Experiment results are reported in Section 6. Finally, we conclude our work in Section 7.

2 Related Work

A variety of approaches have been explored for nonterminal refinement in hierarchical phrase-based translation. These approaches can be categorized into two groups: 1) augmenting the nonterminal symbol X with informative labels, and 2) attaching distributional linguistic knowledge to each nonterminal in hierarchical rules. The former only allows substitution operations with matched labels. The latter normally builds an additional model as a new feature of the log-linear model to incorporate attached knowledge.

Among approaches which directly refine the single label to more fine-grained labels, syntactic and semantic knowledge are explored in various ways. The syntactically augmented translation model (SAMT) proposed by Zollmann and Venugopal (2006) uses syntactic categories extracted from target-side parse trees to augment nonterminals in hierarchical rules. Unfortunately, there is a data sparseness problem in this model due to thousands of extracted syntactic categories. One solution to address this issue is to reduce the number of syntactic categories. Zollmann and Vogel (2011) use word tags, generated by either POS tagger or unsupervised word class induction, instead of syntactic categories. Hanneman and Lavie (2013) coarsen the label set by introducing a label collapsing algorithm to SAMT grammars (Zollmann and Venugopal, 2006). Yet another solution is easing restrictions on label matching. Shen et al. (2009) penalize substitution with unmatched labels while Chiang (2010) uses soft match features to model substitutions with various labels. Similar to Zollmann and Venugopal (2006), Hoang and Koehn (2010) decorate some hierarchical rules with source-side syntax information and use undecorated, decorated, and partially decorated rules in their translation model. Mylonakis and Sima'an (2011) employ source-side syntax-based labels to define a joint probability synchronous grammar. Combinatory Categorial Grammar (CCG) labels or CCG contextual labels are also used to enrich nonterminals (Almaghout et al., 2011; Weese et al., 2012). Li et al. (2012) incorporate head information extracted from source-side dependency structures into translation rules. Besides, semantic knowledge is also used to refine nonterminals. Gao and Vogel (2011) utilize target-side semantic roles to form SRL-aware SCFG rules. Most of approaches introduced here explicitly require syntactic or semantic parsers trained on manually labeled data.

On the other hand, efforts have also been directed towards attaching distributional linguistic knowledge to nonterminals. Venugopal et al. (2009) propose a preference grammar to annotate nonterminals based on preference distributions of syntactic categories. Huang et al. (2010) learn la-

tent syntactic distributions for each nonterminal. They use these distributions to decorate nonterminal Xs in SCFG rules with a real-valued feature vectors and utilize these vectors to measure the similarities between source phrases and applied rules. Similar to this work, Huang et al. (2013) utilize treebank tags based on dependency parsing to learn latent distributions. Cao et al. (2014) attach translation rules with dependency knowledge, which contains both dependency relations inside rules and dependency relations between rules and their contexts.

The difference of our work from these studies is that our semantic representations are learned from unlabeled bilingual (or monolingual) data and do not depend on any linguistic resources, e.g., parsers. We also believe that our model is able to exploit both syntactic and semantic information for nonterminals since vector representations learned in our way are able to capture both syntactic and semantic properties (Turian et al., 2010; Socher et al., 2010).

3 Learning Semantic Representations for Nonterminals

In our framework, semantic representations for nonterminal Xs are automatically induced from word-aligned parallel corpus. In this section, we detail the essential component of our approach, i.e., how to learn semantic vectors for nonterminals and how to project source semantic vectors onto target language semantic space. Before discussing nonterminal representations, we briefly introduce vector representations for words and phrases.

3.1 Prerequisite: Learning Words and Phrases Representations

We employ a neural method, specifically the continuous bag-of-words model (Mikolov et al., 2013a) to learn high-quality vector representations for words. Once we complete the training of the continuous bag-of-words model, word embeddings form an embedding matrix $M \in \mathbb{R}^{d \times |V|}$, where d is a pre-determined embedding dimensionality and each word w in the vocabulary V corresponds to a vector $\vec{v} \in \mathbb{R}^d$. Given the embedding matrix M, mapping words to vectors can be done by simply looking up their respective columns in M.

We further feed these learned word embeddings to recursive autoencoders (RAE) (Socher et al., 2011) for learning phrase representations. In traditional RAE (shown in Figure 1), given two input children representation vectors $\vec{c_1} \in \mathbb{R}^d$ and $\vec{c_2} \in \mathbb{R}^d$, their parent representation $\vec{p}$ can be calculated as follows:

$$\vec{p} = f^{(1)}(W^{(1)}[\vec{c_1}; \vec{c_2}] + b^{(1)}) \quad (1)$$

where $[\vec{c_1}; \vec{c_2}] \in \mathbb{R}^{2d}$ is the concatenation of vectors of two children, $W^{(1)} \in \mathbb{R}^{d \times 2d}$ is a weight matrix, $b^{(1)} \in \mathbb{R}^d$ is a bias term, and $f^{(1)}$ is an element-wise activation function such as $tanh$. The above output representation $\vec{p}$ can be used as a child vector to construct the representation for a larger subphrase. This process is repeated until a binary tree covering the whole input phrase is generated.

In order to evaluate how well the parent vector represents its children, we can reconstruct the children in a reconstruction layer:

$$[\vec{c_1}'; \vec{c_2}'] = f^{(2)}(W^{(2)}\vec{p} + b^{(2)}) \quad (2)$$

where $\vec{c_1}'$ and $\vec{c_2}'$ are the reconstructed children, $W^{(2)}$ is a weight matrix for reconstruction, $b^{(2)}$ is a bias term for reconstruction, and $f^{(2)}$ is an element-wise activation function.

For each node in the generated binary tree, we compute Euclidean distance between the original input vectors and the reconstructed vectors to measure the reconstruction error:

$$E_{rec}([\vec{c_1}; \vec{c_2}]) = \frac{1}{2}\|[\vec{c_1}; \vec{c_2}] - [\vec{c_1}'; \vec{c_2}']\|^2 \quad (3)$$

By minimizing the total reconstruction error over all nonterminal nodes, we can learn parameters of RAE.

Socher et al. (2011) propose a greedy unsupervised RAE as an extension to the above traditional RAE. The main difference is that in the unsupervised RAE there is no tree structure which is given for traditional RAE. It can learn both representations and tree structures of phrases or sentences. In this work, we adopt the unsupervised RAE to learn vector representations for phrases.

3.2 Inducing Nonterminal Representations from Phrase Representations

As we extract hierarchical rules from phrases by replacing subphrases with nonterminal symbols, a nonterminal X is generalized from a number of

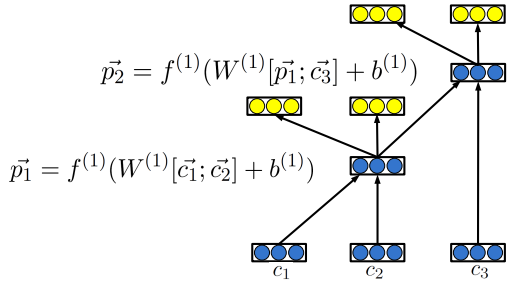


Figure 1: The architecture of a recursive autoencoder, adapted from (Socher et al., 2011). Blue nodes are original vectors and yellow nodes are reconstructed vectors which are used to compute reconstruction errors.

subphrases. We believe that these subphrases determine syntactic and semantic properties of the nonterminal X. We therefore enrich each nonterminal X with a semantic vector induced from vector representations of phrases that are replaced by the nonterminal during rule extraction.

For an SCFG rule, we can learn semantic vectors for nonterminals on both the source and target side. Due to the space limitation, we introduce the procedure of learning nonterminal vectors on the source side. Semantic vectors on the target side can be learned analogically.

For each source-side nonterminal X of a hierarchical rule, we collect all source subphrases replaced by X in a source subphrase set $P = \{p_1, p_2, \cdots, p_m\}$. We also count the number of times of these phrases being replaced by nonterminal X on training data during rule extraction. We collect these numbers in a count set $C = \{c_1, c_2, \cdots, c_m\}$. Based on the phrase set P, count set C and learned phrase vector representations in P, we can compute a semantic vector $\vec{v_x}$ for nonterminal X in each SCFG rule.

We propose two general approaches to obtain semantic vectors for nonterminals: a weighted mean value method and a minimum distance method. Given phrase vector representations $\vec{P_r} = \{\vec{p_1}, \vec{p_2}, \ldots, \vec{p_m}\}$, we calculate the semantic vector for a nonterminal generalized from these phrases as follows.

Weighted mean value method (MV) computes semantic vector $\vec{v_x}$ as:

$$\vec{v_x} = \frac{\sum_{i=1}^{m} c_i \cdot \vec{p_i}}{\sum_{i=1}^{m} c_i} \quad (4)$$

Minimum distance method (MD) finds a point in semantic space to minimize the sum of Euclidean distances of vectors in $\vec{P_r}$ to this point. Formally,

$$\vec{v_x} = \underset{\vec{v_x}}{\operatorname{argmin}} \sum_{i=1}^{m} \sqrt{\sum_{j=1}^{d} (p_{ij} - v_{xj})^2} \quad (5)$$

We use the stochastic gradient descent algorithm to find the minimal distance and the point $\vec{v_x}$. The component v_{xj} can be updated by $v_{xj} \leftarrow v_{xj} + \lambda \frac{\partial f}{\partial v_{xj}}$ where f is $\sum_{i=1}^{m} \sqrt{\sum_{j=1}^{d} (p_{ij} - v_{xj})^2}$ and λ is the learning rate.

Similar to the center of gravity, the semantic vector $\vec{v_x}$ learned by this method acts as a semantic centroid for all vectors of phrases that are substituted by X. Nonterminals in different hierarchical translation rules will have different semantic centroids. These centroids will help translation model capture semantic diversity to a certain degree.

3.3 Mapping Source-Side Representations onto Target-Side Semantic Space

As we discussed in Section 1, directly learning vector representations for target phrases is very costly in practice. Inspired by Mikolov et al. (2013b), we adopt vector projection to alleviate this problem. Different from mapping representations from the source side to the target side by learning a linear matrix on word alignments (Mikolov et al., 2013b), we project source multiword phrase representations onto the target semantic space in a nonlinear manner as we believe that nonlinear relations between languages are more reasonable. Specifically, we use a neural network to achieve this goal. Our neural network is a multilayer feed-forward neural network with one hidden layer. The functional form can be written in the following equation:

$$\vec{p} = \tanh(W^{(4)}(\tanh(W^{(3)} s\vec{r}c) + b^{(3)}) + b^{(4)}) \quad (6)$$

where $s\vec{r}c$ is the input vector which is learned in the source semantic space, $W^{(3)}$ denotes the weight matrix for connections between input and hidden neurons and $W^{(4)}$ denotes the weight matrix for links between hidden neurons and output, $b^{(3)}$ and $b^{(4)}$ are bias terms. To train the neural network, we optimize the following objective:

$$J = \underset{W^{(3)}, W^{(4)}}{\operatorname{argmin}} \frac{1}{N} \sum_{i=1}^{N} \|t\vec{r}g_i - \vec{p_i}\|^2 + R(\theta) \quad (7)$$

where N is the number of training examples, $\vec{trg}_i$ is the target vector representation for the ith example learned by RAE and $\vec{p}_i$ is the output of the neural network for the source vector representation $\vec{src}_i$ of ith example. $R(\theta)$ is the regularizer on parameters:

$$R(\theta) = \frac{\lambda_L}{2}\|W\|^2 \tag{8}$$

where W denotes parameters for parameter matrices $W^{(3)}$, $W^{(4)}$ and bias terms $b^{(3)}$, $b^{(4)}$.

4 Semantic Nonterminal Refinement Model

In this section, we describe our semantic nonterminal refinement model on the basis of induced real-valued semantic vectors for nonterminals.

4.1 Nonterminal Representations in Hierarchical Rules

We incorporate learned semantic representations of nonterminals into hierarchical rules. In particular, ordinary hierarchical rules take the following form:

$$X \rightarrow \langle aX_sb, cX_td \rangle \tag{9}$$

where *a*/*b*, *c*/*d* are strings of terminals on the source and target side, s and t are placeholders denoting the nonterminal X on the source or target side, X_s and X_t are aligned to each other.

Representations for nonterminals can be on either the source or target side. They are attached to hierarchical rules as follows:

$$X \rightarrow \langle aX_sb, cX_td, \vec{v_{xs}}, \vec{v_{xt}} \rangle \tag{10}$$

where $\vec{v_{x.}}$ is the source- or target-side semantic representation for nonterminal. In this way, we keep original translation rules intact and decorate nonterminals with their semantic representations.

4.2 The Model

The proposed semantic nonterminal refinement model estimates the semantic similarity between a phrase p and nonterminal X. The phrase p and nonterminal X will have a high similarity score in the representation space if they are semantically similar. The higher semantic similarity scores are, the more compatible nonterminals are with corresponding phrases.

There is another nonterminal S in glue rules, which are formalized as follows:

$$S \rightarrow \langle S_1X_2, S_1X_2 \rangle \tag{11}$$

$$S \rightarrow \langle X_1, X_1 \rangle \tag{12}$$

This nonterminal S is different from X. We therefore treat it as a special case in the computation of semantic similarity.

In this work, we explore two approaches to compute similarity: one based on cosine similarity and the other based on Euclidean distance.

Given a phrase vector representation $\vec{p}$ and nonterminal X semantic vector $\vec{v_x}$, **Cosine Similarity** (CS) is computed as:

$$cos(\vec{p}, \vec{v_x}) = \frac{\vec{p} \cdot \vec{v_x}}{\|\vec{p}\|\|\vec{v_x}\|} \tag{13}$$

We set α for the Cosine Similarity between the glue rule and its corresponding phrase as follows:

$$SeSim = \begin{cases} cos(\vec{p}, \vec{v_x}) & hierarchical\,rules \\ \alpha & glue\,rules \end{cases} \tag{14}$$

As for **Euclidean Distance** (ED), it is computed according to the following formula:

$$dist(\vec{p}, \vec{v_x}) = \sqrt{\sum_{i=1}^{d}(p_i - v_{xi})^2} \tag{15}$$

and similarly we set β for glue rules:

$$SeSim = \begin{cases} dist(\vec{p}, \vec{v_x}) & hierarchical\,rules \\ \beta & glue\,rules \end{cases} \tag{16}$$

5 Decoding

We incorporate the proposed model as a new feature into the hierarchical phrase-based translation system. Specifically, two features are added into the baseline system:

1. Source-side semantic similarity between source phrases and nonterminals
2. Target-side semantic similarity between target phrases and nonterminals

We compute source- and target-side similarities based on representations of nonterminals and phrasal substitutions for each applied rule, and sum up these similarities to calculate the total score of a derivation on the two features.

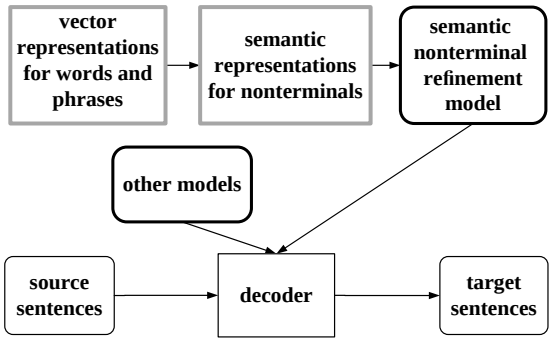


Figure 2: Architecture of SMT system with the proposed semantic nonterminal refinement model.

The integration of the source-side semantic nonterminal refinement model into the decoder is trivial. For the target-side model, however, we have to consider the efficiency issue as we mentioned in Section 1. We introduce two different methods to integrate the target-side model into the decoder: 1) projection and 2) two-pass decoding. In the first integration method, a mapping neural network is trained to map source phrase representations onto the target semantic space as described in Section 3.3. The projection can be linear if we remove the hidden layer in the projection neural network. This is similar to the mapping matrix learned by Mikolov et al. (2013b). We calculate semantic similarities between projected representations of phrases and those of nonterminals. In the two-pass decoding, we collect target phrase candidates from 100-best translations for each source sentence generated by the baseline in the first pass and learn vector representations for these target phrase candidates. Then in the second pass, we decode source sentence with our target semantic nonterminal refinement model using learned target phrase vector representations. If a target phrase appears in the collected set, the target-side semantic nonterminal refinement model will calculate the semantic similarity between the target phrase and the corresponding nonterminal on the target semantic space; otherwise the model will give a penalty. This is because this phrase is not a desirable phrase as it is not used in 100-best translations.

The weights of these two features are tuned by the Minimum Error Rate Training (MERT)(Och, 2003), together with weights of other sub-models on a development set. Figure 2 shows the architecture of SMT system with the proposed semantic nonterminal refinement model.

6 Experiment

In this section, we conducted a series of experiments on Chinese-to-English translation using large-scale bilingual training data, aiming at the following questions:

1. Which approach is better for learning nonterminal representations, weighted mean value or minimum distance?

2. Can the target-side semantic nonterminal refinement model improve translation quality? And which method is better for integrating the target-side semantic model into translation, projection or two-pass decoding?

3. Does the combination of source and target semantic nonterminal refinement models provide further improvement?

6.1 Setup

Our training corpus contains 2.9M sentence pairs with 80.9M Chinese words and 86.4M English words from LDC data[1]. We used NIST MT03 as our development set, NIST MT06 as our development test set and MT08 as our final test set.

We ran Giza++ on the training corpus in both Chinese-to-English and English-to-Chinese directions and applied the "grow-diag-final" refinement rule (Koehn et al., 2003) to obtain word alignments. We used the SRI Language Modeling Toolkit[2] (Stolcke and others, 2002) to train our language models. MERT (Och, 2003) was adopted to tune feature weights of the decoder. We used the case-insensitive BLEU[3] as our evaluation metric. In order to alleviate the instability of MERT , we followed Clark et al. (2011) to perform three runs of MERT and reported average BLEU scores over the three runs for all our experiments.

We used word2vec toolkit[4] to train our word embeddings and set the vector dimension d to 30. In our training experiment, we used the continuous bag-of-words model with a context window of size 5. The monolingual corpus, which was used to pre-train word embeddings, is extracted from

[1]The corpora include LDC2003E14, LDC2004T07, LDC2005T06, LDC2005T10 and LDC2004T08 (Hong Kong Hansards/Laws/News).

[2]http://www.speech.sri.com/projects/srilm/download.html

[3]ftp://jaguar.ncsl.nist.gov/mt/resources/mteval-v11b.pl

[4]https://code.google.com/p/word2vec/

the above parallel corpus in SMT. To train vector representations for multi-word phrases, we randomly selected 1M bilingual sentences [5] as training set and used the unsupervised greedy RAE following (Socher et al., 2011). We used a learning rate of 10^{-3} for our minimum distance method that learned the centroid of phrase representations as the vector representation of the corresponding nonterminal.

For projection neural network in Section 3.3, we set 300 units for the hidden layer and dimensionality of 30 for both input and output vectors. Learning rate was set to 10^{-3} and the regularization coefficient λ_L was set to 10^{-3}. To construct the training set for the projection neural network, we selected phrase pairs from our rule table and used their representations on the source and target side as training examples. We randomly selected 5M examples as training set, 10k examples as development set and 10k examples as test set. The multi-layer projection neural network was trained with the back-propagation and stochastic gradient descent algorithm with a mini-batch size of 5k.

Our baseline system is an in-house hierarchical phrase-based system (Chiang, 2007). The features used in the baseline system includes a 4-gram language model trained on the Xinhua section of the English Gigaword corpus, a 3-gram language model trained on the target part of the bilingual training data, bidirectional translation probabilities, bidirectional lexical weights, a word count, a phrase count and a glue rule count.

In order to compare our proposed models with previous methods on nonterminal refinement, we re-implemented a syntax mismatch model (Syn-Mis) which was used by Huang et al. (2013) and integrated it into hierarchical phrase-based system. Syn-Mis model decorates each nonterminal with a distribution of head POS tags and uses this distribution to measure the degree of syntactic compatibility of translation rules with corresponding source spans. In order to obtain head POS tags for Syn-Mis model, we used the Stanford dependency parser [6] (Chang et al., 2009) to parse Chinese sentences in our training corpus and NIST development/test sets.

[5] We choose bilingual sentences because we want to obtain bilingual training examples to train our projection neural network as described in Section 3.3.

[6] http://nlp.stanford.edu/software/lex-parser.shtml

	MT06	MT08	Avg
Baseline	30.54	23.58	27.06
Syn-Mis	31.23*	24.38*	27.81
MV + CS α = 1.0	31.44+	24.23*	27.84
MV + CS α = 0	31.63*	24.51*	28.07
MV + CS α = -1.0	31.13	24.07*	27.60
MD + ED β = 0	31.02+	23.74	27.38
MD + ED β = 0.5	31.35+	24.08*	27.72
MD + ED β = 1.0	31.06	23.90+	27.48

Table 1: BLEU scores of our models against the baseline and Syn-Mis model. "*" and "+" : significantly better than Baseline at significance level $p < 0.01$ and $p < 0.05$ respectively.

6.2 Different Approaches to Learn Vector Representations for Nonterminals

Our first group of experiments were carried out to investigate which approach is more appropriate to learn semantic vectors for nonterminals. We only used the source-side semantic nonterminal refinement model in these experiments. In order to validate the effectiveness of the proposed approaches for learning nonterminal semantic vectors, we combined the minimum distance method (MD) with the Euclidean Distance (ED) because both of them are distance-based, and combined the weighted mean value method (MV) with the Cosine Similarity model (CS) as they belong to vector-based approaches. We chose α = 1.0, 0, -1.0 and β = 0, 0.5, 1.0 for glue rules to study the impact of these parameters. We compared our model with the baseline and Syn-Mis model.

Results are shown in Table 1. From Table 1, we observe that the proposed two approaches are able to achieve significant improvements over the baseline. (MV + CS) and (MD + ED) achieve up to an absolute improvement of 1.09 and 0.81 (when α = 0 and β = 0.5) BLEU points respectively over the baseline on the development test set MT06. And the approach (MV + CS) with α = 0 outperforms Syn-Mis by 0.4 BLEU points on MT06 without using any syntactic information. The approach (MV + CS) achieves better performance and it is more efficient than (MD + ED) where the computation of semantic centroids is time-consuming. Therefore, we adopt the approach (MV + CS) with α = 0 to learn semantic vectors for nonterminals and compute semantic similarities in the following experiments.

	MT06	MT08	Avg
Baseline	30.54	23.58	27.06
Linear Projection	30.70	23.66	27.18
Nonlinear Projection	31.16	24.11*	27.64
Two-pass decoding	31.29+	24.24*	27.77

Table 2: Comparison of two-pass decoding, linear and nonlinear projection methods for integrating the target-side semantic nonterminal refinement model in terms of BLEU scores. "*" and "+" : significantly better than Baseline at significance level $p < 0.01$ and $p < 0.05$ respectively.

6.3 Effect of the Target Semantic Nonterminal Refinement Models

In the second set of experiments, we further validate the effectiveness of semantic nonterminal vectors learned on the target side. In these experiments, learning vector representations and computing semantic similarities were performed on the target language semantic space. We also compared the two integration methods discussed in Section 5 for the target-side model. With regard to the projection method, we further compared the linear projection (the projection neural network without hidden layer) with the nonlinear projection (with hidden layer). Experiment results are shown in Table 2.

From Table 2, we can see that

- Two-pass decoding achieves the highest BLEU scores, which are higher than those of the baseline by 0.75 and 0.66 BLEU points on MT06 and MT08 respectively. The reason may be that noisy translation candidates are filtered out in the first pass. This finding is consistent with many other multiple-pass systems in natural language processing, e.g., two-pass parsing (Zettlemoyer and Collins, 2007).

- Nonlinear projection achieves an improvement of 0.62 BLEU points over the baseline on MT06. It outperforms linear projection method on both sets. These empirical results support our assumption that nonlinear relations between languages are more reasonable than linear relations.

- The results prove that the target-side semantic nonterminal refinement model is also able to improve the baseline system, although the gain is less than that of the source-side counterpart.

	MT06	MT08	Avg
Baseline	30.54	23.58	27.06
Syn-Mis	31.23*	24.38*	27.81
Src Model [1]	31.63*	24.51*	28.07
Trg Model [2]	31.16	24.11*	27.64
Combined-Model	31.71*	24.72*	28.22

[1] (MV + CS α = 0) is used.
[2] Nonlinear Projection is used.

Table 3: BLEU scores of the combination of the source- and target-side semantic nonterminal refine model. "*" and "+" : significantly better than Baseline at significance level $p < 0.01$ and $p < 0.05$ respectively.

6.4 Combination of the Source and Target Models

Finally, we integrated both the source- and target-side semantic nonterminal refinement models into the baseline system. In this experiment, we adopted nonlinear projection to obtain target semantic vector representations for target phrases. These two models collectively achieve a gain of up to 1.16 BLEU points over the baseline and 0.41 BLEU points over Syn-Mis model on average, which is shown in Table 3.

7 Conclusion

We have presented a framework to refine nonterminal X in hierarchical translation rules with semantic representations. The semantic vectors are derived from vector representations of phrasal substitutions, which are automatically learned using an unsupervised RAE. As the semantic nonterminal refinement model is capable of selecting more semantically similar translation rules, it achieves statistically significant improvements over the baseline on Chinese-to-English translation. Experiment results have shown that

- Using (MV + CS) approach to learn semantic representations for nonterminals can achieve better performance than (MD + ED) in terms of BLEU scores.

- Target-side semantic nonterminal refinement model is able to substantially improve translation quality over the baseline. Two-pass de-

coding method is superior to the projection method.

- The simultaneous incorporation of the source- and target-side models can achieve further improvements over a single-side model.

For the future work, we are interested in learning bilingual representations (Lauly et al., 2014; Gouws et al., 2014) for nonterminals. We also would like to extend our work by using more contextual lexical information to derive semantic vectors for nonterminals.

Acknowledgment

The work was sponsored by the National Natural Science Foundation of China (Grants No. 61403269, 61432013 and 61333018) and Natural Science Foundation of Jiangsu Province (Grant No. BK20140355). We would like to thank three anonymous reviewers for their insightful comments.

References

Hala Almaghout, Jie Jiang, and Andy Way. 2011. Ccg contextual labels in hierarchical phrase-based smt. In *Proceedings of the 15th Annual Conference of the European Association for Machine Translation*.

Phil Blunsom, Edward Grefenstette, Nal Kalchbrenner, et al. 2014. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*. Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics.

Hailong Cao, Dongdong Zhang, Ming Zhou, and Tiejun Zhao. 2014. Soft dependency matching for hierarchical phrase-based machine translation. In *COLING*.

Pi-Chuan Chang, Huihsin Tseng, Dan Jurafsky, and Christopher D Manning. 2009. Discriminative reordering with chinese grammatical relations features. In *Proceedings of the Third Workshop on Syntax and Structure in Statistical Translation*, pages 51–59. Association for Computational Linguistics.

David Chiang. 2007. Hierarchical phrase-based translation. *computational linguistics*, 33(2):201–228.

David Chiang. 2010. Learning to translate with source and target syntax. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1443–1452. Association for Computational Linguistics.

Jonathan H Clark, Chris Dyer, Alon Lavie, and Noah A Smith. 2011. Better hypothesis testing for statistical machine translation: Controlling for optimizer instability. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 176–181. Association for Computational Linguistics.

Qin Gao and Stephan Vogel. 2011. Utilizing target-side semantic role labels to assist hierarchical phrase-based machine translation. In *Proceedings of the Fifth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 107–115. Association for Computational Linguistics.

Stephan Gouws, Yoshua Bengio, and Greg Corrado. 2014. Bilbowa: Fast bilingual distributed representations without word alignments. *arXiv preprint arXiv:1410.2455*.

Greg Hanneman and Alon Lavie. 2013. Improving syntax-augmented machine translation by coarsening the label set. In *HLT-NAACL*, pages 288–297.

Hieu Hoang and Philipp Koehn. 2010. Improved translation with source syntax labels. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, pages 409–417. Association for Computational Linguistics.

Zhongqiang Huang, Martin Čmejrek, and Bowen Zhou. 2010. Soft syntactic constraints for hierarchical phrase-based translation using latent syntactic distributions. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 138–147. Association for Computational Linguistics.

Zhongqiang Huang, Jacob Devlin, and Rabih Zbib. 2013. Factored soft source syntactic constraints for hierarchical machine translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 556–566.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 48–54. Association for Computational Linguistics.

Stanislas Lauly, Hugo Larochelle, Mitesh Khapra, Balaraman Ravindran, Vikas C Raykar, and Amrita Saha. 2014. An autoencoder approach to learning bilingual word representations. In *Advances in Neural Information Processing Systems*, pages 1853–1861.

Junhui Li, Zhaopeng Tu, Guodong Zhou, and Josef van Genabith. 2012. Head-driven hierarchical phrase-based translation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pages 33–37. Association for Computational Linguistics.

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

Tomas Mikolov, Quoc V Le, and Ilya Sutskever. 2013b. Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168*.

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

Jeff Mitchell and Mirella Lapata. 2008. Vector-based models of semantic composition. In *Proceedings of the 46st Annual Meeting on Association for Computational*, pages 236–244.

Markos Mylonakis and Khalil Sima'an. 2011. Learning hierarchical translation structure with linguistic annotations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 642–652. Association for Computational Linguistics.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 160–167. Association for Computational Linguistics.

Libin Shen, Jinxi Xu, Bing Zhang, Spyros Matsoukas, and Ralph Weischedel. 2009. Effective use of linguistic and contextual information for statistical machine translation. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*, pages 72–80. Association for Computational Linguistics.

Richard Socher, Christopher D Manning, and Andrew Y Ng. 2010. Learning continuous phrase representations and syntactic parsing with recursive neural networks. In *Proceedings of the NIPS-2010 Deep Learning and Unsupervised Feature Learning Workshop*, pages 1–9.

Richard Socher, Jeffrey Pennington, Eric H Huang, Andrew Y Ng, and Christopher D Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 151–161. Association for Computational Linguistics.

Andreas Stolcke et al. 2002. Srilm-an extensible language modeling toolkit. In *INTERSPEECH*.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 384–394. Association for Computational Linguistics.

Ashish Venugopal, Andreas Zollmann, Noah A Smith, and Stephan Vogel. 2009. Preference grammars: Softening syntactic constraints to improve statistical machine translation. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 236–244. Association for Computational Linguistics.

Jonathan Weese, Chris Callison-Burch, and Adam Lopez. 2012. Using categorial grammar to label translation rules. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 222–231. Association for Computational Linguistics.

Luke S Zettlemoyer and Michael Collins. 2007. Online learning of relaxed ccg grammars for parsing to logical form. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 678–687.

Andreas Zollmann and Ashish Venugopal. 2006. Syntax augmented machine translation via chart parsing. In *Proceedings of the Workshop on Statistical Machine Translation*, pages 138–141. Association for Computational Linguistics.

Andreas Zollmann and Stephan Vogel. 2011. A word-class approach to labeling pscfg rules for machine translation. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 1–11. Association for Computational Linguistics.

A Comparison between Count and Neural Network Models Based on Joint Translation and Reordering Sequences

Andreas Guta, Tamer Alkhouli, Jan-Thorsten Peter, Joern Wuebker, Hermann Ney
Human Language Technology and Pattern Recognition Group
RWTH Aachen University
Aachen, Germany
{surname}@cs.rwth-aachen.de

Abstract

We propose a conversion of bilingual sentence pairs and the corresponding word alignments into novel linear sequences. These are joint translation and reordering (JTR) uniquely defined sequences, combining interdepending lexical and alignment dependencies on the word level into a single framework. They are constructed in a simple manner while capturing multiple alignments and empty words. JTR sequences can be used to train a variety of models. We investigate the performances of n-gram models with modified Kneser-Ney smoothing, feed-forward and recurrent neural network architectures when estimated on JTR sequences, and compare them to the operation sequence model (Durrani et al., 2013b). Evaluations on the IWSLT German→English, WMT German→English and BOLT Chinese→English tasks show that JTR models improve state-of-the-art phrase-based systems by up to 2.2 BLEU.

1 Introduction

Standard phrase-based machine translation (Och et al., 1999; Zens et al., 2002; Koehn et al., 2003) uses relative frequencies of phrase pairs to estimate a translation model. The phrase table is extracted from a bilingual text aligned on the word level, using e.g. GIZA++ (Och and Ney, 2003). Although the phrase pairs capture internal dependencies between the source and target phrases aligned to each other, they fail to model dependencies that extend beyond phrase boundaries. Phrase-based decoding involves concatenating target phrases. The burden of ensuring that the result is linguistically consistent falls on the language model (LM).

This work proposes word-based translation models that are potentially capable of capturing long-range dependencies. We do this in two steps: First, given bilingual sentence pairs and the associated word alignments, we convert the information into uniquely defined linear sequences. These sequenecs encode both word reordering and translation information. Thus, they are referred to as *joint translation and reordering* (JTR) sequences. Second, we train an n-gram model with modified Kneser-Ney smoothing (Chen and Goodman, 1998) on the resulting JTR sequences. This yields a model that fuses interdepending reordering and translation dependencies into a single framework.

Although JTR n-gram models are closely related to the operation sequence model (OSM) (Durrani et al., 2013b), there are three main differences. To begin with, the OSM employs minimal translation units (MTUs), which are essentially atomic phrases. As the MTUs are extracted sentence-wise, a word can potentially appear in multiple MTUs. In order to avoid overlapping translation units, we define the JTR sequences on the level of words. Consequently, JTR sequences have smaller vocabulary sizes than OSM sequences and lead to models with less sparsity. Moreover, we argue that JTR sequences offer a simpler reordering approach than operation sequences, as they handle reorderings without the need to predict gaps. Finally, when used as an additional model in the log-linear framework of phrase-based decoding, an n-gram model trained on JTR sequences introduces only one single feature to be tuned, whereas the OSM additionally uses 4 supportive features (Durrani et al., 2013b). Experimental results confirm that this simplification does not make JTR models less expressive, as their performance is on par with the OSM.

Due to data sparsity, increasing the n-gram order of count-based models beyond a certain point becomes useless. To address this, we resort to neu-

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1401–1411,
Lisbon, Portugal, 17-21 September 2015.

ral networks (NNs), as they have been successfully applied to machine translation recently (Sundermeyer et al., 2014; Devlin et al., 2014). They are able to score any word combination without requiring additional smoothing techniques. We experiment with feed-forward and recurrent translation networks, benefiting from their smoothing capabilities. To this end, we split the linear sequence into two sequences for the neural translation models to operate on. This is possible due to the simplicity of the JTR sequence. We show that the count and NN models perform well on their own, and that combining them yields even better results.

In this work, we apply n-gram models with modified Kneser-Ney smoothing during phrase-based decoding and neural JTR models in rescoring. However, using a phrase-based system is not required by the model, but only the initial step to demonstrate the strength of JTR models, which can be applied independently of the underlying decoding framework. While the focus of this work is on the development and comparison of the models, the long-term goal is to decode using JTR models without the limitations introduced by phrases, in order to exploit the full potential of JTR models. The JTR models are estimated on word alignments, which we obtain using GIZA++ in this paper. The future aim is to also generate improved word alignments by a joint optimization of both the alignments and the models, similar to the training of IBM models (Brown et al., 1990; Brown et al., 1993). In the long run, we intend to achieve a consistency between decoding and training using the introduced JTR models.

2 Previous Work

In order to address the downsides of the phrase translation model, various approaches have been taken. Mariño et al. (2006) proposed a bilingual language model (BILM) that operates on bilingual n-grams, with an own n-gram decoder requiring monotone alignments. The lexical reordering model introduced in (Tillmann, 2004) was integrated into phrase-based decoding. Crego and Yvon (2010) adapted the approach to BILMs. The bilingual n-grams are further advanced in (Niehues et al., 2011), where they operate on non-monotone alignments within a phrase-based translation framework. Compared to our JTR models, their BILMs treat jointly aligned source words as minimal translation units, ignore unaligned source words and do not include reordering information.

Durrani et al. (2011) developed the OSM which combined dependencies on bilingual word pairs and reordering information into a single framework. It used an own decoder that was based on n-grams of MTUs and predicted single translation or reordering operations. This was further advanced in (Durrani et al., 2013a) by a decoder that was capable of predicting whole sequences of MTUs, similar to a phrase-based decoder. In (Durrani et al., 2013b), a slightly enhanced version of OSM was integrated into the log-linear framework of the Moses system (Koehn et al., 2007). Both the BILM (Stewart et al., 2014) and the OSM (Durrani et al., 2014) can be smoothed using word classes.

Guta et al. (2015) introduced the extended translation model (ETM), which operates on the word level and augments the IBM models by an additional bilingual word pair and a reordering operation. It is implemented into the log-linear framework of a phrase-based decoder and shown to be competitive with a 7-gram OSM.

The JTR n-gram models proposed within this work can be seen as an extension of the ETM. Nevertheless, JTR models utilize linear sequences of dependencies and combine the translation of bilingual word pairs and reoderings into a single model. The ETM, however, features separate models for the translation of individual words and reorderings and provides an explicit treatment of multiple alignments. As they operate on linear sequences, JTR count models can be implemented using existing toolkits for n-gram language models, e.g. the KenLM toolkit (Heafield et al., 2013).

An HMM approach for word-to-phrase alignments was presented in (Deng and Byrne, 2005), showing performance similar to IBM Model 4 on the task of bitext alignment. Feng et al. (2013) propose several models which rely only on the information provided by the source side and predict reorderings. Contrastingly, JTR models incorporate target information as well and predict both translations and reorderings jointly in a single framework.

Zhang et al. (2013) explore different Markov chain orderings for an n-gram model on MTUs in rescoring. Feng and Cohn (2013) present another generative word-based Markov chain translation model which exploits a hierarchical Pitman-Yor process for smoothing, but it is only applied to induce word alignments. Their follow-up work (Feng et al., 2014) introduces a Markov-model on

MTUs, similar to the OSM described above.

Recently, neural machine translation has emerged as an alternative to phrase-based decoding, where NNs are used as standalone models to decode source input. In (Sutskever et al., 2014), a recurrent NN was used to encode a source sequence, and output a target sentence once the source sentence was fully encoded in the network. The network did not have any explicit treatment of alignments. Bahdanau et al. (2015) introduced soft alignments as part of the network architecture. In this work, we make use of hard alignments instead, where we encode the alignments in the source and target sequences, requiring no modifications of existing feed-forward and recurrent NN architectures. Our feed-forward models are based on the architectures proposed in (Devlin et al., 2014), while the recurrent models are based on (Sundermeyer et al., 2014). Further recent research on applying NN models for extended context was carried out in (Le et al., 2012; Auli et al., 2013; Hu et al., 2014). All of these works focus on lexical context and ignore the reordering aspect covered in our work.

3 JTR Sequences

The core idea of this work is the interpretation of a bilingual sentence pair and its word alignment as a linear sequence of K joint translation and reordering (JTR) tokens g_1^K. Formally, the sequence $g_1^K(f_1^J, e_1^I, b_1^I)$ is a uniquely defined interpretation of a given source sentence f_1^J, its translation e_1^I and the inverted alignment b_1^I, where b_i denotes the ordered sequence of source positions j aligned to target position i. We drop the explicit mention of (f_1^J, e_1^I, b_1^I) to allow for a better readability. Each JTR token is either an aligned bilingual word pair $\langle f, e\rangle$ or a reordering class $\Delta_{j'j}$.

Unaligned words on the source and target side are processed as if they were aligned to the empty word ε. Hence, an unaligned source word f generates the token $\langle f, \varepsilon\rangle$, and an unaligned target word e the token $\langle \varepsilon, e\rangle$.

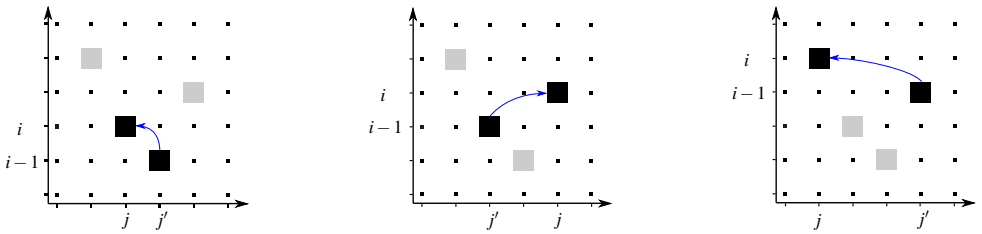
Each word of the source and target sentences is to appear in the corresponding JTR sequence exactly once. For multiply-aligned target words e, the first source word f that is aligned to e generates the token $\langle f, e\rangle$. All other source words f', that are also aligned to e, are processed as if they were aligned to the artificial word σ. Thus, each of these f' generates a token $\langle f', \sigma\rangle$. The same approach is applied to multiply-aligned source words. Similar to Feng and Cohn (2013), we classify the reordered source positions j' and j by $\Delta_{j'j}$:

$$\Delta_{j'j} = \begin{cases} \text{step backward } (\leftarrow), & j = j' - 1 \\ \text{jump forward } (\curvearrowright), & j > j' + 1 \\ \text{jump backward } (\curvearrowleft), & j < j' - 1. \end{cases}$$

The reordering classes are illustrated in Figure 1.

Algorithm 1 JTR Conversion Algorithm

```
1:  procedure JTRCONVERSION(f_1^J, e_1^I, b_1^I)
2:      g_1^K ← ∅
3:      // last translated source position j'
4:      j' ← 0
5:      for i ← 1 to I do
6:          if e_i is unaligned then
7:              // align e_i to the empty word ε
8:              APPEND(g_1^K, ⟨ε, e_i⟩)
9:              continue
10:         // e_i is aligned to at least one source word
11:         j ← first source position in b_i
12:         if j = j' then
13:             // e_i is aligned to the same f_j as e_{i-1}
14:             APPEND(g_1^K, ⟨σ, e_i⟩)
15:             continue
16:         if j ≠ j' + 1 then
17:             // alignment step is non-monotone
18:             REORDERINGS(f_1^J, b_1^I, g_1^K, j', j)
19:         // 1-to-1 translation: f_j is aligned to e_i
20:         APPEND(g_1^K, ⟨f_j, e_i⟩)
21:         j' ← j
22:         // generate all other f_j that are also
23:         // aligned to the current target word e_i
24:         for all remaining j in b_i do
25:             APPEND(g_1^K, ⟨f_j, σ⟩)
26:             j' ← j
27:     // check last alignment step at sentence end
28:     if j' ≠ J then
29:         // last alignment step is non-monotone
30:         REORDERINGS(f_1^J, b_1^I, g_1^K, j', J + 1)
31:     return g_1^K
32:
33: // called when a reordering class is appended
34: procedure REORDERINGS(f_1^J, b_1^I, g_1^K, j', j)
35:     // check if the predecessor is unaligned
36:     if f_{j-1} is unaligned then
37:         // get unaligned predecessors
38:         f_{j_0}^{j-1} ← unaligned predecessors of f_j
39:         // check if the alignment step to the first
40:         // unaligned predecessor is monotone
41:         if j_0 ≠ j' + 1 then
42:             // non-monotone: add reordering class
43:             APPEND(g_1^K, Δ_{j',j_0})
44:         // translate unaligned predecessors by ε
45:         for f ← f_{j_0} to f_{j-1} do
46:             APPEND(g_1^K, ⟨f, ε⟩)
47:     else
48:         // non-monotone: add reordering class
49:         APPEND(g_1^K, Δ_{j',j})
```



(a) step backward (←) (b) jump forward (↷) (c) jump backward (↶)

Figure 1: Overview of the different reordering classes in JTR sequences.

3.1 Sequence Conversion

Algorithm 1 presents the formal conversion of a bilingual sentence pair and its alignment into the corresponding JTR sequence g_1^K. At first, g_1^K is initialized by an empty sequence (line 2). For each target position $i = 1, \ldots, I$ it is extended by at least one token. During the generation process, we store the last visited source position j' (line 4). If a target word e_i is

- unaligned, we align it to the empty word ε and append $\langle \varepsilon, e_i \rangle$ to the current g_1^K (line 8),
- if it is aligned to the same f_j as e_{i-1}, we only add $\langle \sigma, e_i \rangle$ (line 14),
- otherwise we append $\langle f_j, e_i \rangle$ (line 20) and
- in case there are more source words aligned to e_i, we additionally append $\langle f_j, \sigma \rangle$ for each of these (line 24).

Before a token $\langle f_j, e_i \rangle$ is generated, we have to check whether the alignment step from j' to j is monotone (line 16). In case it is not, we have to deal with reorderings (line 34). We define that a token $\langle f_{j-1}, \varepsilon \rangle$ is to be generated right before the generation of the token containing f_j. Thus, if f_{j-1} is not aligned, we first determine the contiguous sequence of unaligned predecessors $f_{j_0}^{j-1}$ (line 38). Next, if the step from j' to j_0 is not monotone, we add the corresponding reordering class (line 43). Afterwards we append all $\langle f_{j_0}, \varepsilon \rangle$ to $\langle f_{j-1}, \varepsilon \rangle$. If f_{j-1} is aligned, we do not have to process unaligned source words and only append the corresponding reordering class (line 49).

Figure 2 illustrates the generation steps of a JTR sequence, whose result is presented in Table 1. The alignment steps are denoted by the arrows connecting the alignment points. The first dashed alignment point indicates the $\langle \varepsilon, , \rangle$ token that is generated right after the $\langle \texttt{Feld}, \texttt{field} \rangle$ token. The second dashed alignment point indicates the $\langle \texttt{ein}, \varepsilon \rangle$ token, which corresponds to the unaligned source word `ein`. Note, that the $\langle \texttt{ein}, \varepsilon \rangle$ token has to be generated right before $\langle ., . \rangle$ is generated. Therefore, there is no forward jump from $\langle \texttt{Code}, \texttt{code} \rangle$ to $\langle ., . \rangle$, but a monotone step to $\langle \texttt{ein}, \varepsilon \rangle$ followed by $\langle ., . \rangle$.

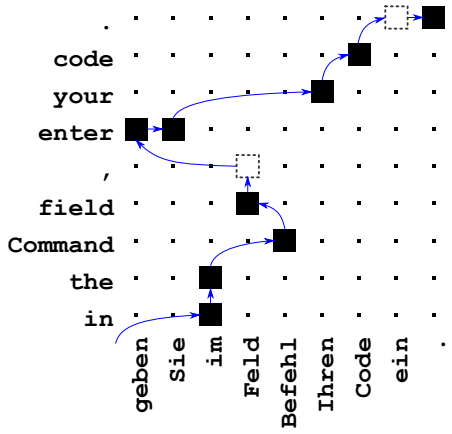


Figure 2: This example illustrates the JTR sequence g_1^K for a German→English sentence pair including the word-to-word alignment.

3.2 Training of Count Models

As the JTR sequence g_1^K is a unique interpretation of a bilingual sentence pair and its alignment, the probability $p(f_1^J, e_1^I, b_1^I)$ can be computed as:

$$p(f_1^J, e_1^I, b_1^I) = p(g_1^K). \tag{1}$$

The probability of g_1^K can be factorized and approximated by an n-gram model.

$$p(g_1^K) = \prod_{k=1}^{K} p(g_k | g_{k-n+1}^{k-1}) \tag{2}$$

Within this work, we first estimate the Viterbi alignment for the bilingual training data using GIZA++ (Och and Ney, 2003). Secondly, the conversion presented in Algorithm 1 is applied to obtain the JTR sequences, on which we estimate an n-gram model with modified Kneser-Ney smoothing as described in (Chen and Goodman, 1998) using the KenLM toolkit[1] (Heafield et al., 2013).

[1] https://kheafield.com/code/kenlm/

k	g_k	s_k	t_k
1	$\curvearrowright$	δ	$\curvearrowright$
2	$\langle$im, in$\rangle$	im	in
3	$\langle\sigma$, the$\rangle$	σ	the
4	$\curvearrowright$	δ	$\curvearrowright$
5	$\langle$Befehl, Command$\rangle$	Befehl	Command
6	$\leftarrow$	δ	$\leftarrow$
7	$\langle$Feld, field$\rangle$	Feld	field
8	$\langle\varepsilon$, ,$\rangle$	ε	,
9	$\curvearrowleft$	δ	$\curvearrowleft$
10	$\langle$geben, enter$\rangle$	geben	enter
11	$\langle$Sie, $\sigma\rangle$	Sie	σ
12	$\curvearrowright$	δ	$\curvearrowright$
13	$\langle$Ihren, your$\rangle$	Ihren	your
14	$\langle$Code, code$\rangle$	Code	code
15	$\langle$ein, $\varepsilon\rangle$	ein	ε
16	$\langle$.,.$\rangle$	.	.

Table 1: The left side of this table presents the JTR tokens g_k corresponding to Figure 2. The right side shows the source and target tokens s_k and t_k obtained from the JTR tokens g_k. They are used for the training of NNs (cf. Section 4).

3.3 Integration into Phrase-based Decoding

Basically, each phrase table entry is annotated with both the word alignment information, which also allows to identify unaligned source words, and the corresponding JTR sequence. The JTR model is added to the log-linear framework as an additional n-gram model. Within the phrase-based decoder, we extend each search state such that it additionally stores the JTR model history.

In comparison to the OSM, the JTR model does not predict gaps. Local reorderings within phrases are handled implicitly. On the other hand, we represent long-range reorderings between phrases by the coverage vector and limit them by reordering constraints.

Phrase-pairs ending with unaligned source words at their right boundary prove to be a problem during decoding. As shown in Subsection 3.1, the conversion from word alignments to JTR sequences assumes that each token corresponding to an unaligned source word is generated immediately before the token corresponding to the closest aligned source position to its right. However, if a phrase ends with an unaligned f_j as its rightmost source word, the generation of the $\langle f_j, \varepsilon\rangle$ token has to be postponed until the next word f_{j+1} is to be translated or, even worse, f_{j+1} has already been translated before.

To address this issue, we constrained the phrase table extraction to discard entries with unaligned source tokens at the right boundary. For IWSLT De→En, this led to a baseline weaker by 0.2 BLEU than the one described in Section 5. In order to have an unconstrained and fair baseline, we thereafter removed this constraint and forced such deletion tokens to be generated at the end of the sequence. Hence, we accept that the JTR model might compute the wrong score in these special cases.

4 Neural Networks

Usually, smoothing techniques are applied to count-based models to handle unseen events. A neural network does not suffer from this, as it is able to score unseen events without additional smoothing techniques. In the following, we will describe how to adapt JTR sequences to be used with feed-forward and recurrent NNs.

The first thing to notice is the vocabulary size, mainly determined by the number of bilingual word pairs, which constituted atomic units in the count-based models. NNs that compute probability values at the output layer evaluate a softmax function that produces normalized scores that sum up to unity. The softmax function is given by:

$$p(e_i|e_1^{i-1}) = \frac{e^{o_{e_i}(e_1^{i-1})}}{\sum_{w=1}^{|V|} e^{o_w(e_1^{i-1})}} \quad (3)$$

where o_{e_i} and o_w are the raw unnormalized output layer values for the words e_i and w, respectively, and $|V|$ is the vocabulary size. The output layer is a function of the context e_1^{i-1}. Computing the denominator is expensive for large vocabularies, as it requires computing the output for all words. Therefore, we split JTR tokens g_k and use individual words as input and output units, such that the NN receives jumps, source and target words as input and outputs target words and jumps. Hence, the resulting neural model is not a LM, but a translation model with different input and output vocabularies. A JTR sequence g_1^K is split into its source and target parts s_1^K and t_1^K. The construction of the JTR source sequence s_1^K proceeds as follows: Whenever a bilingual pair is encountered, the source word is kept and the target word is discarded. In addition, all jump classes are replaced by a special token δ. The JTR target sequence t_1^K is constructed similarly by keeping the target words and dropping source words, and the jump classes are also kept. Table 1 shows the JTR source and target sequences corresponding to JTR sequence of Figure 2.

Due to the design of the JTR sequence, producing the source and target JTR sequences is straightforward. The resulting sequences can then be used with existing NN architectures, without further modifications to the design of the networks. This results in powerful models that require little effort to implement.

4.1 Feed-forward Neural JTR

First, we will apply a feed-forward NN (FFNN) to the JTR sequence. FFNN models resemble count-based models in using a predefined limited context size, but they do not encounter the same smoothing problems. In this work, we use a FFNN similar to that proposed in (Devlin et al., 2014), defined as:

$$p(t_1^K|s_1^K) \approx \prod_{k=1}^{K} p(t_k|t_{k-n}^{k-1}, s_{k-n}^{k}). \quad (4)$$

It scores the JTR target word t_k at position k using the current source word s_k, and the history of n JTR source words. In addition, the n JTR target words preceding t_k are used as context. The FFNN computes the score by looking up the vector embeddings of the source and target context words, concatenating them, then evaluating the rest of the network. We reduce the output layer to a shortlist of the most frequent words, and compute word class probabilities for the remaining words.

4.2 Recurrent Neural JTR

Unlike feed-forward NNs, recurrent NNs (RNNs) enable the use of unbounded context. Following (Sundermeyer et al., 2014), we use bidirectional recurrent NNs (BRNNs) to capture the full JTR source side. The BRNN uses the JTR target side as well as the full JTR source side as context, and it is given by:

$$p(t_1^K|s_1^K) = \prod_{k=1}^{K} p(t_k|t_1^{k-1}, s_1^K) \quad (5)$$

This equation is realized by a network that uses forward and backward recurrent layers to capture the complete source sentence. By a forward layer we imply a recurrent hidden layer that processes a given sequence from left to right, while a backward layer does the processing backwards, from right to left. The source sentence is basically split at a given position k, then past and future representations of the sentence are recursively computed by the forward and backward layers, respectively. To include the target side, we provide the forward layer with the target input t_{k-1} as well, that is, we aggregate the embeddings of the input source word s_k and the input target word t_{k-1} before they are fed into the forward layer. Due to recurrency, the forward layer encodes the parts (t_1^{k-1}, s_1^k), and the backward layer encodes s_k^K, and together they encode (t_1^{k-1}, s_1^K), which is used to score the output target word t_k. For the sake of comparison to FFNN and count models, we also experiment with a recurrent model that does not include future source information, this is obtained by replacing the term s_1^K with s_1^k in Eq. 5. It will be referred to as the unidirectional recurrent neural network (URNN) model in the experiments.

Note that the JTR source and target sides include jump information, therefore, the RNN model described above explicitly models reordering. In contrast, the models proposed in (Sundermeyer et al., 2014) do not include any jumps, and hence do not provide an explicit way of including word reordering. In addition, the JTR RNN models do not require the use of IBM-1 lexica to resolve multiply-aligned words. As discussed in Section 3, these cases are resolved by aligning the multiply-aligned word to the first word on the opposite side.

The integration of the NNs into the decoder is not trivial, due to the dependence on the target context. In the case of RNNs, the context is unbounded, which would affect state recombination, and lead to less variety in the beam used to prune the search space. Therefore, the RNN scores are computed using approximations instead (Auli et al., 2013; Alkhouli et al., 2015). In (Alkhouli et al., 2015), it is shown that approximate RNN integration into the phrase-based decoder has a slight advantage over n-best rescoring. Therefore, we apply RNNs in rescoring in this work, and to allow for a direct comparison between FFNNs and RNNs, we apply FFNNs in rescoring as well.

5 Evaluation

We perform experiments on the large-scale IWSLT 2013[2] (Cettolo et al., 2014) German→English, WMT 2015[3] German→English and the DARPA BOLT Chinese→English tasks. The statistics for the bilingual corpora are shown in Table 2. Word alignments are generated with the GIZA++ toolkit

[2] http://www.iwslt2013.org

[3] http://www.statmt.org/wmt15/

	IWSLT		WMT		BOLT	
	German	**English**	**German**	**English**	**Chinese**	**English**
Sentences	4.32M		4.22M		4.08M	
Run. Words	108M	109M	106M	108M	78M	86M
Vocabulary	836K	792K	814K	773K	384K	817K

Table 2: Statistics for the bilingual training data of the IWSLT 2013 German→English, WMT 2015 German→English, and the DARPA BOLT Chinese→English translation tasks.

(Och and Ney, 2003). We use a standard phrase-based translation system (Koehn et al., 2003). The decoding process is implemented as a beam search. All baselines contain phrasal and lexical smoothing models for both directions, word and phrase penalties, a distance-based reordering model, enhanced low frequency features (Chen et al., 2011), a hierarchical reordering model (HRM) (Galley and Manning, 2008), a word class LM (Wuebker et al., 2013) and an *n*-gram LM. The lexical and phrase translation models of all baseline systems are trained on all provided bilingual data. The log-linear feature weights are tuned with minimum error rate training (MERT) (Och, 2003) on BLEU (Papineni et al., 2001). All systems are evaluated with *MultEval* (Clark et al., 2011). The reported BLEU scores are averaged over three MERT optimization runs.

All LMs, OSMs and count-based JTR models are estimated with the KenLM toolkit (Heafield et al., 2013). The OSM and the count-based JTR model are implemented in the phrasal decoder. NNs are used only in rescoring. The 9-gram FFNNs are trained with two hidden layers. The short lists contain the 10k most frequent words, and all remaining words are clusterd into 1000 word classes. The projecton layer has 17×100 nodes, the first hidden layer 1000 and the second 500. The RNNs have LSTM architectures. The URNN has 2 hidden layers while the BRNN has one forward, one backward and one additional hidden layer. All layers have 200 nodes, while the output layer is class-factored using 2000 classes. For the count-based JTR model and OSM we tuned the *n*-gram size on the tuning set of each task. For the full data, 7-grams were used for the IWSLT and WMT tasks, and 8-grams for BOLT. When using in-domain data, smaller *n*-gram sizes were used. All rescoring experiments used 1000-best lists without duplicates.

5.1 Tasks description

The domain of IWSLT consists of lecture-type talks presented at TED conferences which are also available online[4]. All systems are optimized on the `dev2010` corpus, named `dev` here. Some of the OSM and JTR systems are trained on the TED portions of the data containing 138K sentences. To estimate the 4-gram LM, we additionally make use of parts of the Shuffled News, LDC English Gigaword and 10^9-French-English corpora, selected by a cross-entropy difference criterion (Moore and Lewis, 2010). In total, 1.7 billion running words are taken for LM training. The BOLT Chinese→English task is evaluated on the "discussion forum" domain. The 5-gram LM is trained on 2.9 billion running words in total. The in-domain data consists of a subset of 67.8K sentences and we used a set of 1845 sentences for tuning. The evaluation set `test1` contains 1844 and `test2` 1124 sentences. For the WMT task, we used the target side of the bilingual data and all monolingual data to train a pruned 5-gram LM on a total of 4.4 billion running words. We concatenated the `newstest2011` and `newstest2012` corpora for tuning the systems.

5.2 Results

We start with the IWSLT 2013 German→ English task, where we compare between the different JTR and OSM models. The results are shown in Table 3. When comparing the in-domain *n*-gram JTR model trained using Kneser-Ney smoothing (KN) to OSM, we observe that the *n*-gram KN JTR model improves the baseline by 1.4 BLEU on both `test` and `eval11`. The OSM model performs similarly, with a slight disadvantage on `eval11`. In comparison, the FFNN of Eq. (4) improves the baseline by 0.7–0.9 BLEU, compared to the slightly better 0.8–1.1 BLEU achieved by the URNN. The difference between the FFNN and the

[4] `http://www.ted.com/`

	data	dev	test	eval11
baseline	full	33.3	30.8	35.7
+OSM	TED	34.5	**32.2**	36.8
+FFNN	TED	34.0	31.7	36.4
+URNN	TED	34.2	31.9	36.5
+BRNN	TED	34.4	32.1	36.8
+KN	TED	**34.6**	**32.2**	**37.1**
+BRNN	TED	**35.0**	**32.8**	**37.7**
+OSM	full	34.1	**31.6**	36.5
+FFNN	full	33.9	31.5	36.0
+KN	full	**34.2**	**31.6**	**36.6**
+KN	TED	34.9	32.4	37.1
+FFNN	TED	35.2	32.7	37.2
+FFNN	full	35.1	32.7	37.2
+BRNN	TED	**35.5**	**33.0**	**37.4**
+BRNN	TED	35.4	**33.0**	37.3

Table 3: Results measured in BLEU for the IWSLT German→English task.

	train data	test1	test2
baseline		18.1	17.0
+OSM	indomain	**18.8**	17.2
+FFNN	indomain	18.6	**17.6**
+BRNN	indomain	18.6	**17.6**
+KN	indomain	**18.8**	17.5
+OSM	full	18.5	17.2
+FFNN	full	18.4	**17.4**
+KN	full	**18.8**	17.3
+KN	indomain	19.0	17.7
+FFNN	full	19.2	18.3
+RNN	indomain	**19.3**	**18.4**

Table 4: Results measured in BLEU for the BOLT Chinese→English task.

URNN is that the latter captures the unbounded source and target history that extends until the beginning of the sentences, giving it an advantage over the FFNN. The performance of the URNN can be improved by including the future part of the source sentence, as described in Eq. (5), resulting in the BRNN model. Next, we explore whether the models are additive. When rescoring the *n*-gram KN JTR output with the BRNN, an additional improvement of 0.6 BLEU is obtained. There are two reasons for this: The BRNN includes the future part of the source input when scoring target words. This information is not used by the KN model. Moreover, the BRNN is able to score word combinations unseen in training, while the KN model uses backing off to score unseen events.

When training the KN, FFNN, and OSM models on the full data, we observe less gains in comparison to in-domain data training. However, combining the KN models trained on in-domain and full data gives additional gains, which suggests that although the in-domain model is more adapted to the task, it still can gain from out-of-domain data. Adding the FFNN on top improves the combination. Note here that the FFNN sees the same information as the KN model, but the difference is that the NN operates on the word level rather than the word-pair level. Second, the FFNN is able to handle unseen sequences by design, without the need for the backing off workaround. The BRNN improves the combination more than the FFNN, as the model captures an unbounded source and target history in addition to an unbounded future source context. Combining the KN, FFNN and BRNN JTR models leads to an overall gain of 2.2 BLEU on both dev and test.

Next, we present the BOLT Chinese→English results, shown in Table 4. Comparing *n*-gram KN JTR and OSM trained on the in-domain data shows they perform equally well on test1, improving the baseline by 0.7 BLEU, with a slight advantage for the JTR model on test2. The feed-forward and the recurrent in-domain networks yield the same results in comparison to each other. Training the OSM and JTR models on the full data yields slightly worse results than in-domain training. However, combining the two types of training improves the results. This is shown when adding the in-domain KN JTR model on top of the model trained on full data, improving it by up to 0.4 BLEU. Rescoring with the feed-forward and the recurrent network improves this even further, supporting the previous observation that the *n*-gram KN JTR and NNs complement each other. The combination of the 4 models yields an overall improvement of 1.2–1.4 BLEU.

Finally, we compare KN JTR and OSM models on the WMT German→English task in Table 5. The two models perform almost similar to each other. The JTR model improves the baseline by up to 0.7 BLEU. Rescoring the KN JTR with the FFNN improves it by up to 0.3 BLEU leading to an overall improvement between 0.5 and 1.0 BLEU.

	newstest		
	2013	2014	2015
baseline	28.1	28.6	29.4
+OSM	28.6	**28.9**	**30.0**
+FFNN	28.7	**28.9**	29.7
+KN	**28.8**	**28.9**	29.9
+FFNN	**29.1**	**29.1**	**30.0**

Table 5: Results measured in BLEU for the WMT German→English task.

5.3 Analysis

To investigate the effect of including jump information in the JTR sequence, we trained a BRNN using jump classes and another excluding them. The BRNNs were used in rescoring. Below, we demonstrate the difference between the systems:

source: wir kommen später noch auf diese Leute zurück .
reference: We'll come back to these people later .
Hypothesis 1:
JTR source: wir kommen δ zurück δ später noch auf diese Leute δ .
JTR target: we come $\curvearrowright$ back $\curvearrowleft$ later σ to these people $\curvearrowright$.
Hypothesis 2:
JTR source: wir kommen später noch auf diese Leute zurück .
JTR target: we come later σ on these guys back .

Note the German verb "zurückkommen", which is split into "kommen" and "zurück". German places "kommen" at the second position and "zurück" towards the end of the sentence. Unlike German, the corresponding English phrase "come back" has the words adjacent to each other. We found that the system including jumps prefers the correct translation of the verb, as shown in Hypothesis 1 above. The system translates "kommen" to "come", jumps forward to "zurück", translates it to "back", then jumps back to continue translating the word "später". In contrast, the system that excludes jump classes is blind to this separation of words. It favors Hypothesis 2 which is a strictly monotone translation of the German sentence. This is also reflected by the BLEU scores, where we found the system including jump classes outperforming the one without by up to 0.8 BLEU.

6 Conclusion

We introduced a method that converts bilingual sentence pairs and their word alignments into joint translation and reordering (JTR) sequences. They combine interdepending lexical and alignment dependencies into a single framework. A main advantage of JTR sequences is that a variety of models can be trained on them. Here, we have estimated *n*-gram models with modified Kneser-Ney smoothing, FFNN and RNN architectures on JTR sequences.

We compared our count-based JTR model to the OSM, both used in phrase-based decoding, and showed that the JTR model performed at least as good as OSM, with a slight advantage for JTR. In comparison to the OSM, the JTR model operates on words, leading to a smaller vocabulary size. Moreover, it utilizes simpler reordering structures without gaps and only requires one log-linear feature to be tuned, whereas the OSM needs 5. Due to the flexibility of JTR sequences, we can apply them also to FFNNs and RNNs. Utilizing two count models and applying both networks in rescoring gains the overall highest improvement over the phrase-based system by up to 2.2 BLEU, on the German→English IWSLT task. The combination outperforms OSM by up to 1.2 BLEU on the BOLT Chinese→English tasks.

The JTR models are not dependent on the phrase-based framework, and one of the long-term goals is to perform standalone decoding with the JTR models independently of phrase-based systems. Without the limitations introduced by phrases, we believe that JTR models could perform even better. In addition, we aim to use JTR models to obtain the alignment, which would then be used to train the JTR models in an iterative manner, achieving consistency and hoping for improved models.

Acknowledgements

This work has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement n° 645452 (QT21). This material is partially based upon work supported by the DARPA BOLT project under Contract No. HR0011- 12-C-0015. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of DARPA.

References

Tamer Alkhouli, Felix Rietig, and Hermann Ney. 2015. Investigations on phrase-based decoding with recurrent neural network language and translation models. In *Proceedings of the EMNLP 2015 Tenth Workshop on Statistical Machine Translation*, Lisbon, Portugal, September. to appear.

Michael Auli, Michel Galley, Chris Quirk, and Geoffrey Zweig. 2013. Joint Language and Translation Modeling with Recurrent Neural Networks. In *Conference on Empirical Methods in Natural Language Processing*, pages 1044–1054, Seattle, USA, October.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations*, San Diego, Calefornia, USA, May.

Peter F. Brown, John Cocke, Stephan A. Della Pietra, Vincent J. Della Pietra, Fredrick Jelinek, John D. Lafferty, Robert L. Mercer, and Paul S. Rossin. 1990. A Statistical Approach to Machine Translation. *Computational Linguistics*, 16(2):79–85, June.

Peter F. Brown, Stephan A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. 1993. The Mathematics of Statistical Machine Translation: Parameter Estimation. *Computational Linguistics*, 19(2):263–311, June.

Mauro Cettolo, Jan Niehues, Sebastian Stüker, Luisa Bentivogli, and Marcello Federico. 2014. Report on the 11th iwslt evaluation campaign, iwslt 2014. In *International Workshop on Spoken Language Translation*, pages 2–11, Lake Tahoe, CA, USA, December.

Stanley F. Chen and Joshuo Goodman. 1998. An Empirical Study of Smoothing Techniques for Language Modeling. Technical Report TR-10-98, Computer Science Group, Harvard University, Cambridge, MA, August.

Boxing Chen, Roland Kuhn, George Foster, and Howard Johnson. 2011. Unpacking and transforming feature functions: New ways to smooth phrase tables. In *MT Summit XIII*, pages 269–275, Xiamen, China, September.

Jonathan H. Clark, Chris Dyer, Alon Lavie, and Noah A. Smith. 2011. Better hypothesis testing for statistical machine translation: Controlling for optimizer instability. In *49th Annual Meeting of the Association for Computational Linguistics:shortpapers*, pages 176–181, Portland, Oregon, June.

Josep Maria Crego and François Yvon. 2010. Improving reordering with linguistically informed bilingual n-grams. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010: Posters)*, pages 197–205, Beijing, China.

Yonggang Deng and William Byrne. 2005. Hmm word and phrase alignment for statistical machine translation. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 169–176, Vancouver, British Columbia, Canada, October.

Jacob Devlin, Rabih Zbib, Zhongqiang Huang, Thomas Lamar, Richard Schwartz, and John Makhoul. 2014. Fast and Robust Neural Network Joint Models for Statistical Machine Translation. In *52nd Annual Meeting of the Association for Computational Linguistics*, pages 1370–1380, Baltimore, MD, USA, June.

Nadir Durrani, Helmut Schmid, and Alexander Fraser. 2011. A joint sequence translation model with integrated reordering. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1045–1054, Portland, Oregon, USA, June.

Nadir Durrani, Alexander Fraser, and Helmut Schmid. 2013a. Model with minimal translation units, but decode with phrases. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1–11, Atlanta, Georgia, June.

Nadir Durrani, Alexander Fraser, Helmut Schmid, Hieu Hoang, and Philipp Koehn. 2013b. Can markov models over minimal translation units help phrase-based smt? In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 399–405, Sofia, Bulgaria, August.

Nadir Durrani, Philipp Koehn, Helmut Schmid, and Alexander Fraser. 2014. Investigating the usefulness of generalized word representations in smt. In *COLING*, Dublin, Ireland, August.

Yang Feng and Trevor Cohn. 2013. A markov model of machine translation using non-parametric bayesian inference. In *51st Annual Meeting of the Association for Computational Linguistics*, pages 333–342, Sofia, Bulgaria, August.

Minwei Feng, Jan-Thorsten Peter, and Hermann Ney. 2013. Advancements in reordering models for statistical machine translation. In *Annual Meeting of the Assoc. for Computational Linguistics*, pages 322–332, Sofia, Bulgaria, August.

Yang Feng, Trevor Cohn, and Xinkai Du. 2014. Factored markov translation with robust modeling. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*, pages 151–159, Ann Arbor, Michigan, June.

Michel Galley and Christopher D. Manning. 2008. A simple and effective hierarchical phrase reordering model. In *Proceedings of the Conference on*

Empirical Methods in Natural Language Processing, EMNLP '08, pages 848–856, Stroudsburg, PA, USA. Association for Computational Linguistics.

Andreas Guta, Joern Wuebker, Miguel Graça, Yunsu Kim, and Hermann Ney. 2015. Extended translation models in phrase-based decoding. In *Proceedings of the EMNLP 2015 Tenth Workshop on Statistical Machine Translation*, Lisbon, Portugal, September. to appear.

Kenneth Heafield, Ivan Pouzyrevsky, Jonathan H. Clark, and Philipp Koehn. 2013. Scalable modified Kneser-Ney language model estimation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 690–696, Sofia, Bulgaria, August.

Yuening Hu, Michael Auli, Qin Gao, and Jianfeng Gao. 2014. Minimum translation modeling with recurrent neural networks. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 20–29, Gothenburg, Sweden, April.

P. Koehn, F. J. Och, and D. Marcu. 2003. Statistical Phrase-Based Translation. In *Proceedings of the 2003 Meeting of the North American chapter of the Association for Computational Linguistics (NAACL-03)*, pages 127–133, Edmonton, Alberta.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantine, and Evan Herbst. 2007. Moses: Open Source Toolkit for Statistical Machine Translation. pages 177–180, Prague, Czech Republic, June.

Hai Son Le, Alexandre Allauzen, and François Yvon. 2012. Continuous Space Translation Models with Neural Networks. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 39–48, Montreal, Canada, June.

José B Mariño, Rafael E Banchs, Josep M Crego, Adrià de Gispert, Patrik Lambert, José A R Fonollosa, and Marta R Costa-jussà. 2006. N-gram-based Machine Translation. *Comput. Linguist.*, 32(4):527–549, December.

R.C. Moore and W. Lewis. 2010. Intelligent Selection of Language Model Training Data. In *ACL (Short Papers)*, pages 220–224, Uppsala, Sweden, July.

Jan Niehues, Teresa Herrmann, Stephan Vogel, and Alex Waibel, 2011. *Proceedings of the Sixth Workshop on Statistical Machine Translation*, chapter Wider Context by Using Bilingual Language Models in Machine Translation, pages 198–206.

Franz J. Och and Hermann Ney. 2003. A Systematic Comparison of Various Statistical Alignment Models. *Computational Linguistics*, 29(1):19–51, March.

Franz J. Och, Christoph Tillmann, and Hermann Ney. 1999. Improved Alignment Models for Statistical Machine Translation. In *Proc. Joint SIGDAT Conf. on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 20–28, University of Maryland, College Park, MD, June.

Franz Josef Och. 2003. Minimum Error Rate Training in Statistical Machine Translation. In *Proc. of the 41th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 160–167, Sapporo, Japan, July.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2001. Bleu: a Method for Automatic Evaluation of Machine Translation. IBM Research Report RC22176 (W0109-022), IBM Research Division, Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, NY 10598, September.

Darelene Stewart, Roland Kuhn, Eric Joanis, and George Foster. 2014. Coarse split and lump bilingual languagemodels for richer source information in smt. In *AMTA*, Vancouver, BC, Canada, October.

Martin Sundermeyer, Tamer Alkhouli, Wuebker Wuebker, and Hermann Ney. 2014. Translation Modeling with Bidirectional Recurrent Neural Networks. In *Conference on Empirical Methods on Natural Language Processing*, pages 14–25, Doha, Qatar, October.

Ilya Sutskever, Oriol Vinyals, and Quoc V. V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27*, pages 3104–3112.

Christoph Tillmann. 2004. A unigram orientation model for statistical machine translation. In *Proceedings of HLT-NAACL 2004: Short Papers*, HLT-NAACL-Short '04, pages 101–104, Stroudsburg, PA, USA.

Joern Wuebker, Stephan Peitz, Felix Rietig, and Hermann Ney. 2013. Improving statistical machine translation with word class models. In *Conference on Empirical Methods in Natural Language Processing*, pages 1377–1381, Seattle, USA, October.

Richard Zens, Franz Josef Och, and Hermann Ney. 2002. Phrase-Based Statistical Machine Translation. In *25th German Conf. on Artificial Intelligence (KI2002)*, pages 18–32, Aachen, Germany, September.

Hui Zhang, Kristina Toutanova, Chris Quirk, and Jianfeng Gao. 2013. Beyond left-to-right: Multiple decomposition structures for smt. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 12–21, Atlanta, Georgia, June.

Effective Approaches to Attention-based Neural Machine Translation

Minh-Thang Luong **Hieu Pham** **Christopher D. Manning**
Computer Science Department, Stanford University, Stanford, CA 94305
{lmthang,hyhieu,manning}@stanford.edu

Abstract

An attentional mechanism has lately been used to improve neural machine translation (NMT) by selectively focusing on parts of the source sentence during translation. However, there has been little work exploring useful architectures for attention-based NMT. This paper examines two simple and effective classes of attentional mechanism: a *global* approach which always attends to all source words and a *local* one that only looks at a subset of source words at a time. We demonstrate the effectiveness of both approaches on the WMT translation tasks between English and German in both directions. With local attention, we achieve a significant gain of 5.0 BLEU points over non-attentional systems that already incorporate known techniques such as dropout. Our ensemble model using different attention architectures yields a new state-of-the-art result in the WMT'15 English to German translation task with 25.9 BLEU points, an improvement of 1.0 BLEU points over the existing best system backed by NMT and an n-gram reranker.[1]

1 Introduction

Neural Machine Translation (NMT) achieved state-of-the-art performances in large-scale translation tasks such as from English to French (Luong et al., 2015) and English to German (Jean et al., 2015). NMT is appealing since it requires minimal domain knowledge and is conceptually simple. The model by Luong et al. (2015) reads through all the source words until the end-of-sentence symbol <eos> is reached. It then starts emitting one target word at a time, as illustrated in Figure 1. NMT

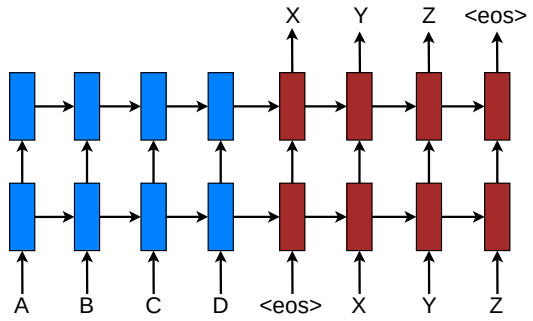


Figure 1: **Neural machine translation** – a stacking recurrent architecture for translating a source sequence A B C D into a target sequence X Y Z. Here, <eos> marks the end of a sentence.

is often a large neural network that is trained in an end-to-end fashion and has the ability to generalize well to very long word sequences. This means the model does not have to explicitly store gigantic phrase tables and language models as in the case of standard MT; hence, NMT has a small memory footprint. Lastly, implementing NMT decoders is easy unlike the highly intricate decoders in standard MT (Koehn et al., 2003).

In parallel, the concept of "attention" has gained popularity recently in training neural networks, allowing models to learn alignments between different modalities, e.g., between image objects and agent actions in the dynamic control problem (Mnih et al., 2014), between speech frames and text in the speech recognition task (Chorowski et al., 2014), or between visual features of a picture and its text description in the image caption generation task (Xu et al., 2015). In the context of NMT, Bahdanau et al. (2015) has successfully applied such attentional mechanism to jointly translate and align words. To the best of our knowledge, there has not been any other work exploring the use of attention-based architectures for NMT.

In this work, we design, with simplicity and effectiveness in mind, two novel types of attention-

[1] All our code and models are publicly available at http://nlp.stanford.edu/projects/nmt.

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1412–1421,
Lisbon, Portugal, 17-21 September 2015.

based models: a *global* approach in which all source words are attended and a *local* one whereby only a subset of source words are considered at a time. The former approach resembles the model of (Bahdanau et al., 2015) but is simpler architecturally. The latter can be viewed as an interesting blend between the *hard* and *soft* attention models proposed in (Xu et al., 2015): it is computationally less expensive than the global model or the soft attention; at the same time, unlike the hard attention, the local attention is differentiable, making it easier to implement and train.[2] Besides, we also examine various alignment functions for our attention-based models.

Experimentally, we demonstrate that both of our approaches are effective in the WMT translation tasks between English and German in both directions. Our attentional models yield a boost of up to 5.0 BLEU over non-attentional systems which already incorporate known techniques such as dropout. For English to German translation, we achieve new state-of-the-art (SOTA) results for both WMT'14 and WMT'15, outperforming previous SOTA systems, backed by NMT models and n-gram LM rerankers, by more than 1.0 BLEU. We conduct extensive analysis to evaluate our models in terms of learning, the ability to handle long sentences, choices of attentional architectures, alignment quality, and translation outputs.

2 Neural Machine Translation

A neural machine translation system is a neural network that directly models the conditional probability $p(y|x)$ of translating a source sentence, $x_1, \ldots, x_n$, to a target sentence, $y_1, \ldots, y_m$.[3] A basic form of NMT consists of two components: (a) an *encoder* which computes a representation $\boldsymbol{s}$ for each source sentence and (b) a *decoder* which generates one target word at a time and hence decomposes the conditional probability as:

$$\log p(y|x) = \sum_{j=1}^{m} \log p\left(y_j|y_{<j}, \boldsymbol{s}\right) \quad (1)$$

A natural choice to model such a decomposition in the decoder is to use a recurrent neural network (RNN) architecture, which most of the recent NMT work such as (Kalchbrenner and Blunsom, 2013; Sutskever et al., 2014; Cho et al., 2014; Bahdanau et al., 2015; Luong et al., 2015; Jean et al., 2015) have in common. They, however, differ in terms of which RNN architectures are used for the decoder and how the encoder computes the source sentence representation $\boldsymbol{s}$.

Kalchbrenner and Blunsom (2013) used an RNN with the standard hidden unit for the decoder and a convolutional neural network for encoding the source sentence representation. On the other hand, both Sutskever et al. (2014) and Luong et al. (2015) stacked multiple layers of an RNN with a Long Short-Term Memory (LSTM) hidden unit for both the encoder and the decoder. Cho et al. (2014), Bahdanau et al. (2015), and Jean et al. (2015) all adopted a different version of the RNN with an LSTM-inspired hidden unit, the gated recurrent unit (GRU), for both components.[4]

In more detail, one can parameterize the probability of decoding each word y_j as:

$$p\left(y_j|y_{<j}, \boldsymbol{s}\right) = \mathrm{softmax}\left(g\left(\boldsymbol{h}_j\right)\right) \quad (2)$$

with g being the transformation function that outputs a vocabulary-sized vector.[5] Here, $\boldsymbol{h}_j$ is the RNN hidden unit, abstractly computed as:

$$\boldsymbol{h}_j = f(\boldsymbol{h}_{j-1}, \boldsymbol{s}), \quad (3)$$

where f computes the current hidden state given the previous hidden state and can be either a vanilla RNN unit, a GRU, or an LSTM unit. In (Kalchbrenner and Blunsom, 2013; Sutskever et al., 2014; Cho et al., 2014; Luong et al., 2015), the source representation $\boldsymbol{s}$ is only used once to initialize the decoder hidden state. On the other hand, in (Bahdanau et al., 2015; Jean et al., 2015) and this work, $\boldsymbol{s}$, in fact, implies a set of source hidden states which are consulted throughout the entire course of the translation process. Such an approach is referred to as an attention mechanism, which we will discuss next.

In this work, following (Sutskever et al., 2014; Luong et al., 2015), we use the stacking LSTM architecture for our NMT systems, as illustrated in Figure 1. We use the LSTM unit defined in (Zaremba et al., 2015). Our training objective is formulated as follows:

$$J_t = \sum_{(x,y)\in D} -\log p(y|x) \quad (4)$$

[2] There is a recent work by Gregor et al. (2015), which is very similar to our local attention and applied to the image generation task. However, as we detail later, our model is much simpler and can achieve good performance for NMT.

[3] All sentences are assumed to terminate with a special "end-of-sentence" token <eos>.

[4] They all used a single RNN layer except for the latter two works which utilized a bidirectional RNN for the encoder.

[5] One can provide g with other inputs such as the currently predicted word y_j as in (Bahdanau et al., 2015).

with D being our parallel training corpus.

3 Attention-based Models

Our various attention-based models are classifed into two broad categories, *global* and *local*. These classes differ in terms of whether the "attention" is placed on all source positions or on only a few source positions. We illustrate these two model types in Figure 2 and 3 respectively.

Common to these two types of models is the fact that at each time step t in the decoding phase, both approaches first take as input the hidden state $\boldsymbol{h}_t$ at the top layer of a stacking LSTM. The goal is then to derive a context vector $\boldsymbol{c}_t$ that captures relevant source-side information to help predict the current target word y_t. While these models differ in how the context vector $\boldsymbol{c}_t$ is derived, they share the same subsequent steps.

Specifically, given the target hidden state $\boldsymbol{h}_t$ and the source-side context vector $\boldsymbol{c}_t$, we employ a simple concatenation layer to combine the information from both vectors to produce an attentional hidden state as follows:

$$\tilde{\boldsymbol{h}}_t = \tanh(\boldsymbol{W_c}[\boldsymbol{c}_t; \boldsymbol{h}_t]) \tag{5}$$

The attentional vector $\tilde{\boldsymbol{h}}_t$ is then fed through the softmax layer to produce the predictive distribution formulated as:

$$p(y_t|y_{<t}, x) = \text{softmax}(\boldsymbol{W_s}\tilde{\boldsymbol{h}}_t) \tag{6}$$

We now detail how each model type computes the source-side context vector $\boldsymbol{c}_t$.

3.1 Global Attention

The idea of a global attentional model is to consider all the hidden states of the encoder when deriving the context vector c_t. In this model type, a variable-length alignment vector $\boldsymbol{a}_t$, whose size equals the number of time steps on the source side, is derived by comparing the current target hidden state $\boldsymbol{h}_t$ with each source hidden state $\bar{\boldsymbol{h}}_s$:

$$\begin{aligned}\boldsymbol{a}_t(s) &= \text{align}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s) \\ &= \frac{\exp\left(\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s)\right)}{\sum_{s'} \exp\left(\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_{s'})\right)}\end{aligned} \tag{7}$$

Here, score is referred as a *content-based* function for which we consider three different alternatives:

$$\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s) = \begin{cases} \boldsymbol{h}_t^\top \bar{\boldsymbol{h}}_s & \textit{dot} \\ \boldsymbol{h}_t^\top \boldsymbol{W_a} \bar{\boldsymbol{h}}_s & \textit{general} \\ \boldsymbol{W_a}[\boldsymbol{h}_t; \bar{\boldsymbol{h}}_s] & \textit{concat} \end{cases} \tag{8}$$

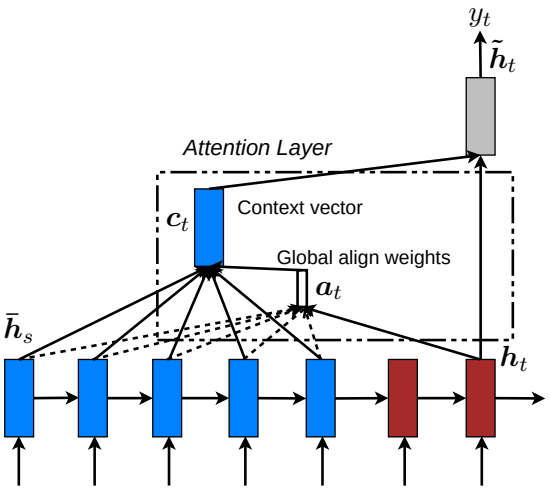


Figure 2: **Global attentional model** – at each time step t, the model infers a *variable-length* alignment weight vector $\boldsymbol{a}_t$ based on the current target state $\boldsymbol{h}_t$ and all source states $\bar{\boldsymbol{h}}_s$. A global context vector $\boldsymbol{c}_t$ is then computed as the weighted average, according to $\boldsymbol{a}_t$, over all the source states.

Besides, in our early attempts to build attention-based models, we use a *location-based* function in which the alignment scores are computed from solely the target hidden state $\boldsymbol{h}_t$ as follows:

$$\boldsymbol{a}_t = \text{softmax}(\boldsymbol{W_a}\boldsymbol{h}_t) \qquad \textit{location} \tag{9}$$

Given the alignment vector as weights, the context vector c_t is computed as the weighted average over all the source hidden states.[6]

Comparison to (Bahdanau et al., 2015) – While our global attention approach is similar in spirit to the model proposed by Bahdanau et al. (2015), there are several key differences which reflect how we have both simplified and generalized from the original model. First, we simply use hidden states at the top LSTM layers in both the encoder and decoder as illustrated in Figure 2. Bahdanau et al. (2015), on the other hand, use the concatenation of the forward and backward source hidden states in the bi-directional encoder and target hidden states in their non-stacking uni-directional decoder. Second, our computation path is simpler; we go from $\boldsymbol{h}_t \rightarrow \boldsymbol{a}_t \rightarrow \boldsymbol{c}_t \rightarrow \tilde{\boldsymbol{h}}_t$ then make a prediction as detailed in Eq. (5), Eq. (6), and Figure 2. On the other hand, at any time t, Bahdanau et al. (2015) build from the previous hidden state $\boldsymbol{h}_{t-1} \rightarrow \boldsymbol{a}_t \rightarrow \boldsymbol{c}_t \rightarrow \boldsymbol{h}_t$, which, in turn,

[6] Eq. (9) implies that all alignment vectors $\boldsymbol{a}_t$ are of the same length. For short sentences, we only use the top part of $\boldsymbol{a}_t$ and for long sentences, we ignore words near the end.

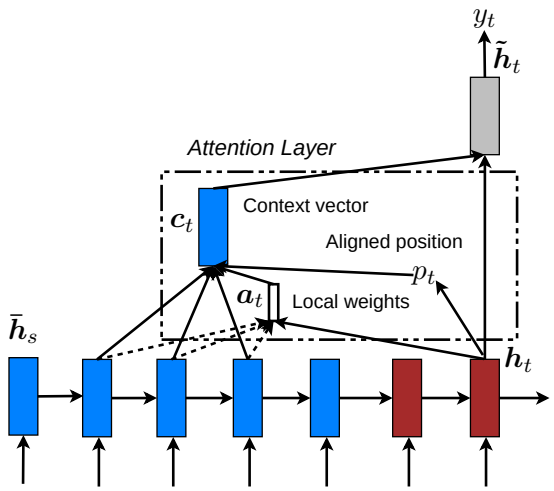


Figure 3: **Local attention model** – the model first predicts a single aligned position p_t for the current target word. A window centered around the source position p_t is then used to compute a context vector $\boldsymbol{c}_t$, a weighted average of the source hidden states in the window. The weights $\boldsymbol{a_t}$ are inferred from the current target state $\boldsymbol{h}_t$ and those source states $\bar{\boldsymbol{h}}_s$ in the window.

goes through a deep-output and a maxout layer before making predictions.[7] Lastly, Bahdanau et al. (2015) only experimented with one alignment function, the *concat* product; whereas we show later that the other alternatives are better.

3.2 Local Attention

The global attention has a drawback that it has to attend to all words on the source side for each target word, which is expensive and can potentially render it impractical to translate longer sequences, e.g., paragraphs or documents. To address this deficiency, we propose a *local* attentional mechanism that chooses to focus only on a small subset of the source positions per target word.

This model takes inspiration from the tradeoff between the *soft* and *hard* attentional models proposed by Xu et al. (2015) to tackle the image caption generation task. In their work, soft attention refers to the global attention approach in which weights are placed "softly" over all patches in the source image. The hard attention, on the other hand, selects one patch of the image to attend to at a time. While less expensive at inference time, the hard attention model is non-differentiable and requires more complicated techniques such as variance reduction or reinforcement learning to train.

[7] We will refer to this difference again in Section 3.3.

Our local attention mechanism selectively focuses on a small window of context and is differentiable. This approach has an advantage of avoiding the expensive computation incurred in the soft attention and at the same time, is easier to train than the hard attention approach. In concrete details, the model first generates an aligned position p_t for each target word at time t. The context vector $\boldsymbol{c}_t$ is then derived as a weighted average over the set of source hidden states within the window $[p_t - D, p_t + D]$; D is empirically selected.[8] Unlike the global approach, the local alignment vector $\boldsymbol{a}_t$ is now fixed-dimensional, i.e., $\in \mathbb{R}^{2D+1}$. We consider two variants of the model as below.

Monotonic alignment (**local-m**) – we simply set $p_t = t$ assuming that source and target sequences are roughly monotonically aligned. The alignment vector $\boldsymbol{a}_t$ is defined according to Eq. (7).[9]

Predictive alignment (**local-p**) – instead of assuming monotonic alignments, our model predicts an aligned position as follows:

$$p_t = S \cdot \text{sigmoid}(\boldsymbol{v}_p^\top \tanh(\boldsymbol{W_p h}_t)), \qquad (10)$$

$\boldsymbol{W_p}$ and $\boldsymbol{v}_p$ are the model parameters which will be learned to predict positions. S is the source sentence length. As a result of sigmoid, $p_t \in [0, S]$. To favor alignment points near p_t, we place a Gaussian distribution centered around p_t . Specifically, our alignment weights are now defined as:

$$\boldsymbol{a}_t(s) = \text{align}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s) \exp\left(-\frac{(s - p_t)^2}{2\sigma^2}\right) \qquad (11)$$

We use the same align function as in Eq. (7) and the standard deviation is empirically set as $\sigma = \frac{D}{2}$. It is important to note that p_t is a *real* nummber; whereas s is an *integer* within the window centered at p_t.[10]

Comparison to (Gregor et al., 2015) – have proposed a *selective attention* mechanism, very similar to our local attention, for the image generation task. Their approach allows the model to select an image patch of varying location and zoom. We, instead, use the same "zoom" for all target positions, which greatly simplifies the formulation and still achieves good performance.

[8] If the window crosses the sentence boundaries, we simply ignore the outside part and consider words in the window.

[9] *local-m* is the same as the global model except that the vector $\boldsymbol{a}_t$ is fixed-length and shorter.

[10] *local-p* is similar to the local-m model except that we dynamically compute p_t and use a Gaussian distribution to modify the original alignment weights $\text{align}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s)$ as shown in Eq. (11). By utilizing p_t to derive $\boldsymbol{a}_t$, we can compute backprop gradients for $\boldsymbol{W_p}$ and $\boldsymbol{v}_p$.

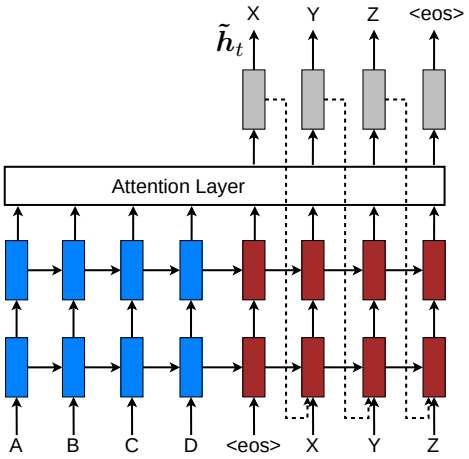


Figure 4: **Input-feeding approach** – Attentional vectors $\tilde{\boldsymbol{h}}_t$ are fed as inputs to the next time steps to inform the model about past alignment decisions.

3.3 Input-feeding Approach

In our proposed global and local approaches, the attentional decisions are made independently, which is suboptimal. Whereas, in standard MT, a *coverage* set is often maintained during the translation process to keep track of which source words have been translated. Likewise, in attentional NMTs, alignment decisions should be made jointly taking into account past alignment information. To address that, we propose an *input-feeding* approach in which attentional vectors $\tilde{\boldsymbol{h}}_t$ are concatenated with inputs at the next time steps as illustrated in Figure 4.[11] The effects of having such connections are two-fold: (a) we hope to make the model fully aware of previous alignment choices and (b) we create a very deep network spanning both horizontally and vertically.

Comparison to other work – Bahdanau et al. (2015) use context vectors, similar to our $\boldsymbol{c}_t$, in building subsequent hidden states, which can also achieve the "coverage" effect. However, there has not been any analysis of whether such connections are useful as done in this work. Also, our approach is more general; as illustrated in Figure 4, it can be applied to general stacking recurrent architectures, including non-attentional models.

Xu et al. (2015) propose a *doubly attentional* approach with an additional constraint added to the training objective to make sure the model pays equal attention to all parts of the image during the caption generation process. Such a constraint can also be useful to capture the coverage set effect in NMT that we mentioned earlier. However, we chose to use the input-feeding approach since it provides flexibility for the model to decide on any attentional constraints it deems suitable.

[11] If n is the number of LSTM cells, the input size of the first LSTM layer is $2n$; those of subsequent layers are n.

4 Experiments

We evaluate the effectiveness of our models on the WMT translation tasks between English and German in both directions. newstest2013 (3000 sentences) is used as a development set to select our hyperparameters. Translation performances are reported in case-sensitive BLEU (Papineni et al., 2002) on newstest2014 (2737 sentences) and newstest2015 (2169 sentences). Following (Luong et al., 2015), we report translation quality using two types of BLEU: (a) *tokenized*[12] BLEU to be comparable with existing NMT work and (b) *NIST*[13] BLEU to be comparable with WMT results.

4.1 Training Details

All our models are trained on the WMT'14 training data consisting of 4.5M sentences pairs (116M English words, 110M German words). Similar to (Jean et al., 2015), we limit our vocabularies to be the top 50K most frequent words for both languages. Words not in these shortlisted vocabularies are converted into a universal token `<unk>`.

When training our NMT systems, following (Bahdanau et al., 2015; Jean et al., 2015), we filter out sentence pairs whose lengths exceed 50 words and shuffle mini-batches as we proceed. Our stacking LSTM models have 4 layers, each with 1000 cells, and 1000-dimensional embeddings. We follow (Sutskever et al., 2014; Luong et al., 2015) in training NMT with similar settings: (a) our parameters are uniformly initialized in $[-0.1, 0.1]$, (b) we train for 10 epochs using plain SGD, (c) a simple learning rate schedule is employed – we start with a learning rate of 1; after 5 epochs, we begin to halve the learning rate every epoch, (d) our mini-batch size is 128, and (e) the normalized gradient is rescaled whenever its norm exceeds 5. Additionally, we also use dropout for our LSTMs as suggested by (Zaremba et al., 2015). For dropout models, we train for 12 epochs and start halving the learning rate after 8 epochs.

Our code is implemented in MATLAB. When

[12] All texts are tokenized with `tokenizer.perl` and BLEU scores are computed with `multi-bleu.perl`.

[13] With the `mteval-v13a` script as per WMT guideline.

System	Ppl	BLEU
Winning WMT'14 system – *phrase-based + large LM* (Buck et al., 2014)		20.7
Existing NMT systems		
RNNsearch (Jean et al., 2015)		16.5
RNNsearch + unk replace (Jean et al., 2015)		19.0
RNNsearch + unk replace + large vocab + *ensemble* 8 models (Jean et al., 2015)		**21.6**
Our NMT systems		
Base	10.6	11.3
Base + reverse	9.9	12.6 (+*1.3*)
Base + reverse + dropout	8.1	14.0 (+*1.4*)
Base + reverse + dropout + global attention (*location*)	7.3	16.8 (+*2.8*)
Base + reverse + dropout + global attention (*location*) + feed input	6.4	18.1 (+*1.3*)
Base + reverse + dropout + local-p attention (*general*) + feed input	5.9	19.0 (+*0.9*)
Base + reverse + dropout + local-p attention (*general*) + feed input + unk replace		20.9 (+*1.9*)
Ensemble 8 models + unk replace		**23.0** (+*2.1*)

Table 1: **WMT'14 English-German results** – shown are the perplexities (ppl) and the *tokenized* BLEU scores of various systems on newstest2014. We highlight the **best** system in bold and give *progressive* improvements in italic between consecutive systems. *local-p* referes to the local attention with predictive alignments. We indicate for each attention model the alignment score function used in pararentheses.

running on a single GPU device Tesla K40, we achieve a speed of 1K *target* words per second. It takes 7–10 days to completely train a model.

4.2 English-German Results

We compare our NMT systems in the English-German task with various other systems. These include the winning system in WMT'14 (Buck et al., 2014), a phrase-based system whose language models were trained on a huge monolingual text, the Common Crawl corpus. For end-to-end neural machine translation systems, to the best of our knowledge, (Jean et al., 2015) is the only work experimenting with this language pair and currently the SOTA system. We only present results for some of our attention models and will later analyze the rest in Section 5.

As shown in Table 1, we achieve progressive improvements when (a) reversing the source sentence, +1.3 BLEU, as proposed in (Sutskever et al., 2014) and (b) using dropout, +1.4 BLEU. On top of that, (c) the global attention approach gives a significant boost of +2.8 BLEU, making our model slightly better than the base attentional system of Bahdanau et al. (2015) (row *RNNSearch*). When (d) using the *input-feeding* approach, we seize another notable gain of +1.3 BLEU and outperform their system. The local attention model with predictive alignments (row *local-p*) proves to be even better, giving us a further improvement of +0.9 BLEU on top of the global attention model. It is interesting to observe the trend previously reported in (Luong et al., 2015) that perplexity strongly correlates with translation quality. In total, we achieve a significant gain of 5.0 BLEU points over the non-attentional baseline, which already includes known techniques such as source reversing and dropout.

The unknown replacement technique proposed in (Luong et al., 2015; Jean et al., 2015) yields another nice gain of +1.9 BLEU, demonstrating that our attentional models do learn useful alignments for unknown works. Finally, by ensembling 8 different models of various settings, e.g., using different attention approaches, with and without dropout etc., we were able to achieve a *new SOTA* result of 23.0 BLEU, outperforming the existing best system (Jean et al., 2015) by +1.4 BLEU.

System	BLEU
SOTA – *NMT + 5-gram rerank* (MILA)	24.9
Our ensemble 8 models + unk replace	**25.9**

Table 2: **WMT'15 English-German results** – *NIST* BLEU scores of the existing WMT'15 SOTA system and our best one on newstest2015.

Latest results in WMT'15 – despite the fact that our models were trained on WMT'14 with slightly less data, we test them on newstest2015 to demonstrate that they can generalize well to different test sets. As shown in Table 2, our best system es-

System	Ppl.	BLEU
WMT'15 systems		
SOTA – *phrase-based* (Edinburgh)		**29.2**
NMT + 5-gram rerank (MILA)		27.6
Our NMT systems		
Base (reverse)	14.3	16.9
+ global (*location*)	12.7	19.1 (+2.2)
+ global (*location*) + feed	10.9	20.1 (+*1.0*)
+ global (*dot*) + drop + feed	9.7	22.8 (+2.7)
+ global (*dot*) + drop + feed + unk		24.9 (+*2.1*)

Table 3: **WMT'15 German-English results** – performances of various systems (similar to Table 1). The *base* system already includes source reversing on which we add *global* attention, *drop*out, input *feed*ing, and *unk* replacement.

tablishes a *new SOTA* performance of 25.9 BLEU, outperforming the existing best system backed by NMT and a 5-gram LM reranker by +1.0 BLEU.

4.3 German-English Results

We carry out a similar set of experiments for the WMT'15 translation task from German to English. While our systems have not yet matched the performance of the SOTA system, we nevertheless show the effectiveness of our approaches with large and progressive gains in terms of BLEU as illustrated in Table 3. The *attentional* mechanism gives us +2.2 BLEU gain and on top of that, we obtain another boost of up to +1.0 BLEU from the *input-feeding* approach. Using a better alignment function, the content-based *dot* product one, together with *dropout* yields another gain of +2.7 BLEU. Lastly, when applying the unknown word replacement technique, we seize an additional +2.1 BLEU, demonstrating the usefulness of attention in aligning rare words.

5 Analysis

We conduct extensive analysis to better understand our models in terms of learning, the ability to handle long sentences, choices of attentional architectures, and alignment quality. All models considered here are English-German NMT systems tested on newstest2014.

5.1 Learning curves

We compare models built on top of one another as listed in Table 1. It is pleasant to observe in Figure 5 a clear separation between non-attentional and attentional models. The input-feeding approach and the local attention model also demonstrate their abilities in driving the test costs lower. The non-attentional model with dropout (the blue + curve) learns slower than other non-dropout models, but as time goes by, it becomes more robust in terms of minimizing test errors.

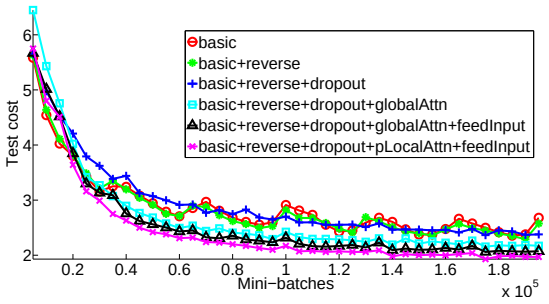


Figure 5: **Learning curves** – test cost (ln perplexity) on newstest2014 for English-German NMTs as training progresses.

5.2 Effects of Translating Long Sentences

We follow (Bahdanau et al., 2015) to group sentences of similar lengths together and compute a BLEU score per group. As demonstrated in Figure 6, our attentional models are more effective than the other non-attentional model in handling long sentences: the translation quality does not degrade as sentences become longer. Our best model (the blue + curve) outperforms all other systems in all length buckets.

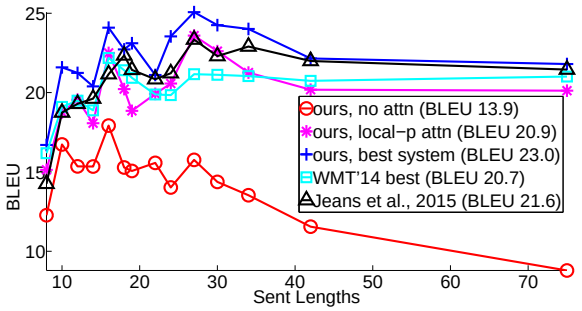


Figure 6: **Length Analysis** – translation qualities of different systems as sentences become longer.

5.3 Choices of Attentional Architectures

We examine different attention models (*global, local-m, local-p*) and different alignment functions (*location, dot, general, concat*) as described in Section 3. Due to limited resources, we cannot run all the possible combinations. However,

System	Ppl	BLEU	
		Before	After unk
global (location)	6.4	18.1	19.3 (+1.2)
global (dot)	6.1	18.6	20.5 (+1.9)
global (general)	6.1	17.3	19.1 (+1.8)
local-m (dot)	>7.0	x	x
local-m (general)	6.2	18.6	20.4 (+1.8)
local-p (dot)	6.6	18.0	19.6 (+1.9)
local-p (general)	**5.9**	**19**	**20.9 (+1.9)**

Table 4: **Attentional Architectures** – performances of different attentional models. We trained two local-m (dot) models; both have ppl > 7.0.

Method	AER
global (location)	0.39
local-m (general)	0.34
local-p (general)	0.36
ensemble	0.34
Berkeley Aligner	0.32

Table 6: **AER scores** – results of various models on the RWTH English-German alignment data.

results in Table 4 do give us some idea about different choices. The *location-based* function does not learn good alignments: the *global (location)* model can only obtain a small gain when performing unknown word replacement compared to using other alignment functions.[14] For *content-based* functions, our implementation of *concat* does not yield good performances and more analysis should be done to understand the reason.[15] It is interesting to observe that *dot* works well for the global attention and *general* is better for the local attention. Among the different models, the local attention model with predictive alignments (*local-p*) is best, both in terms of perplexities and BLEU.

5.4 Alignment Quality

A by-product of attentional models are word alignments. While (Bahdanau et al., 2015) visualized alignments for some sample sentences and observed gains in translation quality as an indication of a working attention model, no work has assessed the alignments learned as a whole. In contrast, we set out to evaluate the alignment quality using the alignment error rate (AER) metric.

Given the gold alignment data provided by RWTH for 508 English-German Europarl sentences, we "force" decode our attentional models to produce translations that match the references. We extract only one-to-one alignments by selecting the source word with the highest alignment weight per target word. Nevertheless, as shown in Table 6, we were able to achieve AER scores comparable to the one-to-many alignments obtained by the Berkeley aligner (Liang et al., 2006).[16]

We also found that the alignments produced by local attention models achieve lower AERs than those of the global one. The AER obtained by the ensemble, while good, is not better than the local-m AER, suggesting the well-known observation that AER and translation scores are not well correlated (Fraser and Marcu, 2007). Due to space constraint, we can only show alignment visualizations in the arXiv version of our paper.[17]

5.5 Sample Translations

We show in Table 5 sample translations in both directions. It it appealing to observe the effect of attentional models in correctly translating names such as "Miranda Kerr" and "Roger Dow". Non-attentional models, while producing sensible names from a language model perspective, lack the direct connections from the source side to make correct translations.

We also observed an interesting case in the second English-German example, which requires translating the *doubly-negated* phrase, "not incompatible". The attentional model correctly produces "nicht ... unvereinbar"; whereas the non-attentional model generates "nicht vereinbar", meaning "not compatible".[18] The attentional model also demonstrates its superiority in translating long sentences as in the last example.

6 Conclusion

In this paper, we propose two simple and effective attentional mechanisms for neural machine

[14]There is a subtle difference in how we retrieve alignments for the different alignment functions. At time step t in which we receive y_{t-1} as input and then compute $\boldsymbol{h}_t, \boldsymbol{a}_t, \boldsymbol{c}_t$, and $\tilde{\boldsymbol{h}}_t$ before predicting y_t, the alignment vector $\boldsymbol{a}_t$ is used as alignment weights for (a) the predicted word y_t in the *location-based* alignment functions and (b) the input word y_{t-1} in the *content-based* functions.

[15]With *concat*, the perplexities achieved by different models are 6.7 (global), 7.1 (local-m), and 7.1 (local-p).

[16]We concatenate the 508 sentence pairs with 1M sentence pairs from WMT and run the Berkeley aligner.

[17]http://arxiv.org/abs/1508.04025

[18]The reference uses a more fancy translation of "incompatible", which is "im Widerspruch zu etwas stehen". Both models, however, failed to translate "passenger experience".

English-German translations	
src	Orlando Bloom and Miranda Kerr still love each other
ref	Orlando Bloom und *Miranda Kerr* lieben sich noch immer
best	Orlando Bloom und *Miranda Kerr* lieben einander noch immer .
base	Orlando Bloom und **Lucas Miranda** lieben einander noch immer .
src	″ We ′ re pleased the FAA recognizes that an enjoyable passenger experience is not incompatible with safety and security , ″ said Roger Dow , CEO of the U.S. Travel Association .
ref	“ Wir freuen uns , dass die FAA erkennt , dass ein angenehmes Passagiererlebnis nicht im Widerspruch zur Sicherheit steht ” , sagte *Roger Dow* , CEO der U.S. Travel Association .
best	″ Wir freuen uns , dass die FAA anerkennt , dass ein angenehmes ist nicht mit Sicherheit und Sicherheit *unvereinbar* ist ″ , sagte *Roger Dow* , CEO der US - die .
base	″ Wir freuen uns über die <unk> , dass ein <unk> <unk> mit Sicherheit nicht **vereinbar** ist mit Sicherheit und Sicherheit ″ , sagte *Roger* **Cameron** , CEO der US - <unk> .
German-English translations	
src	In einem Interview sagte Bloom jedoch , dass er und Kerr sich noch immer lieben .
ref	However , in an interview , Bloom has said that he and *Kerr* still love each other .
best	In an interview , however , Bloom said that he and *Kerr* still love .
base	However , in an interview , Bloom said that he and **Tina** were still <unk> .
src	Wegen der von Berlin und der Europäischen Zentralbank verhängten strengen Sparpolitik in Verbindung mit der Zwangsjacke , in die die jeweilige nationale Wirtschaft durch das Festhalten an der gemeinsamen Währung genötigt wird , sind viele Menschen der Ansicht , das Projekt Europa sei zu weit gegangen
ref	The *austerity imposed by Berlin and the European Central Bank , coupled with the straitjacket* imposed on national economies through adherence to the common currency , has led many people to think Project Europe has gone too far .
best	Because of the strict *austerity measures imposed by Berlin and the European Central Bank in connection with the straitjacket* in which the respective national economy is forced to adhere to the common currency , many people believe that the European project has gone too far .
base	Because of the pressure **imposed by the European Central Bank and the Federal Central Bank with the strict austerity** imposed on the national economy in the face of the single currency , many people believe that the European project has gone too far .

Table 5: **Sample translations** – for each example, we show the source (*src*), the human translation (*ref*), the translation from our best model (*best*), and the translation of a non-attentional model (*base*). We italicize some *correct* translation segments and highlight a few **wrong** ones in bold.

translation: the *global* approach which always looks at all source positions and the *local* one that only attends to a subset of source positions at a time. We test the effectiveness of our models in the WMT translation tasks between English and German in both directions. Our local attention yields large gains of up to 5.0 BLEU over non-attentional models that already incorporate known techniques such as dropout. For the English to German translation direction, our ensemble model has established new state-of-the-art results for both WMT'14 and WMT'15.

We have compared various alignment functions and shed light on which functions are best for which attentional models. Our analysis shows that attention-based NMT models are superior to non-attentional ones in many cases, for example in translating names and handling long sentences.

Acknowledgment

We gratefully acknowledge support from a gift from Bloomberg L.P. and the support of NVIDIA Corporation with the donation of Tesla K40 GPUs. We thank Andrew Ng and his group as well as the Stanford Research Computing for letting us use their computing resources. We thank Russell Stewart for helpful discussions on the models. Lastly, we thank Quoc Le, Ilya Sutskever, Oriol Vinyals, Richard Socher, Michael Kayser, Jiwei Li, Panupong Pasupat, Kelvin Gu, members of the Stanford NLP Group and the annonymous reviewers for their valuable comments and feedback.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

Christian Buck, Kenneth Heafield, and Bas van Ooyen. 2014. N-gram counts and language models from the common crawl. In *LREC*.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *EMNLP*.

Jan Chorowski, Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. End-to-end continuous speech recognition using attention-based recurrent NN: first results. *CoRR*, abs/1412.1602.

Alexander Fraser and Daniel Marcu. 2007. Measuring word alignment quality for statistical machine translation. *Computational Linguistics*, 33(3):293–303.

Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Jimenez Rezende, and Daan Wierstra. 2015. DRAW: A recurrent neural network for image generation. In *ICML*.

Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. On using very large target vocabulary for neural machine translation. In *ACL*.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *EMNLP*.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *NAACL*.

Percy Liang, Ben Taskar, and Dan Klein. 2006. Alignment by agreement. In *NAACL*.

Minh-Thang Luong, Ilya Sutskever, Quoc V. Le, Oriol Vinyals, and Wojciech Zaremba. 2015. Addressing the rare word problem in neural machine translation. In *ACL*.

Volodymyr Mnih, Nicolas Heess, Alex Graves, and Koray Kavukcuoglu. 2014. Recurrent models of visual attention. In *NIPS*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *ACL*.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C. Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2015. Recurrent neural network regularization. In *ICLR*.

Document Modeling with Gated Recurrent Neural Network for Sentiment Classification

Duyu Tang, Bing Qin*, Ting Liu
Harbin Institute of Technology, Harbin, China
{dytang, qinb, tliu}@ir.hit.edu.cn

Abstract

Document level sentiment classification remains a challenge: encoding the intrinsic relations between sentences in the semantic meaning of a document. To address this, we introduce a neural network model to learn vector-based document representation in a unified, bottom-up fashion. The model first learns sentence representation with convolutional neural network or long short-term memory. Afterwards, semantics of sentences and their relations are adaptively encoded in document representation with gated recurrent neural network. We conduct document level sentiment classification on four large-scale review datasets from IMDB and Yelp Dataset Challenge. Experimental results show that: (1) our neural model shows superior performances over several state-of-the-art algorithms; (2) gated recurrent neural network dramatically outperforms standard recurrent neural network in document modeling for sentiment classification.[1]

1 Introduction

Document level sentiment classification is a fundamental task in sentiment analysis, and is crucial to understand user generated content in social networks or product reviews (Manning and Schütze, 1999; Jurafsky and Martin, 2000; Pang and Lee, 2008; Liu, 2012). The task calls for identifying the overall sentiment polarity (e.g. *thumbs up* or *thumbs down*, 1-5 stars on review sites) of a document. In literature, dominant approaches follow (Pang et al., 2002) and exploit machine learning algorithm to build sentiment classifier. Many of them focus on designing hand-crafted features (Qu et al., 2010; Paltoglou and Thelwall, 2010) or learning discriminate features from data, since the performance of a machine learner is heavily dependent on the choice of data representation (Bengio et al., 2015).

Document level sentiment classification remains a significant challenge: how to encode the intrinsic (semantic or syntactic) relations between sentences in the semantic meaning of document. This is crucial for sentiment classification because relations like "contrast" and "cause" have great influences on determining the meaning and the overall polarity of a document. However, existing studies typically fail to effectively capture such information. For example, Pang et al. (2002) and Wang and Manning (2012) represent documents with bag-of-ngrams features and build SVM classifier upon that. Although such feature-driven SVM is an extremely strong performer and hardly to be transcended, its "sparse" and "discrete" characteristics make it clumsy in taking into account of side information like relations between sentences. Recently, Le and Mikolov (2014) exploit neural networks to learn continuous document representation from data. Essentially, they use local ngram information and do not capture semantic relations between sentences. Furthermore, a person asked to do this task will naturally carry it out in a sequential, bottom-up fashion, analyze the meanings of sentences before considering semantic relations between them. This motivates us to develop an end-to-end and bottom-up algorithm to effectively model document representation.

In this paper, we introduce a neural network approach to learn continuous document representation for sentiment classification. The method is on the basis of the principle of compositionality (Frege, 1892), which states that the meaning of a longer expression (e.g. a sentence or a docu-

*Corresponding author.

[1] Codes and datasets are publicly available at http://ir.hit.edu.cn/~dytang.

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1422–1432,
Lisbon, Portugal, 17-21 September 2015.

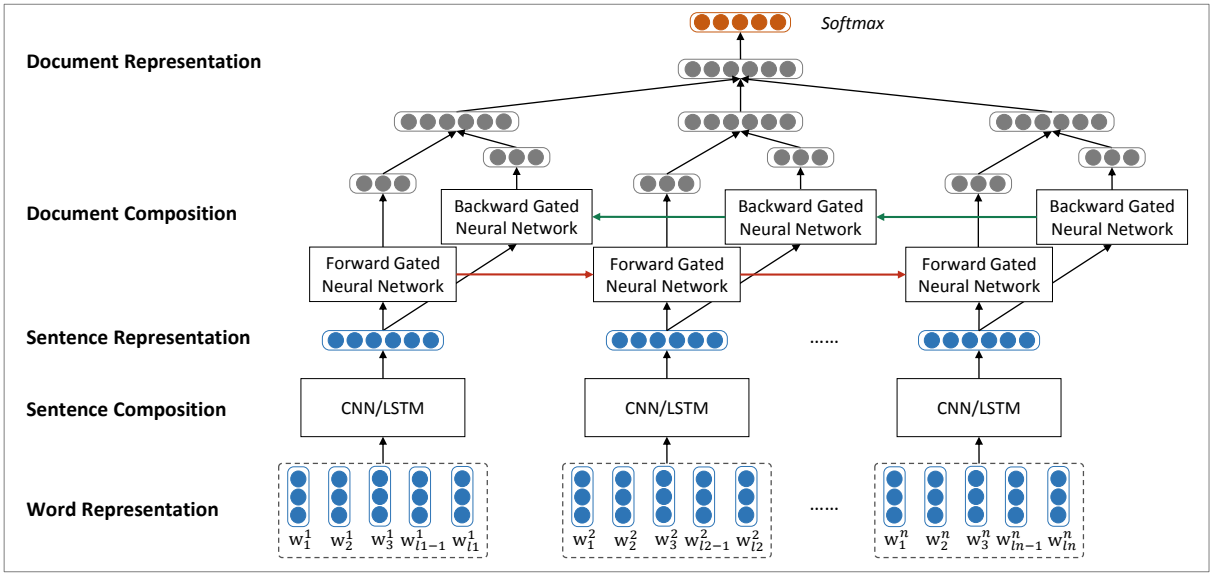


Figure 1: The neural network model for document level sentiment classification. w_i^n stands for the i-th word in the n-th sentence, l_n is sentence length.

ment) depends on the meanings of its constituents. Specifically, the approach models document representation in two steps. In the first step, it uses convolutional neural network (CNN) or long short-term memory (LSTM) to produce sentence representations from word representations. Afterwards, gated recurrent neural network is exploited to adaptively encode semantics of sentences and their inherent relations in document representations. These representations are naturally used as features to classify the sentiment label of each document. The entire model is trained end-to-end with stochastic gradient descent, where the loss function is the cross-entropy error of supervised sentiment classification[2].

We conduct document level sentiment classification on four large-scale review datasets from IMDB[3] and Yelp Dataset Challenge[4]. We compare to neural network models such as paragraph vector (Le and Mikolov, 2014), convolutional neural network, and baselines such as feature-based SVM (Pang et al., 2002), recommendation algorithm JMARS (Diao et al., 2014). Experimental results show that: (1) the proposed neural model shows superior performances over all baseline algorithms; (2) gated recurrent neural network dramatically outperforms standard recurrent neural network in document modeling. The main contributions of this work are as follows:

- We present a neural network approach to encode relations between sentences in document representation for sentiment classification.
- We report empirical results on four large-scale datasets, and show that the approach outperforms state-of-the-art methods for document level sentiment classification.
- We report empirical results that traditional recurrent neural network is weak in modeling document composition, while adding neural gates dramatically improves the classification performance.

2 The Approach

We introduce the proposed neural model in this section, which computes continuous vector representations for documents of variable length. These representations are further used as features to classify the sentiment label of each document. An overview of the approach is displayed in Figure 1.

Our approach models document semantics based on the principle of compositionality (Frege, 1892), which states that the meaning of a longer expression (e.g. a sentence or a document) comes from the meanings of its constituents and the rules used to combine them. Since a document consists of a list of sentences and each sentence is made up of a list of words, the approach models document representation in two stages. It first produces continuous sentence vectors from word represen-

[2]A similar work can be found at: http://deeplearning.net/tutorial/lstm.html

[3]http://www.imdb.com/

[4]http://www.yelp.com/dataset_challenge

tations with sentence composition (Section 2.1). Afterwards, sentence vectors are treated as inputs of document composition to get document representation (Section 2.2). Document representations are then used as features for document level sentiment classification (Section 2.3).

2.1 Sentence Composition

We first describe word vector representation, before presenting a convolutional neural network with multiple filters for sentence composition.

Each word is represented as a low dimensional, continuous and real-valued vector, also known as word embedding (Bengio et al., 2003). All the word vectors are stacked in a word embedding matrix $L_w \in \mathbb{R}^{d \times |V|}$, where d is the dimension of word vector and $|V|$ is vocabulary size. These word vectors can be randomly initialized from a uniform distribution (Socher et al., 2013b), or be pre-trained from text corpus with embedding learning algorithms (Mikolov et al., 2013; Pennington et al., 2014; Tang et al., 2014). We adopt the latter strategy to make better use of semantic and grammatical associations of words.

We use convolutional neural network (CNN) and long short-term memory (LSTM) to compute continuous representations of sentences with semantic composition. CNN and LSTM are state-of-the-art semantic composition models for sentiment classification (Kim, 2014; Kalchbrenner et al., 2014; Johnson and Zhang, 2015; Li et al., 2015a). They learn fixed-length vectors for sentences of varying length, captures words order in a sentence and does not depend on external dependency or constituency parse results. One could also use tree-based composition method such as Recursive Neural Tensor Network (Socher et al., 2013b) or Tree-Structured LSTM (Tai et al., 2015; Zhu et al., 2015) as alternatives.

Specifically, we try CNN with multiple convolutional filters of different widths (Tang et al., 2015) to produce sentence representation. Figure 2 displays the method. We use multiple convolutional filters in order to capture local semantics of n-grams of various granularities, which have been proven effective for sentiment classification. For example, a convolutional filter with a width of 2 essentially captures the semantics of bigrams in a sentence. In this work, we use three convolutional filters whose widths are 1, 2 and 3 to encode the semantics of unigrams, bigrams and trigrams in a sentence. Each filter consists of a list of linear layers with shared parameters. Formally, let us denote a sentence consisting of n words as $\{w_1, w_2, ...w_i, ...w_n\}$, let l_c be the width of a convolutional filter, and let W_c, b_c be the shared parameters of linear layers in the filter. Each word w_i is mapped to its embedding representation $e_i \in \mathbb{R}^d$. The input of a linear layer is the concatenation of word embeddings in a fixed-length window size l_c, which is denoted as $I_c = [e_i; e_{i+1}; ...; e_{i+l_c-1}] \in \mathbb{R}^{d \cdot l_c}$. The output of a linear layer is calculated as

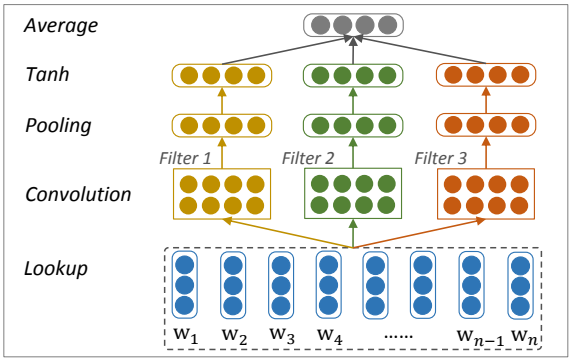


Figure 2: Sentence composition with convolutional neural network.

$$O_c = W_c \cdot I_c + b_c \tag{1}$$

where $W_c \in \mathbb{R}^{l_{oc} \times d \cdot l_c}$, $b_c \in \mathbb{R}^{l_{oc}}$, l_{oc} is the output length of linear layer. To capture global semantics of a sentence, we feed the outputs of linear layers to an *average* pooling layer, resulting in an output vector with fixed-length. We further add hyperbolic tangent ($tanh$) to incorporate pointwise nonlinearity, and *average* the outputs of multiple filters to get sentence representation.

We also try lstm as the sentence level semantic calculator, the performance comparison between these two variations is given in Section 3.

2.2 Document Composition with Gated Recurrent Neural Network

The obtained sentence vectors are fed to a document composition component to calculate the document representation. We present a gated recurrent neural network approach for document composition in this part.

Given the vectors of sentences of variable length as input, document composition produces a fixed-length document vector as output. To this end, a simple strategy is ignoring the order of sen-

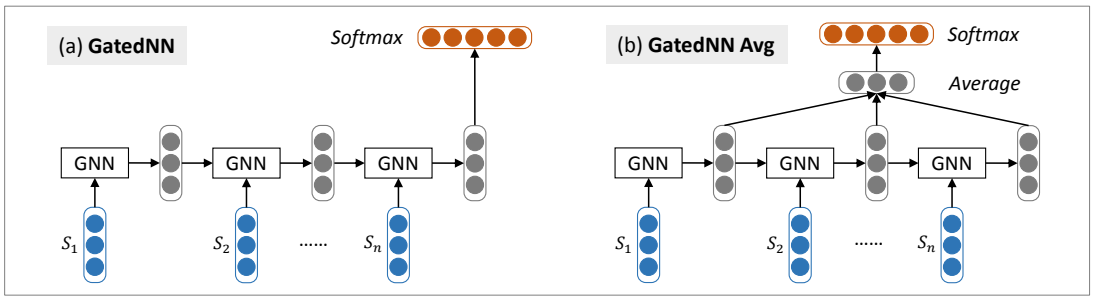


Figure 3: Document modeling with gated recurrent neural network. GNN stands for the basic computational unit of gated recurrent neural network.

tences and averaging sentence vectors as document vector. Despite its computational efficiency, it fails to capture complex linguistic relations (e.g. "cause" and "contrast") between sentences. Convolutional neural network (Denil et al., 2014) is an alternative for document composition, which models local sentence relations with shared parameters of linear layers.

Standard recurrent neural network (RNN) can map vectors of sentences of variable length to a fixed-length vector by recursively transforming current sentence vector s_t with the output vector of the previous step h_{t-1}. The transition function is typically a linear layer followed by pointwise non-linearity layer such as $tanh$.

$$h_t = tanh(W_r \cdot [h_{t-1}; s_t] + b_r) \quad (2)$$

where $W_r \in \mathbb{R}^{l_h \times (l_h + l_{oc})}$, $b_r \in \mathbb{R}^{l_h}$, l_h and l_{oc} are dimensions of hidden vector and sentence vector, respectively. Unfortunately, standard RNN suffers the problem of gradient vanishing or exploding (Bengio et al., 1994; Hochreiter and Schmidhuber, 1997), where gradients may grow or decay exponentially over long sequences. This makes it difficult to model long-distance correlations in a sequence. To address this problem, we develop a gated recurrent neural network for document composition, which works in a sequential way and adaptively encodes sentence semantics in document representations. The approach is analogous to the recently emerged LSTM (Graves et al., 2013; Zaremba and Sutskever, 2014; Sutskever et al., 2014; Xu et al., 2015) and gated neural network (Cho et al., 2014; Chung et al., 2015). Specifically, the transition function of the gated RNN used in this work is calculated as follows.

$$i_t = sigmoid(W_i \cdot [h_{t-1}; s_t] + b_i) \quad (3)$$

$$f_t = sigmoid(W_f \cdot [h_{t-1}; s_t] + b_f) \quad (4)$$

$$g_t = tanh(W_r \cdot [h_{t-1}; s_t] + b_r) \quad (5)$$

$$h_t = tanh(i_t \odot g_t + f_t \odot h_{t-1}) \quad (6)$$

where $\odot$ stands for element-wise multiplication, W_i, W_f, b_i, b_f adaptively select and remove history vector and input vector for semantic composition. The model can be viewed as a LSTM whose output gate is alway on, since we prefer not to discarding any part of the semantics of sentences to get a better document representation. Figure 3 (a) displays a standard sequential way where the last hidden vector is regarded as the document representation for sentiment classification. We can make further extensions such as averaging hidden vectors as document representation, which takes considerations of a hierarchy of historical semantics with different granularities. The method is illustrated in Figure 3 (b), which shares some characteristics with (Zhao et al., 2015). We can go one step further to use preceding histories and following evidences in the same way, and exploit bidirectional (Graves et al., 2013) gated RNN as the calculator. The model is embedded in Figure 1.

2.3 Sentiment Classification

The composed document representations can be naturally regarded as features of documents for sentiment classification without feature engineering. Specifically, we first add a linear layer to transform document vector to real-valued vector whose length is class number C. Afterwards, we add a $softmax$ layer to convert real values to conditional probabilities, which is calculated as follows.

$$P_i = \frac{exp(x_i)}{\sum_{i'=1}^{C} exp(x_{i'})} \quad (7)$$

We conduct experiments in a supervised learning setting, where each document in the training data is accompanied with its gold sentiment label.

Corpus	#docs	#s/d	#w/d	$\|V\|$	#class	Class Distribution
Yelp 2013	335,018	8.90	151.6	211,245	5	.09/.09/.14/.33/.36
Yelp 2014	1,125,457	9.22	156.9	476,191	5	.10/.09/.15/.30/.36
Yelp 2015	1,569,264	8.97	151.9	612,636	5	.10/.09/.14/.30/.37
IMDB	348,415	14.02	325.6	115,831	10	.07/.04/.05/.05/.08/.11/.15/.17/.12/.18

Table 1: Statistical information of Yelp 2013/2014/2015 and IMDB datasets. #docs is the number of documents, #s/d and #w/d represent average number of sentences and average number of words contained in per document, $|V|$ is the vocabulary size of words, #class is the number of classes.

For model training, we use the cross-entropy error between gold sentiment distribution $P^g(d)$ and predicted sentiment distribution $P(d)$ as the loss function.

$$loss = -\sum_{d \in T} \sum_{i=1}^{C} P_i^g(d) \cdot log(P_i(d)) \quad (8)$$

where T is the training data, C is the number of classes, d represents a document. $P^g(d)$ has a *1-of-K* coding scheme, which has the same dimension as the number of classes, and only the dimension corresponding to the ground truth is 1, with all others being 0. We take the derivative of loss function through back-propagation with respect to the whole set of parameters $\theta = [W_c; b_c; W_i; b_i; W_f; b_f; W_r; b_r; W_{softmax}, b_{softmax}]$, and update parameters with stochastic gradient descent. We set the widths of three convolutional filters as 1, 2 and 3, output length of convolutional filter as 50. We learn 200-dimensional word embeddings with SkipGram (Mikolov et al., 2013) on each dataset separately, randomly initialize other parameters from a uniform distribution $U(-0.01, 0.01)$, and set learning rate as 0.03.

3 Experiment

We conduct experiments to empirically evaluate our method by applying it to document level sentiment classification. We describe experimental settings and report empirical results in this section.

3.1 Experimental Setting

We conduct experiments on large-scale datasets consisting of document reviews. Specifically, we use one movie review dataset from IMDB (Diao et al., 2014) and three restaurant review datasets from Yelp Dataset Challenge in 2013, 2014 and 2015. Human labeled review ratings are regarded as gold standard sentiment labels, so that we do not need to manually annotate sentiment labels of documents. We do not consider the cases that rating does not match with review texts (Zhang et al., 2014).

Statistical information of these datasets are given in Table 1. We use the same dataset split as in (Diao et al., 2014) on IMDB dataset, and split Yelp datasets into training, development and testing sets with 80/10/10. We run tokenization and sentence splitting with Stanford CoreNLP (Manning et al., 2014) on all these datasets. We use *accuracy* (Manning and Schütze, 1999; Jurafsky and Martin, 2000) and MSE (Diao et al., 2014) as evaluation metrics, where *accuracy* is a standard metric to measure the overall sentiment classification performance. We use *MSE* to measure the divergences between predicted sentiment labels and ground truth sentiment labels because review labels reflect sentiment strengths (e.g. one star means strong negative and five star means strong positive).

$$MSE = \frac{\sum_i^N (gold_i - predicted_i)^2}{N} \quad (9)$$

3.2 Baseline Methods

We compare our methods (Conv-GRNN and LSTM-GRNN) with the following baseline methods for document level sentiment classification.

(1) *Majority* is a heuristic baseline, which assigns the majority sentiment label in training set to each document in test set.

(2) In *SVM+Ngrams*, we use bag-of-unigrams and bag-of-bigrams as features and train SVM classifier with LibLinear (Fan et al., 2008)[5].

(3) In *TextFeatures*, we implement sophisticated features (Kiritchenko et al., 2014) including word ngrams, character ngrams, sentiment lexicon features, cluster features, et al.

[5]We also try discretized regression (Pang and Lee, 2005) with fixed decision thresholds (e.g. 0.5, 1.5, 2.5, ...). However, its performance is obviously worse than SVM classifier.

	Yelp 2013		Yelp 2014		Yelp 2015		IMDB	
	Accuracy	MSE	Accuracy	MSE	Accuracy	MSE	Accuracy	MSE
Majority	0.356	3.06	0.361	3.28	0.369	3.30	0.179	17.46
SVM + Unigrams	0.589	0.79	0.600	0.78	0.611	0.75	0.399	4.23
SVM + Bigrams	0.576	0.75	0.616	0.65	0.624	0.63	0.409	3.74
SVM + TextFeatures	0.598	0.68	0.618	0.63	0.624	0.60	0.405	3.56
SVM + AverageSG	0.543	1.11	0.557	1.08	0.568	1.04	0.319	5.57
SVM + SSWE	0.535	1.12	0.543	1.13	0.554	1.11	0.262	9.16
JMARS	N/A	–	N/A	–	N/A	–	N/A	4.97
Paragraph Vector	0.577	0.86	0.592	0.70	0.605	0.61	0.341	4.69
Convolutional NN	0.597	0.76	0.610	0.68	0.615	0.68	0.376	3.30
Conv-GRNN	0.637	0.56	0.655	0.51	0.660	0.50	0.425	**2.71**
LSTM-GRNN	**0.651**	**0.50**	**0.671**	**0.48**	**0.676**	**0.49**	**0.453**	3.00

Table 2: Sentiment classification on Yelp 2013/2014/2015 and IMDB datasets. Evaluation metrics are accuracy (higher is better) and MSE (lower is better). The best method in each setting is in **bold**.

(4) In *AverageSG*, we learn 200-dimensional word vectors with *word2vec*[6] (Mikolov et al., 2013), average word embeddings to get document representation, and train a SVM classifier.

(5) We learn sentiment-specific word embeddings (SSWE), and use max/min/average pooling (Tang et al., 2014) to get document representation.

(6) We compare with a state-of-the-art recommendation algorithm JMARS (Diao et al., 2014), which utilizes user and aspects of a review with collaborative filtering and topic modeling.

(7) We implement a convolutional neural network (CNN) baseline as it is a state-of-the-art semantic composition method for sentiment analysis (Kim, 2014; Denil et al., 2014).

(8) We implement a state-of-the-art neural network baseline Paragraph Vector (Le and Mikolov, 2014) because its codes are not officially provided. Window size is tuned on the development set.

3.3 Comparison to Other Methods

Experimental results are given in Table 2. We evaluate each dataset with two metrics, namely accuracy (higher is better) and MSE (lower is better). The best method in each dataset and each evaluation metric is in bold.

From Table 2, we can see that majority is the worst method because it does not capture any textual semantics. SVM classifiers with unigram and bigram features (Pang et al., 2002) are extremely strong, which are almost the strongest performers among all baseline methods. Designing complex features are also effective for document level sentiment classification, however, it does not surpass the bag-of-ngram features significantly as on Twitter corpora (Kiritchenko et al., 2014). Furthermore, the aforementioned bag-of-features are discrete and sparse. For example, the feature dimension of bigrams and TextFeatures on Yelp 2015 dataset are 899K and 4.81M after we filter out low frequent features. Based on them, we try to concatenate several discourse-driven features, but the classification performances remain unchanged.

AverageSG is a straight forward way to compose document representation without feature engineering. Unfortunately, we can see that it does not work in this scenario, which appeals for powerful semantic composition models for document level sentiment classification. We try to make better use of the sentiment information to learn a better SSWE (Tang et al., 2014), e.g. setting a large window size. However, its performance is still worse than context-based word embedding. This stems from the fact that there are many sentiment shifters (e.g. negation or contrast words) in document level reviews, while Tang et al. (2014) learn SSWE by assigning sentiment label of a text to each phrase it contains. How to learn SSWE effectively with document level sentiment supervision remains as an interesting future work.

Since JMARS outputs real-valued outputs, we only evaluate it in terms of MSE. We can see that sophisticated baseline methods such as JMARS, paragraph vector and convolutional NN obtain significant performance boosts over AverageSG by

[6]We use Skipgram as it performs slightly better than CBOW in the experiment. We also try off-the-shell word embeddings from Glove, but its performance is slightly worse than tailored word embedding from each corpus.

	Yelp 2013		Yelp 2014		Yelp 2015		IMDB	
	Accuracy	MSE	Accuracy	MSE	Accuracy	MSE	Accuracy	MSE
Average	0.598	0.65	0.605	0.75	0.614	0.67	0.366	3.91
Recurrent	0.377	1.37	0.306	1.75	0.383	1.67	0.176	12.29
Recurrent Avg	0.582	0.69	0.591	0.70	0.597	0.74	0.344	3.71
Bi Recurrent Avg	0.587	0.73	0.597	0.73	0.577	0.82	0.372	3.32
GatedNN	0.636	0.58	0.656	0.52	0.651	0.51	**0.430**	2.95
GatedNN Avg	0.635	0.57	**0.659**	0.52	0.657	0.56	0.416	2.78
Bi GatedNN Avg	**0.637**	**0.56**	0.655	**0.51**	**0.660**	**0.50**	0.425	**2.71**

Table 3: Sentiment classification on IMDB, Yelp 2013/2014/2015 datasets. Evaluation metrics are accuracy (higher is better) and MSE (lower is better). The best method in each setting is in **bold**.

capturing deeper semantics of texts. Comparing between CNN and AverageSG, we can conclude that deep semantic compositionality is crucial for understanding the semantics and the sentiment of documents. However, it is somewhat disappointing that these models do not significantly outperform discrete bag-of-ngrams and bag-of-features. The reason might lie in that semantic meanings of documents, e.g. relations between sentences, are not well captured. We can see that the proposed method Conv-GRNN and LSTM-GRNN yield the best performance on all four datasets in two evaluation metrics. Compared with CNN, Conv-GRNN shows its superior power in document composition component, which encodes semantics of sentences and their relations in document representation with gated recurrent neural network. We also find that LSTM (almost) consistently performs better than CNN in modeling the sentence representation.

3.4 Model Analysis

As discussed before, document composition contributes a lot to the superior performance of Conv-GRNN and LSTM-GRNN. Therefore, we take Conv-GRNN as an example and compare different neural models for document composition in this part. Specifically, after obtaining sentence vectors with convolutional neural network as described in Section 2.1, we carry out experiments in following settings.

(1) *Average*. Sentence vectors are averaged to get the document vector.

(2) *Recurrent* / *GatedNN*. Sentence vectors are fed to standard (or gated) recurrent neural network in a sequential way from the beginning of the input document. The last hidden vector is regarded as document representation.

(3) *Recurrent Avg* / *GatedNN Avg*. We extend setting (2) by averaging hidden vectors of recurrent neural network as document vector.

(4) *Bi Recurrent Avg* / *Bi GatedNN Avg*. We extend setting (3) by calculating hidden vectors from both preceding histories and following contexts. Bi-directional hidden vectors are averaged as document representation.

Table 3 shows the experimental results. We can see that standard recurrent neural network (RNN) is the worst method, even worse than the simple vector average. This is because RNN suffers from the vanishing gradient problem, stating that the influence of a given input on the hidden layer decays exponentially over time on the network output. In this paper, it means that document representation encodes rare semantics of the beginning sentences. This is further justified by the great improvement of *Recurrent Avg* over *Recurrent*. *Bi Recurrent Avg* and *Recurrent Avg* perform comparably, but disappointingly both of them fail to transcend *Average*. After adding neural gates, *GatedNN* obtains dramatic accuracy improvements over *Recurrent* and significantly outperforms previous settings. The results indicate that *Gated RNN* is capable of handling the vanishing gradient problem to some extend, and it is practical to adaptively model sentence semantics in document representation. *GatedNN Avg* and *Bi GatedNN Avg* obtains comparable performances with *GatedNN*.

4 Related Work

Document level sentiment classification is a fundamental problem in sentiment analysis (Pang and Lee, 2008; Liu, 2012), which aims at identifying the sentiment label of a document (Pang et al., 2002; Turney, 2002). Pang and Lee (2002; 2005)

cast this problem as a classification task, and use machine learning method in a supervised learning framework. Turney (2002) introduces an unsupervised approach by using sentiment words/phrases extracted from syntactic patterns to determine the document polarity. Goldberg and Zhu (2006) place this task in a semi-supervised setting, and use unlabelled reviews with graph-based method. Dominant studies in literature follow Pang et al. (2002) and work on designing effective features for building a powerful sentiment classifier. Representative features include word ngrams (Wang and Manning, 2012), text topic (Ganu et al., 2009), bag-of-opinions (Qu et al., 2010), syntactic relations (Xia and Zong, 2010), sentiment lexicon features (Kiritchenko et al., 2014).

Despite the effectiveness of feature engineering, it is labor intensive and unable to extract and organize the discriminative information from data (Bengio et al., 2015). Recently, neural network emerges as an effective way to learn continuous text representation for sentiment classification. Existing studies in this direction can be divided into two groups. One line of research focuses on learning continuous word embedding. Traditional embedding learning algorithms typically leverage contexts of words in a context-prediction way (Bengio et al., 2003; Mikolov et al., 2013; Baroni et al., 2014). Since these methods typically map words with similar contexts but opposite polarity (e.g. "good" and "bad") to neighboring vectors, several studies (Maas et al., 2011; Labutov and Lipson, 2013; Tang et al., 2014) learn sentiment-specific word embeddings by taking sentiment of texts into account. Another line of research concentrates on semantic composition (Mitchell and Lapata, 2010). Yessenalina and Cardie (2011) represent each word as a matrix and use iterated matrix multiplication as phrase-level composition function. Socher et al. (2013b) introduce a family of recursive neural networks for sentence-level semantic composition. Recursive neural network is extended with global feedbackward (Paulus et al., 2014), feature weight tuning (Li, 2014), deep recursive layer (Irsoy and Cardie, 2014), adaptive composition functions (Dong et al., 2014), combined with Combinatory Categorial Grammar (Hermann and Blunsom, 2013), and used for opinion relation detection (Xu et al., 2014). Glorot et al. (2011) use stacked denoising autoencoder. Convolutional neural networks are widely used for semantic composition (Kim, 2014; Kalchbrenner et al., 2014; Denil et al., 2014; Johnson and Zhang, 2015) by automatically capturing local and global semantics. Le and Mikolov (2014) introduce Paragraph Vector to learn document representation from semantics of words. Sequential model like recurrent neural network or long short-term memory (LSTM) are also verified as strong approaches for semantic composition (Li et al., 2015a).

In this work, we represent document with convolutional-gated recurrent neural network, which adaptively encodes semantics of sentences and their relations. A recent work in (Li et al., 2015b) also investigate LSTM to model document meaning. They verify the effectiveness of LSTM in text generation task.

5 Conclusion

We introduce neural network models (Conv-GRNN and LSTM-GRNN) for document level sentiment classification. The approach encodes semantics of sentences and their relations in document representation, and is effectively trained end-to-end with supervised sentiment classification objectives. We conduct extensive experiments on four review datasets with two evaluation metrics. Empirical results show that our approaches achieve state-of-the-art performances on all these datasets. We also find that (1) traditional recurrent neural network is extremely weak in modeling document composition, while adding neural gates dramatically boosts the performance, (2) LSTM performs better than a multi-filtered CNN in modeling sentence representation.

We briefly discuss some future plans. How to effectively compose sentence meanings to document meaning is a central problem in natural language processing. In this work, we develop neural models in a sequential way, and encode sentence semantics and their relations automatically without using external discourse analysis results. From one perspective, one could carefully define a set of sentiment-sensitive discourse relations (Zhou et al., 2011), such as "contrast", "condition", "cause", etc. Afterwards, relation-specific gated RNN can be developed to explicitly model semantic composition rules for each relation (Socher et al., 2013a). However, defining such a relation scheme is linguistic driven and time consuming, which we leave as future work. From another perspective, one could compose document

representation over discourse tree structures rather than in a sequential way. Accordingly, Recursive Neural Network (Socher et al., 2013b) and Structured LSTM (Tai et al., 2015; Zhu et al., 2015) can be used as composition algorithms. However, existing discourse structure learning algorithms are difficult to scale to massive review texts on the web. How to simultaneously learn document structure and composition function is an interesting future work.

Acknowledgments

The authors give great thanks to Yaming Sun and Jiwei Li for the fruitful discussions. We also would like to thank three anonymous reviewers for their valuable comments and suggestions. This work was supported by the National High Technology Development 863 Program of China (No. 2015AA015407), National Natural Science Foundation of China (No. 61133012 and No. 61273321). Duyu Tang is supported by Baidu Fellowship and IBM Ph.D. Fellowship.

References

Marco Baroni, Georgiana Dinu, and Germán Kruszewski. 2014. Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *ACL*, pages 238–247.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *Neural Networks, IEEE Transactions on*, 5(2):157–166.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155.

Yoshua Bengio, Ian J. Goodfellow, and Aaron Courville. 2015. Deep learning. Book in preparation for MIT Press.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *EMNLP*, pages 1724–1734.

Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2015. Gated feedback recurrent neural networks. *ICML*.

Misha Denil, Alban Demiraj, Nal Kalchbrenner, Phil Blunsom, and Nando de Freitas. 2014. Modelling, visualising and summarising documents with a single convolutional neural network. *arXiv preprint:1406.3830*.

Qiming Diao, Minghui Qiu, Chao-Yuan Wu, Alexander J Smola, Jing Jiang, and Chong Wang. 2014. Jointly modeling aspects, ratings and sentiments for movie recommendation (jmars). In *SIGKDD*, pages 193–202. ACM.

Li Dong, Furu Wei, Ming Zhou, and Ke Xu. 2014. Adaptive multi-compositionality for recursive neural models with applications to sentiment analysis. In *AAAI*, pages 1537–1543.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. Liblinear: A library for large linear classification. *JMLR*.

Gottlob Frege. 1892. On sense and reference. *Ludlow (1997)*, pages 563–584.

Gayatree Ganu, Noemie Elhadad, and Amélie Marian. 2009. Beyond the stars: Improving rating predictions using review text content. In *WebDB*.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *ICML*, pages 513–520.

Andrew B Goldberg and Xiaojin Zhu. 2006. Seeing stars when there aren't many stars: graph-based semi-supervised learning for sentiment categorization. In *GraphBased Method for NLP*, pages 45–52.

Alex Graves, Navdeep Jaitly, and A-R Mohamed. 2013. Hybrid speech recognition with deep bidirectional lstm. In *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*, pages 273–278. IEEE.

Karl Moritz Hermann and Phil Blunsom. 2013. The role of syntax in vector space models of compositional semantics. In *ACL*, pages 894–904.

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

Ozan Irsoy and Claire Cardie. 2014. Deep recursive neural networks for compositionality in language. In *NIPS*, pages 2096–2104.

Rie Johnson and Tong Zhang. 2015. Effective use of word order for text categorization with convolutional neural networks. *NAACL*.

Dan Jurafsky and James H Martin. 2000. *Speech & language processing*. Pearson Education India.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *ACL*, pages 655–665.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *EMNLP*, pages 1746–1751.

Svetlana Kiritchenko, Xiaodan Zhu, and Saif M Mohammad. 2014. Sentiment analysis of short informal texts. *Journal of Artificial Intelligence Research*, pages 723–762.

Igor Labutov and Hod Lipson. 2013. Re-embedding words. In *Annual Meeting of the Association for Computational Linguistics*.

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

Jiwei Li, Dan Jurafsky, and Eudard Hovy. 2015a. When are tree structures necessary for deep learning of representations? *arXiv preprint arXiv:1503.00185*.

Jiwei Li, Minh-Thang Luong, and Dan Jurafsky. 2015b. A hierarchical neural autoencoder for paragraphs and documents. *arXiv preprint arXiv:1506.01057*.

Jiwei Li. 2014. Feature weight tuning for recursive neural networks. *Arxiv preprint*, 1412.3714.

Bing Liu. 2012. Sentiment analysis and opinion mining. *Synthesis Lectures on Human Language Technologies*, 5(1):1–167.

Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *ACL*, pages 142–150.

Christopher D Manning and Hinrich Schütze. 1999. *Foundations of statistical natural language processing*. MIT press.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *ACL*, pages 55–60.

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

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

Georgios Paltoglou and Mike Thelwall. 2010. A study of information retrieval weighting schemes for sentiment analysis. In *Proceedings of Annual Meeting of the Association for Computational Linguistics*, pages 1386–1395.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *ACL*, pages 115–124.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Foundations and trends in information retrieval*, 2(1-2):1–135.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: sentiment classification using machine learning techniques. In *EMNLP*, pages 79–86.

Romain Paulus, Richard Socher, and Christopher D Manning. 2014. Global belief recursive neural networks. In *NIPS*, pages 2888–2896.

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

Lizhen Qu, Georgiana Ifrim, and Gerhard Weikum. 2010. The bag-of-opinions method for review rating prediction from sparse text patterns. In *COLING*, pages 913–921.

Richard Socher, John Bauer, Christopher D. Manning, and Andrew Y. Ng. 2013a. Parsing with compositional vector grammars. In *ACL*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013b. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP*, pages 1631–1642.

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

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *ACL*.

Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. 2014. Learning sentiment-specific word embedding for twitter sentiment classification. In *ACL*, pages 1555–1565.

Duyu Tang, Bing Qin, and Ting Liu. 2015. Learning semantic representations of users and products for document level sentiment classification. In *ACL*, pages 1014–1023.

Peter D Turney. 2002. Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews. In *ACL*, pages 417–424.

Sida Wang and Christopher D Manning. 2012. Baselines and bigrams: Simple, good sentiment and topic classification. In *ACL*, pages 90–94.

Rui Xia and Chengqing Zong. 2010. Exploring the use of word relation features for sentiment classification. In *COLING*, pages 1336–1344.

Liheng Xu, Kang Liu, and Jun Zhao. 2014. Joint opinion relation detection using one-class deep neural network. In *COLING*, pages 677–687.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. *ICML*.

Ainur Yessenalina and Claire Cardie. 2011. Compositional matrix-space models for sentiment analysis. In *EMNLP*, pages 172–182.

Wojciech Zaremba and Ilya Sutskever. 2014. Learning to execute. *arXiv preprint arXiv:1410.4615*.

Yongfeng Zhang, Haochen Zhang, Min Zhang, Yiqun Liu, and Shaoping Ma. 2014. Do users rate or review?: boost phrase-level sentiment labeling with review-level sentiment classification. In *SIGIR*, pages 1027–1030. ACM.

Han Zhao, Zhengdong Lu, and Pascal Poupart. 2015. Self-adaptive hierarchical sentence model. In *IJCAI*.

Lanjun Zhou, Binyang Li, Wei Gao, Zhongyu Wei, and Kam-Fai Wong. 2011. Unsupervised discovery of discourse relations for eliminating intra-sentence polarity ambiguities. In *EMNLP*, pages 162–171, .

Xiaodan Zhu, Parinaz Sobhani, and Hongyu Guo. 2015. Long short-term memory over tree structures. *ICML*.

Fine-grained Opinion Mining with Recurrent Neural Networks and Word Embeddings

Pengfei Liu[1], **Shafiq Joty**[2] **and Helen Meng**[1]
[1]Department of Systems Engineering and Engineering Management,
The Chinese University of Hong Kong, Hong Kong SAR, China
[2]Qatar Computing Research Institute - HBKU, Doha, Qatar
{pfliu, hmmeng}@se.cuhk.edu.hk, sjoty@qf.org.qa

Abstract

The tasks in fine-grained opinion mining can be regarded as either a token-level sequence labeling problem or as a semantic compositional task. We propose a general class of discriminative models based on recurrent neural networks (RNNs) and word embeddings that can be successfully applied to such tasks without any task-specific feature engineering effort. Our experimental results on the task of opinion target identification show that RNNs, without using any hand-crafted features, outperform feature-rich CRF-based models. Our framework is flexible, allows us to incorporate other linguistic features, and achieves results that rival the top performing systems in SemEval-2014.

1 Introduction

Fine-grained opinion mining involves identifying the opinion holder who expresses the opinion, detecting opinion expressions, measuring their intensity and sentiment, and identifying the target or aspect of the opinion (Wiebe et al., 2005). For example, in the sentence "John says, the hard disk is very noisy", John, the opinion holder, expresses a very negative (i.e., sentiment with intensity) opinion towards the target "hard disk" using the opinionated expression "very noisy". A number of NLP applications can benefit from fine-grained opinion mining including opinion summarization and opinion-oriented question answering.

The tasks in fine-grained opinion mining can be regarded as either a token-level sequence labeling problem or as a semantic compositional task at the sequence (e.g., phrase) level. For example, identifying opinion holders, opinion expressions and opinion targets can be formulated as a token-level sequence tagging problem, where the task is to label each word in a sentence using the conventional BIO tagging scheme. For example, Table 1 shows a sentence tagged with BIO scheme for opinion target (middle row) and for opinion expression (bottom row) identification tasks. On the other hand, characterizing intensity and sentiment of an opinionated expression can be regarded as a semantic compositional problem, where the task is to aggregate vector representations of tokens in a meaningful way and later use them for sentiment classification (Socher et al., 2013).

The	**hard**	**disk**	is	*very*	*noisy*
O	B-TARG	I-TARG	O	O	O
O	O	O	O	B-EXPR	I-EXPR

Table 1: An example sentence annotated with BIO labels for opinion target (`TARG` tags) and for opinion expression (`EXPR` tags) extraction.

Conditional random fields (CRFs) (Lafferty et al., 2001) have been quite successful for different fine-grained opinion mining tasks, e.g., opinion expression extraction (Yang and Cardie, 2012). The state-of-the-art model for opinion target extraction is also based on a CRF (Pontiki et al., 2014). However, the success of CRFs depends heavily on the use of an appropriate feature set and feature function expansion, which often requires a lot of engineering effort for each task in hand.

An alternative approach of deep learning automatically learns latent features as distributed vectors and have recently been shown to outperform CRFs on similar tasks. For example, Irsoy and Cardie (2014) apply deep recurrent neural networks (RNNs) to extract opinion expressions from sentences and show that RNNs outperform CRFs. Socher et al. (2013) propose recursive neural networks for a semantic compositional task to identify the sentiments of phrases and sentences hierarchically using the syntactic parse trees.

Meanwhile, recent advances in word embed-

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1433–1443,
Lisbon, Portugal, 17-21 September 2015.

ding induction methods (Collobert and Weston, 2008; Mikolov et al., 2013b) have benefited researchers in two ways: (*i*) they have contributed to significant gains when used as extra word features in existing NLP systems (Turian et al., 2010; Lebret and Lebret, 2013), and (*ii*) they have enabled more effective training of RNNs by providing compact input representations of the words (Mesnil et al., 2013; Irsoy and Cardie, 2014).

Motivated by the recent success of deep learning, in this paper we propose a general class of models based on RNN architecture and word embeddings, that can be successfully applied to fine-grained opinion mining tasks without any task-specific feature engineering effort. We experiment with several important RNN architectures including Elman-RNN, Jordan-RNN, long short term memory (LSTM) and their variations. We acquire pre-trained word embeddings from several external sources to give better initialization to our RNN models. The RNN models then fine-tune the word vectors during training to learn task-specific embeddings. We also present an architecture to incorporate other linguistic features into RNNs.

Our results on the task of opinion target extraction show that word embeddings improve the performance of state-of-the-art CRF models, when included as additional features. They also improve RNNs when used as pre-trained word vectors and fine-tuning them on the task gives the best results. A comparison between models demonstrates that RNNs outperform CRFs, even when they use word embeddings as the only features. Incorporating simple linguistic features into RNNs improves the performance even further. Our best results with LSTM RNN outperform the top performing system on the Laptop dataset and achieve the second best on the Restaurant dataset in SemEval-2014. We make our source code available.[1]

In the remainder of this paper, after discussing related work in Section 2, we present our RNN models in Section 3. In Section 4, we briefly describe the pre-trained word embeddings. The experiments and analysis of results are presented in Section 5. Finally, we summarize our contributions with future directions in Section 6.

2 Related Work

A line of previous research in fine-grained opinion mining focused on detecting opinion (subjective) expressions, e.g., (Wilson et al., 2005; Breck et al., 2007). The common approach was to formulate the problem as a sequence tagging task and use a CRF model. Later approaches extended this to jointly identify opinion holders (Choi et al., 2005), and intensity and polarity (Choi and Cardie, 2010).

Extracting aspect terms or opinion targets have been actively investigated in the past. Typical approaches include association mining to find frequent item sets (i.e., co-occurring words) as candidate aspects (Hu and Liu, 2004), classification-based methods such as hidden Markov model (Jin et al., 2009) and CRF (Shariaty and Moghaddam, 2011; Yang and Cardie, 2012; Yang and Cardie, 2013), as well as topic modeling techniques using Latent Dirichlet Allocation (LDA) model and its variants (Titov and McDonald, 2008; Lin and He, 2009; Moghaddam and Ester, 2012).

Conventional RNNs (e.g., Elman type) and LSTM have been successfully applied to various sequence prediction tasks, such as language modeling (Mikolov et al., 2010; Sundermeyer et al., 2012), speech recognition (Graves and Jaitly, 2014; Sak et al., 2014) and spoken language understanding (Mesnil et al., 2013). For sentiment analysis, Socher et al. (2013) propose to use recursive neural networks to hierarchically compose semantic word vectors based on syntactic parse trees, and use the vectors to identify the sentiments of the phrases and sentences. Le and Zuidema (2015) extended recursive neural networks with LSTM to compute a parent vector in parse trees by combining information of both output and LSTM memory cells from its two children.

Most relevant to our work is the recent work of Irsoy and Cardie (2014), where they apply deep Elman-type RNN to extract opinion expressions and show that deep RNN outperforms CRF, semi-CRF and shallow RNN. They used word embeddings from Google without fine-tuning them.

Although inspired, our work differs from the work of Irsoy and Cardie (2014) in many ways. (*i*) We experiment with not only Elman-type, but also with a Jordan-type and with a more advanced LSTM RNN, and demonstrate the performance of various RNN models. (*ii*) We use not only Google embeddings as pre-trained word vectors, but also other embeddings including SENNA and Amazon, and show their performances. (*iii*) We also fine-tune the embeddings for our task, which is shown to be very crucial. (*iv*) We present an RNN ar-

[1] https://github.com/ppfliu/opinion-target

chitecture to include other linguistic features and show its effectiveness. (*v*) Finally, we present a comprehensive experiment exploring different embedding dimensions and hidden layer sizes for all the variations of the RNNs (i.e., including features and bi-directionality).

3 Recurrent Neural Models

The recurrent neural models in this section compute compositional vector representations for word sequences of arbitrary length. These high-level (i.e., hidden-layer) distributed representations are then used as features to classify each token in the sentence. We first describe the common properties shared among the RNNs below, followed by the descriptions of the specific RNNs.

Each word in the vocabulary V is represented by a D dimensional vector in the shared look-up table $L \in \mathbb{R}^{|V| \times D}$. Note that L is considered as a model parameter to be learned. We can initialize L randomly or by pre-trained word embedding vectors (see Section 4). Given an input sentence $\mathbf{s} = (s_1, \cdots, s_T)$, we first transform it into a feature sequence by mapping each word token $s_t \in \mathbf{s}$ to an index in L. The look-up layer then creates a context vector $\mathbf{x_t} \in \mathbb{R}^{mD}$ covering $m-1$ neighboring tokens for each s_t by concatenating their respective vectors in L. For example, given the context size $m = 3$, the context vector $\mathbf{x_t}$ for the word *disk* in Figure 1 is formed by concatenating the embeddings of *hard*, *disk* and *is*. This window-based approach is intended to capture short-term dependencies between neighboring words in a sentence (Collobert et al., 2011).

The concatenated vector is then passed through non-linear recurrent hidden layers to learn high-level compositional representations, which are in turn fed to the output layer for classification using `softmax`. Formally, the probability of k-th label in the output for classification into K classes:

$$P(y_t = k|\mathbf{s}, \theta) = \frac{exp\ (\mathbf{w}_k^T \mathbf{h_t})}{\sum_{k=1}^{K} exp\ (\mathbf{w}_k^T \mathbf{h_t})} \quad (1)$$

where, $\mathbf{h}_t = \phi(\mathbf{x}_t)$ defines the transformations of $\mathbf{x}_t$ through the hidden layers, and $\mathbf{w}_k$ are the weights from the last hidden layer to the output layer. We fit the models by minimizing the negative log likelihood (NLL) of the training data. The NLL for the sentence $\mathbf{s}$ can be written as:

$$J(\theta) = \sum_{t=1}^{T} \sum_{k=1}^{K} y_{tk}\ log\ P(y_t = k|\mathbf{s}, \theta) \quad (2)$$

where, $y_{tk} = I(y_t = k)$ is an indicator variable to encode the gold labels, i.e., $y_{tk} = 1$ if the gold label $y_t = k$, otherwise 0.[2] The loss function minimizes the cross-entropy between the predicted distribution and the target distribution (i.e., gold labels). The main difference between the models described below is how they compute $\mathbf{h}_t = \phi(\mathbf{x}_t)$.

3.1 Elman-type RNN (Elman, 1990)

In an Elman-type RNN (Fig. 1a), the output of the hidden layer $\mathbf{h}_t$ at time t is computed from a non-linear transformation of the current input $\mathbf{x}_t$ and the previous hidden layer output $\mathbf{h}_{t-1}$. Formally,

$$\mathbf{h}_t = f(U\mathbf{h}_{t-1} + V\mathbf{x}_t + \mathbf{b}) \quad (3)$$

where f is a nonlinear function (e.g., `sigmoid`) applied to the hidden units. U and V are weight matrices between two consecutive hidden layers, and between the input and the hidden layers, respectively, and $\mathbf{b}$ is the bias vector.

This RNN thus creates internal states by remembering previous hidden layer, which allows it to exhibit dynamic temporal behavior. We can interpret $\mathbf{h}_t$ as an intermediate representation summarizing the past, which is in turn used to make a final decision on the current input.

3.2 Jordan-type RNN (Jordan, 1997)

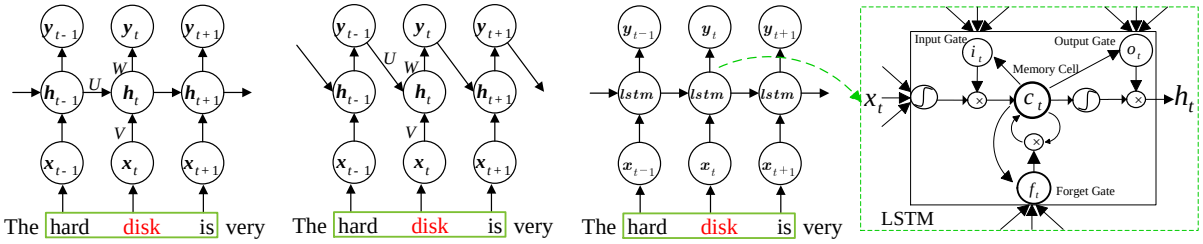
Jordan-type RNNs (Fig. 1b) are similar to Elman-type RNNs except that the hidden layer $\mathbf{h}_t$ at time t is fed from the previous output layer $\mathbf{y}_{t-1}$ instead of the previous hidden layer $\mathbf{h}_{t-1}$. Formally,

$$\mathbf{h}_t = f(U\mathbf{y}_{t-1} + V\mathbf{x}_t + \mathbf{b}) \quad (4)$$

where U, V, $\mathbf{b}$, and f are similarly defined as before. Both Elman-type and Jordan-type RNNs are known as simple RNNs. These types of RNNs are generally trained using stochastic gradient descent (SGD) with backpropagation through time (BPTT), where errors (i.e., gradients) are propagated back through the edges over time.

One common issue with BPTT is that as the errors get propagated, they may soon become very small or very large that can lead to undesired values in weight matrices, causing the training to fail.

[2] This is also known as one-hot vector representation.



(a) Elman-type RNN (b) Jordan-type RNN (c) Long Short-Term Memory (LSTM) RNN

Figure 1: Elman-type, Jordan-type and LSTM RNNs with a lookup-table layer, a hidden layer and an output layer. The concatenated context vector for the word "disk" at time t is $x_t = [x_{hard}, x_{disk}, x_{is}]$ with a context window of size 3. One memory block in the LSTM hidden layer has been enlarged.

This is known as the *vanishing* and the *exploding* gradients problem (Bengio et al., 1994). One simple way to overcome this issue is to use a truncated BPTT (Mikolov, 2012) for restricting the back-propagation to only few steps like 4 or 5. However, this solution limits the RNN to capture long-range dependencies. In the following, we describe an elegant RNN architecture to address this problem.

3.3 Long Short-Term Memory RNN

Long Short-Term Memory or LSTM (Hochreiter and Schmidhuber, 1997) is specifically designed to model long term dependencies in RNNs. The recurrent layer in a standard LSTM is constituted with special (hidden) units called *memory blocks* (Fig. 1c). A memory block is composed of four elements: (*i*) a memory cell c (i.e., a neuron) with a self-connection, (*ii*) an input gate i to control the flow of input signal into the neuron, (*iii*) an output gate o to control the effect of the neuron activation on other neurons, and (*iv*) a forget gate f to allow the neuron to adaptively reset its current state through the self-connection. The following sequence of equations describe how a layer of memory blocks is updated at every time step t:

$$\mathbf{i}_t = \sigma(U_i\mathbf{h}_{t-1} + V_i\mathbf{x}_t + C_i\mathbf{c}_{t-1} + \mathbf{b}_i) \quad (5)$$
$$\mathbf{f}_t = \sigma(U_f\mathbf{h}_{t-1} + V_f\mathbf{x}_t + C_f\mathbf{c}_{t-1} + \mathbf{b}_f) \quad (6)$$
$$\mathbf{c}_t = \mathbf{i}_t \odot g(U_c\mathbf{h}_{t-1} + V_c\mathbf{x}_t + \mathbf{b}_c) + \mathbf{f}_t \odot \mathbf{c}_{t-1} \quad (7)$$
$$\mathbf{o}_t = \sigma(U_o\mathbf{h}_{t-1} + V_o\mathbf{x}_t + C_o\mathbf{c}_t + \mathbf{b}_o) \quad (8)$$
$$\mathbf{h}_t = \mathbf{o}_t \odot h(\mathbf{c}_t) \quad (9)$$

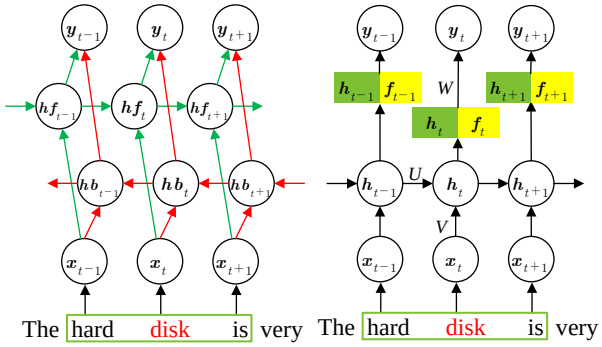
where U_k, V_k and C_k are the weight matrices between two consecutive hidden layers, between the input and the hidden layers, and between two consecutive cell activations, respectively, which are associated with gate k (i.e., input, output, forget and cell), and $\mathbf{b}_k$ is the associated bias vector. The symbol $\odot$ denotes a element-wise product of the two vectors. The gate function σ is the `sigmoid` activation, and g and h are the cell input and cell output activations, typically a `tanh`. LSTMs are generally trained using truncated or full BPTT.

3.4 Bidirectionality

Notice that the RNNs defined above only get information from the past. However, information from the future could also be crucial. In our example sentence (Table 1), to correctly tag the word *hard* as a **B-TARG**, it is beneficial for the RNN to know that the next word is *disk*. Our window-based approach, by considering the neighboring words, already captures short-term dependencies like this from the future. However, it requires tuning to find the right window size, and it disregards long-range dependencies that go beyond the context window, which is typically of size 1 (i.e., no context) to 5 (see Section 5.2). For instance, consider the two sentences: (*i*) *Do you know about the crunchy tuna here, it is to die for.* and (*ii*) *Do you know about the crunchy tuna here, it is imported from Norway.* The phrase *crunchy tuna* is an aspect term in the first (subjective) sentence, but not in the second (objective) one. The RNN models described above will assign the same labels to *crunchy* and *tuna* in both sentences, since the preceding sequences and the context window (of size 1 to 5) are the same.

To capture long-range dependencies from the future as well as from the past, we propose to use bidirectional RNNs (Schuster and Paliwal, 1997), which allow bidirectional links in the network. In an Elman-type bidirectional RNN (Fig. 2a), the forward hidden layer $\overrightarrow{\mathbf{h}_t}$ and the backward hidden layer $\overleftarrow{\mathbf{h}_t}$ at time t are computed as follows:

$$\overrightarrow{\mathbf{h}_t} = f(\overrightarrow{U}\mathbf{h}_{t-1} + \overrightarrow{V}\mathbf{x}_t + \overrightarrow{\mathbf{b}}) \quad (10)$$
$$\overleftarrow{\mathbf{h}_t} = f(\overleftarrow{U}\mathbf{h}_{t-1} + \overleftarrow{V}\mathbf{x}_t + \overleftarrow{\mathbf{b}}) \quad (11)$$



(a) (b)

Figure 2: (a) Bidirectional Elman-type RNN and (b) Linguistic features concatenated with the hidden layer output in Elman-type RNN.

where $\overrightarrow{U}$, $\overrightarrow{V}$ and $\overrightarrow{\mathbf{b}}$ are the forward weight matrices as before; and $\overleftarrow{U}$, $\overleftarrow{V}$ and $\overleftarrow{\mathbf{b}}$ are their backward counterparts. The concatenated vector $\mathbf{h}_t = [\overrightarrow{\mathbf{h}_t}, \overleftarrow{\mathbf{h}_t}]$ is passed to the output layer. We can thus interpret $\mathbf{h}_t$ as an intermediate representation summarizing the past and the future, which is then used to make a final decision on the current input.

Similarly, the unidirectional LSTM RNN can be extended to bidirectional LSTM by allowing bidirectional connections in the hidden layer. This amounts to having a backward counterpart for each of the equations from 5 to 9.

Notice that the forward and the backward computations of bidirectional RNNs are independently done until they are combined in the output layer. This means, during training, after backpropagating the errors from the output layer to the forward and to the backward hidden layers, two independent BPTT can be applied – one to each direction.

3.5 Fine-tuning of Embeddings

In our RNN framework, we intend to avoid manual feature engineering efforts by using word embeddings as the only features. As mentioned before, we can initialize the embeddings randomly and learn them as part of model parameters by backpropagating the errors to the look-up layer. One issue with random initialization is that it may lead the SGD to get stuck in local minima (Murphy, 2012). On the other hand, one can plug the readily available embeddings from external sources (Section 4) in the RNN model and use them as features without tuning them further for the task, as is done in any other machine learning model. However, this approach does not exploit the automatic feature learning capability of NN models, which is one of the main motivations of using them.

In our work, we use the pre-trained word embeddings to better initialize our models, and we fine-tune them for our task in training, which turns out to be quite beneficial (see Section 5.2).

3.6 Incorporating other Linguistic Features

Although NNs learn word features (i.e., embeddings) automatically, we may still be interested in incorporating other linguistic features like part-of-speech (POS) tags and chunk information to guide the training and to learn a better model. However, unlike word embeddings, we want these features to be fixed during training. As shown in Figure 2b, this can be done in our RNNs by feeding these additional features directly to the output layer, and learn their associated weights in training.

4 Word Embeddings

Word embeddings are distributed representations of words, represented as real-valued, dense, and low-dimensional vectors. Each dimension potentially describes syntactic or semantic properties of the word. Here we briefly describe the three types of pre-trained embeddings that we use in our work.

4.1 SENNA Embeddings

Collobert et al. (2011) present a unified NN architecture for various NLP tasks (e.g., POS tagging, chunking, semantic role labeling, named entity recognition) with a window-based approach and a sentence-based approach (i.e., the input layer is a sentence). Each word in the input layer is represented by M features, each of which has an embedding vector associated with it in a *lookup table*. To give their network a better initialization, they learn word embeddings using a non-probabilistic language model, which was trained on English Wikipedia for about 2 months. They released their 50-dimensional word embeddings (vocabulary size $130K$) under the name SENNA.[3]

4.2 Google Embeddings

Mikolov et al. (2013a) propose two log-linear models for computing word embeddings from large corpora efficiently: (*i*) a *bag-of-words* model CBOW that predicts the current word based on the context words, and (*ii*) a *skip-gram* model that predicts surrounding words given the current word.

[3] http://ronan.collobert.com/senna/

They released their pre-trained 300-dimensional word embeddings (vocabulary size $3M$) trained by the skip-gram model on part of Google news dataset containing about 100 billion words.[4]

4.3 Amazon Embeddings

Since we work on customer reviews, which are less formal than Wikipedia and news, we have also trained *domain-specific* embeddings (vocabulary size $1M$) using the CBOW model of *word2vec* tool (Mikolov et al., 2013b) from a large corpus of Amazon reviews.[5] The corpus contains $34,686,770$ reviews (4.7 billion words) on Amazon products from June 1995 to March 2013 (McAuley and Leskovec, 2013). For comparison with SENNA and Google, we learn word embeddings of 50- and 300-dimensions.

5 Experiments

In this section, we present our experimental settings and results for the task of opinion target extraction from customer reviews.

5.1 Experimental Settings

Datasets: In our experiments, we use the two review datasets provided by the *SemEval-2014 task 4: aspect-based sentiment analysis* evaluation campaign (Pontiki et al., 2014), namely the *Laptop* and the *Restaurant* datasets. Table 2 shows some basic statistics about the datasets. The majority of aspect terms have only one word, while about one third of them have multiple words. In both datasets, some sentences have no aspect terms and some have more than one aspect terms. We use the standard train:test split to compare our results with the SemEval best systems. In addition, we show a more general performance of our models on the two datasets based on 10–fold cross validation.

	Laptop		Restaurant	
	Train	Test	Train	Test
Sentences	3045	800	3041	800
Sentence length	15	13	14	14
One-word targets	1494	364	2786	818
Multi-word targets	864	290	907	316
Total targets	2358	654	3693	1134

Table 2: Corpora statistics.

[4]https://code.google.com/p/word2vec/

[5]https://snap.stanford.edu/data/web-Amazon.html

Evaluation Metric: The evaluation metric measures the standard precision, recall and F_1 score based on *exact* matches. This means that a candidate aspect term is considered to be correct only if it exactly matches with the aspect term annotated by the human. In all our experiments when comparing two models, we use paired t-test on the F_1 scores to measure statistical significance and report the corresponding p-value.

CRF Baseline: We use a linear-chain CRF (Lafferty et al., 2001) of order 2 as our baseline, which is the state-of-the-art model for opinion target extraction (Pontiki et al., 2014). Specifically, the CRF generates (binary) feature functions of order 1 and 2; see (Cuong et al., 2014) for higher order CRFs. The features used in the baseline model include the current word, its POS tag, its prefixes and suffixes between one to four characters, its position, its stylistics (e.g., case, digit, symbol, alphanumeric), and its context (i.e., the same features for the two preceding and the two following words). In addition to the hand-crafted features, we also include the three different types of word embeddings described in Section 4.

RNN Settings: We pre-processed each dataset by lowercasing all words and spelling out each digit number as `DIGIT`. We then built the vocabulary from the training set by marking rare words with only one occurrence as `UNKNOWN`, and adding a `PADDDING` word to make context windows for boundary words.

To implement *early stopping* in SGD, we prepared a *validation set* by separating out randomly 10% of the available training data. The remaining 90% is used for training. The weights in the network were initialized by sampling from a small random uniform distribution $\mathcal{U}(-0.2, 0.2)$. The time step in the truncated BPTT was fixed to 6 based on the performance on the validation set; smaller values hurt the performance, while larger values showed no significant gains but increased the training time.

We use a fixed learning rate of 0.01, but we change the batch size depending on the sentence length following Mesnil et al. (2013). The net effect is a variable step size in the SGD. We run SGD for 30 epochs, calculate the F_1 score on the validation set after each epoch, and stop if the accuracy starts to decrease. The size of the context window and the hidden layer are empirically set based on the performance on the validation set. We experimented

with the window size $\in \{1, 3, 5\}$, and found 3 to be the optimal on the validation set. The hidden layer sizes we experimented with are 50, 100, 150, and 200; we report the optimal values in Table 3 (see $|h_l|$ and $|h_r|$ columns).

Linguistic Features in RNNs: In addition to the neural features, we also explore the contribution of simple linguistic features in our RNN models using the architecture described in Section 3.6. Specifically, we encode four POS tag classes (*noun, adjective, verb, adverb*) and BIO-tagged chunk information (*NP, VP, PP, ADJP, ADVP*) as binary features. We feed these extra features directly to the output layer of the RNNs and learn their relative weights. Part-of-speech and phrasal information are arguably the most informative features for identifying aspect terms (i.e., aspect terms are generally noun phrases). BIO tags could be useful to find the right text spans (i.e., aspect terms are unlikely to violate phrasal boundaries).

5.2 Results and Discussion

Table 3 presents our results of aspect term extraction on the *standard testset* in F_1 scores. In Table 4, we show the results on the *whole datasets* based on 10-fold cross validation. RNNs in Table 4 are trained using SENNA embeddings. We perform significance tests on the 10-fold results. In the following, we highlight our main findings.

Contributions of Word Embeddings in CRF: From the first group of results in Table 3, we can observe that even though CRF uses a handful of hand-designed features, including word embeddings still leads to sizable improvements on both datasets. The domain-specific Amazon embeddings (300 dim.) yield more general performance across the datasets, delivering the best gain of absolute 3.54% on the Laptop and the second best on the Restaurant dataset. Google embeddings give the best gain on the Restaurant dataset (absolute 3.08%). The contribution of embeddings in CRF is also validated by the 10-fold results in Table 4 (see first two rows), where SENNA embeddings yield significant improvements – absolute 1.47% on Laptop ($p < 0.03$) and absolute 1.24% on Restaurant ($p < 0.01$). This demonstrates that word embeddings offer generalizations that complement other strong features, and thus should be considered.

CRF vs. RNNs: When we compare the results of

System	Dim.	$\|h_l\|$	Laptop	$\|h_r\|$	Restaurant
CRF Base	-	-	68.66	-	77.28
+SENNA	50	-	71.38	-	78.54
+Amazon	50	-	70.61	-	79.46
+Google	300	-	68.81	-	**80.36**
+Amazon	300	-	**72.20**	-	79.66
Jordan-RNN					
+SENNA	50	200	71.41	200	78.83
+Amazon	50	100	73.21	150	79.01
+Google	300	150	**73.42**	200	**79.89**
+Amazon	300	50	72.43	200	78.30
Elman-RNN					
+SENNA	50	100	73.86	150	79.89
+Amazon	50	100	**74.43**	100	**80.37**
+Google	300	100	72.91	100	79.54
+Amazon	300	200	73.67	100	79.82
Elman-RNN + Feat.					
+SENNA	50	50	73.70	100	81.36
+Amazon	50	200	73.30	50	**81.66**
+Google	300	150	**74.25**	100	80.57
+Amazon	300	50	73.92	100	80.24
Bi-Elman-RNN					
+SENNA	50	100	72.38	100	**80.10**
+Amazon	50	50	**73.93**	50	79.97
+Google	300	50	72.67	100	79.52
+Amazon	300	50	71.12	50	79.09
Bi-Elman-RNN + Feat.					
+SENNA	50	100	73.30	50	80.34
+Amazon	50	50	**74.57**	50	**82.06**
+Google	300	50	74.56	100	78.99
+Amazon	300	50	73.56	100	79.97
LSTM-RNN					
+SENNA	50	100	73.40	150	79.43
+Amazon	50	50	72.44	50	**79.79**
+Google	300	100	72.11	50	79.20
+Amazon	300	50	**73.52**	50	78.99
LSTM-RNN + Feat.					
+SENNA	50	50	73.19	150	80.28
+Amazon	50	100	**75.00**	50	80.82
+Google	300	50	72.19	50	**81.37**
+Amazon	300	100	72.85	100	80.60
Bi-LSTM-RNN					
+SENNA	50	50	72.60	150	**79.89**
+Amazon	50	100	**74.03**	100	79.36
+Google	300	50	70.90	50	78.80
+Amazon	300	150	71.25	150	78.88
Bi-LSTM-RNN + Feat.					
+SENNA	50	100	**74.02**	150	**81.06**
+Amazon	50	100	73.58	50	80.51
+Google	300	100	71.05	50	79.39
+Amazon	300	100	73.81	150	80.67
SemEval-14 top systems					
IHS_RD	-	-	**74.55**	-	79.62
DLIREC	-	-	73.78	-	**84.01**

Table 3: F_1-score performance for CRF baselines, RNNs and SemEval'14 best systems on the standard Laptop and Restaurant testsets. $|h_l|$ and $|h_r|$ columns show the number of hidden units.

Model	Laptop			Restaurant		
	P	R	F_1	P	R	F_1
CRF Base	**79.77**	64.09	70.87	**82.59**	74.63	78.36
+ SENNA	78.23	**67.38**	**72.34**	81.21	**78.12**	**79.60**
Elman-RNN	**82.03**	72.68	76.97	81.96	78.41	80.08
+ Feat.	80.02	**76.60**	**78.22**	81.91	**81.22**	81.52
+ Bidir.	81.92	73.70	77.47	81.69	78.46	79.97
+ Feat. + Bidir.	81.00	75.70	78.17	**82.80**	80.44	**81.57**
LSTM-RNN	**81.92**	73.30	77.14	**83.64**	77.45	80.36
+ Feat.	80.70	**75.82**	**78.00**	81.80	**81.39**	81.54
+ Bidir.	81.31	74.20	77.37	81.66	79.23	80.37
+ Feat. + Bidir.	80.81	74.48	77.27	82.96	80.42	**81.56**

Table 4: 10-fold cross validation results of the models on the two datasets. Elman- and LSTM-RNNs are trained using SENNA embeddings.

System	Dim.	Laptop		Restaurant	
Elman-RNN		-tune	+tune	-tune	+tune
+SENNA	50	60.85	73.86	75.78	79.89
+Amazon	50	15.51	74.43	22.85	80.37
+Random	50	38.26	72.99	56.98	78.44
+Google	300	67.91	72.91	74.73	79.54
+Amazon	300	15.51	73.67	22.85	79.82
Jordan-RNN		-tune	+tune	-tune	+tune
+SENNA	50	58.81	71.41	74.68	78.83
+Amazon	50	15.51	73.21	22.85	79.01
+Random	50	38.05	71.46	55.65	77.38
+Google	300	69.39	73.42	77.33	79.89
+Amazon	300	15.51	72.43	22.85	78.30

Table 5: Effects of fine-tuning in Elman-RNN and Jordan-RNN.

RNNs with those of CRF in Table 3, we see that most of our RNN models outperform CRF models with the maximum absolute gains of 2.80% by **LSTM-RNN+Feat.** on Laptop and 1.70% by **Bi-Elman-RNN+Feat.** on Restaurant. What is remarkable is that RNNs without any hand-crafted features outperform feature-rich CRF models by a good margin – absolute maximum gains of 2.23% by Elman-RNN and 1.83% by **Bi-LSTM-RNN** on Laptop. When we compare their general performance on the 10-folds in Table 4, we observe similar gains, maximum 5.88% on Laptop and 1.97% on Restaurant, which are significant with $p < 6 \times 10^{-6}$ on Laptop and $p < 2 \times 10^{-4}$ on Restaurant. These results demonstrate that RNNs as sequence labelers are more effective than CRFs for fine-grained opinion mining tasks. This can be attributed to RNN's ability to learn better features automatically and to capture long-range sequential dependencies between the output labels.

Comparison among RNN Models: A comparison among the RNN models in Table 3 tells that Elman RNN generally outperforms Jordan RNN. However, bi-directionality and LSTM do not provide clear gains over the simple Elman RNN. In fact, bi-directionality hurts the performance in most cases. This finding contrasts the finding of Irsoy and Cardie (2014) in opinion expression detection task, where bi-directional Elman RNNs outperform their uni-directional counterparts. However, when we analyzed the data, we found it to be unsurprising because aspect terms are generally shorter than opinion expressions. For example, the average length of an aspect term in our Restaurant dataset is 1.4, where the average length of an expressive subjective expression in their MPQA corpus is 3.3. Therefore, the information required to correctly identify aspect terms (e.g., *hard disk*) is already captured by the simple (as opposed to LSTM) unidirectional link and the context window covering the neighboring words. LSTM and Bi-directionality increase the number of parameters in the RNNs, which might contribute to overfitting on this specific task.[6]

[6]Bi-directional links double the number of parameters in RNNs.

As a partial solution to this problem, we experimented with a bi-directional Elman-RNN, where both directions share the same parameters. Therefore, the number of parameters remains the same as the uni-directional one. This modification improves the performance over the non-shared one slightly but not significantly. This demands for better modeling of the two sources of information rather than simple concatenation or sharing.

Contributions of Linguistic Features in RNNs: Although our linguistic features are quite simple (i.e., POS tags and chunk), they give gains on both datasets when incorporated into Elman and LSTM RNNs. The maximum gains on the standard testset (Table 3) are 0.64% on Laptop and 1.96% on Restaurant for Bi-Elman, and 1.48% on Laptop and 1.58% on Restaurant for LSTM. Similar gains are also observed on the 10-folds in Table 4, where the maximum gains are 1.25% on Laptop and 1.44% on Restaurant. These gains are significant with $p < 0.004$ on Laptop and $p < 6 \times 10^{-5}$ on Restaurant. Linguistic features thus complement word embeddings in RNNs.

Importance of Fine-tuning in RNNs: Finally, in order to show the importance of fine-tuning the word embeddings in RNNs on our task, we present in Table 5 the performance of Elman and Jordan RNNs, when the embeddings are used as they are (*'-tune'*), and when they are fine-tuned (*'+tune'*) on the task. The table also shows the contributions of pre-trained embeddings as compared to random initialization. Surprisingly, Amazon embeddings without fine-tuning deliver the worst performance, even lower than the Random initialization. We found that with Amazon embeddings the network gets stuck in a local minimum from the very first epoch.

Other pre-trained (untuned) embeddings improve over the Amazon and Random by providing better initialization. In most cases fine-tuning makes a big difference. For example, the absolute gains for fine-tuning SENNA embeddings in Elman RNN are 13.01% in Laptop and 4.11% in Restaurant. Remarkably, fine-tuning brings both Random and Amazon embeddings close to the best ones.

Comparison with SemEval-2014 Systems: When our RNN results are compared with the top performing systems in the SemEval-2014 (last two rows in Table 3), we see that RNNs without using any linguistic features achieve the second best results on both Laptop and Restaurant datasets. Note that these RNNs only use word embeddings, while IHS_RD and DLIREC use complex features like dependency relations, named entity, sentiment orientation of words, word cluster and many more in their CRF models, most of which are expensive to compute; see (Toh and Wang, 2014; Chernyshevich, 2014). The performance of our RNNs improves when they are given access to very simple features like POS tags and chunks, and **LSTM-RNN+Feat.** achieves the best results on the Laptop dataset.

6 Conclusion and Future Direction

We presented a general class of discriminative models based on recurrent neural network (RNN) architecture and word embeddings, that can be successfully applied to fine-grained opinion mining tasks without any task-specific manual feature engineering effort. We used pre-trained word embeddings from three external sources in different RNN architectures including Elman-type, Jordan-type, LSTM and their several variations.

Our results on the opinion target extraction task demonstrate that word embeddings improve the performance of both CRF and RNN models, however, fine-tuning them in RNNs on the task gives the best results. RNNs outperform CRFs, even when they use word embeddings as the only features. Incorporating simple linguistic features into RNNs improves the performance further. Our best results with LSTM RNN outperform the top performing system on the Laptop dataset and achieve the second best on the Restaurant dataset in SemEval-2014 evaluation campaign. We made our code publicly available for research purposes.

In the future, we would like apply our models to other fine-grained opinion mining tasks including opinion expression detection and characterizing the intensity and sentiment of the opinion expressions. We would also like to explore to what extent these tasks can be jointly modeled in an RNN-based multi-task learning framework.

Acknowledgments

We are grateful to the anonymous reviewers for their insightful comments and suggestions to improve the paper. This research is affiliated with the Stanley Ho Big Data Decision Analytics Research Centre of The Chinese University of Hong Kong.

References

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166.

Eric Breck, Yejin Choi, and Claire Cardie. 2007. Identifying expressions of opinion in context. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, pages 2683–2688. Morgan Kaufmann Publishers Inc.

Maryna Chernyshevich. 2014. IHS R&D Belarus: Cross-domain extraction of product features using conditional random fields. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, page 309.

Yejin Choi and Claire Cardie. 2010. Hierarchical sequential learning for extracting opinions and their attributes. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 269–274. ACL.

Yejin Choi, Claire Cardie, Ellen Riloff, and Siddharth Patwardhan. 2005. Identifying sources of opinions with conditional random fields and extraction patterns. In *Proceedings of HLT/EMNLP*, pages 355–362. ACL.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: deep neural networks with multitask learning. In *Proceedings of ICML*, pages 160–167. ACM.

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

Nguyen Viet Cuong, Nan Ye, Wee Sun Lee, and Hai Leong Chieu. 2014. Conditional random field with high-order dependencies for sequence labeling and segmentation. *The Journal of Machine Learning Research*, 15(1):981–1009.

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

Alex Graves and Navdeep Jaitly. 2014. Towards end-to-end speech recognition with recurrent neural networks. In *Proceedings of ICML*, pages 1764–1772.

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

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of SIGKDD*, pages 168–177. ACM.

Ozan Irsoy and Claire Cardie. 2014. Opinion mining with deep recurrent neural networks. In *Proceedings of EMNLP*, pages 720–728.

Wei Jin, Hung Hay Ho, and Rohini K Srihari. 2009. A novel lexicalized HMM-based learning framework for web opinion mining. In *Proceedings of ICML*, pages 465–472. Citeseer.

Michael I Jordan. 1997. Serial order: A parallel distributed processing approach. *Advances in psychology*, 121:471–495.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proceedings of ICML*, pages 282–289.

Phong Le and Willem Zuidema. 2015. Compositional distributional semantics with long short term memory. In *Proceedings of the joint Conference on Lexical and Computational Semantics (*SEM)*.

Rémi Lebret and Ronan Lebret. 2013. Word emdeddings through Hellinger PCA. *arXiv preprint arXiv:1312.5542*.

Chenghua Lin and Yulan He. 2009. Joint sentiment/topic model for sentiment analysis. In *Proceedings of CIKM*, pages 375–384. ACM.

Julian McAuley and Jure Leskovec. 2013. Hidden factors and hidden topics: understanding rating dimensions with review text. In *Proceedings of the 7th ACM conference on Recommender systems*, pages 165–172. ACM.

Grégoire Mesnil, Xiaodong He, Li Deng, and Yoshua Bengio. 2013. Investigation of recurrent-neural-network architectures and learning methods for spoken language understanding. In *Proceedings of INTERSPEECH*, pages 3771–3775.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Proceedings of INTERSPEECH*, pages 1045–1048.

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

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

Tomas Mikolov, 2012. *Statistical Language Models based on Neural Networks*. PhD thesis, Brno University of Technology.

Samaneh Moghaddam and Martin Ester. 2012. On the design of LDA models for aspect-based opinion mining. In *Proceedings of CIKM*, pages 803–812. ACM.

Kevin Murphy. 2012. *Machine Learning A Probabilistic Perspective*. The MIT Press.

Maria Pontiki, Haris Papageorgiou, Dimitrios Galanis, Ion Androutsopoulos, John Pavlopoulos, and Suresh Manandhar. 2014. Semeval-2014 task 4: Aspect based sentiment analysis. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 27–35.

Hasim Sak, Andrew Senior, and Françoise Beaufays. 2014. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In *Proceedings of INTERSPEECH*, pages 338–342.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Shabnam Shariaty and Samaneh Moghaddam. 2011. Fine-grained opinion mining using conditional random fields. In *International Conference on Data Mining Workshops (ICDMW)*, pages 109–114. IEEE.

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of EMNLP*, pages 1631–1642. Citeseer.

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

Ivan Titov and Ryan McDonald. 2008. Modeling online reviews with multi-grain topic models. In *Proceedings of WWW*, pages 111–120. ACM.

Zhiqiang Toh and Wenting Wang. 2014. DLIREC: Aspect term extraction and term polarity classification system. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, page 235.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of ACL*, pages 384–394. ACL.

Janyce Wiebe, Theresa Wilson, and Claire Cardie. 2005. Annotating expressions of opinions and emotions in language. *Language resources and evaluation*, 39(2-3):165–210.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of HLT/EMNLP*, pages 347–354. ACL.

Bishan Yang and Claire Cardie. 2012. Extracting opinion expressions with semi-Markov conditional random fields. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1335–1345. ACL.

Bishan Yang and Claire Cardie. 2013. Joint inference for fine-grained opinion extraction. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 1640–1649. ACL.

Joint A* CCG Parsing and Semantic Role Labeling

Mike Lewis **Luheng He** **Luke Zettlemoyer**
Computer Science & Engineering
University of Washington
Seattle, WA 98195
{mlewis,luheng,lsz}@cs.washington.edu

Abstract

Joint models of syntactic and semantic parsing have the potential to improve performance on both tasks—but to date, the best results have been achieved with pipelines. We introduce a joint model using CCG, which is motivated by the close link between CCG syntax and semantics. Semantic roles are recovered by labelling the deep dependency structures produced by the grammar. Furthermore, because CCG is lexicalized, we show it is possible to factor the parsing model over words and introduce a new A* parsing algorithm—which we demonstrate is faster and more accurate than adaptive supertagging. Our joint model is the first to substantially improve both syntactic and semantic accuracy over a comparable pipeline, and also achieves state-of-the-art results for a non-ensemble semantic role labelling model.

1 Introduction

Joint models of syntactic and semantic parsing are attractive; they can potentially avoid the error propagation that is inherent in pipelines by using semantic models to inform syntactic attachments. However, in practice, the performance of joint systems for semantic role labelling (SRL) has been substantially beneath that of pipelines (Sutton and McCallum, 2005; Lluís et al., 2009; Johansson, 2009; Titov et al., 2009; Naradowsky et al., 2012; Lluís et al., 2013; Henderson et al., 2013). In this paper, we present the first approach to break this trend, by building on the close relationship of syntax and semantics in CCG grammars to enable both (1) a simple but highly effective joint model and (2) an efficient A* parsing algorithm.

Semantic dependencies can span an unbounded number of syntactic dependencies, causing significant inference and sparsity challenges for joint models. For example, in the Figure 1, the semantic dependency between *He* and *deny* spans three syntactic edges. This fact makes it difficult to jointly parse syntactic and semantic dependencies with dynamic programs, and means that dependency path features can be sparse. Syntactic dependencies also often have ambiguous semantic interpretations—for example in *He opened the door* and *The door opened*, the syntactic subject corresponds to different semantic roles.

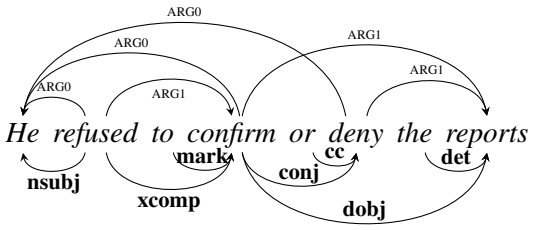


Figure 1: Mismatch between syntactic and semantic dependencies.

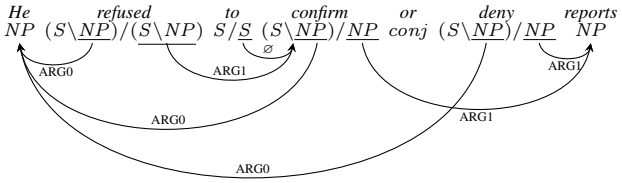


Figure 2: Dependencies produced by a CCG parse. SRL dependencies can be recovered by labelling the edges with a semantic role or ∅. Figure 3 shows a CCG derivation for these dependencies.

We address these challenges with a new joint model of CCG syntactic parsing and semantic role labelling. The CCG formalism is particularly well suited; it models both short- and long-range syntactic dependencies which correspond directly to the semantic roles we aim to recover. The joint model simply involves labelling a subset of these dependencies with the appropriate roles, as seen in Figure 2. This labelling decision can be easily integrated into existing parsing algorithms. CCG also helps resolve cases where interpretation depends on the valency of the pred-

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1444–1454,
Lisbon, Portugal, 17-21 September 2015.

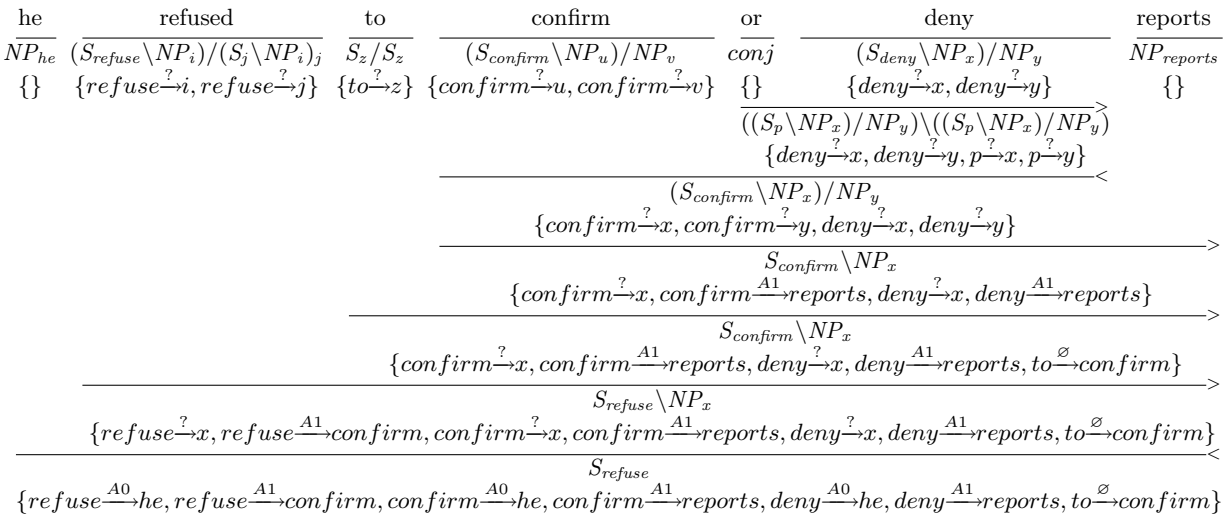


Figure 3: Jointly building a CCG parse and semantic dependencies representation. Subscripts beneath categories denote heads, which are unified when spans combine. One dependency is created for each argument of each lexical category. In our approach, dependency labels are initially underspecified (represented $f \xrightarrow{?} a$) until an attachment is determined by the derivation and a label is chosen by the model.

icate, such as ergative verbs, by learning lexical entries that pair syntactic arguments with semantic roles, such as $open : S\backslash NP_{ARG1}$ and $open : (S\backslash NP_{ARG0})/NP_{ARG1}$. Figure 3 shows a detailed trace of how the example from Figure 2 is parsed with our model.

We also present a new A* algorithm for the joint model. Because CCG is strongly lexicalized, we are able to introduce a new type of *extended lexical entries* that allows us to factor the model over words and develop effective new upper bounds on the Viterbi outside parse score. A* parsing algorithms have previously been developed for models with tree-structured syntactic dependencies (Klein and Manning, 2003; Auli and Lopez, 2011b), and models with no bi-lexical dependencies, including supertag-factored CCGs (Lewis and Steedman, 2014a). We generalize these techniques to SRL-style graph-structured dependencies.

Experiments demonstrate that our model not only outperforms pipeline semantic role labelling models, but improves the quality of the syntactic parser. PropBank SRL performance is 1.6 points higher than comparable existing work, and semantic features improve syntactic accuracy by 1.6 points. Our A* algorithm is 5 times faster than CKY parsing, with no loss in accuracy. The combination of CCG-based joint modelling and A* decoding gives an efficient, accurate, and linguistically principled parser.[1]

[1]The parser is available from: `https://github.com/mikelewis0/EasySRL`

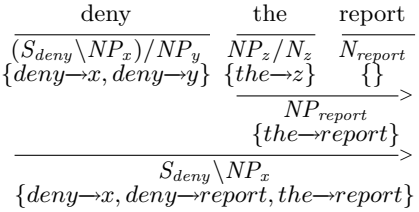
2 Background

2.1 CCG Dependencies

CCG parses define an implicit dependency graph, by associating each argument of each category with a dependency. In contrast to Stanford dependency trees, CCG dependency graphs can be non-projective, and words can have multiple parents. For example, in Figure 2, *He* is an argument of *refuse*, *confirm* and *deny*.

To create dependencies, categories are marked with headedness information—which we denote with subscripts. For example, in the category $(S_{deny}\backslash NP_x)/NP_y$, the final head for the sentence S will be *deny*, and that two dependencies will be introduced from *deny* to unspecified arguments x and y. During parsing, variables are unified with heads, creating fully specified dependencies.

Re-using variables allows dependencies to propagate. For example, the determiner category NP_i/N_i marks that the head of the resulting NP is equal to the head of its N argument (e.g. the head of *the report* is *report*), as in the following parse:



The same mechanism allows long-range arguments to be propagated. Figure 3 shows several long-range arguments, such as how co-indexation

propagates the subject of *deny* to *he*. For more details, see Hockenmaier (2003).

2.2 A* CCG parsing

A* parsing searches for an optimal parse, without building a complete chart (in contrast to CKY parsing). This is particularly attractive for CCG, because the formalism's lexical and derivational ambiguity causes the parse charts to be very dense, in practice. Lewis and Steedman (2014a) showed that A* CCG parsing can be highly efficient, but used a restricted model without bi-lexical features.

In A* parsing, entries y are added to the chart in order of their cost $f(y) = g(y) + h(y)$, where $g(y)$ is the inside score for the entry, and $h(y)$ is an upper bound on the Viterbi outside score. Because partial parses are built in order of increasing $f(y)$, the first complete parse added to the chart is guaranteed to be optimal. The key to making A* parsing efficient is computing tight upper bounds on the outside score. In Lewis and Steedman's supertag-factored model, the bound can be computed as the sum of the highest-scoring supertags in the outside parse.

The agenda is initialized with items representing every category for every word. Then, after each item is added to the chart, the agenda is updated with all binary and unary rules that can be applied to the new item. For more details, see Lewis and Steedman (2014a).

3 Model

Our joint model of CCG and SRL parsing simply involves labelling CCG syntactic dependencies (which are implicit in CCG parses), with SRL roles (or null). This formulation allows us to easily adapt the log-linear CCG parsing model of Clark and Curran (2007) to the joint setting by working with extended dependencies that including syntactic and semantic information.

More formally, we can define a notion of consistency to specify the labelling of syntactic dependencies. A set of semantic dependencies π is consistent with a CCG derivation d if each semantic dependency corresponds to a single syntactic dependency—for example, see the dependencies in Figure 3. The CCG derivation in Figure 3 would also be consistent with the SRL dependency $refuse \xrightarrow{ARG2} he$, but not $refuse \xrightarrow{ARG1} reports$ (because the derivation does not produce the required syntactic dependency).

Now, we can define a log-linear model over pairs of consistent CCG derivations d and SRL dependencies π:

$$p(d, \pi|x) = \frac{e^{\theta \cdot \phi(x,d,\pi)}}{\sum_{(d',\pi') \in GEN(x)} e^{\theta \cdot \phi(x,d',\pi')}}$$

where $GEN(x)$ is the space of consistent pairs for sentence x.

3.1 Dependencies

Because we enforce consistency, we can work with joint dependencies that are a combination of CCG and SRL dependencies. We denote a dependency as a 6-tuple of functor f, lexical category c, argument number n, preposition p, argument a and semantic role r: $\langle f, c, n, p, a, r \rangle$. The first three of these (f,c,n) are lexically specified, but a and r are lexically underspecified until the attachment is determined by the derivation, and a label is chosen by the model. For example, in Figure 3, the lexical entry for *deny* has two underspecified dependencies: $\langle deny, (S\backslash\mathbf{NP})/NP, 1, \varnothing, ?, ? \rangle$ and $\langle deny, (S\backslash NP)/\mathbf{NP}, 2, \varnothing, ?, ? \rangle$.[2] At the end of the derivation, the dependencies are specified as: $\langle deny, (S\backslash\mathbf{NP})/NP, 1, \varnothing, he, ARG0 \rangle$ and $\langle deny, (S\backslash NP)/\mathbf{NP}, 2, \varnothing, reports, ARG1 \rangle$.

The preposition p is lexically underspecified for PP arguments, but otherwise $\varnothing$. Prepositions are marked lexically on $PP/*$ categories, and then propagated, for example:

$$\dfrac{\dfrac{\dfrac{\begin{array}{c}\text{fly}\\ (S_{fly}\backslash NP_x)/PP_y^p\\ \{fly \xrightarrow{?} x, fly \xrightarrow{?} y\}\end{array} \quad \dfrac{\begin{array}{c}\text{to}\\ PP_z^{to}/NP_z\\ \{to \xrightarrow{?} z\}\end{array} \quad \begin{array}{c}\text{Lisbon}\\ NP_{Lisbon}\\ \{\}\end{array}}{PP^{to}_{Lisbon}\ \{to \xrightarrow{\varnothing} Lisbon\}}<}{S_{fly}\backslash NP_x\ \{fly \xrightarrow{?} x, fly \xrightarrow{ARG1} Lisbon, to \xrightarrow{\varnothing} Lisbon\}}<$$

In the above example, the dependency from *fly* to *Lisbon* corresponds to the tuple: $\langle fly, (S\backslash NP)/\mathbf{PP}, 2, to, Lisbon, ARG1 \rangle$. Propagating both the noun and preposition from prepositional phrases allows the model to use both for features, and improved results.

3.2 Features

We use the following feature classes, which decompose over categories, dependencies, and local rule instantiations.

[2] Bold-face is used to highlight the argument of a category corresponding to a dependency. For example $(S\backslash\mathbf{NP})/NP$ denotes the first (subject) dependency of a transitive verb.

3.2.1 Supertagging Features

Supertagging features ϕ^{CAT} score categories for words. A single feature is used, which is the (unnormalized) score from Lewis and Steedman (2014b)'s supertagging model. The supertagger outputs a distribution over categories for each word independently. The model is trained on supertags extracted from an adaptation of CCGrebank (Honnibal et al., 2010). The adaptation makes minor changes to better match PropBank.

3.2.2 Preposition Features

Preposition features ϕ^{PP} score whether the nth argument of the word at index f with category c should take a PP argument headed by preposition p. We use features $x_f{+}p$, $l_f{+}c$ and $l_f{+}c{+}n{+}p$, where x_f is the fth word, and l_f is its lemma.

3.2.3 Labelling Features

Labelling features ϕ^{ROLE} determine whether the nth argument of a word with lemma l, and category c should take a role r. We use $n{+}r$, $c{+}n{+}r$, $c{+}n{+}p{+}r$, $l{+}c{+}n{+}p{+}r$, $n{+}r$, r, $l{+}p{+}r$, $c{+}n{+}r$, $l{+}r$, $h{+}r$, where h is an indicator of whether l is hyphenated, and p is the preposition of PP arguments.

3.2.4 Dependency Features

Dependency features ϕ^{DEP} score dependencies, based on the functor index f, argument index a and role r. We use $l_f{+}r{+}l_a$, $l_f{+}r{+}c_a$, $d{+}r$, $c_{f+o}{+}r$, $c_{a+o}{+}r$, where o is an offset from $-3 \ldots +3$, d is the distance between f and a, c_i is a cluster or POS tag for word i, and l_i is the lemma of word i. Clusters were created from pre-trained word embeddings (Levy and Goldberg (2014)) using k-means, with $k = 20, 50, 200, 1000, 2500$. These features only apply when the role $r \neq \varnothing$.

3.2.5 Derivation Features

Derivation features ϕ^{DERIV} score properties of the syntactic derivation. To simplify computation of the upper bounds for the A* parsing algorithm given in Section 5, the weights of these features are constrained to be ≤ 0. For simplicity, we only use features for unary rules—for example, a feature records when the unary rule $N \rightarrow NP$ converts a determiner-less N to a NP.

4 Lexical Factorization

In this section, we show how to factor the model into *extended lexical entries*. This factorization allows us to efficiently compute upper bounds on the scores of partial parses, which is crucial to the A* algorithm described in Section 5.

The key observation is that our supertagging, labelling and dependency features can each be associated with exactly one word (using the functor for the dependency features). Therefore, we can equivalently view the complete parse as a series of *extended lexical entries*: $y = y_0 ... y_N$. Extended lexical entries are composed of a lexical category, and a role and attachment for each argument of the category. The set of extended lexical entries specifies the yield of the parse. For example, the parse from Figure 2 can be represented as the following extended lexical entries:

he	$\vdash NP$
refused	$\vdash (S\backslash NP_{ARG0=he})/(S\backslash NP)_{ARG1=confirm}$
to	$\vdash S/S_{\varnothing=confirm}$
confirm	$\vdash (S\backslash NP_{ARG0=he})/NP_{ARG1=reports}$
or	$\vdash conj$
deny	$\vdash (S\backslash NP_{ARG0=he})/NP_{ARG1=reports}$
reports	$\vdash NP$

The score for a sentence x and joint SRL-CCG parse y, $\theta \cdot \phi(x, y)$, can be decomposed into scores of its extended lexical entries y_i, plus a score for the derivation features:

$$\theta \cdot \phi(x, y) = \sum_{y_i \in y} \theta \cdot \phi(x, y_i) + \theta \cdot \phi^{DERIV}(x, y)$$

The space of possible extended lexical entries for word x_i is defined by $GENLEX(x_i)$, which is expressed with a CFG, as shown in Figure 4a.

The features defined in Section 3.2 decompose over the rules of $GENLEX$—so it can be weighted with the globally trained feature weights.

Expressing $GENLEX(x_i)$ as a CFG allows us to efficiently compute upper bounds on the score of y_i when it is only partially specified, using the Viterbi algorithm—which we will make use of in Section 5. Making the attachment choice independent of the syntactic category greatly improves the efficiency of these calculations.

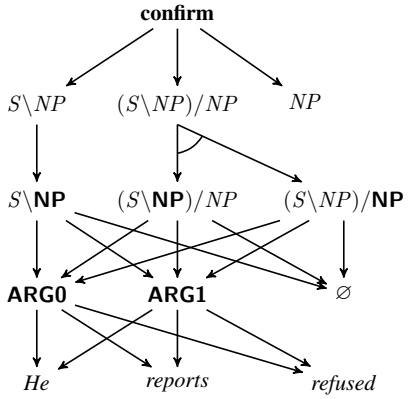
5 A* Parsing

To efficiently decode our model, we introduce a new A* parsing algorithm. As discussed in Section 2.2, the key to efficient A* parsing is computing tight upper bounds on the Viterbi outside score of partial parses, with the function h.

The intuition behind our algorithm is that we can use the lexical factorization of Section 4 to compute upper bounds for words individually,

Lexical Category Choice	
x_i	$\rightarrow NP \mid S\backslash NP \mid (S\backslash NP)/PP \mid \ldots$
One dependency is created for every argument of the category	
$S\backslash NP$	$\rightarrow S\backslash \mathbf{NP}$
$(S\backslash NP)/PP$	$\rightarrow (S\backslash \mathbf{NP})/PP,\ \ (S\backslash NP)/\mathbf{PP}$
$\ldots$	
Preposition choice for PP arguments	
$(S\backslash NP)/\mathbf{PP}$	$\rightarrow (S\backslash NP)/\mathbf{PP^{in}} \mid (S\backslash NP)/\mathbf{PP^{for}} \mid \ldots$
$\ldots$	
Semantic role label choice for the argument	
$S\backslash \mathbf{NP}$	$\rightarrow S\backslash \mathbf{NP_{ARG0}} \mid S\backslash \mathbf{NP_{ARG1}} \mid \ldots$
$(S\backslash \mathbf{NP})/PP$	$\rightarrow (S\backslash \mathbf{NP_{ARG0}})/PP \mid (S\backslash \mathbf{NP_{ARG1}})/PP \ldots$
$(S\backslash NP)/\mathbf{PP^X}$	$\rightarrow (S\backslash NP)/\mathbf{PP^X_{ARG0}} \mid (S\backslash NP)/\mathbf{PP^X_{ARG1}} \cdots$
$\ldots$	
Attachment choice	
$ARG0$	$\rightarrow x_0 \mid \ldots \mid x_{i-1} \mid x_{i+1} \mid \ldots \mid x_N$
$ARG1$	$\rightarrow x_0 \mid \ldots \mid x_{i-1} \mid x_{i+1} \mid \ldots \mid x_N$
$\ldots$	

(a) The grammar $GENLEX(x_i)$



(b) Visualization of a fragment of $GENLEX(confirm)$

Figure 4: (a) The grammar $GENLEX(x_i)$, which defines the space of extended lexical entries, and (b) a visualization of a fragment of $GENLEX(confirm)$. Extended lexical entries, including **confirm** $\vdash (S\backslash NP_{ARG0=he})/NP_{ARG1=reports}$ and **confirm** $\vdash NP$, are specified by choosing one category (top level in both a and b), enumerating all arguments (second level), selecting the preposition for PP arguments (when present), selecting a semantic role label for each, and finally choosing the argument head word. The features are local to the grammar rules, enabling efficient dynamic programs for upper bound computations on partially specified entries, such as **confirm** $\vdash (S\backslash NP_{r=a})/NP_{ARG1=reports}$.

then create an upper bound for the parse as a sum of upper bounds for words. The bound is not exact, because the grammar may not allow the combination of the best lexical entry for each word.

Section 5.1 gives a declarative definition of h for any partial parse, and 5.2 explains how to efficiently compute h during parsing.

5.1 Upper Bounds for Partial Parses

This section defines the upper bound on the Viterbi outside score $h(y_{i,j})$ for any partial parse $y_{i,j}$ of span $i \ldots j$. For example, in the parse in Figure 3, $y_{3,5}$ is the partial parse of *confirm or deny* with category $(S\backslash NP)/NP$.

As explained in Section 4, a parse can be decomposed into a series of extended lexical entries. Similarly, a partial parse can be viewed as a series of *partially specified* extended lexical entries $y_0 \ldots y_N$. For example, in Figure 3, the partial parse of the span *confirm or deny reports*, the extended lexical entries for the words outside the span (*He*, *refused* and *to*) are completely unspecified. The extended lexical entries for words inside the span have specified categories, but can contain underspecified dependencies:

confirm	$\vdash (S\backslash NP_{r=a})/NP_{ARG1=reports}$
or	$\vdash conj$
deny	$\vdash (S\backslash NP_{r'=a'})/NP_{ARG1=reports}$
reports	$\vdash NP$

Therefore, we can compute an upper bound for the outside score of a partial parse as a sum of the upper bounds of the unspecified components of each extended lexical entry. Note that because the derivation features are constrained to be ≤ 0, they do not affect the calculation of the upper bound.

We can then find an upper bound for completely unspecified spans using by summing the upper bounds for the words. We can pre-compute an upper bound for the span $h_{i,j}$ for every span i, j as:

$$h_{i,j} = \sum_{k=i}^{j} \max_{y_k \in GENLEX(x_k)} \theta \cdot \phi(x, y_k)$$

The max can be efficiently computed using the Viterbi algorithm on $GENLEX(x_k)$ (as described in Section 4).

The upper bound on the outside score of a partial parse is then the sum of the upper bounds of the words outside the parse, and the sum of the scores of the best possible specifications for each underspecified dependency:

$$h(y_{i,j}) = \sum_{\langle f,c,n,?,?,? \rangle \in \mathbf{deps}(y_{i,j})} \max_{a',p',r'} \theta \cdot \phi(\langle f, c, n, p', a', r' \rangle) + h_{0,i-1} + h_{j+1,N}$$

where $\mathbf{deps}(y)$ returns the underspecified dependencies from partial parse.

For example, in Figure 3, the upper bound for the outside score of the partial parse of *confirm or deny reports* is the sum of the upper bounds

for the other words independently ($h_{0,3}$) added to the score of the best attachments and roles for the subjects of *confirm* and *deny* independently.

5.2 Additive Updates

During parsing, the upper bound h can be efficiently computed recursively with additive updates. Initially h is the sum of the upper bounds for each word independently $h_{0,N}$. Then, when specifying categories or labelling dependencies, the score is updated.

- When specifying a category for a word x_i, the Viterbi score of $GENLEX(x_i)$ is subtracted from h, and the sum of Viterbi scores for each of the category's (underspecified) dependencies is added to h.
- When specifying a semantic role for a dependency with functor f, causing the dependency to be fully specified, the Viterbi score for the node representing that dependency in $GENLEX(x_f)$ is subtracted from h.
- When a binary rule combines two partial parses $y_{i,j}$ and $y_{j+1,k}$, the bound on the outside score is updated by summing the bounds of the outside scores of the child parses, and subtracting the overall upper bound for the sentence (to avoid double-counting):
  $h(y_{i,k}) = h(y_{i,j}) + h(y_{j+1,k}) - h_{0,N}$
  This update can be derived from the definition of h.

These are the only cases where h is updated.

6 Training

During training, we optimize parameters θ for the marginal likelihood of the semantic dependencies π_i given a sentence x_i, treating syntactic derivations d as a latent variable and using L_2 regularization.

$$L(\theta) = \log \prod_i p_\theta(\pi_i|x_i) - \frac{\sum_j \theta_j^2}{2\sigma^2}$$
$$= \sum_i \log \frac{\sum_{d \in \Delta(x_i,\pi_i)} e^{\theta \cdot \phi(x_i,d,\pi_i)}}{\sum_{(d',\pi') \in GEN(x_i)} e^{\theta \cdot \phi(x_i,d',\pi')}} - \frac{\sum_j \theta_j^2}{2\sigma^2}$$

where $GEN(x_i)$ is the set of consistent CCG and SRL parses for a sentence x_i (see Section 3), and $\Delta(x_i, \pi_i)$ is the set of CCG derivations that are maximally consistent with gold SRL parses π_i. More formally, if $\textbf{labelled}(y)$ returns the set of labelled dependencies from parse y, then:

$$\Delta(x, \pi) = \arg\max_{y \in GEN(x)} |\pi \cap \textbf{labelled}(y)|$$

The arguments of PropBank dependencies can span multiple words, so CCG dependencies are marked as equivalent if their argument is anywhere within the PropBank span.

The approach is closely related to the hybrid dependency model (Clark and Curran, 2007). However the CCGbank dependencies used by Clark and Curran's model constrain all lexical and attachment decisions (only allowing 'spurious' derivational ambiguity) whereas our use of semantic dependencies models most of the syntactic parse as latent.

6.1 Hyperparameters

Calculating the gradient of the loss function requires computing the marginal probability of the correct parses. Computing marginal probabilities exactly involves summing over all possible CCG parses, which is intractable. Instead, following Clark and Curran, we limit the size of training charts using a variable-width supertagger beam β, initially 10^{-3}. If the chart size exceeds 100000 nodes, we double β and re-parse. For computing Δ, we use a more restrictive beam of $\beta = 0.01$ to improve the quality of the positive training examples. We optimize using L-BFGS (Liu and Nocedal, 1989), with $\sigma^2 = 0.06$.

6.2 Pruning

To improve efficiency, we compute a number of thresholds, by aligning gold CCGbank dependencies with PropBank. If an argument of a category occurs with a particular semantic role less than 3 times in the aligned data, it is pruned from the training and decoding charts. We also filter infrequent features before training. We count the number of times each feature occurs in the aligned data, and filter features occurring less than 3 times. During decoding, we prune lexical categories whose probability under the supertagging model is less than a factor of 10^{-2} of that of the best category for that word. If the chart size exceeds 20000 nodes, we back off to a pipeline model (roughly 3% of sentences). Finally, we build and use a tag dictionary in the same way as Lewis and Steedman (2014a).

7 Experiments

7.1 Experimental Setup

We used PropBank Section 00 for development, Sections 02-21 for training, and Section 23 for testing. The **Pipeline** baseline first parses with a supertag-factored A* model, and chooses a semantic role for each CCG dependency with a log-linear classifier. The classifier uses the role and attachment features used by the parser.

7.2 Semantic Role Labelling

We evaluate our parser as a dependency-based SRL model on PropBank, comparing with CoNLL-2008 systems (Surdeanu et al., 2008).

Comparison systems Following Täckström et al. (2015), we compare with the best 'single parser' SRL models, which use only a single syntactic parse.[3]

Much recent work has evaluated using gold predicate identification (Hajič et al., 2009; FitzGerald et al., 2015). This setting is particularly unrealistic for our joint model, where gold predicate identification would be a highly useful feature for the supertagger; we only compare with models that use automatic predicate identification. To the best of our knowledge, the best SRL results with automatic predicate identification were achieved at CoNLL-2008.

A number of other models have been evaluated on CoNLL-2008 data. While we cannot compare directly on our metric, the best reported joint model (Johansson, 2009) scored 1.4 points lower than the Che et al. (2008) system we compare to on the CoNLL metric on PropBank. Other joint models, such as those of Titov et al. (2009) and Lluís et al. (2013), achieve similar performance.

Evaluation Metric Comparing fairly with existing work is complicated by the mismatch between heads found by our model and those used in other evaluations. Headedness decisions are often arbitrary—for example, whether a prepositional phrase is headed by the noun or preposition—and different choices were made in the design of CCGbank and the CoNLL-2008 headedness rules.

To solve this problem, we introduce a new *within-constituent* metric, which awards dependencies as correct if they attach anywhere within the original PropBank-annotated argument spans. For example, if the PropBank annotates that the *ARG*0 of *owned* is *by Google*, a dependency to either *by* or *Google* is judged correct. We compute new scores for the CoNLL-2008 submissions on our metric, filtering reference and continuation arguments (which are artifacts of the CoNLL conversion of PropBank, but not required by our metric), and nominal predicates based on POS tag. The ranking of the top 5 CoNLL-2008 open-track models is identical under our metric and the original one (up to statistical significance), suggesting that our metric is equally discriminative. However, perhaps interestingly, the ranking of Vickrey and Koller (2008) does improve—likely due to the use of a syntactic formalism with different headedness rules. For simplicity, we do not include predicate senses in the evaluation.

[3]The best models use reranking with powerful global features (Toutanova et al., 2008; Johansson and Nugues, 2008) or ensemble methods (Surdeanu et al., 2007; Punyakanok et al., 2008). These techniques have the potential to improve any SRL system, including ours, at some expense in speed.

	PropBank			Brown		
Model	P	R	F1	P	R	F1
Vickrey	**87.3**	77.3	82.0	**74.0**	64.5	68.9
Che	85.3	78.6	81.8	71.1	65.7	68.0
Zhao	82.4	79.8	81.1	66.6	64.9	65.7
Riedel	83.6	74.7	78.9	69.3	62.7	65.8
Pipeline	79.2	73.9	76.4	69.3	64.0	66.1
Joint	84.8	**82.2**	**83.5**	71.2	**69.2**	**70.2**

Table 1: Comparison with the best single-parser SRL models on PropBank from CoNLL-2008. The comparison models are Vickrey and Koller (2008), Che et al. (2008), Zhao and Kit (2008) and Riedel and Meza-Ruiz (2008).

Results are given in Table 1 and show that our joint model greatly outperforms the pipeline version, demonstrating the value of joint reasoning. It also scores 1.5 points higher than the best comparable models in-domain, and 1.3 points higher out-of-domain. To the best of our knowledge, this is the first joint syntax/SRL model to outperform strong pipeline models.

7.3 Efficiency Experiments

We explore whether our A* decoding is more efficient than alternatives, including both algorithmic and feature computation. While the A* search builds very small charts, the features must be precomputed for the heuristic. In CKY parsing, features can be computed lazily.

Model	Sentences per second	PropBank F1
Pipeline A*	**38.3**	76.4
Joint CKY	6.0	**83.5**
Joint AST	20.3	83.0
Joint A*	31.3	**83.5**

Table 2: Parser speed on PropBank Section 23

We compare with a CKY parsing over the same space. If no parse is found, or the chart size exceeds 400000 nodes, we back off to the pipeline (tuned so that the backoff is used roughly as often as for the A* parser). We also compare with adaptive supertagging (AST, Clark and Curran (2007)), which is the same except for first attempting to parse with a restrictive supertagger beam $\beta = 0.1$.

Table 2 shows the results. The A* pipeline is fast, but inaccurate. CKY is 5 times slower than A* in the same space, whereas adaptive supertagging trades accuracy for speed. The best previously reported speed improvement using A* parsing is a factor of 1.2 times faster (Auli and Lopez, 2011b). Our new A* algorithm dominates existing alternatives in both speed and accuracy.

7.4 Syntactic Parsing

We also evaluate our model for CCG parsing accuracy, using CCGrebank (Honnibal et al., 2010), and comparing with a C&C parser model adapted to this dataset. Results are shown in Table 3. Of course, given our latent syntax, and the fact we have no model of CCGbank dependencies, we do not expect state-of-the-art accuracy. However, the 1.6 point improvement over the pipeline shows that SRL dependencies are useful for disambiguating syntactic attachment decisions. Many errors were caused by disagreements between CCGbank and PropBank—PropBank is likely to be more accurate as it was hand-annotated, rather than automatically converted from the Penn Treebank. In effect, our latent model of syntax is successfully learning a grammar that better produces the correct semantics.

Table 4 shows the syntactic dependencies which are improved most by modelling semantics. Unsurprisingly, verb arguments and adjuncts relations show large improvements. However, we also see more accurate attachment of relative clauses. While we do not model relative clauses explicitly, correctly attaching them is essential for propagating certain verb arguments (e.g. the subject of *broke* in *the glass on the table that broke*).

Model	P	R	F1
C&C	81.8	81.2	81.5
Pipeline	76.6	77.7	77.2
Joint	78.3	79.4	78.8

Table 3: Labelled F1 for CCGbank dependencies on CCGrebank Section 23

Dependency	Type	Δ F1
$((S_{dcl}\backslash NP)/\mathbf{PP})/NP$		+12.6
$((S_{dcl}\backslash NP)/PP)/\mathbf{NP}$		+11.0
$((S_b\backslash NP)/\mathbf{PP})/NP$	Verb	+10.7
$(S_{dcl}\backslash NP)/\mathbf{PP}$	arguments	+8.1
$(S_{ng}\backslash \mathbf{NP})/NP$		+7.6
$(S_{pt}\backslash \mathbf{NP})/NP$		+6.8
$((S\backslash NP)\backslash(\mathbf{S}\backslash\mathbf{NP}))/NP$	Verb	+16.9
$((S\backslash NP)\backslash(\mathbf{S}\backslash\mathbf{NP}))/S_{dcl}$	adjuncts	+11.9
$(N\backslash\mathbf{N})/(S_{dcl}\backslash NP)$	Relative	+12.5
$(NP\backslash\mathbf{NP})/(S_{dcl}\backslash NP)$	clauses	+8.5

Table 4: CCG syntactic dependencies with the largest change in F1 between the Pipeline and Joint models (of those occurring at least 100 times in the development set).

7.5 Error Analysis

Table 5 gives an analysis of recall errors from the first 100 sentences of PropBank Section 00. One third of errors were syntactic attachment errors. A further 14% were triggered by cases where the parser found the correct attachment, but gave an adjunct a core argument CCG category (or vice versa). Many of these decisions are not obvious—for example in *rose sharply*, PropBank considers *sharply* to be an argument and not an adverb. 21% of errors were caused by choosing the wrong SRL

Error	Percentage
Attachment error	33%
Correct attachment, wrong label	21%
Correct attachment, unlabelled	20%
Argument/adjunct distinction	14%
Problematic constructions	9%
Dubious annotation	4%

Table 5: Error analysis of recall errors from 100 development set sentences.

label. Another 20% were caused by predicates assigning arguments the null semantic role ∅. The major cause of these errors is predicates that act syntactically like adjectives (e.g. *publishing* in *Dutch publishing group*) where syntactic cues are weak. Finally, 9% involved long-range arguments that our current grammar is unable to project. One common case is constructions like *X has a plan to buy Y*, where the grammar does not propagate the subject of *buy* to *X*. Further improvements to CCGbank may help to resolve these cases.

8 Related Work

Joint syntactic and SRL models There have been many proposals for jointly parsing syntactic and semantic dependencies. Lluís et al. (2013) introduce a joint arc-factored model for parsing syntactic and semantic dependencies, using dual-decomposition to maximize agreement between the models. SRL performance is slightly worse than a pipeline version. Naradowsky et al. (2012) introduce a SRL model with latent syntax representations, by modelling a latent dependency tree during training, which is marginalized out at test time. However, performance at English SRL is roughly 7 points beneath state of the art. Other notable models include those of Johansson (2009) and Titov et al. (2009).

CCG parsing Our log-linear model is closely related to that of Clark and Curran (2007), but we model SRL dependencies instead of CCG dependencies. The best CCG parsing results were achieved by Auli and Lopez (2011a), who, like us, score CCG parses based jointly on supertagging and dependency model scores. Decoding their model requires dual-decomposition, to maximize agreement between the separate models. We avoid the need for this technique by using a unigram supertagging model, rather than a sequence model.

CCG semantics Work on semantic parsing has mapped sentences onto semantic representations with latent CCGs (Zettlemoyer and Collins, 2009; Kwiatkowski et al., 2010; Kwiatkowski et al., 2013) for restricted domains. Recent work has scaled these techniques to wide-coverage datasets (Artzi et al., 2015). Krishnamurthy and Mitchell (2014) also explore joint CCG syntactic and semantic parsing. They use a smaller semantic lexicon, containing 130 predicates, rather than the 3257 PropBank verbs. In contrast to our results, jointly modelling the semantics lowers their model's syntactic accuracy.

Other CCG-based SRL models haved used CCG dependencies as features for predicting semantic roles (Gildea and Hockenmaier, 2003; Boxwell et al., 2009), but performance is limited by relying on 1-best parses—a problem we resolved with a joint model.

A* parsing A* parsing has previously been explored for less general models than ours. Klein and Manning (2003) and Auli and Lopez (2011b) use A* parsing for models with tree-structured dependencies. The best reported speed improvement is parsing 1.2 times faster, whereas we improve by a factor of 5. Our model also allows the more complex graph-structured dependencies required for semantic role labelling. Lewis and Steedman (2014a) demonstrate an efficient A* algorithm for CCG, but cannot model dependencies.

9 Conclusions and Future Work

We have shown that using CCG can allow joint models of syntax and semantics to outperform pipelines, and achieve state-of-the-art results on PropBank SRL. Our new A* parsing algorithm is 5 times faster than CKY parsing, without loss of accuracy. Using latent syntax allows us to train the model purely from semantic dependencies, enabling future work to train against other annotations such as FrameNet (Baker et al., 1998), Ontonotes (Hovy et al., 2006) or QA-SRL (He et al., 2015). The semantic labels provided by PropBank can also be integrated into wide-coverage CCG semantic parsers (Bos, 2008; Lewis and Steedman, 2013) to improve performance on downstream applications.

Acknowledgements

This research was supported in part by the NSF (IIS-1252835), DARPA under the DEFT program through the AFRL (FA8750-13-2-0019), an Allen Distinguished Investigator Award, and a gift from Google. We thank Nicholas Fitzgerald, Dan Garrette, Kenton Lee, Swabha Swayamdipta, Mark Yatskar and the anonymous reviewers for their helpful comments on earlier versions of this paper, and Matthew Honnibal for useful discussions.

References

Yoav Artzi, Kenton Lee, and Luke Zettlemoyer. 2015. Broad-coverage CCG Semantic Parsing with AMR. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Michael Auli and Adam Lopez. 2011a. A Comparison of Loopy Belief Propagation and Dual Decomposition for Integrated CCG Supertagging and Parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*.

Michael Auli and Adam Lopez. 2011b. Efficient CCG parsing: A* versus Adaptive Supertagging. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*.

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet Project. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics - Volume 1*.

Johan Bos. 2008. Wide-coverage Semantic Analysis with Boxer. In Johan Bos and Rodolfo Delmonte, editors, *Semantics in Text Processing. STEP 2008 Conference Proceedings*, Research in Computational Semantics.

Stephen A. Boxwell, Dennis Mehay, and Chris Brew. 2009. Brutus: A Semantic Role Labeling System Incorporating CCG, CFG, and Dependency Features. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1*.

Wanxiang Che, Zhenghua Li, Yuxuan Hu, Yongqiang Li, Bing Qin, Ting Liu, and Sheng Li. 2008. A cascaded Syntactic and Semantic Dependency Parsing System. In *Proceedings of the Twelfth Conference on Computational Natural Language Learning*.

Stephen Clark and James R Curran. 2007. Wide-coverage Efficient Statistical Parsing with CCG and Log-Linear Models. *Computational Linguistics*, 33(4).

Nicholas FitzGerald, Oscar Täckström, Kuzman Ganchev, and Dipanjan Das. 2015. Semantic Role Labeling with Neural Network Factors. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Daniel Gildea and Julia Hockenmaier. 2003. Identifying Semantic Roles Using Combinatory Categorial Grammar. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*.

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, Pavel Straňák, Mihai Surdeanu, Nianwen Xue, and Yi Zhang. 2009. The CoNLL-2009 Shared Task: Syntactic and Semantic Dependencies in Multiple Languages. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning: Shared Task*.

Luheng He, Mike Lewis, and Luke Zettlemoyer. 2015. Question-answer driven semantic role labeling: Using natural language to annotate natural language. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

James Henderson, Paola Merlo, Ivan Titov, and Gabriele Musillo. 2013. Multi-lingual Joint Parsing of Syntactic and Semantic Dependencies with a Latent Variable Model. *Computational Linguistics*.

Julia Hockenmaier. 2003. *Data and models for statistical parsing with Combinatory Categorial Grammar*. Ph.D. thesis, University of Edinburgh. College of Science and Engineering. School of Informatics.

M. Honnibal, J.R. Curran, and J. Bos. 2010. Rebanking CCGbank for Improved NP Interpretation. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*.

Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. OntoNotes: The 90% Solution. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*.

Richard Johansson and Pierre Nugues. 2008. Dependency-based Syntactic-Semantic Analysis with Propbank and Nombank. In *Proceedings of the Twelfth Conference on Computational Natural Language Learning*.

Richard Johansson. 2009. Statistical Bistratal Dependency Parsing. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2*.

Dan Klein and Christopher D Manning. 2003. A* Parsing: Fast Exact Viterbi Parse Selection. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*.

Jayant Krishnamurthy and Tom M. Mitchell. 2014. Joint Syntactic and Semantic Parsing with Combinatory Categorial Grammar. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*.

Tom Kwiatkowski, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2010. Inducing Probabilistic CCG Grammars from Logical Form with Higher-order Unification. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1223–1233.

Tom Kwiatkowski, Eunsol Choi, Yoav Artzi, and Luke Zettlemoyer. 2013. Scaling Semantic Parsers with On-the-Fly Ontology Matching. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*.

Omer Levy and Yoav Goldberg. 2014. Dependency-Based Word Embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*.

Mike Lewis and Mark Steedman. 2013. Combined Distributional and Logical Semantics. *Transactions of the Association for Computational Linguistics*, 1.

Mike Lewis and Mark Steedman. 2014a. A* CCG Parsing with a Supertag-factored Model. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*.

Mike Lewis and Mark Steedman. 2014b. Improved CCG parsing with Semi-supervised Supertagging. *Transactions of the Association for Computational Linguistics*.

D. C. Liu and J. Nocedal. 1989. On the Limited Memory BFGS Method for Large Scale Optimization. *Math. Program.*, 45(3).

Xavier Lluís, Stefan Bott, and Lluís Màrquez. 2009. A Second-order Joint Eisner Model for Syntactic and Semantic Dependency Parsing. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning: Shared Task*.

Xavier Lluís, Xavier Carreras, and Lluís Màrquez. 2013. Joint Arc-factored Parsing of Syntactic and Semantic Dependencies. *Transactions of the Association of Computational Linguistics – Volume 1*, pages 219–230.

Jason Naradowsky, Sebastian Riedel, and David A. Smith. 2012. Improving NLP Through Marginalization of Hidden Syntactic Structure. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*.

Vasin Punyakanok, Dan Roth, and Wen-tau Yih. 2008. The Importance of Syntactic Parsing and Inference in Semantic Role Labeling. *Computational Linguistics*, 34(2).

Sebastian Riedel and Ivan Meza-Ruiz. 2008. Collective Semantic Role Labelling with Markov Logic. In *Proceedings of the Twelfth Conference on Computational Natural Language Learning*.

Mihai Surdeanu, Lluís Màrquez, Xavier Carreras, and Pere R. Comas. 2007. Combination Strategies for Semantic Role Labeling. *Journal of Artificial Intelligence Research*, 29(1).

Mihai Surdeanu, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. The CoNLL-2008 Shared Task on Joint Parsing of Syntactic and Semantic Dependencies. In *Proceedings of the Twelfth Conference on Computational Natural Language Learning*.

Charles Sutton and Andrew McCallum. 2005. Joint Parsing and Semantic Role Labeling. In *Proceedings of the Ninth Conference on Computational Natural Language Learning*.

Oscar Täckström, Kuzman Ganchev, and Dipanjan Das. 2015. Efficient Inference and Structured Learning for Semantic Role Labeling. *Transactions of the Association for Computational Linguistics*, 3:29–41.

Ivan Titov, James Henderson, Paola Merlo, and Gabriele Musillo. 2009. Online Projectivisation for Synchronous Parsing of Semantic and Syntactic Dependencies. In *In Proceedings of the International Joint Conference on Artificial Intelligence*.

Kristina Toutanova, Aria Haghighi, and Christopher D. Manning. 2008. A Global Joint Model for Semantic Role Labeling. *Computational Linguistics*, 34(2).

David Vickrey and Daphne Koller. 2008. Applying Sentence Simplification to the CoNLL-2008 Shared Task. In *Proceedings of the Twelfth Conference on Computational Natural Language Learning*.

Luke Zettlemoyer and Michael Collins. 2009. Learning Context-dependent Mappings from Sentences to Logical Form. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2*.

Hai Zhao and Chunyu Kit. 2008. Parsing Syntactic and Semantic Dependencies with Two Single-stage Maximum Entropy Models. In *Proceedings of the Twelfth Conference on Computational Natural Language Learning*.

Improving Semantic Parsing with Enriched Synchronous Context-Free Grammar

Junhui Li[1] **Muhua Zhu**[2] **Wei Lu**[3] **Guodong Zhou**[1]
[1]Natural Language Processing Lab, Soochow University, China
{lijunhui, gdzhou}@suda.edu.cn
[2] Alibaba Inc., Hangzhou, China
muhua.zmh@alibaba-inc.com
[3] Information Systems Technology and Design,
Singapore University of Technology and Design
luwei@sutd.edu.sg

Abstract

Semantic parsing maps a sentence in natural language into a structured meaning representation. Previous studies show that semantic parsing with synchronous context-free grammars (SCFGs) achieves favorable performance over most other alternatives. Motivated by the observation that the performance of semantic parsing with SCFGs is closely tied to the translation rules, this paper explores extending translation rules with high quality and increased coverage in three ways. First, we introduce structure informed non-terminals, better guiding the parsing in favor of well formed structure, instead of using a uninformed non-terminal in SCFGs. Second, we examine the difference between word alignments for semantic parsing and statistical machine translation (SMT) to better adapt word alignment in SMT to semantic parsing. Finally, we address the unknown word translation issue via synthetic translation rules. Evaluation on the standard GeoQuery benchmark dataset shows that our approach achieves the state-of-the-art across various languages, including English, German and Greek.

1 Introduction

Semantic parsing, the task of mapping natural language (NL) sentences into a formal meaning representation language (MRL), has recently received a significant amount of attention with various models proposed over the past few years. Consider the NL sentence paired with its corresponding MRL in Figure 1(a). Semantic parsing can be naturally viewed as a statistical machine translation (SMT) task, which translates a sentence in NL (i.e., the source language in SMT) into its meaning representation in MRL (i.e., the target language in SMT). Indeed, many attempts have been made to directly apply statistical machine translation (SMT) systems (or methodologies) to semantic parsing (Papineni et al., 1997; Macherey et al., 2001; Wong and Mooney, 2006; Andreas et al., 2013). However, although recent studies (Wong and Mooney, 2006; Andreas et al., 2013) show that semantic parsing with SCFGs, which form the basis of most existing statistical syntax-based translation models (Yamada and Knight, 2001; Chiang, 2007), achieves favorable results, this approach is still behind the most recent state-of-the-art. For details, please see performance comparison in Andreas et al. (2013) and Lu (2014).

NL: What is the area of Seattle
MRL: answer(area_1(cityid('seattle', _)))
(a) before pre-processing

NL': what be the area of seattle
MRL': answer@1 area_1@1 cityid@2 seattle@s _@0
(b) after pre-processing

Figure 1: Example of a sentence pair in NL and MRL.

The key issues behind the limited success of applying SMT systems directly to semantic parsing lie in the difference between semantic parsing and SMT: *MRL is not a real natural language with different properties from natural language*. First, MRL is machine-interpretable and thus strictly structured with the meaning representation in a nested structure of functions and arguments. Second, the two languages are intrinsically asymmetric since each token in MRL carries specific mean-

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1455–1465,
Lisbon, Portugal, 17-21 September 2015.

ing[1] while this does not hold in NL since auxiliary words and some function words usually have no counterparts in MRL. Third and finally, the expressions in NL are more flexible with respect to lexicon selection and token ordering. For example, since sentences in NL *'could you tell me the states that utah borders'*, *'what states does utah border'*, and *'utah borders what states'* convey the same meaning, they should have the same expression in MRL.

Motivated by the above observations, we believe that semantic parsing with *standard* SMT components is not an ideal approach. Alternatively, this paper proposes an effective, yet simple way to enrich SCFG in hierarchical phrase-based SMT for better semantic parsing. Specifically, since the translation rules play a critical role in SMT, we explore to improve translation rule quality and increase its coverage in three ways. First, we enrich non-terminal symbols as to capture contextual and structured information. The enrichment of non-terminal symbols not only guides the translation in favor of well formed structures, but also is beneficial to translation. Second, we examine the difference between word alignments for semantic parsing and SMT to better adapt word alignment in SMT to semantic parsing. Third, unlike most existing SMT systems that keep unknown words untranslated and intact in translation, we exploit the translation of unknown words via synthetic translation rules. Evaluation on GeoQuery benchmark dataset shows that our approach obtains consistent improvement and achieves the state-of-the-art across various languages, including English, German and Greek.

2 Background: Semantic Parsing as Statistical Machine Translation

In this section, we present the framework of semantic parsing as SMT, which was proposed in Andreas et al. (2013).

Pre-Processing Various semantic formalisms have been considered for semantic parsing. Examples include the variable-free semantic representations (that is, the meaning representation for each utterance is tree-shaped), the lambda calculus expressions, and dependency-based compositional semantic representations. In this work, we specifically focus on the variable-free semantic representations, as shown in Figure 1. On the target side, we convert these meaning representations to series of strings similar to NL. To do so, we simply take a preorder traversal of every functional form, and label every function with the number of arguments it takes. Figure 1(b) shows an example of converted meaning representation, where each token is in the format of $A@B$ where *A* is the symbol while *B* is either *s* indicating that the symbol is a string or a number indicating the symbol's arity (constants, including strings, are treated as zero-argument functions).

On the source side, we perform stemming (for English and German) and lowercasing to overcome data sparseness.

Hereafter, we refer to the pre-processed NL and MRL as NL$'$ and MRL$'$ respectively.

Translation Given a corpus of NL$'$ sentences paired with MRL$'$, we learn a semantic parser by adopting a string-to-string translation system. Typical components in such a translation system include word alignments between the source and the target languages, translation rule extraction, language model learning, parameter tuning and decoding. For more details about each component, please refer to (Chiang, 2007). In the rest of this paper, we refer to the source language (side) as NL$'$, and the target language (side) as MRL$'$.

Post-Processing We convert MRL$'$ back into MRL by recovering parentheses and commas to reconstruct the corresponding tree structure in MRL. This can be easily done by examining each symbol's arity. It eliminates any possible ambiguity from the tree reconstruction: given any sequence of tokens in MRL$'$, we can always reconstruct the tree structure (if one exists). For those translations that can not be successfully converted, we call them *ill-formed* translations.

3 Semantic Parsing with Enriched SCFG

In this section, we present the details of our enriched SCFG for semantic parsing.

3.1 Enriched SCFG

In hierarchical phrase-based (HPB) translation models, synchronous rules take the form $X \rightarrow \langle \gamma, \alpha, \sim \rangle$, where X is the non-terminal symbol, γ and α are strings of lexical items and non-terminals in the source and target side respectively, and $\sim$ indicates the one-to-one cor-

[1] As seen in Section 2, delimiters, including parentheses and commas which do not carry any meaning will be removed in pre-processing and be recovered in post-processing

respondence between non-terminals in γ and α. From an aligned phrase pair *<state that border, state@1 next_to_2@1>* in Figure 2(a), for example, we can get a synchronous rule $X \rightarrow \langle \text{state } X_{[1]}, \text{state@1 } X_{[1]} \rangle$, where we use boxed indices to indicate which nonterminal occurrences are linked by $\sim$. The fact that SCFGs in HPB models contain only one type of non-terminal symbol[2] is responsible for ill-formed translation (e.g., *answer@1 state@1*). To this end, we enrich the non-terminals to capture the tree structure information, guiding the translation in favor of well-formed translations. The enrichment of non-terminals is two-fold: first, it can handle MRL with a nested structure to guarantee the well-formed translations; second, related studies in SMT have shown that introducing multiple non-terminal symbols in SCFGs benefits translation (Zollmann and Venugopal, 2006; Li et al., 2012).

Given a word sequence e_j^i from position i to position j in MRL′, we enrich the non-terminal symbol X to reflect the internal structure of the word sequence of e_j^i . A correct translation rule selection therefore not only maps source terminals into target terminals, but is both constrained and guided by structure information in the non-terminals. As mentioned earlier, we regard the nested structure in MRL′ as function-argument structure, where each function takes one or more arguments as input while its return serves as an argument to the outside function. As in Figure 1, function *cityid* holds two arguments and returns as an argument to function *area_1*. For a word sequence e_j^i, we examine its completeness, which is defined as:

Definition 1. *For word sequence e_j^i, it is regarded as complete if it satisfies 1) every function (if exists) meets its argument requirement; and 2) it can serve as **one** argument to another function.*

We use symbol *C* to label word sequences which are complete. For an incomplete word sequence, we examine 1) the number of arguments it requires on the right to be complete; and 2) the arity of a function it requires on the left to be complete. Then the sequence is labeled as $(\text{C}\backslash\text{F}m)/An$, indicating it requires n arguments on the right and a function with m arities on the left. [3]

[2]In practice, non-terminal symbol S is used in glue rules. However, this is not relevant in the present discussion.

[3]This is similar to the naming convention in combinatory categorial grammar (CCG) (Steedman, 2000)

texas, stateid@1 texas@s *C*→ ⟨texas, stateid@1 texas@s⟩
seattle, seattle@s _@0 *C*\F2→ ⟨seattle, seattle@s _@0⟩
that border, next_to_2@1 *C/A1*→ ⟨that border, next_to_2@1⟩
state that border, state@1 next_to_2@1 *C/A1*→ ⟨state *C/A1*$_{[1]}$, state@1 *C/A1*$_{[1]}$⟩

(a) Examples of phrase pairs in enriched SCFG.

state that border , state@1 next_to_2@1 *C/A1*→ ⟨*C/A1*$_{[1]}$ *C/A1*$_{[2]}$, *C/A1*$_{[1]}$ *C/A1*$_{[2]}$⟩
state that border texas have the highest population, l..one@1 p..1@1 state@1 n..@1 s..@1 t..@s *C*→ ⟨*C*$_{[1]}$ *C/A1*$_{[2]}$, *C/A1*$_{[2]}$ *C*$_{[1]}$⟩

(b) Examples of glue rules in enriched SCFG.

Table 1: Example of translation rules in enriched SCFG, where underline and underwave indicate the first and the second phrases respectively.

Specifically, we omit $\backslash\text{F}m$ and $/An$ if $m = 0$ and $n = 0$ respectively.[4]

Table 1(a) demonstrates examples of phrase pairs in our enriched SCFG. For instance, word sequence *stateid@1 texas@s* is complete, and thus labeled as *C*. Similarly, to be complete, word sequence *next_to_2@1* requires one argument on the right side, labeled as $C/A1$ accordingly.

When extracting translation rules from aligned datasets, we follow Chiang (2007) except that we use enriched non-terminal symbols rather than X. Each translation rule is associated with a set of translation model features $\{\phi_i\}$, including phrase translation probability $p(\alpha \mid \gamma)$ and its inverse $p(\gamma \mid \alpha)$, the lexical translation probability $p_{lex}(\alpha \mid \gamma)$ and its inverse $p_{lex}(\gamma \mid \alpha)$, and a rule penalty that learns the preference for longer or shorter derivations.

Inverted Glue Rules In SMT decoding (Chiang, 2007), if no rule (e.g., a rule whose left-hand side is *X*) can be applied or the length of the potential source span is larger than a pre-defined length (e.g., 10 as in Chiang (2007)), a glue rule (either $S \rightarrow \langle X_{[1]}, X_{[1]} \rangle$ or $S \rightarrow \langle S_{[1]} X_{[2]}, S_{[1]} X_{[2]} \rangle$) will be used to simply stitch two consequent translated phrases together in a *monotone* way. Although this will reduce computational and modeling challenges, it obviously prevents some reasonable translation derivations because in certain cases, the order of phrases may be inverted on the target

[4]If $m = 0$, it indicates that no function is needed.

side. In this work, we additionally use an inverted glue rule which combines two non-terminals in a *swapped* way. Each glue rule, either *straight* or *inverted*, contains only two non-terminal symbols and is associated with two features, including phrase translation probability $p(\alpha \mid \gamma)$, and a glue rule penalty. Table 1(b) shows examples of a straight and an inverted glue rules. Moreover, these glue rules can be applied to any two neighboring translation nodes if the non-terminal symbols are matched.

3.2 Word Alignment for Semantic Parsing

Word alignment is an essential step for rule extraction in SMT, where recognizing that *wo shi* in Chinese is a good translation for *I am* in English requires establishing a correspondence between *wo* and *I*, and between *shi* and *am*. In the SMT community, researchers have developed standard, proven alignment tools such as GIZA++ (Och and Ney, 2003), which can be used to train IBM Models 1-5. However, there is one fundamental problem with the IBM models (Brown et al., 1993): each word on one side can be traced back to exactly one particular on the other word (or the *null* token which indicates the word aligns to no word on the other side). Figure 2(a) shows an example of GIZA++ alignment output from source side to target side, from which we can see that each source word aligns to exactly one target word. While alignment of multiple target words to one source word is common in SMT, a trick is then to run IBM model training in both directions. Then two resulting word alignments can be symmetrized, for instance, taking the intersection or the union of alignment points of each alignment. For example, Figure 2(b) shows GIZA++ alignment output from target side to source side while Figure 2(c) shows the symmetrization result with widely used grow-diag-final-and strategy.

Although symmetrization of word alignments works for SMT, can it be applied to semantic parsing? There are reasons to be doubtful. Word alignment for semantic parsing differs from alignment for SMT in several important aspects, at least including:

1. It is intrinsically asymmetric: within the semantic formalism used in this paper, NL′ is often longer than MRL′, and commonly contains words which have no counterpart in MRL′.
2. Little training data is available. SMT alignment models are typically trained in unsupervised fashion, inducing lexical correspondences from massive quantities of sentence-aligned bitexts.

Consequently, the symmetrization of word alignments may not work perfectly for semantic parsing. According to word alignment in Figure 2(c), a phrase extractor will generate a phrase pair ⟨*have the highest, largest_one@1*⟩, which is non-intuitive. By contrast, a more useful and general phrase pair ⟨*highest, largest_one@1*⟩ is typically excluded because *largest_one@1* aligns to all of *have*, *the*, and *highest*. Similarly, another useful phrase pair ⟨*texas, texas@s*⟩ is prohibited since *texas* aligns to both *stateid@1* and *texas@s*.

Ideally a new semantic parsing aligner should be able to capture the semantic equivalence. Unfortunately we are not aware of any research on alignment for semantic parsing, possibly due to lack of a paucity of high quality, publicly available data from which to learn. Instead of developing new alignment algorithm for semantic parsing, we make use of all the alignments as shown in Figure 2. That is to say, we triple the training data with each sentence pair having three alignments, i.e., two alignments in both directions, and the symmetrization alignment.[5] The advantages include: first, considering more possible alignments would increase the phrase coverage, especially when the training data is little; second, including the alignment from both directions would alleviate the error propagation caused by mis-aligned stop words (e.g., *be*, *the* in NL′ and *stateid@1* in MRL′). As a result, the phrase extractor will include phrase pairs of both ⟨*highest, largest_one@1*⟩ and ⟨*texas, texas@s*⟩. Our experiment shows that using the combination of all the three alignments achieve better performance than using any one, or any combination of two. Moreover, we found that we could achieve comparable performance even with manual alignment.

[5] Combining multiple alignments from different alignment models usually improves translation performance in SMT (Tu et al., 2012). However, our preliminary experiments showed that this did not yield higher improvement in semantic parsing, which in turn also demonstrates the difference in alignments for semantic parsing and SMT.

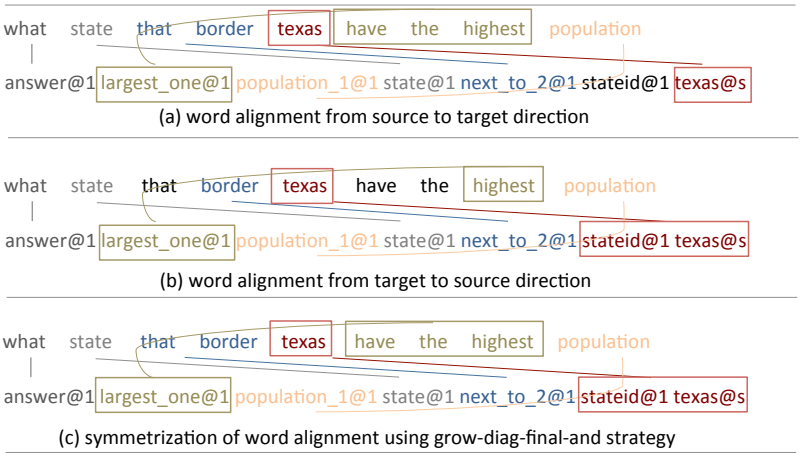


Figure 2: Example of a sentence pair with different alignments.

3.3 Synthetic Translation Rules for Unknown Word Translation

Most NLP tasks face the problem of unknown words, especially if only little training data is available. For example, it is estimated that 5.7% sentences in the (English) test data in our experiments have unknown words. Unknown words usually remain intact in the translation in most machine translation systems (Koehn et al., 2007; Dyer et al., 2010), resulting in the fact that certain translations can not be converted back to tree structures. This indicates that in semantic parsing the translation of a word can be from two categories: 1) a token in MRL; or 2) *null* (i.e., not translated at all), we generate synthetic translation rules for unknown word translation.

As a baseline, we simply skip unknown words as Kwiatkowski et al. (2010) by adding translation rules that translate them to *null* in MRL′. Each such rule is accompanied with one feature indicating that it is a translation rule for unknown word.

Alternatively, taking advantage of publicly available resources, we generate synthetic translation rules for unknown words pivoted by their semantically close words. Algorithm 1 illustrates the process to generate synthetic translation rules for unknown word translation. Given an unknown word w_u, it generates its synthetic rules in two steps: 1) finding top n (e.g., 5 as in our experiments) close words via Word2Vec;[6] and 2) generating synthetic translation rules based on the close words. Note that it may generate a synthetic rule with *null* at the target side since the lexical translation table derived from aligned training data contains translation to *null*. Each synthetic translation rule for unknown words is associated with three features returned from function $generate\_rule$.

Algorithm 1: Generating synthetic translation rules for unknown words

Input: Unknown word w_u in the source language
Source side training data vocabulary: W
Lexical translation tables T1 and T2 (two directions)
Output: Synthetic translation rule set R for w_u
1. **foreach** word w_i in W
2. $s_i = sim(w_u, w_i)$
3. get the top n words $WB = \{wb_1 ... wb_n\}$ with the highest $\{s_i\}$
4. $R = \phi$
5. **foreach** wb_i in WB
6. **foreach** t_j such $\langle wb_i, t_j \rangle$ in T1 and T2
7. $R \cup = generate\_rule(w_u, wb_i, t_j, T_1, T_2)$
8. **return** R

sim: returns the similarity between w_u and w_i.

$generate\_rule$: returns rule $\langle w_u, t_j \rangle$ with a feature indicating the similarity between w_u and wb_i, and two features indicating the lexical translation probabilities from wb_i to t_j and the way around.

4 Experimentation

In this section, we test our approach on the GeoQuery dataset, which is publicly available.

4.1 Experimental Settings

Data GeoQuery dataset consists of 880 questions paired with their corresponding tree structured semantic representations. Following the experimental setup in Jones et al. (2012), we use the 600 question pairs to train and tune our SMT de-

[6] It is available at http://code.google.com/p/word2vec/. We use Word2Vec rather than other linguistic resources like WordNet because the approach can be easily adopted to other languages only if there exists large monolingual data to train Word2Vec models.

coder, and evaluated on the remaining 280. Note that there is another version of GeoQuery dataset where the semantic representation is annotated with lambda calculus expressions and which is extensively studied (Zettlemoyer and Collins, 2005; Wong and Mooney, 2007; Liang et al., 2011; Kwiatkowski et al., 2013). Performance on the version of lambda calculus is higher than that on the tree structured version, however, the results obtained over the two versions are not directly comparable.

SMT Setting We use cdec (Dyer et al., 2010) as our HPB decoder. As mentioned above, 600 instances are used to train and tune our decoder. To get fair results, we split the 600 instances into 10 folds, each having 60 instances. Then for each fold, we use it as the tuning data while the other 540 instances and the NP list are used as training data.[7] We use IRSTLM toolkit (Federico et al., 2008) to train a 5-gram LM on the MRL′ side of the training data, using modified Kneser-Ney smoothing. We use Mira (Chiang et al., 2008) to tune the parameters of the system to maximize BLEU (Papineni et al., 2002). When extracting translation rules from aligned training data, we include both *tight* and *untight* phrases.

Evaluation We use the standard evaluation criteria for evaluation by executing both the predicted MRL and the gold standard against the database and obtaining their respective answer. Specifically, we convert a translation from MRL′ into MRL (if exists). The translation then is considered correct if and only if its MRL retrieves the same answers as the gold standard MRL (Jones et al., 2012), allowing for a fair comparison between our systems and previous works. As in Jones et al. (2012), we report accuracy, i.e. the percentage of translations with correct answers, and F1, i.e. the harmonic mean of precision (the proportion of correct answers out of translations with an answer) and recall (the proportion of correct answers out of all translations). In this section, we report our performance scores and analysis numbers averaged on our 10 SMT models.

[7]The NP list is from GeoQUery dataset in Jones et al. (2012), which contains MRs for every noun phrase that appears in the NL utterances of each language. As in Andreas et al. (2013), the NP list is included by appending all entries as extra training sentences with 50 times the weight of regular training examples, to ensure that they are learned as translation rules.

SCFG	alignment	Acc.	F1
non-enriched	src2tgt	75.0	82.5
	tgt2src	78.5	82.5
	gdfa	77.5	83.5
	src2tgt + tgt2src	81.4	85.0
	src2tgt + gdfa	77.1	83.1
	tgt2src + gdfa	80.6	83.9
	all	81.5	85.2
	gold	82.4	86.2
enriched	src2tgt	76.3	82.9
	tgt2src	82.0	85.2
	gdfa	78.9	83.9
	src2tgt + tgt2src	82.6	85.9
	src2tgt + gdfa	78.8	83.7
	tgt2src + gdfa	83.1	86.1
	all	82.9	86.1
	gold	84.1	87.1

Table 2: Performance of our (non-) enriched SCFG systems with different alignment settings.

4.2 Experimental Results

Table 2 shows the results of (non-) enriched SCFG systems over different alignment settings. In Table 2, *src2tgt* and *tgt2src* indicate alignment of source to target direction and alignment of target to source direction, respectively; *gdfa* indicates symmetrization of alignment with grow-diag-final-and strategy; *src2tgt+tgt2src* indicates doubling the training data with each sentence pair having both *src2tgt* and *tgt2src* alignments, similar for *src2tgt+gdfa* and *tgt2src+gdfa*; *all* indicates tripling the training data with each sentence pair having three alignments. Finally, *gold* indicates using gold alignment. [8]

Effect of Enriched SCFG From Table 2, we observe that enriched SCFG systems outperform non-enriched SCFG systems over all alignment settings, indicating the effect of enriching nonterminals. In particular for *tgt2src* alignment, it obtains improvements of 3.5% in accuracy and 2.7% in F1.

As mentioned earlier, the non-enriched SCFG system may result in ill-formed translations, which can not be converted back to tree structure. One natural way to overcome this issue, as in Andreas et al. (2013), would be to simply filter n-best translation till a well-formed one is found. However, we see very limited performance changes in accuracy and F1, suggesting that the effect of using n-best translation is very limited. For example, after using n-best translation, the non-enriched SCFG system with *all* alignment obtains 82.0 in accuracy (increased from 81.5) and 84.5 in

[8]We manually aligned sentence pairs in NL′ and MRL′.

alignment	Rec.	Pre.	F1
src2tgt	96.3	68.8	80.3
tgt2src	90.0	77.0	83.0
gdfa	95.1	62.7	75.6

Table 3: Alignment performance.

F1 (reduced from 85.2).

Effect of Word Alignment With respect to the performance over different alignment settings, we have the following observations from Table 2:

- Semantic parsing is substantially sensitive to alignment. Surprisingly, *gdfa* alignment, which is widely adopted in SMT, is inferior to *tgt2src* alignment. As expected, *src2tgt* alignment achieves the worst performance.

- Thanks to the increased coverage, doubling the training data (e.g., rows of *src2tgt+tgt2src*, *src2tgt+gdfa*, and *tgt2src+gdfa*) usually outperforms its corresponding single alignment. Moreover, tripling the training data (e.g., rows of *all*) achieves slightly better performance than any way of doubling the training data. This is expected since the *gdfa* alignment actually comes from the alignments of *src2tgt* and *tgt2src*, thus doubling the training with *src2tgt* and *tgt2src* have already included most aligns in *gdfa* alignment.

- Our approach of tripling the training data achieves comparable performance to the one with gold alignment, suggesting that instead of developing a brand new algorithm for semantic parsing alignment, we can simply make use of GIZA++ alignment output.

In terms of the *src2tgt*, *tgt2src* and *gdfa* alignments, the trend of the results is consistent over both non-enriched and enriched SCFG systems: the systems with *tgt2src* alignment work best while the systems with *src2tgt* alignment work worst. Next we look at the non-enriched SCFG systems to explore the behavior differences among the three alignments.

We examine the alignment accuracy against the gold alignment on training data (except the NP list part). As shown in Table 3, *src2tgt* has the highest recall while *tgt2src* has the highest precision. This is partly due to: 1) In *src2tgt* alignment, each source word aligns to exactly one particular target word (or the *null* token), resulting in frequent

		enriched + gdfa	
		correct	wrong
non-enriched + gdfa	correct	211	6
	wrong	10	53
enriched + all	correct	215	17
	wrong	6	42

Table 4: Confusion matrices of three SMT systems on English test sentences.

alignment errors for source side words that have no counterpart in target side. For example, both words of *the* and *be* on source side, which play functional roles in NL, rather than semantic roles, align to 15 different target words. 2) Except for a few words on target side, including *stateid@1*, *all@0* which have strong occurrence patterns (e.g., *stateid@1* is always followed by a state name), each word has counterpart on source side.

As to have a clearer understanding on the individual contribution of using enriched non-terminals and multiple word alignments, Table 4 presents two confusion matrices which show numbers of sentences that are correctly/wrongly parsed by three SMT systems on English test sentences. It shows that, for example, 211 sentences are correctly parsed by both non-enriched and enriched SCFG systems with *gdfa* alignment. Moving from performance of the non-enriched SMT system with *gdfa* alignment to that of the enriched SMT system with *all* alignment, we observe that on average more than half of the improvement comes from using multiple word alignments, the rest from using enriched non-terminals.

Effect of Unknown Word Translation Since each of our SMT model is actually trained on 540 instances (plus the NP list), the rate of unknown words in the test data tends to be higher than that in a system trained with the whole 600 instances. Based on the system of enriched SCFG with *all* alignment, Table 5 shows the results of applying unknown word translation. It shows that translating all unknown words into *null* obtains 2.4 points in accuracy over the system without it (e.g., 85.3 *vs.* 82.9). However, the slight improvement in F1 (e.g., 86.3 *vs.* 86.1) suggests that there are many scenarios that translating unknown words into *null* is incorrect. Fortunately, our semantic approach is partially able to generate correct translation rules for those unknown words which have translation in MRL′. Actually, the effect of our approach is highly dependent on the quality of the close words found via Word2Vec. With a manual examination

System	Acc.	F1
No unknown word translation	82.9	86.1
null: baseline	85.3	86.3
semantic: ours	86.3	87.1

Table 5: Performance with unknown word translation.

on the test data, we found that 11 out of all 17 unknown words should be translated into a corresponding token in MRL. For 8 of them, the synthetic translation rule set returned by Algorithm 1 contains correct translation rules.

Effect across Different Languages We have also tested our approach on the same dataset with other three languages. Specifically, while we are not aware of public resources to looking for semantically close words in German, Greek and Thai, we translate unknown words into *null* for the three languages. Table 6 shows the performance over four different languages. It shows that our approach, including enriched SCFG, tripling training data with three alignments, and unknown word translation, obtains consistent improvement over the four languages.

Decoding Time Analysis We analyze the effect on the decoding time of our approach, which is closely related to the size of phrase tables. Firstly, splitting non-terminal X into enriched ones increases the size of phrase tables. [9] This is not surprising since a phrase with non-terminal X (e.g., *the X* on the source side) may be further specified as multiple phrases with various non-terminals (e.g., *the C*, *the C/A1*, etc.). As a result, the average number of phrases per sentence (in English test data, hereafter) increases from 453 to 893 while the decoding time of the SMT decoder increases from 0.11 seconds to 0.19 seconds per sentence on average. Secondly, using multiple alignments also leads to larger phrase tables. This is illustrated by the increase of average number of phrases per sentence from 893 to 2055 while the decoding time moves from 0.19 seconds to 0.38 seconds per sentence on average. Finally, finding similar words via Word2Vec, however, is quite fast since this is bounded by the vocabulary size of our training set. Thanks to the small size of unknown words, adding unknown word translation rules has a very limited impact on the size of phrase table, consequently negligible changes on decoding time.

[9] In cdec, we generate a phrase table for each sentence.

5 Related Work

While there has been substantial work on semantic parsing, we focus our discussions on several approaches (e.g., SCFG approach, hybrid tree approach, and others approaches) that focus on the variable-free semantic representations.

WASP (Wong and Mooney, 2006) was strongly influenced by SMT techniques. Although WASP was also using multiple non-terminal symbols in SCFG to guarantee well-formed translations, our work differs from theirs in at least three ways. First, we use a different inventory of non-terminal symbols from theirs which was derived from MRL parses in the GeoQuery dataset. Second, to avoid the issues caused by word alignment between NL and MRL, we triple training data with each sentence pair having multiple alignments. However, WASP used a sequence of productions to represent MRL before running GIZA++. Third, we use typical features in HPB SMT (e.g., phrase translation probabilities, lexical translation probabilities, language model feature, etc.) while WASP used rule identity features. SMT-SemParse (Andreas et al., 2013) adapted standard SMT components for semantic parsing. The present work is based on theirs with all the extensions detailed in Section 3.

HYBRIDTREE+ (Lu et al., 2008) learned a synchronous generative model which simultaneously generated a NL sentence and an MRL tree. tsVB (Jones et al., 2012) used tree transducers, which were similar to the hybrid tree structures, to learn a generative process under a Bayesian framework. RHT (Lu, 2014) defined distributions over *relaxed hybrid tree* structures that jointly represented both sentences and semantics. Most recently, f-RHT (Lu, 2015) introduced constrained semantic forests to improve RHT model.

SCISSOR (Ge and Mooney, 2005) augmented syntactic parse tree with semantic information and then performed integrated semantic and syntactic parsing to NL sentences. KRISP (Mooney, 2006) used string classifiers to label substrings of an NL with entities from the meaning representation. UBL (Kwiatkowski et al., 2010) performed semantic parsing with an automatically-induced CCG lexicon.

Table 7 shows the evaluation results of our system as well as those of several other comparable related works which share the same experiment setup as ours. We can observe from Table 7 that semantic parsing with SMT components gives

System	English		German		Greek		Thai	
	Acc.	F1	Acc.	F1	Acc.	F1	Acc.	F1
non-enriched + gdfa	77.5	83.5	66.0	74.9	65.6	74.1	65.4	72.4
non-enriched + all	81.5	85.2	72.1	76.8	75.2	80.5	72.7	76.4
enriched + gdfa	78.9	83.9	66.7	74.6	67.8	76.1	68.5	74.1
enriched + all	82.9	86.1	75.4	79.5	76.5	81.2	75.2	77.9
enriched + all + unknown word translation	86.3	87.1	79.1	80.3	80.5	81.6	76.3	77.9

Table 6: Performance for the multilingual GeoQuery test set.

System	English		German		Greek		Thai	
	Acc.	F1	Acc.	F1	Acc.	F1	Acc.	F1
WASP	71.1	77.7	65.7	74.9	70.7	78.6	71.4	75.0
SMT-SemParse	80.5	81.8	68.9	71.8	69.1	72.3	70.4	70.7
HYBRIDTREE+	76.8	81.0	62.1	68.5	69.3	74.6	73.6	76.7
tsVB	79.3	79.3	74.6	74.6	75.4	75.4	78.2	78.2
RHT	83.6	83.6	74.3	74.3	78.2	78.2	79.3	79.3
f-RHT	**86.8**	86.8	75.7	75.7	79.3	79.3	**80.7**	**80.7**
UBL	82.1	82.1	73.6	73.7	75.0	75.0	66.4	66.4
this work	86.3	**87.1**	**79.1**	**80.3**	**80.5**	**81.6**	76.3	77.9

Table 7: Performance comparison for the multilingual GeoQuery test set. The performance of WASP, HYBRIDTREE+, tsVB and UBL is taken from Jones et al. (2012).

competitive performance when all the extensions (described in Section 3) are used. Specifically, it significantly outperforms the semantic parser with standard SMT components (Andreas et al., 2013). Our approach reports the best accuracy and F1 scores on English, German, and Greek. While we are able to obtain improvement on Thai, the performance is still lower than those of RHT and TREETRANS. This is probably because of the low quality of word alignment output between this Asian language and MRL.

6 Conclusion and Future Work

In this paper, we have presented an enriched SCFG approach for semantic parsing which realizes the potential of the SMT approach. The performance improvement is contributed from the extension of translation rules with informative symbols and increased coverage. Such an extension share a similar spirit as generalization of a CCG lexicon for CCG-based semantic parser (Kwiatkowski et al., 2011; Wang et al., 2014). Experiments on benchmark data have shown that our model is competitive to previous work and achieves state-of-the-art performance across a few different languages.

Recently the research of semantic parsing in open domain with weakly (or un-) supervised setups, under different settings where the goal was to optimize the performance of certain downstream NLP tasks such as answering questions, has received a significant amount of attention (Poon and Domingos, 2009; Clarke et al., 2010; Berant et al., 2013; Berant and Liang, 2014). One direction of our future work is to extend the current framework to support the generation of synthetic translation rules from weaker signals (e.g., from question-answer pairs), rather than from aligned parallel data.

We also noticed recent advance in tree-based SMT. Applying such string-to-tree or tree-to-tree translation models (Yamada and Knight, 2001; Shen et al., 2008) to semantic parsing will naturally resolve the inconsistent semantic structure issue, though they require additional information to generate tree labels on the target side. However, due to the constraint that each target phrase needs to map to a syntactic constituent, phrase tables in tree-based translation models usually suffer from the low coverage issue, especially if the training data size is small. Therefore, another direction of our future work is to explore specific problems that will emerge when employing tree-based SMT systems to semantic parsing, and provide solutions to them.

Acknowledgments

The authors would like to thank Zhaopeng Tu, S. Matthew English and three anonymous reviewers for providing helpful suggestions, and also acknowledge Jacob Andreas for help in running SMT-SemParse. Junhui Li and Guodong Zhou were supported partially by NSFC grants, No.61273320, No.61331011, No.61305088, No.61401295. Wei Lu was supported by SUTD grant SRG ISTD 2013 064.

References

Jacob Andreas, Andreas Vlachos, and Stephen Clark. 2013. Semantic parsing as machine translation. In *Proceedings of ACL 2013*, pages 47–52.

Jonathan Berant and Percy Liang. 2014. Semantic parsing via paraphrasing. In *Proceedings of ACL 2014*, pages 1415–1425.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Proceedings of EMNLP 2013*, pages 1533–1544.

Peter E. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–313.

David Chiang, Yuval Marton, and Philip Resnik. 2008. Online large-margin training of syntactic and structural translation features. In *Proceedings of EMNLP 2008*, pages 224–233.

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.

James Clarke, Dan Goldwasser, Ming-Wei Chang, and Dan Roth. 2010. Driving semantic parsing from the worlds response. In *Proceedings of CoNLL 2010*, pages 18–27.

Chris Dyer, Adam Lopez, Juri Ganitkevitch, Jonathan Weese, Ferhan Ture, Phil Blunsom, Hendra Setiawan, Vladimir Eidelman, and Philip Resnik. 2010. cdec: A decoder, alignment, and learning framework for finite-state and context-free translation models. In *Proceedings of ACL 2010 System Demonstrations*, pages 7–12.

Marcello Federico, Nicola Bertoldi, and Mauro Cettolo. 2008. IRSTLM: an open source toolkit for handling large scale language models. In *Proceedings of Interspeech 2008*, pages 1618–1621.

Ruifang Ge and Raymond Mooney. 2005. A statistical semantic parser that integrates syntax and semantics. In *Proceedings of CoNLL 2006*, pages 9–16.

Bevan Jones, Mark Johnson, and Sharon Goldwater. 2012. Semantic parsing with bayesian tree transducers. In *Proceedings of ACL 2012*, pages 488–496.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of ACL 2010 System Demonstrations*, pages 177–180.

Tom Kwiatkowski, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2010. Inducing probabilistic CCG grammars from logical form with higher-order unification. In *Proceedings of EMNLP 2010*, pages 1223–1233.

Tom Kwiatkowski, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2011. Lexical generalization in ccg grammar induction for semantic parsing. In *Proceedings of EMNLP 2011*, pages 1512–1523.

Tom Kwiatkowski, Eunsol Choi, Yoav Artzi, and Luke Zettlemoyer. 2013. Scaling semantic parsers with on-the-fly ontology matching. In *Proceedings of EMNLP 2013*, pages 1545–1556.

Junhui Li, Zhaopeng Tu, Guodong Zhou, and Josef van Genabith. 2012. Using syntactic head information in hierarchical phrase-based translation. In *Proceedings of WMT 2012*, pages 232–242.

Percy Liang, Michael I. Jordan, and Dan Klein. 2011. Learning dependency-based compositional semantics. In *Proceedings of ACL 2011*, pages 590–599.

Wei Lu, Hwee Tou Ng, Wee Sun Lee, and Luke S. Zettlemoyer. 2008. A generative model for parsing natural language to meaning representations. In *Proceedings of EMNLP 2008*, pages 783–792.

Wei Lu. 2014. Semantic parsing with relaxed hybrid trees. In *Proceedings of EMNLP 2014*, pages 1308–1318.

Wei Lu. 2015. Constrained semantic forests for improved discriminative semantic parsing. In *Proceedings of ACL-IJCNLP 2015*, pages 737–742.

Klaus Macherey, Franz Josef Och, and Hermann Ney. 2001. Natural language understanding using statistical machine translation. In *Proceedings of EuroSpeech 2001*, pages 2205–2208.

Rohit J. Kate; Raymond J. Mooney. 2006. Using string-kernels for learning semantic parsers. In *Proceedings of ACL-COLING 2006*, pages 913–920.

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

Kishore A. Papineni, Salim Roukos, and Todd Ward. 1997. Feature-based language understanding. In *Proceedings of EuroSpeech 1997*, pages 1435–1438.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of ACL 2002*, pages 311–318.

Hoifung Poon and Pedro Domingos. 2009. Unsupervised semantic parsing. In *Proceedings of EMNLP 2009*, pages 1–10.

Libin Shen, Jinxi Xu, and Ralph Weischedel. 2008. A new string-to-dependency machine translation algorithm with a target dependency language model. In *Proceedings of ACL 2008*, pages 577–585.

Mark Steedman. 2000. *The syntactic process*. The MIT Press.

Zhaopeng Tu, Yang Liu, Yifan He, Josef van Genabith, Qun Liu, and Shouxun Lin. 2012. Combining multiple alignments to improve machine translation. In *Proceedings of COLING 2012: Posters*, pages 1249–1260.

Adrienne Wang, Tom Kwiatkowski, and Luke Zettlemoyer. 2014. Morpho-syntactic lexical generalization for ccg semantic parsing. In *Proceedings of EMNLP 2014*, pages 1284–1295.

Yuk Wah Wong and Raymond Mooney. 2006. Learning for semantic parsing with statistical machine translation. In *Proceedings of HLT-NAACL 2006*, pages 439–446.

Yuk Wah Wong and Raymond Mooney. 2007. Learning synchronous grammars for semantic parsing with lambda calculus. In *Proceedings of ACL 2007*, pages 960–967.

Kenji Yamada and Kevin Knight. 2001. A syntax-based statistical translation model. In *Proceedings of ACL 2001*, pages 523–530.

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of UAI 2005*, pages 658–666.

Andreas Zollmann and Ashish Venugopal. 2006. Syntax augmented machine translation via chart parsing. In *Proceedings of WMT 2006*, pages 138–141.

Solving Geometry Problems: Combining Text and Diagram Interpretation

Minjoon Seo, Hannaneh Hajishirzi, Ali Farhadi, Oren Etzioni, Clint Malcolm
University of Washington, Allen Institute for Artificial Intelligence
{minjoon,hannaneh,clintm}@washington.edu,{alif,orene}@allenai.org

Abstract

This paper introduces GEOS, the first automated system to solve unaltered SAT geometry questions by combining text understanding and diagram interpretation. We model the problem of understanding geometry questions as submodular optimization, and identify a formal problem description likely to be compatible with both the question text and diagram. GEOS then feeds the description to a geometric solver that attempts to determine the correct answer. In our experiments, GEOS achieves a 49% score on official SAT questions, and a score of 61% on practice questions.[1] Finally, we show that by integrating textual and visual information, GEOS boosts the accuracy of dependency and semantic parsing of the question text.

1 Introduction

This paper introduces the first fully-automated system for solving unaletered SAT-level geometric word problems, each of which consists of text and the corresponding diagram (Figure 1). The geometry domain has a long history in AI, but previous work has focused on geometric theorem proving (Feigenbaum and Feldman, 1963) or geometric analogies (Evans, 1964). Arithmetic and algebraic word problems have attracted several NLP researchers (Kushman et al., 2014; Hosseini et al., 2014; Roy et al., 2015), but *geometric* word problems were first explored only last year by Seo et al. (2014). Still, this system merely aligned diagram elements with their textual mentions (e.g., "Circle O")—it did not attempt to fully represent geometry problems or solve them. Answering geometry questions requires a method that interpert question text and diagrams in concert.

[1]The source code, the dataset and the annotations are publicly available at geometry.allenai.org.

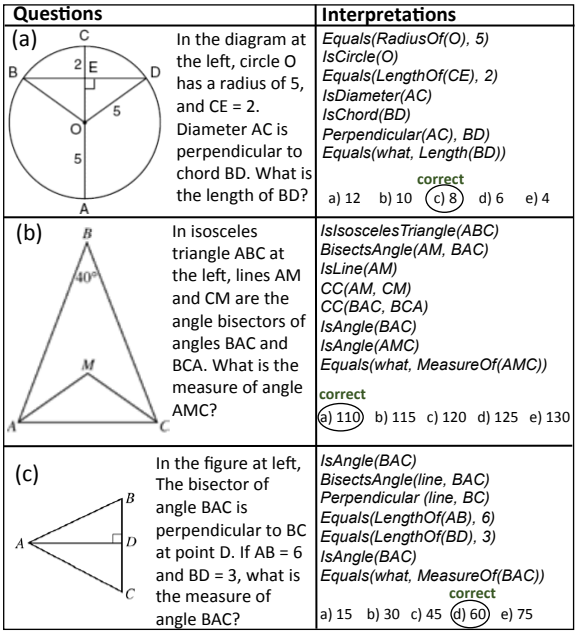


Figure 1: Questions (left column) and interpretations (right column) derived by GEOS.

The geometry genre has several distinctive characteristics. First, diagrams provide essential information absent from question text. In Figure 1 problem (a), for example, the unstated fact that lines BD and AC intersect at E is necessary to solve the problem. Second, the text often includes difficult references to diagram elements. For example, in the sentence "In the diagram, the longer line is tangent to the circle", resolving the referent of the phrase "longer line" is challenging. Third, the text often contains implicit relations. For example, in the sentence "AB is 5", the relations `IsLine(AB)` and `length(AB)=5` are implicit. Fourth, geometric terms can be ambiguous as well. For instance, radius can be a type identifier in "the length of radius AO is 5", or a predicate in "AO is the radius of circle O". Fifth, identifying the correct arguments for each relation is challenging. For example, in sentence "Lines AB and CD are perpendicular to EF", the parser has to determine *what* is perpendicular to EF—line AB? line

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1466–1476,
Lisbon, Portugal, 17-21 September 2015.

CD? Or both AB and CD? Finally, it is hard to obtain large number of SAT-level geometry questions; Learning from a few examples makes this a particularly challenging NLP problem.

This paper introduces GEOS, a system that maps geometry word problems into a logical representation that is compatible with both the problem text and the accompanying diagram (Figure 1). We cast the mapping problem as the problem of selecting the subset of relations that is most likely to correspond to each question.

We compute the mapping in three main steps (Figure 2). First, GEOS uses text- and diagram-parsing to overgenerate a set of relations that potentially correspond to the question text, and associates a score with each. Second, GEOS generates a set of relations (with scores) that corresponds to the diagram. Third, GEOS selects a subset of the relations that maximizes the joint text and diagram scores. We cast this maximization as a submodular optimization problem, which enables GEOS to use a close-to-optimal greedy algorithm. Finally, we feed the derived formal model of the problem to a geometric solver, which computes the answer to the question.

GEOS is able to solve *unseen* and *unaltered* multiple-choice geometry questions. We report on experiments where GEOS achieves a 49% score on official SAT questions, and a score of 61% on practice questions, providing the first results of this kind. Our contributions include: (1) designing and implementing the first end-to-end system that solves SAT plane geometry problems; (2) formalizing the problem of interpreting geometry questions as a submodular optimization problem; and (3) providing the first empirical results on the geometry genre, making the data and software available for future work.

2 Related Work

Semantic parsing is an important area of NLP research (Zettlemoyer and Collins, 2005; Ge and Mooney, 2006; Flanigan et al., 2014; Eisenstein et al., 2009; Kate and Mooney, 2007; Goldwasser and Roth, 2011; Poon and Domingos, 2009; Berant and Liang, 2014; Kwiatkowski et al., 2013; Reddy et al., 2014). However, semantic parsers do not tackle diagrams—a critical element of the geometry genre. In addition, the overall number of available geometry questions is quite small compared to the size of typical NLP corpora, making it challenging to learn semantic parsers directly from geometry questions. Relation extraction is another area of NLP that is related to our task (Cowie and Lehnert, 1996; Culotta and Sorensen, 2004). Again, both diagrams and small corpora are problematic for this body of work.

Our work is part of grounded language acquisition research (Branavan et al., 2012; Vogel and Jurafsky, 2010; Chen et al., 2010; Hajishirzi et al., 2011; Liang et al., 2009; Koncel-Kedziorski et al., 2014; Bordes et al., 2010; Kim and Mooney, 2013; Angeli and Manning, 2014; Hixon et al., 2015; Koncel-Kedziorski et al., 2014; Artzi and Zettlemoyer, 2013) that involves mapping text to a restricted formalism (instead of a full, domain independent representation). In the geometry domain, we recover the entities (e.g., circles) from diagrams, derive relations compatible with both text and diagram, and re-score relations derived from text parsing using diagram information. Casting the interpretation problem as selecting the most likely subset of literals can be generalized to grounded semantic parsing domains such as navigational instructions.

Coupling images and the corresponding text has attracted attention in both vision and NLP (Farhadi et al., 2010; Kulkarni et al., 2011; Gupta and Mooney, 2010; Gong et al., 2014; Fang et al., 2014). We build on this powerful paradigm, but instead of generating captions we show how processing multimodal information help improve textual or visual interpretations for solving geometry questions.

Diagram understanding has been explored since early days in AI (Lin et al., 1985; Hegarty and Just, 1989; Novak, 1995; O'Gorman and Kasturi, 1995; Bulko, 1988; Srihari, 1994; Lovett and Forbus, 2012). Most previous approaches differ from our method because they address the twin problems of diagram understanding and text understanding in isolation. Often, previous work relies on manual identification of visual primitives, or on rule-based system for text analysis. The closest work to ours is the recent work of Seo et al. (2014) that aligns geometric shapes with their textual mentions, but does not identify geometric relations or solve geometry problems.

3 Problem Formulation

A geometry question is a tuple (t, d, c) consisting of a text t in natural language, a diagram d

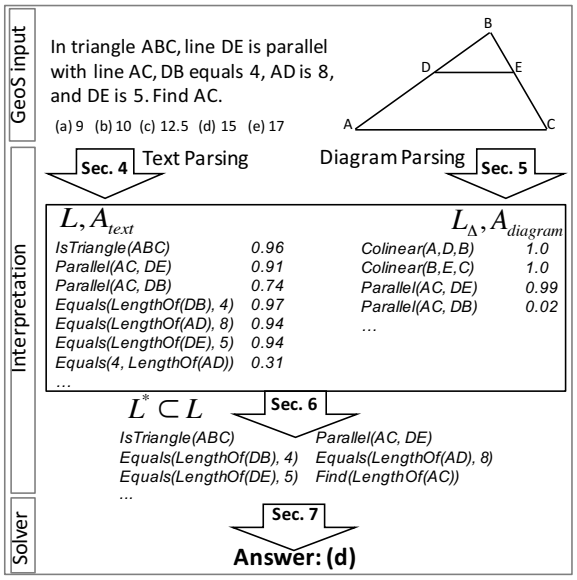


Figure 2: Overview of our method for solving geometry questions.

in raster graphics, and multiple choice answers $c = \{c_1, \ldots, c_M\}$ ($M = 5$ in SAT). Answering a geometry question is to find a correct choice c_i.

Our method, GEOS, consists of two steps (Figure 2): (1) *interpreting* a geometry question by deriving a logical expression that represents the meaning of the text and the diagram, and (2) *solving* the geometry question by checking the satisfiablity of the derived logical expression. In this paper we mainly focus on *interpreting* geometry questions and use a standard algebraic solver (see section 7 for a brief description of the solver).

Definitions: We formally represent logical expressions in the geometry domain with the language Ω, a subset of typed first-order logic that includes:

- *constants*, corresponding to known numbers (e.g., `5` and `2` in Figure 1) or entities with known geometric coordinates.
- *variables*, corresponding to unknown numbers or geometrical entities in the question (e.g., `O` and `CE` in Figure 1).
- *predicates*, corresponding to geometric or arithmetic relations (e.g., `Equals`, `IsDiameter`, `IsTangent`).
- *functions*, corresponding to properties of geometrical entities (e.g., `LengthOf`, `AreaOf`) or arithmetic operations (e.g., `SumOf`, `RatioOf`).

Each element in the geometry language has either boolean (e.g., `true`), numeric (e.g., `4`), or entity (e.g., `line`, `circle`) type. We refer to all symbols in the language Ω as *concepts*.

We use the term *literal* to refer to the application of a predicate to a sequence of arguments (e.g., `IsTriangle(ABC)`). Literals are possibly negated atomic formulas in the language Ω. Logical formulas contain constants, variables, functions, existential quantifiers and conjunctions over literals (e.g., $\exists$`x, IsTriangle(x)`$\wedge$`IsIsosceles(x)`).

Interpretation is the task of mapping a new geometry question with each choice, (t, d, c_m), into a logical formula γ in Ω. More formally, the goal is to find $\gamma^* = \arg\max_{\gamma \in \Gamma} score(\gamma; t, d, c_m)$ where Γ is the set of all logical formulas in Ω and $score$ measures the interpretation score of the formula according to both text and diagram. The problem of deriving the best formula γ^* can be modeled as a combinatorial search in the space of literals L (note that each logical formula γ is represented as a conjunction over literals l_i).

GEOS efficiently searches this combinatorial space taking advantage of a submodular set function that scores a subset of literals using both text and diagram. The best subset of literals is the one that has a high *affinity* with both text and diagram and is *coherent* i.e., does not suffer from redundancy (see Section 6). More formally,[2]

$$L^* = \arg\max_{L' \subset L} \lambda \underbrace{\mathcal{A}(L', t, d)}_{\text{Affinity}} + \underbrace{\mathcal{H}(L', t, d)}_{\text{Coherence}}, \quad (1)$$

where $\mathcal{A}(L', t, d)$ measures the affinity of the literals in L' with both the text and the diagram, $\mathcal{H}(L', t, d)$ measures the coverage of the literals in L' compared to the text and discourages redundancies, and λ is a trade-off parameter between $\mathcal{A}$ and $\mathcal{H}$.

The affinity $\mathcal{A}$ is decomposed into text-based affinity, $\mathcal{A}_{text}$, and diagram-based affinity, $\mathcal{A}_{diagram}$. The text-based affinity closely mirrors the linguistic structure of the sentences as well as type matches in the geometry language Ω. For modeling the text score for each literal, we learn a log-linear model. The diagram-based affinity $\mathcal{A}_{diagram}$ grounds literals into the diagram, and scores literals according to the diagram parse. We describe the details on how to compute $\mathcal{A}_{text}$ in section 4 and $\mathcal{A}_{diagram}$ in section 5.

4 Text Parser

The text-based scoring function $\mathcal{A}_{text}(L, t)$ computes the affinity score between the set of liter-

[2]We omit the argument c_m for the ease of notation.

1. $\{l_j\}, \mathcal{A}_{text} \leftarrow$ TEXT PARSING(language Ω, text-choice pair (t, c_i)) (Section 4)
   (i) **concept identification:** initialize a hypergraph G with concept nodes.
   (ii) **relation identification:** add a hyperedge (relation) r_j between two or three related concept nodes and assign a weight $\mathcal{A}_{text}(r_j, t; \theta)$ based on the learned classifier.
   (iii) **literals parsing:** obtain all subtrees of G, which are equivalent to all possible literals, $\{l'_j\}$. Let $\mathcal{A}_{text}(l_j, t) = \sum_j \mathcal{A}_{text}(r_j, t; \theta)$ for all r_j in the literal l_i.
   (iv) **relation completion:** obtain a complete literal l_j for each (under-specified) l'_j, dealing with implication and coordinating conjunctions.
2. $L_\Delta, \mathcal{A}_{diagram} \leftarrow$ DIAGRAM PARSING(diagram image d, literals $\{l_j\}$) (Section 5)
3. $L^* \leftarrow$ GREEDY MAXIMIZATION(literals $L = \{l_j\}$, score functions $\mathcal{A}_{text}$ and $\mathcal{A}_{diagram}$) (Section 6)
   (i) **initialization**: $L' \leftarrow \{\}$
   (ii) **greedy addition**: $add(L', l_j)$ s.t. $l_j = \text{argmax}_{l_j \in L \setminus L'} \mathcal{F}(L' \cup \{l_j\}) - \mathcal{F}(L')$, where $\mathcal{F} = \lambda\mathcal{A} + \mathcal{H}$
   (iii) **iteration**: repeat step (ii) while the gain is positive.
4. Answer $c^* \leftarrow$ one of choices s.t. $L^* \cup L_\Delta$ are simultaneously satisfiable according to SOLVER (Section 7)

Figure 3: Solving geometry questions with GEOS.

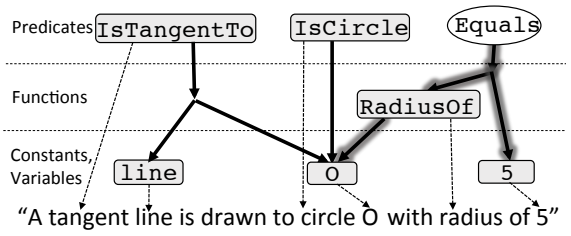


Figure 4: Hypergraph representation of the sentence "A tangent line is drawn to circle O with radius of 5".

als L and the question text t. This score is the sum of the affinity scores of individual literals $l_j \in L$ i.e., $\mathcal{A}_{text}(L, t) = \sum_j \mathcal{A}_{text}(l_j, t)$ where $\mathcal{A}_{text}(l_j, t) \mapsto [-\infty, 0]$.[3] GEOS learns a discriminative model $\mathcal{A}_{text}(l_j, t; \theta)$ that scores the affinity of every literal $l_j \in L$ and the question text t through supervised learning from training data.

We represent literals using a hypergraph (Figure 4) (Klein and Manning, 2005; Flanigan et al., 2014). Each node in the graph corresponds to a concept in the geometry language (i.e. constants, variables, functions, or predicates). The edges capture the *relations* between concepts; concept nodes are connected if one concept is the argument of the other in the geometry language. In order to interpret the question text (Figure 3 step 1), GEOS first identifies concepts evoked by the words or phrases in the input text. Then, it learns the affinity scores which are the weights of edges in the hypergraph. It finally completes relations so that type matches are satistfied in the formal language.

4.1 Concept Identification

Concepts are defined as symbols in the geometry language Ω. The concept identification stage maps words or phrases to their corresponding concepts in the geometry language. Note that a phrase can be mapped to several concepts. For instance, in the sentence "ABCD is a square with an area of 1", the word "square" is a noun referring to some object, so it maps to a variable `square`. In a similar sentence "square ABCD has an area 1", the word "square" describes the variable `ABCD`, so it maps to a predicate `IsSquare`.

GEOS builds a *lexicon* from training data that maps stemmed words and phrases to the concepts in the geometry language Ω. The lexicon is derived from all correspondences between geometry keywords and concepts in the geometry language as well as phrases and concepts from manual annotations in the training data. For instance, the lexicon contains ("square", {`square`, `IsSquare`}) including all possible concepts for the phrase "square". Note that GEOS does not make any hard decision on which identification is correct in this stage, and defers it to the relation identification stage (Section 4.2). To identify numbers and explicit variables (e.g. "5", "AB", "O"), GEOS uses regular expressions. For an input text t, GEOS assigns one node in the graph (Figure 4) for each concept identified by the lexicon.

4.2 Relation Identification

A relation is a directed hyperedge between concept nodes. A hyperedge connects two nodes (for unary relations such as the edge between `RadiusOf` and `O` in Figure 4) or three nodes (for binary relations such as the hyperedge between `Equals` and its two arguments `RadiusOf` and `5` in Figure 4).

We use a discriminative model (logistic regression) to predict the probability of a relation r_i being correct in text t: $P_\theta(y_i|r_i, t) = \frac{1}{1+\exp(f_{text}(r_i,t)\cdot\theta)}$, where $y_i \in \{0, 1\}$ is the label

[3] For the ease of notation, we use $\mathcal{A}_{text}$ as a function taking sets of literals or a literal.

Dependency tree distance	Shortest distance between the words of the concept nodes in the dependency tree. We use indicator features for distances of -3 to 3. Positive distance shows if the child word is at the right of the parent's in the sentence, and negative otherwise.
Word distance	Distance between the words of the concept nodes in the sentence.
Dependency tree edge label	Indicator functions for the outgoing edges of the parent and child for the shortest path between them.
Part of speech tag	Indicator functions for the POS tags of the parent and the child.
Relation type	Indicator functions for unary / binary parent and child nodes.
Return type	Indicator functions for the return types of the parent and the child nodes. For example, return type of `Equals` is `boolean`, and that of `LengthOf` is `numeric`.

Table 1: The features of the unary relations. The features of the binary relations is computed in a similar way.

(a)	sentence:	"What is the perimeter of ABCE?"
	intermediate:	`∃ what, ABCE: Bridged(what, PerimeterOf(ABCE))`
	final:	`∃ what, ABCE: Equals(what, PerimeterOf(ABCE))`
(b)	sentence:	"AM and CM bisect BAC and BCA."
	intermediate:	`∃ AM, CM, BAC, BCA: BisectsAngle(AM, BAC) ∧ CC(AM, CM) ∧ CC(BAC, BCA)`
	final:	`∃ AM, CM, BAC, BCA: BisectsAngle(AM, BAC) ∧ BisectsAngle(CM, BCA)`

Figure 5: Showing the two-stage learning with the intermediate representation that demonstrates implication.

for r_i being correct in t, $f_{text}(r_i, t)$ is a feature vector of t and r_i, and θ is a vector of parameters to be learned. We define the affinity score of r_i by $\mathcal{A}_{text}(r_i, t; \theta) = \log P_\theta(y_i | r_i, t)$. The weight of the corresponding hyperedge is the relation's affinity score. We learn θ using the maximum likelihood estimation of the training data (details in Section 8), with L2 regularization.

We train two separate models for learning unary and binary relations. The training data consists of sentence-relation-label tuples (t, r, y); for instance, ("A tangent line is drawn to circle O", `IsTangent(line, O)`, 1) is a positive training example. All incorrect relations in the sentences of the training data are negative examples (e.g. ("A tangent line is drawn to circle O", `IsCircle(line)`, 0)).

The features for the unary and binary models are shown in Table 1 for the text t and the relation r_i. We use two main feature categories. Structural features: these features capture the syntactic cues of the text in the form of text distance, dependency tree labels, and part of speech tags for the words associated with the concepts in the relation. Geometry language features: these features capture the cues available in the geometry language Ω in the form of the types and the truth values of the corresponding concepts in the relation.

At inference, GEOS uses the learned models to calculate the affinity scores of all the literals derived from the text t. The affinity score of each literal l_j is calculated from the edge (relation) weights in the corresponding subgraph, i.e. $\mathcal{A}_{text}(l_j, t) = \sum_i \mathcal{A}_{text}(r_i, t; \theta)$ for all r_i in the literal l_j.

4.3 Relation Completion

So far, we have explained how to score the affinities between explicit relations and the question text. Geometry questions usually include *implicit* concepts. For instance, "Circle O has a radius of 5" implies the `Equals` relationship between "Radius of circle O" and "5". In addition, geometry questions include *coordinating conjunctions* between entities. In "AM and CM bisect BAC and BCA", "bisect" is shared by two lines and two angles (Figure 5 (b)). Also, consider two sentences: "AB and CD are perpendicular" and "AB is perpendicular to CD". Both have the same semantic annotation but very different syntactic structures.

It is difficult to directly fit the syntactic structure of question sentences into the formal language Ω for *implications* and *coordinating conjunctions*, especially due to small training data. We, instead, adopt a two-stage learning inspired by recent work in semantic parsing (Kwiatkowski et al., 2013). Our solution assumes an intermediate representation that is syntactically sound but possibly *underspecified*. The intermediate representation closely mirrors the linguistic structure of the sentences. In addition, it can easily be transferred to the formal representation in the geometry language Ω.

Figure 5 shows how implications and coordinating conjunctions are modeled in the intermediate representation. `Bridged` in Figure 5 (a) indicates that there is a special relation (edge) between the two concepts (e.g., `what` and `PerimeterOf`), but the alignment to the geometry language $\mathcal{L}$ is not clear. `CC` in Figure 5 (b) indicates that there is a special relation between two concepts that are connected by "and" in the sentence. GEOS *completes*

the under-specified relations by mapping them to the corresponding well-defined relations in the formal language.

Implication: We train a log-linear classifier to identify if a `Bridged` relation (implied concept) exists between two concepts. Intuitively, the classification score indicates the likelihood that certain two concepts (e.g., `What` and `PerimeterOf`) are *bridged*. For training, positive examples are pairs of concepts whose underlying relation is underspecified, and negative examples are all other pairs of concepts that are not bridged. For instance, (`what`, `PerimeterOf`) is a positive training example for the `bridged` relation. We use the same features in Table 1 for the classifier.

We then use a deterministic rule to map *bridged* relations in the intermediate representation to the correct *completed* relations in the final representation. In particular, we map `bridged` to `Equals` if the two children concepts are of type `number`, and to `IsA` if the concepts are of type entity (e.g. `point`, `line`, `circle`).

Coordinating Conjunctions: `CC` relations model coordinating conjunctions in the intermediate representation. For example, Figure 5 (b) shows the conjunction between the two angles `BAC` and `BCA`. We train a log-linear classifier for the `CC` relations, where the setup of the model is identical to that of the binary relation model in Section 4.2.

After we obtain a list of `CC(x,y)` in the intermediate representation, we use deterministic rules to coordinate the entities x and y in each `CC` relation (Figure 5 (b)). First, GEOS forms a set $\{x, y\}$ for every two concepts x and y that appear in `CC(x,y)` and transforms every x and y in other literals to $\{x, y\}$. Second, GEOS transforms the relations with expansion and distribution rules (Figure 3 Step 1 (iv)). For instance, `Perpendicular({x,y})` will be transferred to `Perpendicular(x, y)` (expansion rule), and `LengthOf{x,y})` will be transferred to `LengthOf(x)` $\wedge$ `LengthOf(y)` (distribution rule).

5 Diagram Parser

We use the publicly available diagram parser (Seo et al., 2014) to obtain the set of all visual elements (points, lines, circles, etc.), their coordinates, their relationships in the diagram, and their alignment with entity references in the text (e.g. "line AB", "circle O"). The diagram parser serves two purposes: (a) computing the diagram score as a measure of the affinity of each literal with the diagram; (b) obtaining high-confidence visual literals which cannot be obtained from the text.

Diagram score: For each literal l_j from the text parsing, we obtain its diagram score $\mathcal{A}_{diagram}(l_j, d) \mapsto [-\infty, 0]$. GEOS grounds each literal derived from the text by replacing every variable (entity or numerical variable) in the relation to the corresponding variable from the diagram parse. The score function is the relaxed indicator function of whether a literal is true according to the diagram. For instance, in Figure 1 (a), consider the literal $l =$ `Perpendicular(AC, BD)`. In order to obtain its diagram score, we compute the angle between the lines AC and BD in the diagram and compare it with $\pi/2$. The closer the two values, the higher the score (closer to 0), and the farther they are, the lower the score. Note that the variables `AC` and `BD` are grounded into the diagram before we obtain the score; that is, they are matched with the actual corresponding lines AC and BD in the diagram.

The diagram parser is not able to evaluate the correctness of some literals, in which case their diagram scores are *undefined*. For instance, `Equals(LengthOf(AB), 5)` cannot be evaluated in the diagram because the scales in the diagram (pixel) and the text are different. For another example, `Equals(what, RadiusOf(circle))` cannot be evaluated because it contains an ungrounded (query) variable, `what`. When the diagram score of a literal l_j is undefined, GEOS lets $\mathcal{A}_{diagram}(l_j) = \mathcal{A}_{text}(l_j)$.

If the diagram score of a literal is very low, then it is highly likely that the literal is false. For example, in Figure 2, `Parallel(AC, DB)` has a very low diagram score, 0.02, and is apparently false in the diagram. Concretely, if for some literal l_j, $\mathcal{A}_{diagram}(l_i) < \epsilon$, then GEOS disregards the text score of l_i by replacing $\mathcal{A}_{text}(l_j)$ with $\mathcal{A}_{diagram}(l_j)$. On the other hand, even if the diagram score of a literal is very high, it is still possible that the literal is false, because many diagrams are not drawn to scale. Hence, GEOS adds both text and diagram scores in order to score literals (Section 6).

High-confidence visual literals: Diagrams often contain critical information that is not present in the text. For instance, to solve the question in Figure 1, one has to know that the points `A`, `E`, and `C` are colinear. In addition, diagrams include numer-

ical labels (e.g. one of the labels in Figure 1(b) indicates the measure of the angle ABC = 40 degrees). This kind of information is confidently parsed with the diagram parser by Seo et al. (2014). We denote the set of the high-confidence literals by L_Δ that are passed to the solver (Section 7).

6 Optimization

Here, we describe the details of the objective function (Equation 1) and how to efficiently maximize it. The integrated affinity score of a set of literals L' (the first term in Equation 1) is defined as:

$$\mathcal{A}(L', t, d) = \sum_{l'_j \in L'} \left[\mathcal{A}_{text}(l'_j, t) + \mathcal{A}_{diagram}(l'_j, d)\right]$$

where $\mathcal{A}_{text}$ and $\mathcal{A}_{diagram}$ are the text and diagram affinities of l'_j, respectively.

To encourage GEOS to pick a subset of literals that cover the concepts in the question text and, at the same time, avoid redundancies, we define the coherence function as:

$$\mathcal{H}(L', t, d) = N_{\text{covered}}(L') - R_{\text{redundant}}(L')$$

where N_{covered} is the number of the concept nodes used by the literals in L', and $N_{\text{redundant}}$ is the number of redundancies among the concept nodes of the literals. To account for the different scales between $\mathcal{A}$ and $\mathcal{H}$, we use the trade-off parameter λ in Equation 1 learned on the validation dataset.

Maximizing the objective function in Equation 1 is an NP-hard combinatorial optimization problem. However, we show that our objective function is submodular (see Appendix (Section 11) for the proof of submodularity). This means that there exists a greedy method that can provide a reliable approximation. GEOS greedily maximizes Equation 1 by starting from an empty set of literals and adding the next literal l_j that maximizes the gain of the objective function until the gain becomes negative (details of the algorithm and the gain function are explained in Figure 3 step 3).

7 Solver

We now have the best set of literals L^* from the optimization, and the high-confidence visual literals L_Δ from the diagram parser. In this step, GEOS determines if an assignment exists to the variables X in $L^* \cup L_\Delta$ that simultaneously satisfies all of the literals. This is known as the problem of automated geometry theorem proving in computational geometry (Alvin et al., 2014).

We use a numerical method to check the satisfiablity of literals. For each literal l_j in $L^* \cup L_\Delta$, we define a relaxed indicator function $g_j : S \mapsto z_j \in [-\infty, 0]$. The function $z_j = g_j(S)$ indicates the relaxed satisfiability of l_j given an assignment S to the variables X. The literal l_j is completely satisfied if $g_j(S) = 0$. We formulate the problem of satisfiability of literals as the task of finding the assignment S^* to X such that sum of all indicator functions $g_j(S^*)$ is maximized, i.e. $S^* = \arg\max_S \sum_j g_j(S)$. We use the basing-hopping algorithm (Wales and Doye, 1997) with sequential least squares programming (Kraft, 1988) to globally maximize the sum of the indicator functions. If there exists an assignment such that $\sum_j g_j(S) = 0$, then GEOS finds an assignment to X that satisfies all literals. If such assignment does not exist, then GEOS concludes that the literals are not satisfiable simultaneously. GEOS chooses to answer a geometry question if the literals of exactly one answer choice are simultaneously satisfiable.

8 Experimental Setup

Logical Language Ω: Ω consists of 13 types of entities and 94 function and predicates observed in our development set of geometry questions.

Implementation details: Sentences in geometry questions often contain in-line mathematical expressions, such as "If AB=x+5, what is x?". These mathematical expressions cause general purpose parsers to fail. GEOS uses an equation analyzer and pre-processes question text by replacing "=" with "equals", and replacing mathematical terms (e.g., "x+5") with a dummy noun so that the dependency parser does not fail.

GEOS uses Stanford dependency parser (Chen and Manning, 2014) to obtain syntactic information, which is used to compute features for relation identification (Table 1). For diagram parsing, similar to Seo et al. (2014), we assume that GEOS has access to ground truth optical character recognition for labels in the diagrams. For optimization, we tune the parameters λ to 0.5, based on the training examples.[4]

Dataset: We built a dataset of SAT plane geometry questions where every question has a tex-

[4] In our dataset, the number of all possible literals for each sentence is at most 1000.

	Total	Training	Practice	Official
Questions	186	67	64	55
Sentences	336	121	110	105
Words	4343	1435	1310	1598
Literals	577	176	189	212
Binary relations	337	110	108	119
Unary relations	437	141	150	146

Table 2: Data and annotation statistics

tual description in English accompanied by a diagram and multiple choices. Questions and answers are compiled from previous official SAT exams and practice exams offered by the College Board (Board, 2014). In addition, we use a portion of the publicly available high-school plane geometry questions (Seo et al., 2014) as our training set.

We annotate ground-truth logical forms for all questions in the dataset. Table 2 shows details of the data and annotation statistics. For evaluating dependency parsing, we annotate 50 questions with the ground truth dependency tree structures of all sentences in the questions. [5]

Baselines: *Rule-based text parsing* + GEOS *diagram* solves geometry questions using literals extracted from a manually defined set of rules over the textual dependency parser, and scored by diagram. For this baseline, we manually designed 12 high-precision rules based on the development set. Each rule compares the dependency tree of each sentence to pre-defined templates, and if a template pattern is matched, the rule outputs the relation or function structure corresponding to that template. For example, a rule assigns a relation *parent(child-1, child-2)* for a triplet of *(parent, child-1, child-2)* where *child-1* is the subject of *parent* and *child-2* is the object of the parent.

GEOS *without text parsing* solves geometry questions using a simple heuristic. With simple textual processing, this baseline extracts numerical relations from the question text and then computes the scale between the units in the question and the pixels in the diagram. This baseline rounds the number to the closest choice available in the multiple choices.

GEOS *without diagram parsing* solves geometry questions only relying on the literals interpreted from the text. It outputs all literals whose text scores are higher than a tuned threshold, 0.6 on the training set.

GEOS *without relation completion* solves geometry questions when text parsing does not use the intermediate representation and does not include the relation completion step.

9 Experiments

We evaluate our method on three tasks: solving geometry question, interpreting geometry questions, and dependency parsing.

Solving Geometry Questions: Table 3 compares the score of GEOS in solving geometry questions in practice and official SAT questions with that of baselines. SAT's grading scheme penalizes a wrong answer with a negative score of 0.25. We report the SAT score as the percentage of correctly answered questions penalized by the wrong answers. For official questions, GEOS answers 27 questions correctly, 1 questions incorrectly, and leaves 27 un-answered, which gives it a score of 26.75 out of 55, or 49%. Thus, GEOS's precision exceeds 96% on the 51% of questions that it chooses to answer. For practice SAT questions, GEOS scores 61%.[6]

In order to understand the effect of individual components of GEOS, we compare the full method with a few ablations. GEOS significantly outperforms the two baselines GEOS *without text parsing* and GEOS *without diagram parsing*, demonstrating that GEOS benefits from both text and diagram parsing. In order to understand the text parsing component, we compare GEOS with *Rule-based text parsing* + GEOS *Diagram* and GEOS *without relation completion*. The results show that our method of learning to interpret literals from the text is substantially better than the rule-based baseline. In addition, the relation completion step, which relies on the intermediate representation, helps to improve text interpretation.

Error Analysis: In order to understand the errors made by GEOS, we use *oracle text parsing* and *oracle diagram parsing* (Table 3). Roughly 38% of the errors are due to failures in text parsing, and about 46% of errors are due to failures in diagram parsing. Among them, about 15% of errors were due to failures in both diagram and text parsing. For an example of text parsing failure, the literals in Figure 6 (a) are not scored accurately due to missing coreference relations (Hajishirzi et al., 2013). The rest of errors are due to problems that require more complex reasoning (Figure 6 (b)).

[5] The source code, the dataset and the annotations are publicly available at `geometry.allenai.org`.

[6] Typically, 50th percentile (penalized) score in SAT math section is 27 out of 54 (50%).

	SAT score (%)	
Method	Practice	Official
GEOS w/o diagram parsing	7	5
GEOS w/o text parsing	10	10
Rule-based text parsing + GEOS diagram	31	24
GEOS w/o relation completion	42	33
GEOS	**61**	**49**
Oracle text parsing + GEOS diagram parsing	78	75
GEOS text parsing + oracle diagram parsing	81	79
Oracle text parsing + oracle diagram parsing	88	84

Table 3: SAT scores of solving geometry questions.

	P	R	F1
Rule-based text parsing	**0.99**	0.23	0.37
GEOS w/o diagram	0.57	**0.82**	0.67
GEOS	0.92	0.76	**0.83**

Table 4: Precision and recall of text interpretation.

Interpreting Question Texts: Table 4 details the precision and recall of GEOS in deriving literals for geometry question texts for official SAT questions. The rule-based text parsing baseline achieves a high precision, but at the cost of lower recall. On the other hand, the baseline GEOS *without diagram* achieves a high recall, but at the cost of lower precision. Nevertheless, GEOS attains substantially higher F1 score compared to both baselines, which is the key factor in solving the questions. Direct application of a generic semantic parser (Berant et al., 2013) with full supervision does not perform well in the geometry domain, mainly due to lack of enough training data. Our initial investigations show the performance of 33% F1 in the official set.

Improving Dependency Parsing: Table 5 shows the results of different methods in dependency parsing. GEOS returns a dependency parse tree by selecting the dependency tree that maximizes the text score in the objective function from the top 50 trees produced by a generic dependency parser, Stanford parser (Chen and Manning, 2014). Note that Stanford parser cannot handle mathematical symbols and equations. We report the results of a baseline that extends the Stanford dependency parser by adding a pre-processing step to separate the mathematical expressions from the plain sentences (Section 8).

We evaluate the performance of GEOS against the best tree returned by Stanford parser by reporting the fraction of the questions whose dependency parse structures match the ground truth annotations. Our results show an improvement of 16% over the Stanford dependency parser when equipped with the equation analyzer. For example, in "AB is perpendicular to CD at E", the Stanford dependency parser predicts that "E" depends on "CD", while GEOS predicts the correct parse in which "E" depends on "perpendicular".

	Accuracy
Stanford dep parse	0.05
Stanford dep parse + eq. analyzer	0.64
GEOS	**0.78**

Table 5: Accuracy of dependency parsing.

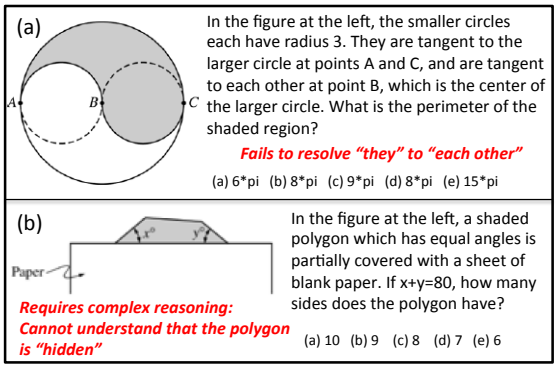


Figure 6: Examples of Failure: reasons are in red.

10 Conclusion

This paper introduced GEOS, an automated system that combines diagram and text interpretation to solve geometry problems. Solving geometry questions was inspired by two important trends in the current NLP literature. The first is in designing methods for grounded language acquisition to map text to a restricted formalism (instead of a full, domain independent representation). We demonstrate a new algorithm for learning to map text to a geometry language with a small amount of training data. The second is designing methods in coupling language and vision and show how processing multimodal information help improve textual or visual interpretations.

Our experiments on unseen SAT geometry problems achieve a score of 49% of official questions and a score of 61% on practice questions, providing a baseline for future work. Future work includes expanding the geometry language and the reasoning to address a broader set of geometry questions, reducing the amount of supervision, learning the relevant geometry knowledge, and scaling up the dataset.

Acknowledgements. The research was supported by the Allen Institute for AI, Allen Distinguished Investigator Award, and NSF (IIS-1352249). We thank Dan Weld, Luke Zettlemoyer, Aria Haghighi, Mark Hopkins, Eunsol Choi, and the anonymous reviewers for helpful comments.

References

Chris Alvin, Sumit Gulwani, Rupak Majumdar, and Supratik Mukhopadhyay. 2014. Synthesis of geometry proof problems. In *AAAI*.

Gabor Angeli and Christopher D. Manning. 2014. Naturalli: Natural logic inference for common sense reasoning. In *EMNLP*.

Yoav Artzi and Luke Zettlemoyer. 2013. Weakly supervised learning of semantic parsers for mapping instructions to actions. *TACL*, 1.

J. Berant and P. Liang. 2014. Semantic parsing via paraphrasing. In *ACL*.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *EMNLP*.

College Board. 2014. The college board.

Antoine Bordes, Nicolas Usunier, and Jason Weston. 2010. Label ranking under ambiguous supervision for learning semantic correspondences. In *ICML*.

SRK Branavan, Nate Kushman, Tao Lei, and Regina Barzilay. 2012. Learning high-level planning from text. In *ACL*.

William C. Bulko. 1988. Understanding text with an accompanying diagram. In *IEA/AIE*.

Danqi Chen and Christopher D Manning. 2014. A fast and accurate dependency parser using neural networks. In *EMNLP*.

David Chen, Joohyun Kim, and Raymond Mooney. 2010. Training a multilingual sportscaster: Using perceptual context to learn language. *JAIR*, 37.

Jim Cowie and Wendy Lehnert. 1996. Information extraction. *Communications of the ACM*, 39(1).

Aron Culotta and Jeffrey Sorensen. 2004. Dependency tree kernels for relation extraction. In *ACL*.

Jacob Eisenstein, James Clarke, Dan Goldwasser, and Dan Roth. 2009. Reading to learn: Constructing features from semantic abstracts. In *EMNLP*.

Thomas G Evans. 1964. A heuristic program to solve geometric-analogy problems. In *Proceedings of the April 21-23, 1964, spring joint computer conference*.

Hao Fang, Saurabh Gupta, Forrest Iandola, Rupesh Srivastava, Li Deng, Piotr Dollár, Jianfeng Gao, Xiaodong He, Margaret Mitchell, John Platt, et al. 2014. From captions to visual concepts and back. In *CVPR*.

Ali Farhadi, Mohsen Hejrati, Mohammad Amin Sadeghi, Peter Young, Cyrus Rashtchian, Julia Hockenmaier, and David Forsyth. 2010. Every picture tells a story: Generating sentences from images. In *ECCV*.

E.A. Feigenbaum and J. Feldman, editors. 1963. *Computers and Thought*. McGraw Hill, New York.

Jeffrey Flanigan, Sam Thomson, Jaime Carbonell, Chris Dyer, and Noah A. Smith. 2014. A discriminative graph-based parser for the abstract meaning representation. In *ACL*.

Ruifang Ge and Raymond J. Mooney. 2006. Discriminative reranking for semantic parsing. In *ACL*.

Dan Goldwasser and Dan Roth. 2011. Learning from natural instructions. In *IJCAI*.

Yunchao Gong, Liwei Wang, Micah Hodosh, Julia Hockenmaier, and Svetlana Lazebnik. 2014. Improving image-sentence embeddings using large weakly annotated photo collections. In *ECCV*.

Sonal Gupta and Raymond J. Mooney. 2010. Using closed captions as supervision for video activity recognition. In *AAAI*.

Hannaneh Hajishirzi, Julia Hockenmaier, Erik T. Mueller, and Eyal Amir. 2011. Reasoning about robocup soccer narratives. In *UAI*.

Hannaneh Hajishirzi, Leila Zilles, Daniel S Weld, and Luke S Zettlemoyer. 2013. Joint coreference resolution and named-entity linking with multi-pass sieves. In *EMNLP*.

Mary Hegarty and Marcel Adam Just. 1989. 10 understanding machines from text and diagrams. *Knowledge acquisition from text and pictures*.

Ben Hixon, Peter Clark, and Hannaneh Hajishirzi. 2015. Learning knowledge graphs for question answering through conversational dialog. In *NAACL*.

Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. 2014. Learning to solve arithmetic word problems with verb categorization. In *EMNLP*.

Rohit J. Kate and Raymond J. Mooney. 2007. Learning language semantics from ambiguous supervision. In *AAAI*.

Joohyun Kim and Raymond J. Mooney. 2013. Adapting discriminative reranking to grounded language learning. In *ACL*.

Dan Klein and Christopher D Manning. 2005. Parsing and hypergraphs. In *New developments in parsing technology*. Springer.

R Koncel-Kedziorski, Hannaneh Hajishirzi, and Ali Farhadi. 2014. Multi-resolution language grounding with weak supervision. In *EMNLP*.

Dieter et. al. Kraft. 1988. *A software package for sequential quadratic programming*. DFVLR Obersfaffeuhofen, Germany.

Girish Kulkarni, Visruth Premraj, Sagnik Dhar, Siming Li, Yejin Choi, Alexander C Berg, and Tamara L Berg. 2011. Baby talk: Understanding and generating image descriptions. In *CVPR*.

Nate Kushman, Yoav Artzi, Luke Zettlemoyer, and Regina Barzilay. 2014. Learning to automatically solve algebra word problems. In *ACL*.

T Kwiatkowski, E Choi, Y Artzi, and L Zettlemoyer. 2013. Scaling semantic parsers with on-the-fly ontology matching. In *EMNLP*.

Percy Liang, Michael I. Jordan, and Dan Klein. 2009. Learning semantic correspondences with less supervision. In *ACLAFNLP*.

Xinggang Lin, Shigeyoshi Shimotsuji, Michihiko Minoh, and Toshiyuki Sakai. 1985. Efficient diagram understanding with characteristic pattern detection. *CVGIP*, 30(1).

A. Lovett and K. Forbus. 2012. Modeling multiple strategies for solving geometric analogy problems. In *CCS*.

Gordon Novak. 1995. Diagrams for solving physical problems. *Diagrammatic reasoning: Cognitive and computational perspectives*.

Lawrence O'Gorman and Rangachar Kasturi. 1995. *Document image analysis*, volume 39. Citeseer.

Hoifung Poon and Pedro Domingos. 2009. Unsupervised semantic parsing. In *EMNLP*.

Siva Reddy, Mirella Lapata, and Mark Steedman. 2014. Large-scale semantic parsing without question-answer pairs. *TACL*, 2(Oct).

S. Roy, T. Vieira, and D. Roth. 2015. Reasoning about quantities in natural language.

Min Joon Seo, Hannaneh Hajishirzi, Ali Farhadi, and Oren Etzioni. 2014. Diagram understanding in geometry questions. In *AAAI*.

Rohini K Srihari. 1994. Computational models for integrating linguistic and visual information: A survey. *Artificial Intelligence Review*, 8(5-6).

Adam Vogel and Daniel Jurafsky. 2010. Learning to follow navigational directions. In *ACL*.

David J Wales and Jonathan PK Doye. 1997. Global optimization by basin-hopping and the lowest energy structures of lennard-jones clusters containing up to 110 atoms. *The Journal of Physical Chemistry A*, 101(28).

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *UAI*.

11 Appendix: Proof of Submodularity of Equation 1

We prove that the objective function in equation (1), $\lambda\mathcal{A}(L') + \mathcal{H}(L')$ is submodular by showing that $\mathcal{A}(L')$ and $\mathcal{H}(L')$ are submodular functions.

Submodularity of $\mathcal{A}$. Consider $L' \subset L$, and a new literal to be added, $l_i \in L \setminus L'$. By the definition of $\mathcal{A}$, it is clear that $\mathcal{A}(L' \cup \{l_j\}) = \mathcal{A}(L') + \mathcal{A}(\{l_j\})$. Hence, for all $L'' \subset L' \subset L$,

$$\mathcal{A}(L'' \cup \{l_j\}) - \mathcal{A}(L'') = \mathcal{A}(L' \cup \{l_j\}) - \mathcal{A}(L')$$

. Thus $\mathcal{A}$ is submodular.

Submodularity of $\mathcal{H}$. We prove that the coverage function, $\mathcal{H}_{\text{cov}}$, and the negation of the redundancy function, $-\mathcal{H}_{\text{red}}$ are submodular independently, and thus derive that their sum is submodular. For both, consider we are given $L'' \subset L' \subset L$, and a new literal $l_j \in L \setminus L'$. Also, let K'' and K' denote the the sets of concepts covered by L'' and L', respectively, and let K_j denote the set of concepts covered by l_j.

Coverage: Since $K'' \subset K'$, $|K'' \cup K_j| - |K''| \geq |K' \cup K_j| - |K'|$, which is equivalent to

$$\begin{aligned}&\mathcal{H}_{\text{cov}}(L'' \cup \{l_j\}) - \mathcal{H}_{\text{cov}}(L'') \\ &\quad \geq \mathcal{H}_{\text{cov}}(L' \cup \{l_j\}) - \mathcal{H}_{\text{cov}}(L')\end{aligned}$$

Redundancy: Note that $\mathcal{H}_{\text{red}}(L'' \cup \{l_j\}) - \mathcal{H}_{\text{red}}(L'') = |K'' \cap K_j|$, and similarly, $\mathcal{H}_{\text{red}}(L' \cup \{l_j\}) - \mathcal{H}_{\text{red}}(L') = |K' \cap K_j|$. Since $K'' \subset K'$, thus $|K'' \cap K_j| \leq |K' \cap K_j|$. Hence,

$$\begin{aligned}&\mathcal{H}_{\text{red}}(L'' \cup \{l_j\}) - \mathcal{H}_{\text{red}}(L'') \\ &\quad \leq \mathcal{H}_{\text{red}}(L' \cup \{l_j\}) - \mathcal{H}_{\text{red}}(L'),\end{aligned}$$

By negating both sides, we derive that the negation of the redundancy function is submodular.

Do You See What I Mean?
Visual Resolution of Linguistic Ambiguities

Yevgeni Berzak
CSAIL MIT
berzak@mit.edu

Andrei Barbu
CSAIL MIT
andrei@0xab.com

Daniel Harari
CSAIL MIT
hararid@mit.edu

Boris Katz
CSAIL MIT
boris@mit.edu

Shimon Ullman
Weizmann Institute of Science
shimon.ullman@weizmann.ac.il

Abstract

Understanding language goes hand in hand with the ability to integrate complex contextual information obtained via perception. In this work, we present a novel task for grounded language understanding: disambiguating a sentence given a visual scene which depicts one of the possible interpretations of that sentence. To this end, we introduce a new multimodal corpus containing ambiguous sentences, representing a wide range of syntactic, semantic and discourse ambiguities, coupled with videos that visualize the different interpretations for each sentence. We address this task by extending a vision model which determines if a sentence is depicted by a video. We demonstrate how such a model can be adjusted to recognize different interpretations of the same underlying sentence, allowing to disambiguate sentences in a unified fashion across the different ambiguity types.

1 Introduction

Ambiguity is one of the defining characteristics of human languages, and language understanding crucially relies on the ability to obtain unambiguous representations of linguistic content. While some ambiguities can be resolved using intra-linguistic contextual cues, the disambiguation of many linguistic constructions requires integration of world knowledge and perceptual information obtained from other modalities.

In this work, we focus on the problem of grounding language in the visual modality, and introduce a novel task for language understanding which requires resolving linguistic ambiguities by utilizing the visual context in which the linguistic content is expressed. This type of inference is frequently called for in human communication that occurs in a visual environment, and is crucial for language acquisition, when much of the linguistic content refers to the visual surroundings of the child (Snow, 1972).

Our task is also fundamental to the problem of grounding vision in language, by focusing on phenomena of linguistic ambiguity, which are prevalent in language, but typically overlooked when using language as a medium for expressing understanding of visual content. Due to such ambiguities, a superficially appropriate description of a visual scene may in fact not be sufficient for demonstrating a correct understanding of the relevant visual content. Our task addresses this issue by introducing a deep validation protocol for visual understanding, requiring not only providing a surface description of a visual activity but also demonstrating structural understanding at the levels of syntax, semantics and discourse.

To enable the systematic study of visually grounded processing of ambiguous language, we create a new corpus, *LAVA* (Language and Vision Ambiguities). This corpus contains sentences with linguistic ambiguities that can only be resolved using external information. The sentences are paired with short videos that visualize different interpretations of each sentence. Our sentences encompass a wide range of syntactic, semantic and dis-

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1477–1487,
Lisbon, Portugal, 17-21 September 2015.

course ambiguities, including ambiguous prepositional and verb phrase attachments, conjunctions, logical forms, anaphora and ellipsis. Overall, the corpus contains 237 sentences, with 2 to 3 interpretations per sentence, and an average of 3.37 videos that depict visual variations of each sentence interpretation, corresponding to a total of 1679 videos.

Using this corpus, we address the problem of selecting the interpretation of an ambiguous sentence that matches the content of a given video. Our approach for tackling this task extends the *sentence tracker* introduced in (Siddharth et al., 2014). The sentence tracker produces a score which determines if a sentence is depicted by a video. This earlier work had no concept of ambiguities; it assumed that every sentence had a single interpretation. We extend this approach to represent multiple interpretations of a sentence, enabling us to pick the interpretation that is most compatible with the video.

To summarize, the contributions of this paper are threefold. First, we introduce a new task for visually grounded language understanding, in which an ambiguous sentence has to be disambiguated using a visual depiction of the sentence's content. Second, we release a multimodal corpus of sentences coupled with videos which covers a wide range of linguistic ambiguities, and enables a systematic study of linguistic ambiguities in visual contexts. Finally, we present a computational model which disambiguates the sentences in our corpus with an accuracy of 75.36%.

2 Related Work

Previous language and vision studies focused on the development of multimodal word and sentence representations (Bruni et al., 2012; Socher et al., 2013; Silberer and Lapata, 2014; Gong et al., 2014; Lazaridou et al., 2015), as well as methods for describing images and videos in natural language (Farhadi et al., 2010; Kulkarni et al., 2011; Mitchell et al., 2012; Socher et al., 2014; Thomason et al., 2014; Karpathy and Fei-Fei, 2014; Siddharth et al., 2014; Venugopalan et al., 2015; Vinyals et al., 2015). While these studies handle important challenges in multimodal processing of language and vision, they do not provide explicit modeling of linguistic ambiguities.

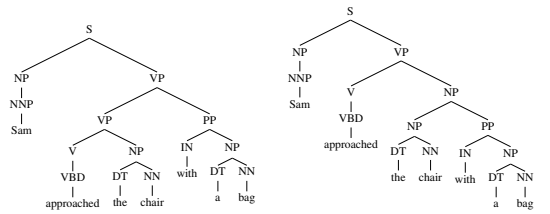
Previous work relating ambiguity in language to the visual modality addressed the problem of word sense disambiguation (Barnard et al., 2003). However, this work is limited to context independent interpretation of individual words, and does not consider structure-related ambiguities. Discourse ambiguities were previously studied in work on multimodal coreference resolution (Ramanathan et al., 2014; Kong et al., 2014). Our work expands this line of research, and addresses further discourse ambiguities in the interpretation of ellipsis. More importantly, to the best of our knowledge our study is the first to present a systematic treatment of syntactic and semantic sentence level ambiguities in the context of language and vision.

The interactions between linguistic and visual information in human sentence processing have been extensively studied in psycholinguistics and cognitive psychology (Tanenhaus et al., 1995). A considerable fraction of this work focused on the processing of ambiguous language (Spivey et al., 2002; Coco and Keller, 2015), providing evidence for the importance of visual information for linguistic ambiguity resolution by humans. Such information is also vital during language acquisition, when much of the linguistic content perceived by the child refers to their immediate visual environment (Snow, 1972). Over time, children develop mechanisms for grounded disambiguation of language, manifested among others by the usage of iconic gestures when communicating ambiguous linguistic content (Kidd and Holler, 2009). Our study leverages such insights to develop a complementary framework that enables addressing the challenge of visually grounded disambiguation of language in the realm of artificial intelligence.

3 Task

In this work we provide a concrete framework for the study of language understanding with visual context by introducing the task of grounded language disambiguation. This task requires to choose the correct linguistic representation of a sentence given a visual context depicted in a video. Specifically, provided with a sentence, n candidate interpretations of that sentence and a video that depicts the content of the sentence, one needs to choose the interpretation that corresponds to the content of the video.

To illustrate this task, consider the example in figure 1, where we are given the sentence "Sam approached the chair with a bag" along with two different linguistic interpretations. In the first in-

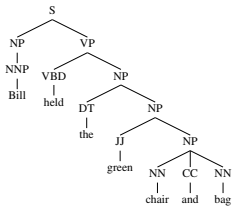


(a) First interpretation

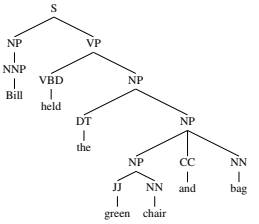
(b) Second interpretation

(c) Visual context

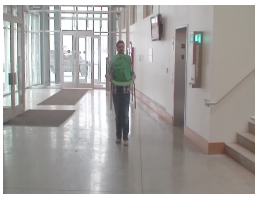
Figure 1: An example of the visually grounded language disambiguation task. Given the sentence "Sam approached the chair with a bag", two potential parses, (a) and (b), correspond to two different semantic interpretations. In the first interpretation Sam has the bag, while in the second reading the bag is on the chair. The task is to select the correct interpretation given the visual context (c).



(a) First interpretation



(b) Second interpretation



(c) First visual context

(d) Second visual context

Figure 2: Linguistic and visual interpretations of the sentence "Bill held the green chair and bag". In the first interpretation (a,c) both the chair and bag are green, while in the second interpretation (b,d) only the chair is green and the bag has a different color.

terpretation, which corresponds to parse 1(a), Sam has the bag. In the second interpretation associated with parse 1(b), the bag is on the chair rather than with Sam. Given the visual context from figure 1(c), the task is to choose which interpretation is most appropriate for the sentence.

4 Approach Overview

To address the grounded language disambiguation task, we use a compositional approach for determining if a specific interpretation of a sentence is depicted by a video. In this framework, described in detail in section 6, a sentence and an accompanying interpretation encoded in first order logic, give rise to a grounded model that matches a video against the provided sentence interpretation.

The model is comprised of Hidden Markov Models (HMMs) which encode the semantics of words, and trackers which locate objects in video frames. To represent an interpretation of a sentence, word models are combined with trackers through a cross-product which respects the semantic representation of the sentence to create a single model which recognizes that interpretation.

Given a sentence, we construct an HMM based representation for each interpretation of that sentence. We then detect candidate locations for objects in every frame of the video. Together the reforestation for the sentence and the candidate object locations are combined to form a model which can determine if a given interpretation is depicted by the video. We test each interpretation and report the interpretation with highest likelihood.

5 Corpus

To enable a systematic study of linguistic ambiguities that are grounded in vision, we compiled a corpus with ambiguous sentences describing visual actions. The sentences are formulated such that the correct linguistic interpretation of each sentence can only be determined using external, non-linguistic, information about the depicted activity. For example, in the sentence "Bill held the green chair and bag", the correct scope of "green" can only be determined by integrating additional information about the color of the bag. This information is provided in the accompanying videos, which visualize the possible interpretations of each sentence. Figure 2 presents the syntactic parses for this example along with frames from the respective videos. Although our videos contain visual uncertainty, they are not ambiguous with respect to the linguistic interpretation they are presenting, and hence a video always corresponds to a single candidate representation of a sentence.

The corpus covers a wide range of well

known syntactic, semantic and discourse ambiguity classes. While the ambiguities are associated with various types, different sentence interpretations always represent distinct sentence meanings, and are hence encoded semantically using first order logic. For syntactic and discourse ambiguities we also provide an additional, ambiguity type specific encoding as described below.

- **Syntax** Syntactic ambiguities include Prepositional Phrase (PP) attachments, Verb Phrase (VP) attachments, and ambiguities in the interpretation of conjunctions. In addition to logical forms, sentences with syntactic ambiguities are also accompanied with Context Free Grammar (CFG) parses of the candidate interpretations, generated from a deterministic CFG parser.

- **Semantics** The corpus addresses several classes of semantic quantification ambiguities, in which a syntactically unambiguous sentence may correspond to different logical forms. For each such sentence we provide the respective logical forms.

- **Discourse** The corpus contains two types of discourse ambiguities, Pronoun Anaphora and Ellipsis, offering examples comprising two sentences. In **anaphora** ambiguity cases, an ambiguous pronoun in the second sentence is given its candidate antecedents in the first sentence, as well as a corresponding logical form for the meaning of the second sentence. In **ellipsis** cases, a part of the second sentence, which can constitute either the subject and the verb, or the verb and the object, is omitted. We provide both interpretations of the omission in the form of a single unambiguous sentence, and its logical form, which combines the meanings of the first and the second sentences.

Table 2 lists examples of the different ambiguity classes, along with the candidate interpretations of each example.

The corpus is generated using Part of Speech (POS) tag sequence templates. For each template, the POS tags are replaced with lexical items from the corpus lexicon, described in table 3, using all the visually applicable assignments. This generation process yields an overall of 237 sentences,

	Ambiguity	Templates	#
Syntax	PP	NNP V DT [JJ] NN_1 IN DT [JJ] NN_2.	48
	VP	NNP_1 V [IN] NNP_2 V [JJ] NN.	60
	Conjunction	NNP_1 [and NNP_2] V DT JJ NN_1 and NN_2. NNP V DT NN_1 or DT NN_2 and DT NN_3.	40
Semantics	Logical Form	NNP_1 and NNP_2 V a NN. Someone V the NNS.	35
Discourse	Anaphora	NNP V DT NN_1 and DT NN_2. It is JJ.	36
	Ellipsis	NNP_1 V NNP_2. Also NNP_3.	18

Table 1: POS templates for generating the sentences in our corpus. The rightmost column represents the number of sentences in each category. The sentences are produced by replacing the POS tags with all the visually applicable assignments of lexical items from the corpus lexicon shown in table 3.

of which 213 sentences have 2 candidate interpretations, and 24 sentences have 3 interpretations. Table 1 presents the corpus templates for each ambiguity class, along with the number of sentences generated from each template.

The corpus videos are filmed in an indoor environment containing background objects and pedestrians. To account for the manner of performing actions, videos are shot twice with different actors. Whenever applicable, we also filmed the actions from two different directions (e.g. approach from the left, and approach from the right). Finally, all videos were shot with two cameras from two different view points. Taking these variations into account, the resulting video corpus contains 7.1 videos per sentence and 3.37 videos per sentence interpretation, corresponding to a total of 1679 videos. The average video length is 3.02 seconds (90.78 frames), with in an overall of 1.4 hours of footage (152434 frames).

A custom corpus is required for this task because no existing corpus, containing either videos or images, systematically covers multimodal ambiguities. Datasets such as *UCF Sports* (Rodriguez et al., 2008), *YouTube* (Liu et al., 2009), and *HMDB* (Kuehne et al., 2011) which come out of the activity recognition community are accompanied by action labels, not sentences, and do not control for the content of the videos aside from the principal action being performed. Datasets for image and video captioning, such as *MSCOCO* (Lin et al., 2014) and *TACOS* (Regneri et al., 2013),

	Ambiguity	Example	Linguistic interpretations	Visual setups
Syntax	PP	Claire left the green chair with a yellow bag.	Claire [left the green chair] [with a yellow bag].	The bag is with Claire.
			Claire left [the green chair with a yellow bag].	Bag is on the chair.
	VP	Claire looked at Bill picking up a chair.	Claire looked at [Bill [picking up a chair]].	Bill picks up the chair.
			Claire [looked at Bill] [picking up a chair].	Claire picks up the chair.
	Conjunction	Claire held a green bag and chair.	Claire held a [green [bag and chair]].	The chair is green.
			Claire held a [[green bag] and [chair]].	The chair is not green.
		Claire held the chair or the bag and the telescope.	Claire held [[the chair] or [the bag and the telescope]].	Claire holds the chair.
			Claire held [[the chair or the bag] and [the telescope]].	Claire holds the chair and the telescope.
Semantics	Logical Form	Claire and Bill moved a chair.	**chair**(x), **move**(Claire, x), **move**(Bill, x)	Claire and Bill move the same chair.
			chair(x), **chair**(y), **move**(Claire, x), **move**(Bill, y), $x \neq y$	Claire and Bill move different chairs.
		Someone moved the two chairs.	**chair**(x), **chair**(y), $x \neq y$, **person**(u), **move**(u, x), **move**(u, y)	One person moves both chairs.
			chair(x), **chair**(y), $x \neq y$, **person**(u), **person**(v), $u \neq v$, **move**(u, x), **move**(v, y)	Each chair moved by a different person.
Discourse	Anaphora	Claire held the bag and the chair. It is yellow.	It = bag	The bag is yellow.
			It = chair	The chair is yellow.
	Ellipsis	Claire looked at Bill. Also Sam.	Claire looked at Bill and Sam.	Claire looks at Bill and Sam.
			Claire and Sam looked at Bill.	Claire and Sam look at Bill.

Table 2: An overview of the different ambiguity types, along with examples of ambiguous sentences with their linguistic and visual interpretations. Note that similarly to semantic ambiguities, syntactic and discourse ambiguities are also provided with first order logic formulas for the resulting sentence interpretations. Table 4 shows additional examples for each ambiguity type, with frames from sample videos corresponding to the different interpretations of each sentence.

Syntactic Category	Visual Category	Words
Nouns	Objects, People	chair, bag, telescope, someone, proper names
Verbs	Actions	pick up, put down, hold, move (transitive), look at, approach, leave
Prepositions	Spacial Relations	with, left of, right of, on
Adjectives	Visual Properties	yellow, green

Table 3: The lexicon used to instantiate the templates in figure 1 in order to generate the corpus.

aim to control for more aspects of the videos than just the main action being performed but they do not provide the range of ambiguities discussed here. The closest dataset is that of Siddharth et al. (2014) as it controls for object appearance, color, action, and direction of motion, making it more likely to be suitable for evaluating disambiguation tasks. Unfortunately, that dataset was designed to avoid ambiguities, and therefore is not suitable for evaluating the work described here.

6 Model

To perform the disambiguation task, we extend the sentence recognition model of Siddharth et al. (2014) which represents sentences as compositions of words. Given a sentence, its first order logic interpretation and a video, our model produces a score which determines if the sentence is depicted by the video. It simultaneously tracks the participants in the events described by the sentence while recognizing the events themselves. This allows it to be flexible in the presence of noise by integrating top-down information from the sentence with bottom-up information from object and property detectors. Each word in the query sentence is represented by an HMM (Baum et al., 1970), which recognizes tracks (i.e. paths of detections in a video for a specific object) that satisfy the semantics of the given word. In essence, this model can be described as having two layers, one in which object tracking occurs and one in which words observe tracks and filter tracks that do not satisfy the word constraints.

Given a sentence interpretation, we construct a sentence-specific model which recognizes if a video depicts the sentence as follows. Each predicate in the first order logic formula has a corresponding HMM, which can recognize if that predicate is true of a video given its arguments. Each variable has a corresponding tracker which attempts to physically locate the bounding box corresponding to that variable in each frame of a

PP Attachment	Sam looked at Bill with a telescope.
VP Attachment	Bill approached the person holding a green chair.
Conjunction	Sam and Bill picked up the yellow bag and chair.
Logical Form	Someone put down the bags.
Anaphora	Sam picked up the bag and the chair. It is yellow.
Ellipsis	Sam left Bill. Also Clark.

Table 4: Examples of the six ambiguity classes described in table 2. The example sentences have at least two interpretations, which are depicted by different videos. Three frames from each such video are shown on the left and on the right below each sentence.

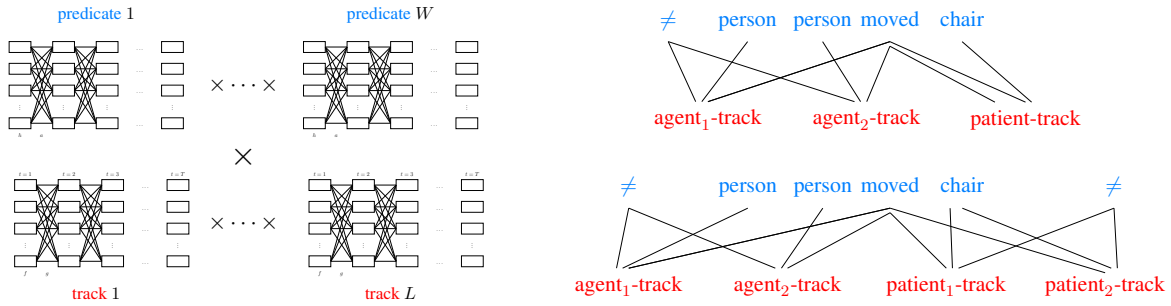


Figure 3: (left) Tracker lattices for every sentence participant are combined with predicate HMMs. The MAP estimate in the resulting cross-product lattice simultaneously finds the best tracks and the best state sequences for every predicate. (right) Two interpretations of the sentence "Claire and Bill moved a chair" having different first order logic formulas. The top interpretation corresponds to Bill and Claire moving the same chair, while the bottom one describes them moving different chairs. Predicates are highlighted in blue at the top and variables are highlighted in red at the bottom. Each predicate has a corresponding HMM which recognizes its presence in a video. Each variable has a corresponding tracker which locates it in a video. Lines connect predicates and the variables which fill their argument slots. Some predicates, such as *move* and $\neq$, take multiple arguments. Some predicates, such as *move*, are applied multiple times between different pairs of variables.

video. This creates a bipartite graph: HMMs that represent predicates are connected to trackers that represent variables. The trackers themselves are similar to the HMMs, in that they comprise a lattice of potential bounding boxes in every frame. To construct a joint model for a sentence interpretation, we take the cross product of HMMs and trackers, taking only those cross products dictated by the structure of the formula corresponding to the desired interpretation. Given a video, we employ an object detector to generate candidate detections in each frame, construct trackers which select one of these detections in each frame, and finally construct the overall model from HMMs and trackers.

Provided an interpretation and its corresponding formula composed of P predicates and V variables, along with a collection of object detections, $b^{\text{frame}}_{\text{detection index}}$, in each frame of a video of length T the model computes the score of the video-sentence pair by finding the optimal detection for each participant in every frame. This is in essence the Viterbi algorithm (Viterbi, 1971), the MAP algorithm for HMMs, applied to finding optimal object detections $j^{\text{frame}}_{\text{variable}}$ for each participant, and the optimal state $k^{\text{frame}}_{\text{predicate}}$ for each predicate HMM, in every frame. Each detection is scored by its confidence from the object detector, f and each object track is scored by a motion coherence metric g which determines if the motion of the track agrees with the underlying optical flow. Each predicate, p, is scored by the probability of observing a particular detection in a given state h_p, and by the probability of transitioning between states a_p. The structure of the formula and the fact that multiple predicates often refer to the same variables is recorded by θ, a mapping between predicates and their arguments. The model computes the MAP estimate as:

$$\max_{j_1^1,\ldots,j_1^T} \cdots \max_{j_V^1,\ldots,j_V^T} \max_{k_1^1,\ldots,k_1^T} \cdots \max_{k_P^1,\ldots,k_P^T} \sum_{v=1}^{V}\sum_{t=1}^{T} f(b^t_{j_v^t}) + \sum_{t=2}^{T} g(b^{t-1}_{j_v^{t-1}}, b^t_{j_v^t}) + \sum_{p=1}^{P}\sum_{t=1}^{T} h_p(k_p^t, b^t_{j^t_{\theta_p^1}}, b^t_{j^t_{\theta_p^2}}) + \sum_{t=2}^{T} a_p(k_p^{t-1}, k_p^t)$$

for sentences which have words that refer to at most two tracks (i.e. transitive verbs or binary predicates) but is trivially extended to arbitrary arities. Figure 3 provides a visual overview of the model as a cross-product of tracker models and word models.

Our model extends the approach of Siddharth et al. (2014) in several ways. First, we depart from the dependency based representation used in that work, and recast the model to encode first order logic formulas. Note that some complex first order logic formulas cannot be directly encoded in the model and require additional inference steps. This extension enables us to represent ambiguities in which a given sentence has multiple logical interpretations for the same syntactic parse.

Second, we introduce several model components which are not specific to disambiguation, but are required to encode linguistic constructions that are present in our corpus and could not be handled by the model of Siddharth et al. (2014). These new components are the predicate "not equal", disjunction, and conjunction. The key addition among these components is support for the new predicate "not equal", which enforces that two tracks, i.e. objects, are distinct from each other. For example, in the sentence "Claire and Bill moved a chair" one would want to ensure that the two movers are distinct entities. In earlier work, this was not required because the sentences tested in that work were designed to distinguish objects based on constraints rather than identity. In other words, there might have been two different people but they were distinguished in the sentence by their actions or appearance. To faithfully recognize that two actors are moving the chair in the earlier example, we must ensure that they are disjoint from each other. In order to do this we create a new HMM for this predicate, which assigns low probability to tracks that heavily overlap, forcing the model to fit two different actors in the previous example. By combining the new first order logic based semantic representation in lieu of a syntactic representation with a more expressive model, we can encode the sentence interpretations required to perform the disambiguation task.

Figure 3(left) shows an example of two different interpretations of the above discussed sentence "Claire and Bill moved a chair". Object trackers, which correspond to variables in the first order logic representation of the sentence interpretation, are shown in red. Predicates which constrain the possible bindings of the trackers, corresponding to predicates in the representation of the sentence, are shown in blue. Links represent the argument structure of the first order logic formula, and determine the cross products that are taken between the predicate HMMs and tracker lattices in order to form the joint model which recognizes the entire interpretation in a video.

The resulting model provides a single unified formalism for representing all the ambiguities in table 2. Moreover, this approach can be tuned to different levels of specificity. We can create models that are specific to one interpretation of a sentence or that are generic, and accept multiple interpretations by eliding constraints that are not common between the different interpretations. This allows the model, like humans, to defer deciding on a particular interpretation or to infer that multiple interpretation of the sentence are plausible.

7 Experimental Results

We tested the performance of the model described in the previous section on the *LAVA* dataset presented in section 5. Each video in the dataset was pre-processed with object detectors for humans, bags, chairs, and telescopes. We employed a mixture of CNN (Krizhevsky et al., 2012) and DPM (Felzenszwalb et al., 2010) detectors, trained on held out sections of our corpus. For each object class we generated proposals from both the CNN and the DPM detectors, and trained a scoring function to map both results into the same space. The scoring function consisted of a sigmoid over the confidence of the detectors trained on the same held out portion of the training set. As none of the disambiguation examples discussed here rely on the specific identity of the actors, we did not detect their identity. Instead, any sentence which contains names was automatically converted to one which contains arbitrary "person" labels.

The sentences in our corpus have either two or three interpretations. Each interpretation has one or more associated videos where the scene was shot from a different angle, carried out either by different actors, with different objects, or in different directions of motion. For each sentence-video pair, we performed a 1-out-of-2 or 1-out-of-3 classification task to determine which of the interpretations of the corresponding sentence best fits that video. Overall chance performance on our dataset is 49.04%, slightly lower than 50% due to the 1-out-of-3 classification examples.

The model presented here achieved an accuracy of 75.36% over the entire corpus averaged across all error categories. This demonstrates that the model is largely capable of capturing the underlying task and that similar compositional cross-modal models may do the same. For each of the 3 major ambiguity classes we had an accuracy of 84.26% for syntactic ambiguities, 72.28% for semantic ambiguities, and 64.44% for discourse ambiguities.

The most significant source of model failures are poor object detections. Objects are often rotated and presented at angles that are difficult to recognize. Certain object classes like the telescope

are much more difficult to recognize due to their small size and the fact that hands tend to largely occlude them. This accounts for the degraded performance of the semantic ambiguities relative to the syntactic ambiguities, as many more semantic ambiguities involved the telescope. Object detector performance is similarly responsible for the lower performance of the discourse ambiguities which relied much more on the accuracy of the person detector as many sentences involve only people interacting with each other without any additional objects. This degrades performance by removing a helpful constraint for inference, according to which people tend to be close to the objects they are manipulating. In addition, these sentences introduced more visual uncertainty as they often involved three actors.

The remaining errors are due to the event models. HMMs can fixate on short sequences of events which seem as if they are part of an action, but in fact are just noise or the prefix of another action. Ideally, one would want an event model which has a global view of the action, if an object went up from the beginning to the end of the video while a person was holding it, it's likely that the object was being picked up. The event models used here cannot enforce this constraint, they merely assert that the object was moving up for some number of frames; an event which can happen due to noise in the object detectors. Enforcing such local constraints instead of the global constraint of the motion of the object over the video makes joint tracking and event recognition tractable in the framework presented here but can lead to errors. Finding models which strike a better balance between local information and global constraints while maintaining tractable inference remains an area of future work.

8 Conclusion

We present a novel framework for studying ambiguous utterances expressed in a visual context. In particular, we formulate a new task for resolving structural ambiguities using visual signal. This is a fundamental task for humans, involving complex cognitive processing, and is a key challenge for language acquisition during childhood. We release a multimodal corpus that enables to address this task, as well as support further investigation of ambiguity related phenomena in visually grounded language processing. Finally, we present a unified approach for resolving ambiguous descriptions of videos, achieving good performance on our corpus.

While our current investigation focuses on structural *inference*, we intend to extend this line of work to *learning* scenarios, in which the agent has to deduce the meaning of words and sentences from structurally ambiguous input. Furthermore, our framework can be beneficial for image and video retrieval applications in which the query is expressed in natural language. Given an ambiguous query, our approach will enable matching and clustering the retrieved results according to the different query interpretations.

Acknowledgments

This material is based upon work supported by the Center for Brains, Minds, and Machines (CBMM), funded by NSF STC award CCF-1231216. SU was also supported by ERC Advanced Grant 269627 Digital Baby.

References

Kobus Barnard, Matthew Johnson, and David Forsyth. 2003. Word sense disambiguation with pictures. In *Proceedings of the HLT-NAACL 2003 workshop on Learning word meaning from non-linguistic data-Volume 6*, pages 1–5. Association for Computational Linguistics.

L. E. Baum, T. Petrie, G. Soules, and N. Weiss. 1970. A maximization technique occuring in the statistical analysis of probabilistic functions of Markov chains. *The Annals of Mathematical Statistics*, 41(1):164–171.

Elia Bruni, Gemma Boleda, Marco Baroni, and Nam-Khanh Tran. 2012. Distributional semantics in technicolor. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 136–145. Association for Computational Linguistics.

Moreno I Coco and Frank Keller. 2015. The interaction of visual and linguistic saliency during syntactic ambiguity resolution. *The Quarterly Journal of Experimental Psychology*, 68(1):46–74.

Ali Farhadi, Mohsen Hejrati, Mohammad Amin Sadeghi, Peter Young, Cyrus Rashtchian, Julia Hockenmaier, and David Forsyth. 2010. Every picture tells a story: Generating sentences from images. In *Computer Vision–ECCV 2010*, pages 15–29. Springer.

Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan. 2010. Object detection with discriminatively trained part-based models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(9):1627–1645.

Yunchao Gong, Liwei Wang, Micah Hodosh, Julia Hockenmaier, and Svetlana Lazebnik. 2014. Improving image-sentence embeddings using large weakly annotated photo collections. In *Computer Vision–ECCV 2014*, pages 529–545. Springer.

Andrej Karpathy and Li Fei-Fei. 2014. Deep visual-semantic alignments for generating image descriptions. *arXiv preprint arXiv:1412.2306*.

Evan Kidd and Judith Holler. 2009. Children's use of gesture to resolve lexical ambiguity. *Developmental Science*, 12(6):903–913.

Chen Kong, Dahua Lin, Mayank Bansal, Raquel Urtasun, and Sanja Fidler. 2014. What are you talking about? text-to-image coreference. In *Computer Vision and Pattern Recognition (CVPR)*, pages 3558–3565. IEEE.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

Hildegard Kuehne, Hueihan Jhuang, Estíbaliz Garrote, Tomaso Poggio, and Thomas Serre. 2011. Hmdb: a large video database for human motion recognition. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2556–2563. IEEE.

G Kulkarni, V Premraj, S Dhar, Siming Li, Yejin Choi, AC Berg, and TL Berg. 2011. Baby talk: Understanding and generating simple image descriptions. In *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1601–1608. IEEE Computer Society.

Angeliki Lazaridou, Nghia The Pham, and Marco Baroni. 2015. Combining language and vision with a multimodal skip-gram model. *CoRR*, abs/1501.02598.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014*, pages 740–755. Springer.

Jingen Liu, Jiebo Luo, and Mubarak Shah. 2009. Recognizing realistic actions from videos in the wild. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1996–2003. IEEE.

Margaret Mitchell, Xufeng Han, Jesse Dodge, Alyssa Mensch, Amit Goyal, Alex Berg, Kota Yamaguchi, Tamara Berg, Karl Stratos, and Hal Daumé III. 2012. Midge: Generating image descriptions from computer vision detections. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 747–756. Association for Computational Linguistics.

Vignesh Ramanathan, Armand Joulin, Percy Liang, and Li Fei-Fei. 2014. Linking people in videos with their names using coreference resolution. In *Computer Vision–ECCV 2014*, pages 95–110. Springer.

Michaela Regneri, Marcus Rohrbach, Dominikus Wetzel, Stefan Thater, Bernt Schiele, and Manfred Pinkal. 2013. Grounding action descriptions in videos. *Transactions of the Association for Computational Linguistics*, 1:25–36.

Mikel D. Rodriguez, Javed Ahmed, and Mubarak Shah. 2008. Action MACH A Spatio-temporal Maximum Average Correlation Height Filter for Action Recognition. In *Computer Vision and Pattern Recognition*, pages 1–8.

Narayanaswamy Siddharth, Andrei Barbu, and Jeffrey Mark Siskind. 2014. Seeing what you're told: Sentence-guided activity recognition in video. In *Computer Vision and Pattern Recognition (CVPR)*, pages 732–739. IEEE.

Carina Silberer and Mirella Lapata. 2014. Learning grounded meaning representations with autoencoders. In *Proceedings of ACL*, pages 721–732.

Catherine E Snow. 1972. Mothers' speech to children learning language. *Child development*, pages 549–565.

Richard Socher, Milind Ganjoo, Christopher D Manning, and Andrew Ng. 2013. Zero-shot learning through cross-modal transfer. In *Advances in Neural Information Processing Systems*, pages 935–943.

Richard Socher, Andrej Karpathy, Quoc V Le, Christopher D Manning, and Andrew Y Ng. 2014. Grounded compositional semantics for finding and describing images with sentences. *Transactions of the Association for Computational Linguistics*, 2:207–218.

Michael J Spivey, Michael K Tanenhaus, Kathleen M Eberhard, and Julie C Sedivy. 2002. Eye movements and spoken language comprehension: Effects of visual context on syntactic ambiguity resolution. *Cognitive psychology*, 45(4):447–481.

Michael K Tanenhaus, Michael J Spivey-Knowlton, Kathleen M Eberhard, and Julie C Sedivy. 1995. Integration of visual and linguistic information in spoken language comprehension. *Science*, 268(5217):1632–1634.

Jesse Thomason, Subhashini Venugopalan, Sergio Guadarrama, Kate Saenko, and Raymond Mooney. 2014. Integrating language and vision to generate natural language descriptions of videos in the wild. In *Proceedings of the 25th International Conference on Computational Linguistics (COLING), August*.

Subhashini Venugopalan, Huijuan Xu, Jeff Donahue, Marcus Rohrbach, Raymond Mooney, and Kate Saenko. 2015. Translating videos to natural language using deep recurrent neural networks. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Denver, Colorado.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural image caption generator. In *Computer Vision and Pattern Recognition (CVPR)*.

A. J. Viterbi. 1971. Convolutional codes and their performance in communication systems. *Communications of the IEEE*, 19:751–772, October.

Efficient and Expressive Knowledge Base Completion Using Subgraph Feature Extraction

Matt Gardner and Tom Mitchell
Carnegie Mellon University
mg1@cs.cmu.edu, tom.mitchell@cmu.edu

Abstract

We explore some of the practicalities of using random walk inference methods, such as the Path Ranking Algorithm (PRA), for the task of knowledge base completion. We show that the random walk probabilities computed (at great expense) by PRA provide no discernible benefit to performance on this task, so they can safely be dropped. This allows us to define a simpler algorithm for generating feature matrices from graphs, which we call subgraph feature extraction (SFE). In addition to being conceptually simpler than PRA, SFE is much more efficient, reducing computation by an order of magnitude, and more expressive, allowing for much richer features than paths between two nodes in a graph. We show experimentally that this technique gives substantially better performance than PRA and its variants, improving mean average precision from .432 to .528 on a knowledge base completion task using the NELL KB.

1 Introduction

Knowledge bases (KBs), such as Freebase (Bollacker et al., 2008), NELL (Mitchell et al., 2015), and DBPedia (Mendes et al., 2012) contain large collections of facts about things, people, and places in the world. These knowledge bases are useful for various tasks, including training relation extractors and semantic parsers (Hoffmann et al., 2011; Krishnamurthy and Mitchell, 2012), and question answering (Berant et al., 2013). While these knowledge bases may be very large, they are still quite incomplete, missing large percentages of facts about common or popular entities (West et al., 2014; Choi et al., 2015). The task of knowledge base completion—filling in missing facts by examining the facts already in the KB, or by looking in a corpus—is one attempt to mitigate the problems of this knowledge sparsity.

In this work we examine one method for performing knowledge base completion that is currently in use: the Path Ranking Algorithm (PRA) (Lao et al., 2011; Dong et al., 2014). PRA is a method for performing link prediction in a graph with labeled edges by computing feature matrices over node pairs in the graph. The method has a strong connection to logical inference (Gardner et al., 2015), as the feature space considered by PRA consists of a restricted class of horn clauses found in the graph. While PRA can be applied to any link prediction task in a graph, its primary use has been in KB completion (Lao et al., 2012; Gardner et al., 2013; Gardner et al., 2014).

PRA is a two-step process, where the first step finds potential path types between node pairs to use as features in a statistical model, and the second step computes random walk probabilities associated with each path type and node pair (these are the values in a feature matrix). This second step is very computationally intensive, requiring time proportional to the average out-degree of the graph to the power of the path length for *each cell* in the computed feature matrix. In this paper we consider whether this computational effort is well-spent, or whether we might more profitably spend computation in other ways. We propose a new way of generating feature matrices over node pairs in a graph that aims to improve both the efficiency and the expressivity of the model relative to PRA.

Our technique, which we call *subgraph feature extraction* (SFE), is similar to only doing the first step of PRA. Given a set of node pairs in a graph, we first do a local search to characterize the graph around each node. We then run a set of feature extractors over these local subgraphs to obtain feature vectors for each node pair. In the simplest case, where the feature extractors only

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1488–1498,
Lisbon, Portugal, 17-21 September 2015.

look for paths connecting the two nodes, the feature space is equivalent to PRA's, and this is the same as running PRA and binarizing the resultant feature vectors. However, because we do not have to compute random walk probabilities associated with each path type in the feature matrix, we can extract much more expressive features, including features which are not representable as paths in the graph at all. In addition, we can do a more exhaustive search to characterize the local graph, using a breadth-first search instead of random walks. SFE is a much simpler method than PRA for obtaining feature matrices over node pairs in a graph. Despite its simplicity, however, we show experimentally that it substantially outperforms PRA, both in terms of running time and prediction performance. SFE decreases running time over PRA by an order of magnitude, it improves mean average precision from .432 to .528 on the NELL KB, and it improves mean reciprocal rank from .850 to .933.

In the remainder of this paper, we first describe PRA in more detail. We then situate our methods in the context of related work, and provide additional experimental motivation for the improvements described in this paper. We then formally define SFE and the feature extractors we used, and finally we present an experimental comparison between PRA and SFE on the NELL KB. The code and data used in this paper is available at http://rtw.ml.cmu.edu/emnlp2015_sfe/.

2 The Path Ranking Algorithm

The path ranking algorithm was introduced by Lao and Cohen (2010). It is a two-step process for generating a feature matrix over node pairs in a graph. The first step finds a set of potentially useful path types that connect the node pairs, which become the columns of the feature matrix. The second step then computes the values in the feature matrix by finding random walk probabilities as described below. Once the feature matrix has been computed, it can be used with whatever classification model is desired (or even incorporated as one of many factors in a structured prediction model), though almost all prior work with PRA simply uses logistic regression.

More formally, consider a graph $\mathcal{G}$ with nodes $\mathcal{N}$, edges $\mathcal{E}$, and edge labels $\mathcal{R}$, and a set of node pairs $(s_j, t_j) \in \mathcal{D}$ that are instances of some relationship of interest. PRA will generate a feature vector for each (s_j, t_j) pair, where each feature is some sequence of edge labels -e_1-e_2-. . .-e_l-. If the edge sequence, or *path type*, corresponding to the feature exists between the source and target nodes in the graph, the value of that feature in the feature vector will be non-zero.

Because the feature space considered by PRA is so large,[1] and because computing the feature values is so computationally intensive, the first step PRA must perform is *feature selection*, which is done using random walks over the graph. In this step of PRA, we find path types π that are likely to be useful in predicting new instances of the relation represented by the input node pairs. These path types are found by performing random walks on the graph $\mathcal{G}$ starting at the source and target nodes in $\mathcal{D}$ and recording which paths connect some source node with its target.[2] Note that these are *two-sided*, *unconstrained* random walks: the walks from sources and targets can be joined on intermediate nodes to get a larger set of paths that connect the source and target nodes. Once connectivity statistics have been computed in this way, k path types are selected as features. Lao et al. (2011) use measures of the precision and recall of each feature in this selection, while Gardner et al. (2014) simply pick those most frequently seen.

Once a set of path features has been selected, the second step of PRA is to compute values for each cell in the feature matrix. Recall that rows in this matrix correspond to node pairs, and the columns correspond to the path types found in the first step. The cell value assigned by PRA is the probability of arriving at the target node of a node pair, given that a random walk began at the source node and was constrained to follow the path type: $\mathrm{p}(t|s, \pi)$. There are several ways of computing this probability. The most straightforward method is to use a path-constrained breadth-first search to exhaustively enumerate all possible targets given a source node and a path type, count how frequently each target is seen, and normalize the distribution. This calculates the desired probability exactly, but at the cost of doing a breadth-first search (with

[1] The feature space consists of the set of all possible edge label sequences, with cardinality $\sum_{i=1}^{l} |\mathcal{R}|^i$, assuming a bound l on the maximum path length.

[2] A deterministic algorithm, such as a breadth-first search, could obviously be used here instead of random walks, and indeed Lao's original work did use a more exhaustive search. However, when moving to the larger graphs corresponding to the NELL and Freebase KBs, Lao (2011) (and all future work) switched to using random walks, because the graph was too large.

complexity proportional to the average per-edge-label out-degree to the power of the path length) per source node per path type.

There are three methods that can potentially reduce the computational complexity of this probability calculation. The first is to use random walks to approximate the probability via rejection sampling: for each path type and source node, a number of random walks are performed, attempting to follow the edge sequence corresponding to the path type. If a node is reached where it is no longer possible to follow the path type, the random walk is restarted. This does not reduce the time necessary to get an arbitrarily good approximation, but it does allow us to decrease computation time, even getting a fixed complexity, at the cost of accepting some error in our probability estimates. Second, Lao (2012) showed that when the target node of a query is known, the exponent can be cut in half by using a two-sided BFS. In this method, some careful bookkeeping is done with dynamic programming such that the probability can be computed correctly when the two-sided search meets at an intermediate node. Lao's dynamic programming technique is only applicable when the target node is known, however, and only cuts the exponent in half—this is still quite computationally intensive. Lastly, we could replace the BFS with a multiplication of adjacency matrices, which performs the same computation. The efficiency gain comes from the fact that we can just do the multiplication once per path type, instead of once per path type per source node. However, to correctly compute the probabilities for a (source, target) pair, we need to exclude from the graph the edge connecting that training instance. This means that the matrix computed for each path type *should* be different for each training instance, and so we either lose our efficiency gain or we accept incorrect probability estimates. In this work we use the rejection sampling technique.

As mentioned above, once the feature matrix has been computed in the second step of PRA, one can use any kind of classifier desired to learn a model and make predictions on test data.

3 Related Work

The task of knowledge base completion has seen a lot of attention in recent years, with entire workshops devoted to it (Suchanek et al., 2013). We will touch on three broad categories related to KB completion: the task of relation extraction, embedding methods for KB completion, and graph methods for KB completion.

Relation extraction. Relation extraction and knowledge base completion have the same goal: to predict new instances of relations in a formal knowledge base such as Freebase or NELL. The difference is that relation extraction focuses on determining what relationship is expressed by a particular sentence, while knowledge base completion tries to predict which relationships hold between which entities. A relation extraction system can be used for knowledge base completion, but typical KB completion methods do not make predictions on single sentences. This is easily seen in the line of work known as distantly-supervised relation extraction (Mintz et al., 2009; Hoffmann et al., 2011; Surdeanu et al., 2012); these models use the relation instances in a knowledge base as their only supervision, performing some heuristic mapping of the entities in text to the knowledge base, then using that mapping to train extractors for each relation in the KB. The cost of using these methods is that is it generally difficult to incorporate richer features from the knowledge base when predicting whether a particular sentence expresses a relation, and so techniques that make fuller use of the KB can perform better on the KB completion task (Weston et al., 2013; Riedel et al., 2013).

Embedding methods for KB completion. There has been much recent work that attempts to perform KB completion by learning an embedded representation of entities and relations in the KB and then using these representations to infer missing relationships. Some of earliest work along these lines were the RESCAL model (Nickel et al., 2011) and Structured Embeddings (Bordes et al., 2011). These were soon followed by TransE (Bordes et al., 2013), Neural Tensor Networks (Socher et al., 2013), and many variants on all of these algorithms (Chang et al., 2014; García-Durán et al., 2014; Wang et al., 2014). These methods perform well when there is structural redundancy in the knowledge base tensor, but when the tensor (or individual relations in the tensor) has high rank, learning good embeddings can be challenging. The ARE model (Nickel et al., 2014) attempted to address this by only making the embeddings capture the residual of the tensor that cannot be readily predicted from the graph-based techniques mentioned below.

Dataset	Method	MAP
Freebase	Probabilities	.337
	Binarized	.344
NELL	Probabilities	.303
	Binarized	.319

Table 1: Using binary feature values instead of random walk probabilities gives statistically indistinguishable performance. The p-value on the Freebase data is .55, while it is .25 for NELL.

Graph-based methods for KB completion. A separate line of research into KB completion can be broadly construed as performing some kind of inference over graphs in order to predict missing instances in a knowledge base. Markov logic networks (Richardson and Domingos, 2006) fall into this category, as does ProPPR (Wang et al., 2013) and many other logic-based systems. PRA, the main subject of this paper, also fits in this line of work. Work specifically with PRA has ranged from incorporating a parsed corpus as additional evidence when doing random walk inference (Lao et al., 2012), to introducing better representations of the text corpus (Gardner et al., 2013; Gardner et al., 2014), and using PRA in a broader context as part of Google's Knowledge Vault (Dong et al., 2014). An interesting piece of work that combines embedding methods with graph-based methods is that of Neelakantan et al. (2015), which uses a recursive neural network to create embedded representations of PRA-style paths.

4 Motivation

We motivate our modifications to PRA with three observations. First, it appears that binarizing the feature matrix produced by PRA, removing most of the information gained in PRA's second step, has no significant impact on prediction performance in knowledge base completion tasks. We show this on the NELL KB and the Freebase KB in Table 1.[3] The fact that random walk probabilities carry no additional information for this task over binary features is surprising, and it shows that the second step of PRA spends a lot of its computation for no discernible gain in performance.

Second, Neelakantan et al. (2015) presented experiments showing a substantial increase in performance from using a much larger set of features in a PRA-like model.[4] All of their experiments used binary features, so this is not a direct comparison of random walk probabilities versus binarized features, but it shows that increasing the feature size beyond the point that is computationally feasible with random walk probabilities seems useful. Additionally, they showed that using path bigram features, where each sequential pair of edges types in each path was added as an additional feature to the model, gave a significant increase in performance. These kind of features are not representable in the traditional formulation of PRA.

Lastly, the method used to compute the random walk probabilities—rejection sampling—makes the inclusion of more expressive features problematic. Consider the path bigrams mentioned above; one could conceivably compute a probability for a path type that only specifies that the last edge type in the path must be r, but it would be incredibly inefficient with rejection sampling, as most of the samples would end up rejected (leaving aside the additional issues of an unspecified path length). In contrast, if the features simply signify whether a particular path type exists in the graph, without any associated probability, these kinds of features are very easy to compute.

Given this motivation, our work attempts to improve both the efficiency and the expressivity of PRA by removing the second step of the algorithm. Efficiency is improved because the second step is the most computationally expensive, and expressivity is improved by allowing features that cannot be reasonably computed with rejection sampling. We show experimentally that the techniques we introduce do indeed improve performance quite substantially.

5 Subgraph Feature Extraction

In this section we discuss how SFE constructs feature matrices over node pairs in a graph using just a single search over the graph for each node (which is comparable to only using the first step of PRA). As outlined in Section 2, the first step of PRA does a series of random walks from each source and target node (s_j, t_j) in a dataset $\mathcal{D}$. In

[3]The NELL data and experimental protocol is described in Section 6.1. The Freebase data consists of 24 relations from the Freebase KB; we used the same data used by Gardner et al. (2014).

[4]Note that while Neelakantan et al. called the baseline they were comparing to "PRA", they only used the first step of the algorithm to produce path types, and thus did not really compare against PRA per se. It is their version of "PRA" that we formalize and expand as SFE in this work.

PRA these random walks are used to find a relatively small set of potentially useful path types for which more specific random walk probabilities are then computed, at great expense. In our method, subgraph feature extraction (SFE), we stop after this first set of random walks and instead construct a binary feature matrix.

More formally, for each node n in the data (where n could be either a source node or a target node), SFE constructs a subgraph centered around that node using k random walks. Each random walk that leaves n follows some path type π and ends at some intermediate node i. We keep all of these (π, i) pairs as the characterization of the subgraph around n, and we will refer to this subgraph as G_n. To construct a feature vector for a source-target pair (s_j, t_j), SFE takes the subgraphs G_{s_j} and G_{t_j} and merges them on the intermediate nodes i. That is, if an intermediate node i is present in both G_{s_j} and G_{t_j}, SFE takes the path types π corresponding to i and combines them (reversing the path type coming from the target node t_j). If some intermediate node for the source s_j happens to be t_j, no combination of path types is necessary (and similarly if an intermediate node for the target t_j is s_j—the path only needs to be reversed in this case). This creates a feature space that is exactly the same as that constructed by PRA: sequences of edge types that connect a source node to a target node. To construct the feature vector SFE just takes all of these combined path types as binary features for (s_j, t_j). Note, however, that we need not restrict ourselves to only using the same feature space as PRA; Section 5.1 will examine extracting more expressive features from these subgraphs.

This method for generating a feature matrix over node pairs in a graph is much simpler and less computationally expensive than PRA, and from looking at Table 1 we would expect that it would perform on par with PRA with drastically reduced computation costs. Some experimentation shows that it is not that simple. Table 2 shows a comparison between PRA and SFE on 10 NELL relations.[5] SFE has a higher mean average precision, but the difference is not statistically significant. There is a large variance in SFE's performance, and on some relations PRA performs better.

We examined the feature matrices computed by these methods and discovered that the reason for the inconsistency of SFE's improvement is because its random walks are all *unconstrained*. Consider the case of a node with a very high degree, say 1000. If we only do 200 random walks from this node, we cannot possibly get a complete characterization of the graph even one step away from the node. If a particularly informative path is <CITYINSTATE, STATEINCOUNTRY>, and both the city from which a random walk starts and the intermediate state node have very high degree, the probability of actually finding this path type using unconstrained random walks is quite low. This is the benefit gained by the *path-constrained* random walks performed by PRA; PRA leverages training instances with relatively low degree and aggregation across a large number of instances to find path types that are potentially useful. Once they are found, significant computational effort goes into discovering whether each path type exists for *all* (s, t) pairs. It is this computational effort that allows the path type <CITYINSTATE, STATEINCOUNTRY> to have a non-zero value even for very highly connected nodes.

Method	MAP	Ave. Features
PRA	.3704	835
SFE	.4007	8275

Table 2: Comparison of PRA and SFE on 10 NELL relations. The difference shown is not statistically significant.

How do we mitigate this issue, so that SFE can consistently find these path types? It seems the only option without resorting to a similar two-step process to what PRA uses is to do a more exhaustive search. PRA uses random walks to improve scalability on very large graphs, particularly because the second step of the algorithm is so expensive. However, if we are only doing a single search, and the graph fits in memory, a few steps of a breadth-first search (BFS) per node is not infeasible. We can make the BFS more tractable by excluding edge types whose fan out is too high. For example, at a type node in Freebase, there could be thousands of edges of type /TYPE/OBJECT/TYPE; if there are a large number of edges of the same type leaving a node, we do not include those edges in the BFS. Note that because the type node will still be counted as an intermediate node in the subgraph, we can still find paths that go through that

[5]The data and evaluation methods are described more fully in Section 6.1. These experiments were conducted on a different development split of the same data.

Method	MAP	Ave. Features	Time
PRA	.3704	835	44 min.
SFE-RW	.4007	8275	6 min.
SFE-BFS	.4799	237853	5 min.

Table 3: Comparison of PRA and SFE (with PRA-style features) on 10 NELL relations. SFE-RW is not statistically better than PRA, but SFE-BFS is ($p < 0.05$).

node; we just do not continue searching if the out-degree of a particular edge type is too high.

When using a BFS instead of random walks to obtain the subgraphs G_{s_j} and G_{t_j} for each node pair, we saw a dramatic increase in the number of path type features found and a substantial increase in performance.[6] These results are shown in Table 3; SFE-RW is our SFE implementation using random walks, and SFE-BFS uses a BFS.

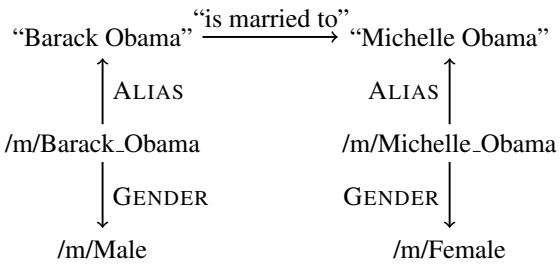
5.1 More expressive features

The description above shows how to recreate the feature space used by PRA using our simpler subgraph feature extraction technique. As we have mentioned, however, we need not restrict ourselves to merely recreating PRA's feature space. Eliminating random walk probabilities allows us to extract a much richer set of features from the subgraphs around each node, and here we present the feature extractors we have experimented with. Figure 1 contains an example graph that we will refer to when describing these features.

PRA-style features. We explained these features in Section 5, but we repeat them here for consistency, and use the example to make the feature extraction process more clear. Relying on the notation introduced earlier, these features are generated by intersecting the subgraphs G_s and G_t on the intermediate nodes. That is, when the subgraphs share an intermediate node, we combine the path types found from the source and target to that node. In the example in Figure 1, there are two common intermediate nodes ("Barack Obama" and "Michelle Obama"), and combining the path types corresponding to those nodes gives the same path type: -ALIAS-"is married to"-ALIAS^{-1}-.

Path bigram features. In Section 4, we mentioned that Neelakantan et al. (2015) experimented with using path bigrams as features. We include those features here as well. For any path π between a source node s and a target node t, we create a feature for each relation bigram in the path type. In the example in Figure 1, this would result in the features "BIGRAM:@START@-ALIAS", "BIGRAM:ALIAS-is married to", "BIGRAM:is married to-ALIAS", and "BIGRAM:ALIAS-@END@".

[6] One should not read too much into the decrease in running time between SFE-RW and SFE-BFS, however, as it was mostly an implementation detail.



Subgraph for /m/Barack_Obama	
π	i
-ALIAS-	"Barack Obama"
-GENDER-	/m/Male
-ALIAS-"is married to"-	"Michelle Obama"

Subgraph for /m/Michelle_Obama	
π	i
-ALIAS-	"Michelle Obama"
-GENDER-	/m/Female
-ALIAS-"is married to"$^{-1}$-	"Barack Obama"

Figure 1: An example graph, with subgraphs extracted for two nodes.

One-sided features. We use *one-sided path* to describe a sequence of edges that starts at a source or target node in the data, but does not necessarily terminate at a corresponding target or source node, as PRA features do. Following the notation introduced in Section 5, we use as features each (π, i) pair in the subgraph characterizations G_s and G_t, along with whether the feature came from the source node or the target node. The motivation for these one-sided path types is to better model which sources and targets are good candidates for participating in a particular relation. For example, not all cities participate in the relation CITYCAPITALOFCOUNTRY, even though the domain of the relation is all cities. A city that has a large number of sports teams may be more likely to be a capital city, and these one-sided features could easily capture that kind of information.

Example one-sided features from the example in Figure 1 would be "SOURCE:-GENDER-:male", "TARGET:-GENDER-:female",

"SOURCE:-ALIAS-:Barack Obama", and "SOURCE:-ALIAS-is married to-:Michelle Obama".

One-sided feature comparisons. We can expand on the one-sided features introduced above by allowing for comparisons of these features in certain circumstances. For example, if both the source and target nodes have an age or gender encoded in the graph, we might profitably use comparisons of these values to make better predictions.

Drawing again on the notation from Section 5, we can formalize these features as analogous to the pairwise PRA features. To get the PRA features, we intersect the intermediate nodes i from the subgraphs G_s and G_t, and combine the path types π when we find common intermediate nodes. To get these comparison features, we instead intersect the subgraphs on the path types, and combine the intermediate nodes when there are common path types. That is, if we see a common path type, such as -GENDER-, we will construct a feature representing a comparison between the intermediate node for the source and the target. If the values are the same, this information can be captured with a PRA feature, but it cannot be easily captured by PRA when the values are different.

In the example in Figure 1, there are two common path types: -ALIAS-, and -GENDER-. The feature generated from the path type -GENDER- would be "COMPARISON:-GENDER-:/m/Male:/m/Female".

Vector space similarity features. Gardner et al. (2014) introduced a modification of PRA's random walks to incorporate vector space similarity between the relations in the graph. On the data they were using, a graph that combined a formal knowledge base with textual relations extracted from text, they found that this technique gave a substantial performance improvement. The vector space random walks only affected the second step of PRA, however, and we have removed that step in SFE. While it is not as conceptually clean as the vector space random walks, we can obtain a similar effect with a simple feature transformation using the vectors for each relation. We obtain vector representations of relations through factorization of the knowledge base tensor as did Gardner et al., and replace each edge type in a PRA-style path with edges that are similar to it in the vector space. We also introduce a special "any edge" symbol, and say that all other edge types are similar to this edge type.

To reduce the combinatorial explosion of the feature space that this feature extractor creates, we only allow replacing one relation at a time with a similar relation. In the example graph in Figure 1, and assuming that "spouse of" is found to be similar to "is married to", some of the features extracted would be the following: "VECSIM:-ALIAS-is married to-ALIAS-", "VECSIM:-ALIAS-spouse of-ALIAS-", "VECSIM:-ALIAS-@ANY_REL@-ALIAS-", and "VECSIM:-@ANY_REL@-is married to-ALIAS-". Note that the first of those features, "VECSIM:-ALIAS-is married to-ALIAS-", is necessary even though it just duplicates the original PRA-style feature. This allows path types with different but similar relations to generate the same features.

Any-Relation features. It turns out that much of the benefit gained from Gardner et al.'s vector space similarity features came from allowing any path type that used a surface edge to match *any* other surface edge with non-zero probability.[7] To test whether the vector space similarity features give us any benefit over just replacing relations with dummy symbols, we add a feature extractor that is identical to the one above, assuming an empty vector similarity mapping. The features extracted from Figure 1 would thus be "ANYREL:-@ANY_REL@-is married to-ALIAS", "ANYREL:-ALIAS-@ANY_REL@-ALIAS", "ANYREL:-ALIAS-is married to-@ANY_REL@".

6 Experiments

Here we present experimental results evaluating the feature extractors we presented, and a comparison between SFE and PRA. As we showed in Section 5 that using a breadth-first search to obtain subgraphs is superior to using random walks, all of the experiments presented here use the BFS implementation of SFE.

6.1 Data

To evaluate SFE and the feature extractors we introduced, we learned models for 10 relations in the NELL KB. We used the same data as Gardner et al. (2014), using both the formal KB relations and the surface relations extracted from text in our

[7] Replacing all surface edges with a single dummy relation gives performance close to vector space PRA. The vector space walks do statistically outperform this, but the extra gain is small.

graph. We used logistic regression with elastic net (L1 and L2) regularization. We tuned the L1 and L2 parameters for each method on a random development split of the data, then used a new split of the data to run the final tests presented here.

The evaluation metrics we use are mean average precision (MAP) and mean reciprocal rank (MRR). We judge statistical significance using a paired permutation test, where the average precision[8] on each relation is used as paired data.

6.2 On Obtaining Negative Evidence

One important practical issue for most uses of PRA is the selection of negative examples for training a model. Typically a knowledge base only contains positive examples of a relation, and it is not clear a priori what the best method is for obtaining negative evidence. Prior work with PRA makes a closed world assumption, treating any (s, t) pair not seen in the knowledge base as a negative example. Negative instances are selected when performing the second step of PRA—if a random walk from a source ends at a target that is not a known correct target for that source, that source-target pair is used as a negative example.

SFE only scores (source, target) pairs; it has no mechanism similar to PRA's that will find potential targets given a source node. We thus need a new way of finding negative examples, both at training time and at test time. We used a simple technique to find negative examples from a graph given a set of positive examples, and we used this to obtain the training and testing data used in the experiments below. Our technique takes each source and target node in the given positive examples and finds other nodes in the same category that are close in terms of personalized page rank (PPR). We then sample new (source, target) pairs from these lists of similar nodes, weighted by their PPR score (while also allowing the original source and target to be sampled). These become our negative examples, both at training and at testing time.

Because this is changing the negative evidence available to PRA at training time, we wanted to be sure we were not unfairly hindering PRA in our comparisons. If it is in fact better to let PRA find its own negative examples at training time, instead of the ones sampled based on personalized page rank, then we should let PRA get its own negative evidence. We thus ran an experiment to see under which training regime PRA performs better. We created a test set with both positive and negative examples as described in the paragraph above, and at training time we compared two techniques: (1) letting PRA find its own negative examples through its random walks, and (2) only using the negative examples selected by PPR. As can be seen in Table 4, the difference between the two training conditions is very small, and it is not statistically significant. Because there is no significant difference between the two conditions, in the experiments that follow we give both PRA and SFE the same training data, created through the PPR-based sampling technique described above.

Method	MAP
PRA's random walks	.359
PPR-based sampling	.363

Table 4: Comparing methods for obtaining negative evidence available at training time. The difference seen is not statistically significant ($p = .77$).

6.3 Results

We first examine the effect of each of the feature types introduced in Section 5.1. The results are shown in Table 5. We can see that, for this data, the comparisons and one-sided features did not improve performance (and the decreases are not statistically significant). Bigram features do appear to improve performance, though the improvement was not consistent enough across relations to achieve statistical significance. The vector similarity features do improve performance, with p-values hovering right at 0.05 when comparing against only PRA features and PRA + bigram features. The any rel features, however, do statistically improve over all other methods ($p <= 0.01$) except the PRA + vec sim result ($p = .21$).

Finally, we present a comparison between PRA, PRA with vector space random walks, and the best SFE result from the ablation study. This is shown in Table 6. SFE significantly outperforms PRA, both with and without the vector space random walks presented by Gardner et al. (2014).

6.4 Discussion

When using only PRA-style features with SFE, the highest weighted features were almost always those of the form -ALIAS-[some textual relation]-ALIAS^{-1}-. For example, for the relation WRITER-

[8] Average precision is equivalent to the area under a precision/recall curve.

Feature Types	MAP	MRR	Features
PRA-style features	.431	.806	240k
+ Comparisons	.405	.833	558k
+ One-sided	.389	.800	1,227k
+ One-sided + Comps.	.387	.817	1,544k
+ Bigrams	.483	1.00	320k
+ Vector similarity	.514	.910	3,993k
+ Bigrams + vec sim.	.490	.950	4,073k
+ Any Rel	**.528**	.933	649k

Table 5: SFE feature ablation study. All rows use PRA features. PRA + any rel is statistically better than all other methods except PRA + vec sim, and most of the other differences are not significant.

Method	MAP	MRR
PRA	.362	.717
Vector space PRA	.432	.850
SFE (PRA + any rel features)	**.528**	.933

Table 6: Results of final comparison between SFE and PRA, with and without vector space similarity features. SFE is statistically better than both PRA methods ($p < 0.005$).

WROTEBOOK, the textual relations used in this feature might be "wrote", "describes in", "writes in", and "expresses in". These are the same feature types that PRA itself finds to have the highest weight, also, though SFE finds many more of them than PRA does, as PRA has to do aggressive feature selection. For this particular dataset, where the graph consists of edges from a formal KB mixed with edges from extracted textual relations, these kinds of features are by far the most useful, and most of the improvements seen by the additional feature types we used with SFE come from more compactly encoding these features.

For example, the path bigram features can encode the fact that there exists a path from the source to the target that begins or ends with an ALIAS edge. This captures in just two features all path types of the form -ALIAS-[some textual relation]-ALIAS^{-1}-, and those two bigram features are almost always the highest weighted features in models where they are used.

However, the bigram features do not capture those path types exactly. The Any-Rel features were designed in part specifically for this path type, and they capture it exactly with a single feature. For all 10 relations, the feature "ANYREL:-ALIAS-@ANY_REL@-ALIAS^{-1}" is the highest weighted feature. This is because, for the relations we experimented with, knowing that *some* relationship is expressed in text between a particular pair of KB entities is a very strong indication of a single KB relation. There are only so many possible relationships between cities and countries, for instance. These features are much less informative between entity types where more than one relation is possible, such as between people.

While the bigram and any-rel features capture succintly whether textual relations are present between two entities, the one-sided features are more useful for determining whether an entity fits into the domain or range of a particular relation. We saw a few features that did this, capturing fine-grained entity types. Most of the features, however, tended towards memorizing (and thus overfitting) the training data, as these features contained the names of the training entities. We believe this overfitting to be the main reason these features did not improve performance, along with the fact that the relations we tested do not need much domain or range modeling (as opposed to, e.g., SPOUSEOF or CITYCAPITALOFCOUNTRY).

7 Conclusion

We have explored several practical issues that arise when using the path ranking algorithm for knowledge base completion. An analysis of several of these issues led us to propose a simpler algorithm, which we called subgraph feature extraction, which characterizes the subgraph around node pairs and extracts features from that subgraph. SFE is both significantly faster and performs better than PRA on this task. We showed experimentally that we can reduce running time by an order of magnitude, while at the same time improving mean average precision from .432 to .528 and mean reciprocal rank from .850 to .933. This thus constitutes the best published results for knowledge base completion on NELL data. The code and data used in the experiments in this paper are available at http://rtw.ml.cmu.edu/emnlp2015_sfe/.

Acknowledgements

This work was supported in part by NSF grant IIS1247489, in part by support as a Yahoo! Fellow, and in part by DARPA contract FA87501320005.

References

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *EMNLP*, pages 1533–1544.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of SIGMOD*.

Antoine Bordes, Jason Weston, Ronan Collobert, Yoshua Bengio, et al. 2011. Learning structured embeddings of knowledge bases. In *AAAI*.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems*, pages 2787–2795.

Kai-Wei Chang, Wen-tau Yih, Bishan Yang, and Christopher Meek. 2014. Typed tensor decomposition of knowledge bases for relation extraction. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1568–1579.

Eunsol Choi, Tom Kwiatkowski, and Luke Zettlemoyer. 2015. Scalable semantic parsing with partial ontologies. In *Proceedings of ACL*.

Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. 2014. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 601–610. ACM.

Alberto García-Durán, Antoine Bordes, and Nicolas Usunier. 2014. Effective blending of two and three-way interactions for modeling multi-relational data. In *Machine Learning and Knowledge Discovery in Databases*, pages 434–449. Springer.

Matt Gardner, Partha Talukdar, Bryan Kisiel, and Tom Mitchell. 2013. Improving learning and inference in a large knowledge-base using latent syntactic cues. In *Proceedings of EMNLP*. Association for Computational Linguistics.

Matt Gardner, Partha Talukdar, Jayant Krishnamurthy, and Tom Mitchell. 2014. Incorporating vector space similarity in random walk inference over knowledge bases. In *Proceedings of EMNLP*. Association for Computational Linguistics.

Matt Gardner, Partha Talukdar, and Tom Mitchell. 2015. Combining vector space embeddings with symbolic logical inference over open-domain text. In *2015 AAAI Spring Symposium Series*.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 541–550. Association for Computational Linguistics.

Jayant Krishnamurthy and Tom M Mitchell. 2012. Weakly supervised training of semantic parsers. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 754–765. Association for Computational Linguistics.

Ni Lao and William W Cohen. 2010. Relational retrieval using a combination of path-constrained random walks. *Machine learning*, 81(1):53–67.

Ni Lao, Tom Mitchell, and William W Cohen. 2011. Random walk inference and learning in a large scale knowledge base. In *Proceedings of EMNLP*. Association for Computational Linguistics.

Ni Lao, Amarnag Subramanya, Fernando Pereira, and William W Cohen. 2012. Reading the web with learned syntactic-semantic inference rules. In *Proceedings of EMNLP-CoNLL*.

Ni Lao. 2012. *Efficient Random Walk Inference with Knowledge Bases*. Ph.D. thesis, Carnegie Mellon University.

Pablo N. Mendes, Max Jakob, and Christian Bizer. 2012. Dbpedia for nlp: A multilingual cross-domain knowledge base. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 1003–1011. Association for Computational Linguistics.

Tom M. Mitchell, William Cohen, Estevam Hruschka, Partha Talukdar, Justin Betteridge, Andrew Carlson, Bhavana Dalvi, Matthew Gardner, Bryan Kisiel, Jayant Krishnamurthy, Ni Lao, Kathryn Mazaitis, Thahir Mohamed, Ndapa Nakashole, Emmanouil Antonios Platanios, Alan Ritter, Mehdi Samadi, Burr Settles, Richard Wang, Derry Wijaya, Abhinav Gupta, Xinlei Chen, Abulhair Saparov, Malcolm Greaves, and Joel Welling. 2015. Never-ending learning. In *AAAI 2015*. Association for the Advancement of Artificial Intelligence.

Arvind Neelakantan, Benjamin Roth, and Andrew McCallum. 2015. Compositional vector space models for knowledge base completion. In *ACL 2015*. Association for Computational Linguistics.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 809–816.

Maximilian Nickel, Xueyan Jiang, and Volker Tresp. 2014. Reducing the rank in relational factorization models by including observable patterns. In *Advances in Neural Information Processing Systems*, pages 1179–1187.

Matthew Richardson and Pedro Domingos. 2006. Markov logic networks. *Machine learning*, 62(1-2):107–136.

Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In *Proceedings of NAACL-HLT*.

Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *Advances in Neural Information Processing Systems*, pages 926–934.

Fabian Suchanek, James Fan, Raphael Hoffmann, Sebastian Riedel, and Partha Pratim Talukdar. 2013. Advances in automated knowledge base construction. *SIGMOD Records journal, March*.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D Manning. 2012. Multi-instance multi-label learning for relation extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 455–465. Association for Computational Linguistics.

William Yang Wang, Kathryn Mazaitis, and William W. Cohen. 2013. Programming with personalized pagerank: A locally groundable first-order probabilistic logic. In *Proceedings of the 22Nd ACM International Conference on Conference on Information & Knowledge Management*, CIKM '13, pages 2129–2138, New York, NY, USA. ACM.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, pages 1112–1119.

Robert West, Evgeniy Gabrilovich, Kevin Murphy, Shaohua Sun, Rahul Gupta, and Dekang Lin. 2014. Knowledge base completion via search-based question answering. In *WWW*.

Jason Weston, Antoine Bordes, Oksana Yakhnenko, and Nicolas Usunier. 2013. Connecting language and knowledge bases with embedding models for relation extraction. In *Proceedings of EMNLP*.

Representing Text for Joint Embedding of Text and Knowledge Bases

Kristina Toutanova
Microsoft Research
Redmond, WA, USA

Danqi Chen*
Computer Science Department
Stanford University

Patrick Pantel
Microsoft Research
Redmond, WA, USA

Hoifung Poon
Microsoft Research
Redmond, WA, USA

Pallavi Choudhury
Microsoft Research
Redmond, WA, USA

Michael Gamon
Microsoft Research
Redmond, WA, USA

Abstract

Models that learn to represent textual and knowledge base relations in the same continuous latent space are able to perform joint inferences among the two kinds of relations and obtain high accuracy on knowledge base completion (Riedel et al., 2013). In this paper we propose a model that captures the compositional structure of textual relations, and jointly optimizes entity, knowledge base, and textual relation representations. The proposed model significantly improves performance over a model that does not share parameters among textual relations with common sub-structure.

1 Introduction

Representing information about real-world entities and their relations in structured knowledge base (KB) form enables numerous applications. Large, collaboratively created knowledge bases have recently become available e.g., Freebase (Bollacker et al., 2008), YAGO (Suchanek et al., 2007), and DBPedia (Auer et al., 2007), but even though they are impressively large, their coverage is far from complete. This has motivated research in automatically deriving new facts to extend a manually built knowledge base, by using information from the existing knowledge base, textual mentions of entities, and semi-structured data such as tables and web forms (Nickel et al., 2015).

In this paper we build upon the work of Riedel et al. (2013), which jointly learns continuous representations for knowledge base and textual relations. This common representation in the same vector space can serve as a kind of "universal schema" which admits joint inferences among

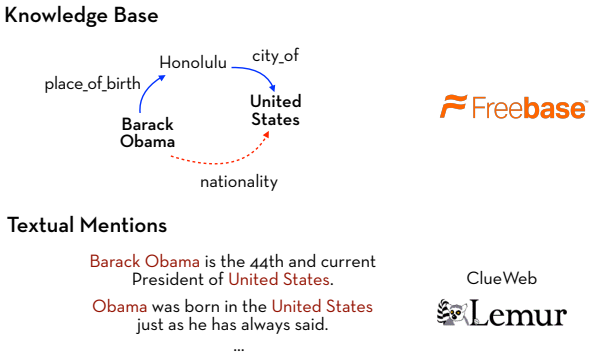


Figure 1: A knowledge base fragment coupled with textual mentions of pairs of entities.

KBs and text. The textual relations represent the relationships between entities expressed in individual sentences (see Figure 1 for an example). Riedel et al. (2013) represented each textual mention of an entity pair by the lexicalized dependency path between the two entities (see Figure 2). Each such path is treated as a separate relation in a combined knowledge graph including both KB and textual relations. Following prior work in latent feature models for knowledge base completion, every textual relation receives its own continuous representation, learned from the pattern of its co-occurrences in the knowledge graph.

However, largely synonymous textual relations often share common sub-structure, and are composed of similar words and dependency arcs. For example, Table 1 shows a collection of dependency paths co-occurring with the *person/organizations_founded* relation.

In this paper we model this sub-structure and share parameters among related dependency paths, using a unified loss function learning entity and relation representations to maximize performance on the knowledge base link prediction task.

We evaluate our approach on the FB15k-237 dataset, a knowledge base derived from the Free-

*This research was conducted during the author's internship at Microsoft Research.

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1499–1509,
Lisbon, Portugal, 17-21 September 2015.

base subset FB15k (Bordes et al., 2013) and filtered to remove highly redundant relations (Toutanova and Chen, 2015). The knowledge base is paired with textual mentions for all entity pairs derived from ClueWeb12[1] with Freebase entity mention annotations (Gabrilovich et al., 2013).

We show that using a convolutional neural network to derive continuous representations for textual relations boosts the overall performance on link prediction, with larger improvement on entity pairs that have textual mentions.

2 Related Work

There has been a growing body of work on learning to predict relations between entities without requiring sentence-level annotations of textual mentions at training time. We group such related work into three groups based on whether KB, text, or both sources of information are used. Additionally, we discuss related work in the area of supervised relation extraction using continuous representations of text, even though we do not use supervision at the level of textual mentions.

Knowledge base completion

Nickel et al. (2015) provide a broad overview of machine learning models for knowledge graphs, including models based on observed graph features such as the path ranking algorithm (Lao et al., 2011), models based on continuous representations (latent features), and model combinations (Dong et al., 2014). These models predict new facts in a given knowledge base, based on information from existing entities and relations. From this line of work, most relevant to our study is prior work evaluating continuous representation models on the FB15k dataset. Yang et al. (2015) showed that a simple variant of a bilinear model DISTMULT outperformed TRANSE (Bordes et al., 2013) and more richly parameterized models on this dataset. We therefore build upon the best performing prior model DISTMULT from this line of work, as well as additional models E and F developed in the context of text-augmented knowledge graphs (Riedel et al., 2013), and extend them to incorporate compositional representations of textual relations.

[1] http://lemurproject.org/clueweb12/FACC1/

Relation extraction using distant supervision

A number of works have focused on extracting new instances of relations using information from textual mentions, without sophisticated modeling of prior knowledge from the knowledge base. Mintz et al. (2009) demonstrated that both surface context and dependency path context were helpful for the task, but did not model the compositional sub-structure of this context. Other work proposed more sophisticated models that reason about sentence-level hidden variables (Riedel et al., 2010; Hoffmann et al., 2011; Surdeanu et al., 2012) or model the noise arising from the incompleteness of knowledge bases and text collections (Ritter et al., 2013), *inter alia*. Our work focuses on representing the compositional structure of sentential context for learning joint continuous representations of text and knowledge bases.

Combining knowledge base and text information

A combination of knowledge base and textual information was first shown to outperform either source alone in the framework of path-ranking algorithms in a combined knowledge base and text graph (Lao et al., 2012). To alleviate the sparsity of textual relations arising in such a combined graph, (Gardner et al., 2013; Gardner et al., 2014) showed how to incorporate clusters or continuous representations of textual relations. Note that these vector representations are based on the co-occurrence patterns for the textual relations and not on their compositional structure. Co-occurrence based textual relation representations were also learned in (Neelakantan et al., 2015). Wang et al. (2014a) combined knowledge base and text information by embedding knowledge base entities and the words in their names in the same vector space, but did not model the textual co-occurrences of entity pairs and the expressed textual relations. Weston et al. (2013) combined continuous representations from a knowledge base and textual mentions for prediction of new relations. The two representations were trained independently of each other and using different loss functions, and were only combined at inference time. Additionally, the employed representations of text were non-compositional.

In this work we train continuous representations of knowledge base and textual relations jointly, which allows for deeper interactions between the

sources of information. We directly build on the universal schema approach of Riedel et al. (2013) as well as the universal schema extension of the DISTMULT model mentioned previously, to improve the representations of textual relations by capturing their compositional structure. Additionally, we evaluate the approach on a dataset that contains rich prior information from the training knowledge base, as well as a wealth of textual information from a large document collection.

Continuous representations for supervised relation extraction

In contrast to the work reviewed so far, work on sentence-level relation extraction using direct supervision has focused heavily on representing sentence context. Models using hand-crafted features have evolved for more than a decade, and recently, models using continuous representations have been found to achieve new state-of-the-art performance (Zeng et al., 2014; Gormley et al., 2015). Compared to work on representation learning for sentence-level context, such as this recent work using LSTM models on constituency or dependency trees (Tai et al., 2015), our approach using a one-hidden-layer convolutional neural network is relatively simple. However, even such a simple approach has been shown to be very competitive (Kim, 2014).

3 Models for knowledge base completion

We begin by introducing notation to define the task, largely following the terminology in Nickel et al. (2015). We assume knowledge bases are represented using RDF triples, in the form (*subject*, *predicate*, *object*), where the subject and object are entities and the predicate is the type of relation. For example, the KB fragment shown in Figure 1 is shown as a knowledge graph, where the entities are the nodes, and the relations are shown as directed labeled edges: we see three entities participating in three relation instances indicated by the edges. For brevity, we will denote triples by (e_s, r, e_o), where e_s and e_o denote the subject and object entities, respectively.

The task is, given a training KB consisting of entities with some relations between them, to predict new relations (links) that do not appear in the training KB. More specifically, we will build models that rank candidate entities for given queries $(e_s, r, ?)$ or $(?, r, e_o)$, which ask about the object or subject of a given relation.

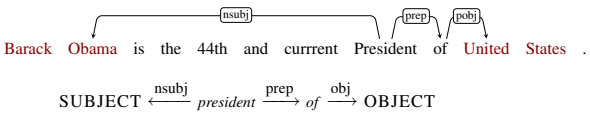


Figure 2: Textual relation extracted from an entity pair mention.

This task setting has been used in models for KB completion previously, e.g. (Dong et al., 2014; Gardner et al., 2014), even though it has not been standard in evaluations of distant supervision for relation extraction (Mintz et al., 2009; Riedel et al., 2013). The advantage of this evaluation setting is that it enables automatic evaluation without requiring humans to label candidate extractions, while making only a *local* closed world assumption for the completeness of the knowledge base — i.e., if one object e_o for a certain subject / relation pair (e_s, r) is present in the knowledge base, it is assumed likely that all other objects (e_s, r, e'_o) will be present. Such an assumption is particularly justified for nearly functional relations.

To incorporate textual information, we follow prior work (Lao et al., 2012; Riedel et al., 2013) and represent both textual and knowledge base relations in a single graph of "universal" relations. The textual relations are represented as full lexicalized dependency paths, as illustrated in Figure 2. An instance of the textual relation SUBJECT $\xleftarrow{\text{nsubj}}$ *president* $\xrightarrow{\text{prep}}$ *of* $\xrightarrow{\text{obj}}$ OBJECT connecting the entities BARACK OBAMA and UNITED STATES, is added to the knowledge graph based on this sentential occurrence.

To present the models for knowledge base completion based on such combined knowledge graphs, we first introduce some notation. Let $\mathcal{E}$ denote the set of entities in the knowledge graph and let $\mathcal{R}$ denote the set of relation types. We denote each possible triple as $T = (e_s, r, e_o)$ where $e_s, e_o \in \mathcal{E}, r \in \mathcal{R}$, and model its presence with a binary random variable $y_T \in \{0, 1\}$ which indicates whether the triple exists. The models we build score possible triples (e_s, r, e_o) using *continuous representations* (*latent features*) of the three elements of the triple. The models use scoring function $f(e_s, r, e_o)$ to represent the model's confidence in the existence of the triple. We present the models and then the loss function used to train

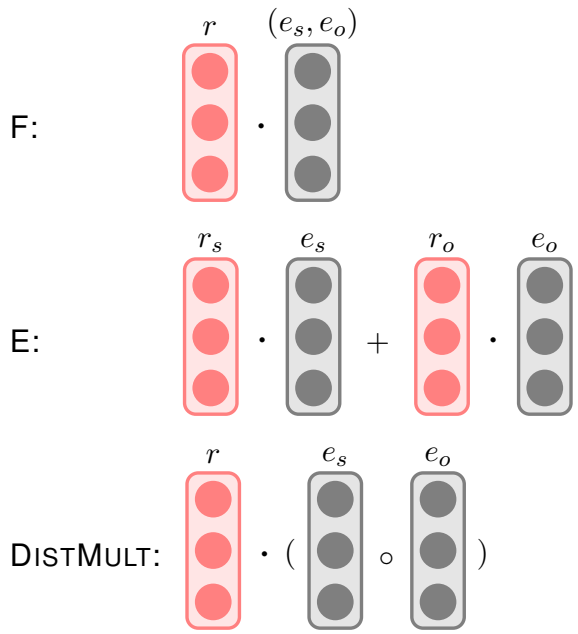


Figure 3: The continuous representations for model F, E and DISTMULT.

their parameters.

3.1 Basic Models

We begin with presenting the three models from prior work that this research builds upon. They all learn latent continuous representations of relations and entities or entity pairs, and score possible triples based on the learned continuous representations. Each of the models can be defined on a knowledge graph containing entities and KB relations only, or on a knowledge graph additionally containing textual relations. We use models F and E from (Riedel et al., 2013) where they were used for a combined KB+text graph, and model DISTMULT from (Yang et al., 2015), which was originally used for a knowledge graph containing only KB relations.

As shown in Figure 3, model F learns a K-dimensional latent feature vector for each candidate entity pair (e_s, e_o), as well as a same-dimensional vector for each relation r, and the scoring function is simply defined as their inner product: $f(e_s, r, e_o) = v(r)^\intercal v(e_s, e_o)$. Therefore, different pairs sharing the same entity would not share parameters in this model.

Model E does not have parameters for entity pairs, and instead has parameters for individual entities. It aims to capture the compatibility between entities and the subject and object positions of relations. For each relation type r, the model learns two latent feature vectors $v(r_s)$ and $v(r_o)$ of dimension K. For each entity (node) e_i, the model also learns a latent feature vector of the same dimensionality. The score of a candidate triple (e_s, r, e_o) is defined as $f(e_s, r, e_o) = v(r_s)^\intercal v(e_s) + v(r_o)^\intercal v(e_o)$. It can be seen that when a subject entity is fixed in a query $(e_s, r, ?)$, the ranking of candidate object entity fillers according to f does not depend on the subject entity but only on the relation type r.

The third model DISTMULT, is a special form of a bilinear model like RESCAL (Nickel et al., 2011), where the non-diagonal entries in the relation matrices are assumed to be zero. This model was proposed in Yang et al. (2015) and was shown to outperform prior work on the FB15k dataset. In this model, each entity e_i and each relation r is assigned a latent feature vector of dimension K. The score of a candidate triple (e_s, r, e_o) is defined as $f(e_s, r, e_o) = v(r)^\intercal \left(v(e_s) \circ v(e_o)\right)$, where $\circ$ denotes the element-wise vector product. In this model, entity pairs which share an entity also share parameters, and the ranking of candidate objects for queries $(e_s, r, ?)$ depends on the subject entity.

Denote $N_e = |\mathcal{E}|, N_r = |\mathcal{R}|$, and K = dimension of latent feature vectors, then model E has $KN_e + 2KN_r$ parameters and model DISTMULT has $KN_e + KN_r$ parameters. Model F has $KN_e^2 + KN_r$ parameters, although most entity pairs will not co-occur in the knowledge base or text.

In the basic models, knowledge base and textual relations are treated uniformly, and each textual relation receives its own latent representation of dimensionality K. When textual relations are added to the training knowledge graph, the total number of relations $|\mathcal{R}|$ grows substantially (it increases from 237 to more than 2.7 million for the dataset in this study), resulting in a substantial increase in the total number of independent parameters.

Note that in all of these models queries about the arguments of knowledge base relations $(e_s, r, ?)$ are answered by scoring functions looking only at the entity and KB relation representations, without using representations of textual mentions. The textual mention information and representations are only used at training time to improve the learned representations of KB relations and entities.

3.2 CONV: Compositional Representations of Textual Relations

In the standard latent feature models discussed above, each textual relation is treated as an atomic unit receiving its own set of latent features. However, many textual relations differ only slightly in the words or dependency arcs used to express the relation. For example, Table 1 shows several textual patterns that co-occurr with the relation *person/organizations_founded* in the training KB. While some dependency paths occur frequently, many very closely related ones have been observed only once. The statistical strength of the model could be improved if similar dependency paths have a shared parameterization. We build on work using similar intuitions for other tasks and learn compositional representations of textual relations based on their internal structure, so that the derived representations are accurate for the task of predicting knowledge base relations.

We use a convolutional neural network applied to the lexicalized dependency paths treated as a sequence of words and dependency arcs with direction. Figure 4 depicts the neural network architecture. In the first layer, each word or directed labeled arc is mapped to a continuous representation using an embedding matrix $\mathbf{V}$. In the hidden layer, every window of three elements is mapped to a hidden vector using position-specific maps $\mathbf{W}$, a bias vector $\mathbf{b}$, and a $\tanh$ activation function. A max-pooling operation over the sequence is applied to derive the final continuous representation for the dependency path.

The CONV representation of textual relations can be used to augment any of the three basic models. The difference between a basic model and its CONV-augmented variant is in the parameterization of textual mentions. The basic models learn distinct latent feature vectors of dimensionality K for all textual relation types, whereas the CONV models derive the K-dimensional latent feature vectors for textual relation types as the activation at the top layer of the convolutional network in Figure 4, given the corresponding lexicalized dependency path as input.

3.3 Training loss function

All basic and CONV-augmented models use the same training loss function. Our loss function is motivated by the link prediction task and the performance measures used. As previously mentioned, the task is to predict the subject or object entity for given held-out triples (e_s, r, e_o), i.e., to rank all entities with respect to their likelihood of filling the respective position in the triple[2]. We would thus like the model to score correct triples (e_s, r, e_o) higher than incorrect triples (e', r, e_o) and (e_s, r, e') which differ from the correct triple by one entity. Several approaches (Nickel et al., 2015) use a margin-based loss function. We use an approximation to the negative log-likelihood of the correct entity filler instead[3]. We define the conditional probabilities $p(e_o|e_s, r)$ and $p(e_s|r, e_o)$ for object and subject entities given the relation and the other argument as follows:

$$p(e_o|e_s, r; \Theta) = \frac{e^{f(e_s,r,e_o;\Theta)}}{\sum_{e' \in Neg(e_s,r,?)} e^{f(e_s,r,e';\Theta)}}$$

Conditional probabilities for subject entities $p(e_s|e_o, r; \Theta)$ are defined analogously. Here Θ denotes all the parameters of latent features. The denominator is defined using a set of entities that do not fill the object position in any relation triple $(e_s, r, ?)$ in the training knowledge graph. Since the number of such entities is impractically large, we sample negative triples from the full set. We also limit the candidate entities to ones that have types consistent with the position in the relation triple (Chang et al., 2014; Yang et al., 2015), where the types are approximated following Toutanova and Chen (2015). Additionally, since the task of predicting textual relations is auxiliary to the main task, we use a weighting factor τ for the loss on predicting the arguments of textual relations (Toutanova and Chen, 2015).

Denote $\mathcal{T}$ as a set of triples, we define the loss $L(\mathcal{T}; \Theta)$ as:

$$\begin{aligned} L(\mathcal{T}; \Theta) \;=\; & -\sum_{(e_s,r,e_o) \in \mathcal{T}} \log p(e_o|e_s, r; \Theta) \\ & -\sum_{(e_s,r,e_o) \in \mathcal{T}} \log p(e_s|e_o, r; \Theta) \end{aligned}$$

Let $\mathcal{T}_{\text{KB}}$ and $\mathcal{T}_{\text{text}}$ represent the set of knowledge base triples and textual relation triples respectively. The final training loss function is de-

[2] Our experimental comparison focuses on predicting object entities only, but we consider both argument types in the training loss function.

[3] Note that both margin-based and likelihood-based loss functions are susceptible to noise from potential selection of false negative examples. An empirical comparison of training loss functions would be interesting.

Textual Pattern	Count
SUBJECT $\xrightarrow{\text{appos}}$ founder $\xrightarrow{\text{prep}}$ of $\xrightarrow{\text{pobj}}$ OBJECT	12
SUBJECT $\xleftarrow{\text{nsubj}}$ co-founded $\xrightarrow{\text{dobj}}$ OBJECT	3
SUBJECT $\xrightarrow{\text{appos}}$ co-founder $\xrightarrow{\text{prep}}$ of $\xrightarrow{\text{pobj}}$ OBJECT	3
SUBJECT $\xrightarrow{\text{conj}}$ co-founder $\xrightarrow{\text{prep}}$ of $\xrightarrow{\text{pobj}}$ OBJECT	3
SUBJECT $\xleftarrow{\text{pobj}}$ with $\xleftarrow{\text{prep}}$ co-founded $\xrightarrow{\text{dobj}}$ OBJECT	2
SUBJECT $\xleftarrow{\text{nsubj}}$ signed $\xrightarrow{\text{xcomp}}$ establishing $\xrightarrow{\text{dobj}}$ OBJECT	2
SUBJECT $\xleftarrow{\text{pobj}}$ with $\xleftarrow{\text{prep}}$ founders $\xrightarrow{\text{prep}}$ of $\xrightarrow{\text{pobj}}$ OBJECT	2
SUBJECT $\xrightarrow{\text{appos}}$ founders $\xrightarrow{\text{prep}}$ of $\xrightarrow{\text{pobj}}$ OBJECT	2
SUBJECT $\xleftarrow{\text{nsubj}}$ one $\xrightarrow{\text{prep}}$ of $\xrightarrow{\text{pobj}}$ founders $\xrightarrow{\text{prep}}$ of $\xrightarrow{\text{pobj}}$ OBJECT	2
SUBJECT $\xleftarrow{\text{nsubj}}$ founded $\xrightarrow{\text{dobj}}$ production $\xrightarrow{\text{conj}}$ OBJECT	2
SUBJECT $\xleftarrow{\text{appos}}$ partner $\xleftarrow{\text{pobj}}$ with $\xleftarrow{\text{prep}}$ founded $\xrightarrow{\text{dobj}}$ production $\xrightarrow{\text{conj}}$ OBJECT	2
SUBJECT $\xleftarrow{\text{pobj}}$ by $\xleftarrow{\text{prep}}$ co-founded $\xleftarrow{\text{rcmod}}$ OBJECT	1
SUBJECT $\xleftarrow{\text{nn}}$ co-founder $\xrightarrow{\text{prep}}$ of $\xrightarrow{\text{pobj}}$ OBJECT	1
SUBJECT $\xrightarrow{\text{dep}}$ co-founder $\xrightarrow{\text{prep}}$ of $\xrightarrow{\text{pobj}}$ OBJECT	1
SUBJECT $\xleftarrow{\text{nsubj}}$ helped $\xrightarrow{\text{xcomp}}$ establish $\xrightarrow{\text{dobj}}$ OBJECT	1
SUBJECT $\xleftarrow{\text{nsubj}}$ signed $\xrightarrow{\text{xcomp}}$ creating $\xrightarrow{\text{dobj}}$ OBJECT	1

Table 1: Textual patterns occurring with entity pairs in a *person/organizations_founded* relationship. The count indicates the number of training set instances that have this KB relation, which co-occur with each textual pattern.

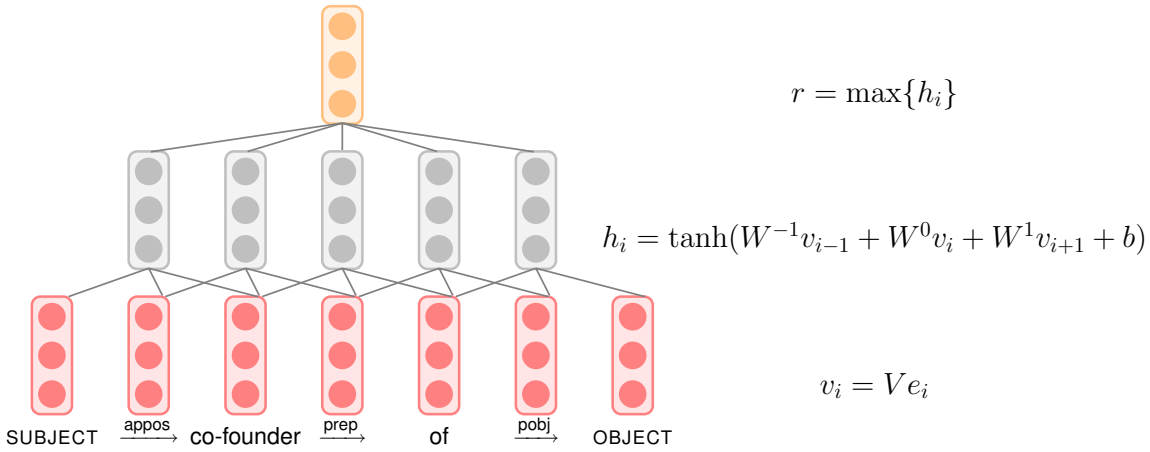


Figure 4: The convolutional neural network architecture for representing textual relations.

fined as:

$$L(\mathcal{T}_{\text{KB}}; \Theta) + \tau L(\mathcal{T}_{\text{text}}; \Theta) + \lambda \|\Theta\|^2,$$

where λ is the regularization parameter, and τ is the weighing factor of the textual relations.

The parameters of all models are trained using a batch training algorithm. The gradients of the basic models are straightforward to compute, and the gradients of the convolutional network parameters for the CONV-augmented models are also not hard to derive using back-propagation.

4 Experiments

Dataset and Evaluation Protocol

We use the FB15k-237 [4] dataset, which is a subset of FB15k (Bordes et al., 2013) that excludes redundant relations and direct training links for held-out triples, with the goal of making the task more realistic (Toutanova and Chen, 2015). The FB15k dataset has been used in multiple studies on knowledge base completion (Wang et al., 2014b; Yang et al., 2015). Textual relations for

[4] Check the first author's website for a release of the dataset.

FB15k-237 are extracted from 200 million sentences in the ClueWeb12 corpus coupled with Freebase mention annotations (Gabrilovich et al., 2013), and include textual links of all co-occurring entities from the KB set. After pruning[5], there are 2.7 million unique textual relations that are added to the knowledge graph. The set of textual relations is larger than the set used in Toutanova and Chen (2015) (25,000 versus 2.7 million), leading to improved performance.

The number of relations and triples in the training, validation and test portions of the data are given in Table 2. The two rows list statistics for the KB and text portions of the data separately. The 2.7 million textual relations occur in 3.9 million text triples. Almost all entities occur in textual relations (13,937 out of 14,541). The numbers of triples for textual relations are shown as zero for the validation and test sets because we don't evaluate on prediction of textual relations (all text triples are used in training). The percentage of KB triples that have textual relations for their pair of entities is 40.5% for the training, 26.6% for the validation, and 28.1% for the test set. While 26.6% of the validation set triples have textual mentions, the percentage with textual relations that have been seen in the training set is 18.4%. Having a mention increases the chance that a random entity pair has a relation from 0.1% to 5.0% — a fifty-fold increase.

Given a set of triples in a set disjoint from a training knowledge graph, we test models on predicting the object of each triple, given the subject and relation type. We rank all entities in the training knowledge base in order of their likelihood of filling the argument position. We report the mean reciprocal rank (MRR) of the correct entity, as well as HITS@10 — the percentage of test triples for which the correct entity is ranked in the top 10. We use *filtered* measures following the protocol proposed in Bordes et al. (2013) — that is, when we rank entities for a given position, we remove all other entities that are known to be part of an existing triple in the training, validation, or test set. This avoids penalizing the model for ranking other correct fillers higher than the tested entity.

[5]The full set of 37 million textual patterns connecting the entity pairs of interest was pruned based on the count of patterns and their tri-grams, and their precision in indicating that entity pairs have KB relations.

Implementation details

We used a value of $\lambda = 1$ for the weight of the L_2 penalty for the main results in Table 3, and present some results on the impact of λ at the end of this section. We used batch optimization after initial experiments with AdaGrad showed inferior performance. L-BFGS (Liu and Nocedal, 1989) and RProp (Riedmiller and Braun, 1993) were found to converge to similar function values, with RProp converging significantly faster. We thus used RProp for optimization. We initialized the KB+text models from the KB-only models and also from random initial values (sampled from a Gaussian distribution), and stopped optimization when the overall MRR on the validation set decreased. For each model type, we chose the better of random and KB-only initialization. The word embeddings in the CONV models were initialized using the 50-dimensional vectors from Turian et al. (2010) in the main experiments, with a slight positive impact. The effect of initialization is discussed at the end of the section.

The number of negative examples for each triple was set to 200. Performance improved substantially when the number of negative examples was increased and reached a plateau around 200. We chose the optimal number of latent feature dimensions via a grid search to optimize MRR on the validation set, testing the values 5, 10, 15, 35, 50, 100, 200 and 500. We also performed a grid search over the values of the parameter τ, testing values in the set $\{0.01, 0.1, 0.25, 0.5, 1\}$. The best dimension for latent feature vectors was 10 for most KB-only models (not including model F), and 5 for the two model configurations including F. We used $K = 10$ for all KB+text models, as higher dimension was also not helpful for them.

Experimental results

In Table 3 we show the performance of different models and their combinations[6], both when using textual mentions (*KB+text*), and when using only knowledge base relations (*KB only*). In the KB+text setting, we evaluate the contribution of the CONV representations of the textual relations. The upper portion of the Table shows the performance of models that have been trained using knowledge graphs including only knowledge

[6]Different models are combined by simply defining a combined scoring function which adds the scores from individual models. Combined models are trained jointly.

	# Relations	# Entities	# Triples in Train / Validation / Test
KB	237	14,541	272,115 / 17,535 / 20, 466
Text	2,740k	13,937	3,978k / 0 / 0

Table 2: The statistics of dataset FB15k-237.

Model	Overall		With mentions		Without mentions	
	MRR	HITS@10	MRR	HITS@10	MRR	HITS@10
KB only						
F	16.9	24.5	26.4	49.1	13.3	15.5
E	33.2	47.6	25.5	37.8	36.0	51.2
DISTMULT	35.7	52.3	26.0	39.0	39.3	57.2
E+DISTMULT	**37.3**	**55.2**	**28.6**	**42.9**	40.5	**59.8**
F+E+DISTMULT	33.8	50.1	15.0	26.1	**40.7**	59.0
KB and text						
F ($\tau = 1$)	19.4	27.9	**35.4**	61.6	13.4	15.5
CONV-F ($\tau = 1$)	19.2	28.4	34.9	**63.7**	13.3	15.4
E ($\tau = 0$)	33.2	47.6	25.5	37.8	36.0	51.2
CONV-E ($\tau = 0$)	33.2	47.6	25.5	37.8	36.0	51.2
DISTMULT ($\tau = 0.01$)	36.1	52.7	26.5	39.5	39.6	57.5
CONV-DISTMULT ($\tau = 0.25$)	36.6	53.5	28.3	43.4	39.7	57.2
E + DISTMULT ($\tau = 0.01$)	37.7	55.7	28.9	43.4	40.9	60.2
CONV-E + CONV-DISTMULT ($\tau = 0.25$)	**40.1**	**58.1**	33.9	49.9	**42.4**	**61.1**

Table 3: Results on FB15k-237 for KB only and KB+text inference, with basic models versus the proposed CONV-augmented models. The values of the hyper-parameter τ (as shown in the Table) were chosen to maximize MRR on the validation set. The reported numbers were obtained for the test set.

base relations, and are not using any information from textual mentions. The lower portion of the Table shows the performance when textual relations are added to the training knowledge graph and the corresponding training loss function. Note that all models predict based on the learned knowledge base relation and entity representations, and the textual relations are only used at training time when they can impact these representations.

The performance of all models is shown as an overall MRR (scaled by 100) and HITS@10, as well as performance on the subset of triples that have textual mentions (column *With mentions*), and ones that do not (column *Without mentions*). Around 28% of the test triples have mentions and contribute toward the measures in the *With mentions* column, and the other 72% of the test triples contribute to the *Without mentions* column.

For the KB-only models, we see the performance of each individual model F, E, and DISTMULT. Model F was the best performing single model from (Riedel et al., 2013), but it does not perform well when textual mentions are not used. In our implementation of model F, we created entity pair parameters only for entity pairs that co-occur in the text data (Riedel et al. (2013) also trained pairwise vectors for co-occuring entities only, but all of the training and test tuples in their study were co-occurring)[7]. Without textual information, model F is performing essentially randomly, because entity pairs in the test sets do not occur in training set relations (by construction of the dataset). Model E is able to do surprisingly well, given that it is making predictions for each object position of a relation without considering the given subject of the relation. DISTMULT is the best performing single model. Unlike model F, it is able to share parameters among entity pairs with common subject or object entities, and, unlike model E, it captures some dependencies between the subject and object entities of a relation. The combination of models E+DISTMULT improves performance, but combining model F with the other two is not helpful.

The lower portion of Table 3 shows results when textual relations are added to the training knowledge graph. The basic models treat the textual relations as atomic and learn a separate latent feature vector for each textual relation. The CONV- models use the compositional representations of tex-

[7]Learning entity pair parameters for all entity pairs would result in 2.2 billion parameters for vectors with dimensionality 10 for our dataset. This was infeasible and was also not found useful based on experiments with vectors of lower dimensionality.

tual relations learned using the convolutional neural network architecture shown in Figure 4. We show the performance of each individual model and its corresponding variant with a CONV parameterization. For each model, we also show the optimal value of τ, the weight of the textual relations loss. Model F is able to benefit from textual relations and its performance increases by 2.5 points in MRR, with the gain in performance being particularly large on test triples with textual mentions. Model F is essentially limiting its space of considered argument fillers to ones that have co-occurred with the given subject entity. This gives it an advantage on test triples with textual mentions, but model F still does relatively very poorly overall when taking into account the much more numerous test triples without textual mentions. The CONV parameterization performs slightly worse in MRR, but slightly better in HITS@10, compared to the atomic parameterization. For model E and its CONV variant, we see that text does not help as its performance using text is the same as that when not using text and the optimal weight of the text is zero. Model DISTMULT benefits from text, and its convolutional text variant CONV-DISTMULT outperforms the basic model, with the gain being larger on test triples with mentions.

The best model overall, as in the KB-only case, is E+DISTMULT. The basic model benefits from text slightly and the model with compositional representations of textual patterns CONV-E+CONV-DISTMULT, improves the performance further, by 2.4 MRR overall, and by 5 MRR on triples with textual mentions. It is interesting that the text and the compositional representations helped most for this combined model. One hypothesis is that model E, which provides a prior over relation arguments, is needed in combination with DISTMULT to prevent the prediction of unlikely arguments based on noisy inference from textual patterns and their individual words and dependency links.

Hyperparameter Sensitivity

To gain insight into the sensitivity of the model to hyper-parameters and initialization, we report on experiments starting with the best model CONV-E + CONV-DISTMULT from Table 3 and varying one parameter at a time. This model has weight of the textual relations loss $\tau = 0.25$, weight of the L_2 penalty $\lambda = 1$, convolution window size of three, and is initialized randomly for the entity and KB relation vectors, and from pre-trained embeddings for word vectors (Turian et al., 2010). The overall MRR of the model is 40.4 on the validation set (test results are shown in the Table).

When the weight of τ is changed to 1 (i.e., equal contribution of textual and KB relations), the overall MRR goes down to 39.6 from 40.4, indicating the usefulness of weighting the two kinds of relations non-uniformly. When λ is reduced to 0.04, MRR is 40.0 and when λ is increased to 25, MRR goes down to 38.9. This indicates the L_2 penalty hyper-parameter has a large impact on performance. When we initialize the word embeddings randomly instead of using pre-trained word vectors, performance drops only slightly to 40.3. If we initialize from a model trained using KB-only information, performance goes down substantially to 38.7. This indicates that initialization is important and there is a small gain from using pre-trained word embeddings. There was a drop in performance to MRR 40.2 when using a window size of one for the convolutional architecture in Figure 4, and an increase to 40.6 when using a window size of five.

5 Conclusion and Future Work

Here we explored an alternative representation of textual relations for latent feature models that learn to represent knowledge base and textual relations in the same vector space. We showed that given the large degree of sharing of sub-structure in the textual relations, it was beneficial to compose their continuous representations out of the representations of their component words and dependency arc links. We applied a convolutional neural network model and trained it jointly with a model mapping entities and knowledge base relations to the same vector space, obtaining substantial improvements over an approach that treats the textual relations as atomic units having independent parameterization.

Acknowledgements

We would like to thank the anonymous reviewers for their suggestions, and Jianfeng Gao, Scott Wen-tau Yih, and Wei Xu for useful discussions.

References

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. *Dbpedia: A nucleus for a web of open data*. Springer.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250. ACM.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems (NIPS)*.

Kai-Wei Chang, Wen-tau Yih, Bishan Yang, and Christopher Meek. 2014. Typed tensor decomposition of knowledge bases for relation extraction. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. 2014. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *International Conference on Knowledge Discovery and Data Mining (KDD)*.

Evgeniy Gabrilovich, Michael Ringgaard, and Amarnag Subramanya. 2013. FACC1: Freebase annotation of ClueWeb corpora, Version 1 (release date 2013-06-26, format version 1, correction level 0).

Matt Gardner, Partha Pratim Talukdar, Bryan Kisiel, and Tom Mitchell. 2013. Improving learning and inference in a large knowledge-base using latent syntactic cues. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Matt Gardner, Partha Talukdar, Jayant Krishnamurthy, and Tom Mitchell. 2014. Incorporating vector space similarity in random walk inference over knowledge bases. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Matthew R Gormley, Mo Yu, and Mark Dredze. 2015. Improved relation extraction with feature-rich compositional embedding models. *arXiv preprint arXiv:1505.02419*.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S. Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *Association for Computational Linguistics (ACL)*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Ni Lao, Tom Mitchell, and William W Cohen. 2011. Random walk inference and learning in a large scale knowledge base. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Ni Lao, Amarnag Subramanya, Fernando Pereira, and William W Cohen. 2012. Reading the web with learned syntactic-semantic inference rules. In *Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP/CoNLL)*.

Dong C Liu and Jorge Nocedal. 1989. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1-3):503–528.

Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Association for Computational Linguistics and International Joint Conference on Natural Language Processing (ACL-IJCNLP)*.

Arvind Neelakantan, Benjamin Roth, and Andrew McCallum. 2015. Compositional vector space models for knowledge base completion. In *ACL*.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 809–816.

Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. 2015. A review of relational machine learning for knowledge graphs: From multi-relational link prediction to automated knowledge graph construction. *arXiv preprint arXiv:1503.00759*.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *Machine Learning and Knowledge Discovery in Databases*, pages 148–163. Springer.

Sebastian Riedel, Limin Yao, Benjamin M. Marlin, and Andrew McCallum. 2013. Relation extraction with matrix factorization and universal schemas. In *North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Martin Riedmiller and Heinrich Braun. 1993. A direct adaptive method for faster backpropagation learning: The rprop algorithm. In *Neural Networks, 1993., IEEE International Conference on*, pages 586–591. IEEE.

Alan Ritter, Luke Zettlemoyer, Oren Etzioni, et al. 2013. Modeling missing data in distant supervision for information extraction. *Transactions of the Association for Computational Linguistics*, 1:367–378.

Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*, pages 697–706. ACM.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D. Manning. 2012. Multi-instance multi-label learning for relation extraction. In *Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP/CoNLL)*.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*.

Kristina Toutanova and Danqi Chen. 2015. Observed versus latent features for knowledge base and text inference. In *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality*, pages 57–66. Association for Computational Linguistics.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 384–394. Association for Computational Linguistics.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014a. Knowledge graph and text jointly embedding. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). Association for Computational Linguistics*, pages 1591–1601.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014b. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, pages 1112–1119.

Jason Weston, Antoine Bordes, Oksana Yakhnenko, and Nicolas Usunier. 2013. Connecting language and knowledge bases with embedding models for relation extraction. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding entities and relations for learning and inference in knowledge bases. In *International Conference on Learning Representations (ICLR)*.

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

A Utility Model of Authors in the Scientific Community

Yanchuan Sim
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213, USA
ysim@cs.cmu.edu

Bryan R. Routledge
Tepper School of Business
Carnegie Mellon University
Pittsburgh, PA 15213, USA
routledge@cmu.edu

Noah A. Smith
Computer Science & Engineering
University of Washington
Seattle, WA 98195, USA
nasmith@cs.washington.edu

Abstract

Authoring a scientific paper is a complex process involving many decisions. We introduce a probabilistic model of some of the important aspects of that process: that authors have individual preferences, that writing a paper requires trading off among the preferences of authors as well as extrinsic rewards in the form of community response to their papers, that preferences (of individuals and the community) and tradeoffs vary over time. Variants of our model lead to improved predictive accuracy of citations given texts and texts given authors. Further, our model's posterior suggests an interesting relationship between seniority and author choices.

1 Introduction

Why do we write? As researchers, we write papers to report new scientific findings, but this is not the whole story. Authoring a paper involves a huge amount of decision-making that may be influenced by factors such as institutional incentives, attention-seeking, and pleasure derived from research on topics that excite us.

We propose that text collections and associated metadata can be analyzed to reveal optimizing behavior by authors. Specifically, we consider the ACL Anthology Network Corpus (Radev et al., 2013), along with author and citation metadata. Our main contribution is a method that infers two kinds of quantities about an author: her associations with interpretable research topics, which might correspond to relative expertise or merely to preferences among topics to write about; and a tradeoff coefficient that estimates the extent to which she writes papers that will be cited versus papers close to her preferences.

The method is based on a probabilistic model that incorporates assumptions about how authors decide what to write, how joint decisions work when papers are coauthored, and how individual and community preferences shift over time. Central to our model is a low-dimensional topic representation shared by authors (in defining preferences), papers (i.e., what they are "about"), and the community as a whole (in responding with citations). This method can be used to make predictions; empirically, we find that:

1. topics discovered by generative models outperform a strong text regression baseline (Yogatama et al., 2011) for citation count prediction;
2. such models do better at that task *without* modeling author utility as we propose; and
3. the author utility model leads to better predictive accuracy when answering the question, "given a set of authors, what are they likely to write?"

This method can also be used for exploration and to generate hypotheses. We provide an intriguing example relating author tradeoffs to age within the research community.

2 Notation and Representations

In the following, a document d will be represented by a vector $\boldsymbol{\theta}_d \in \mathbb{R}^K$. The dimensions of this vector might correspond to elements of a vocabulary, giving a "bag of words" encoding; in this work they correspond to latent topics.

Document d is assumed to elicit from the scientific community an observable response y_d, which might correspond to the number of citations (or downloads) of the paper.

Each author a is associated with a vector $\boldsymbol{\eta}_a \in \mathbb{R}^K$, with dimensions indexed the same as documents. Below, we will refer to this vector as a's "preferences," though it is important to remember that they could also capture an author's *expertise*,

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1510–1519,
Lisbon, Portugal, 17-21 September 2015.

and the model makes no attempt to distinguish between them. We use "preferences" because it is a weaker theoretical commitment.

3 Author Utility Model

We describe the components of our model—author utility (§3.1), coauthorship (§3.2), topics (§3.3), and temporal dynamics (§3.4)—then give the full form in §3.5.

3.1 Modeling Utility

Our main assumption about author a is that she is an optimizer: when writing document d she seeks to increase the response y_d while keeping the contents of d, $\boldsymbol{\theta}_d$, "close" to her preferences $\boldsymbol{\eta}_a$. We encode her objectives as a utility function to be maximized with respect to $\boldsymbol{\theta}_d$:

$$U(\boldsymbol{\theta}_d) = \kappa_a y_d - \frac{1}{2}\|\boldsymbol{\theta}_d - (\boldsymbol{\eta}_a + \boldsymbol{\epsilon}_{d,a})\|_2^2 \quad (1)$$

where $\boldsymbol{\epsilon}_{d,a}$ is an author-paper-specific idiosyncratic randomness that is unobserved to us but assumed known to the author. (This is a common assumption in discrete choice models. It is often called a "random utility model.")

Notice the tradeoff between maximizing the response y_d and staying close to one's preferences. We capture these competing objectives by formulating the latter as a squared Euclidean distance between $\boldsymbol{\eta}_a$ and $\boldsymbol{\theta}_d$, and encoding the tradeoff between extrinsic (citation-seeking) and intrinsic (preference-satisfying) objectives as the (positive) coefficient κ_a. If κ_a is large, a might be understood as a citation-maximizing agent; if κ_a is small, a might appear to care much more about certain kinds of papers ($\boldsymbol{\eta}_a$) than about citation.

This utility function considers only two particular facets of author writing behavior; it does not take into account other factors that may contribute to an author's objective. For this reason, some care is required in interpreting quantities like κ_a. For example, divergence between a particular $\boldsymbol{\eta}_a$ and $\boldsymbol{\theta}_d$ might suggest that a is open to new topics, not merely hungry for citations. Other motivations, such as reputation (notoriously difficult to measure), funding maintenance, and the preferences of peer referees are not captured in this model. Similarly for preferences $\boldsymbol{\eta}_a$, a large value in this vector might reflect a's skill or the preferences of a's sponsors rather than a's personal interest the topic.

Next, we model the response y_d. We assume that responses are driven largely by topics, with some noise, so that

$$y_d = \boldsymbol{\beta}^\top \boldsymbol{\theta}_d + \xi_d \quad (2)$$

where $\xi_d \sim \mathcal{N}(0, 1)$. Because the community's interest in different topics varies over time, $\boldsymbol{\beta}$ is given temporal dynamics, discussed in §3.4.

Under this assumption, the author's *expected* utility assuming she is aware of $\boldsymbol{\beta}$ (often called "rational expectations" in discrete choice models), is:

$$\mathbb{E}[U(\boldsymbol{\theta}_d)] = \kappa_a \boldsymbol{\beta}^\top \boldsymbol{\theta}_d - \frac{1}{2}\|\boldsymbol{\theta}_d - (\boldsymbol{\eta}_a + \boldsymbol{\epsilon}_{d,a})\|_2^2 \quad (3)$$

(This is obtained by plugging the expected value of y_d, from Eq. 2, into Eq. 1.)

An author's decision will therefore be

$$\hat{\boldsymbol{\theta}}_d = \arg\max_{\boldsymbol{\theta}} \kappa_a \boldsymbol{\beta}^\top \boldsymbol{\theta} - \frac{1}{2}\|\boldsymbol{\theta} - (\boldsymbol{\eta}_a + \boldsymbol{\epsilon}_{d,a})\|_2^2 \quad (4)$$

Optimality implies that $\hat{\boldsymbol{\theta}}_d$ solves the first-order equations

$$\kappa_a \beta_j - (\hat{\theta}_{d,j} - (\eta_{a,j} + \epsilon_{d,a,j})) = 0, \ \forall 1 \le j \le K \quad (5)$$

Eq. 5 highlights the tradeoff the author faces: when $\beta_j > 0$, the author will write more on $\theta_{d,j}$, while straying too far from $\eta_{a,j}$ incurs a penalty.

3.2 Modeling Coauthorship

Matters become more complicated when multiple authors write a paper together. Suppose the document d is authored by set of authors $\boldsymbol{a}_d$. We model the joint expected utility of $\boldsymbol{a}_d$ in writing $\boldsymbol{\theta}_d$ as the average of the group's utility.[1]

$$\mathbb{E}[U(\boldsymbol{\theta}_d)] = \frac{1}{|\boldsymbol{a}_d|}\sum_{a \in \boldsymbol{a}_d}\Big(\kappa_a \boldsymbol{\beta}^\top \boldsymbol{\theta}_d - \frac{1}{2} c_{d,a}\|\boldsymbol{\theta}_d - (\boldsymbol{\eta}_a + \boldsymbol{\epsilon}_{d,a})\|_2^2\Big) \quad (6)$$

where the "cost" term is scaled by $c_{d,a}$, denoting the fractional "contribution" of author a to document d. Thus, $\sum_{a \in \boldsymbol{a}_d} c_{d,a} = 1$, and we treat $\boldsymbol{c}_d$ as a latent categorical distribution to be inferred. The first-order equation becomes

$$\sum_{a \in \boldsymbol{a}_d} \kappa_a \boldsymbol{\beta} - c_{d,a}(\boldsymbol{\theta}_d - (\boldsymbol{\eta}_a + \boldsymbol{\epsilon}_{d,a})) = \mathbf{0} \quad (7)$$

[1] This assumption is a convenient starting place, but we can imagine revisiting it in future work. For example, an economist and a linguist with different expertise might derive "utility" from the collaboration that is non-linear in each one's individual preferences (Anderson, 2012). Further, contributions by complementary authors are not expected to be independent of each other.

3.3 Modeling Document Content

As noted before, there are many possible ways to represent and model document content $\boldsymbol{\theta}_d$. We treat $\boldsymbol{\theta}_d$ as (an encoding of) a mixture of topics. Following considerable past work, a "topic" is defined as a categorical distribution over observable tokens (Blei et al., 2003; Hofmann, 1999). Let $\boldsymbol{w}_d$ be the observed bag of tokens constituting document d. We assume each token is drawn from a mixture over topics:

$$p(\boldsymbol{w}_d \mid \boldsymbol{\theta}_d) = \sum_{\boldsymbol{z}_d} \prod_{i=1}^{N_d} p(z_{d,i} \mid \boldsymbol{\theta}_d) p(w_{d,i} \mid \boldsymbol{\phi}_{z_{d,i}})$$

where N_d is the number of tokens in document d, $z_{d,i}$ is the topic assignment for d's ith token $w_{d,i}$, and $\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_K$ are topic-term distributions. Note that $\boldsymbol{\theta}_d \in \mathbb{R}^K$; we define $p(z \mid \boldsymbol{\theta}_d)$ as a categorical draw from the softmax-transformed $\boldsymbol{\theta}_d$ (Blei and Lafferty, 2007).

Using topic mixtures instead of a bag of words provides us with a low-dimensional interpretable representation that is useful for analyzing authors' behaviors and preferences. Each dimension j of an author's preference is grounded in topic j. If we ignore document responses, this component of model closely resembles the author-topic model (Rosen-Zvi et al., 2004), except that we assume a different prior for the topic mixtures.

3.4 Modeling Temporal Dynamics

Individual preferences shift over time, as do those of the research community. We extend our model to allow variation at different timesteps. Let $t \in \langle 1, \ldots, T \rangle$ index timesteps (in our experiments, each t is a calendar year). We let $\boldsymbol{\beta}^{(t)}$, $\boldsymbol{\eta}_a^{(t)}$, and $\kappa_a^{(t)}$ denote the community's response coefficients, author a's preferences, and author a's tradeoff coefficient at timestep t.

Again, we must take care in interpreting these quantities. Do changes in community interest drive authors to adjust their preferences or expertise? Or do changing author preferences aggregate into community-wide shifts? Or do changes in the economy or funding availability change authors' tradeoffs? Our model cannot differentiate among these different causal patterns. Our method is useful for tracking these changes, but it does not provide an explanation for *why* they take place.

Modeling the temporal dynamics of a vector-valued random variable can be accomplished using a multivariate Gaussian distribution. Following Yogatama et al. (2011), we assume the prior for $\boldsymbol{\beta}_j^{(\cdot)} = \langle \beta_j^{(1)}, \ldots, \beta_j^{(T)} \rangle$ has a tridiagonal precision matrix $\Lambda(\lambda, \alpha) \in \mathbb{R}^{T \times T}$:

$$\Lambda(\lambda, \alpha) = \lambda \begin{pmatrix} 1+\alpha & -\alpha & 0 & 0 & \ldots \\ -\alpha & 1+2\alpha & -\alpha & 0 & \ldots \\ 0 & -\alpha & 1+2\alpha & -\alpha & \ldots \\ 0 & 0 & -\alpha & 1+2\alpha & \ldots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix}$$

The two hyperparameters α and λ capture, respectively, autocorrelation (the tendency of $\beta_j^{(t+1)}$ to be similar to $\beta_j^{(t)}$) and overall variance. This approach to modeling time series allows us to capture temporal dynamics while sharing statistical strength of evidence across all time steps.

We use the notation $\mathcal{T}(\lambda, \alpha) \equiv \mathcal{N}(\mathbf{0}, \Lambda(\lambda, \alpha))$ for this multivariate Gaussian distribution, instances of which are used as priors over response coefficients $\boldsymbol{\beta}$, author preferences $\boldsymbol{\eta}_a$, and (transformed) author tradeoffs $\log \kappa_a$.

.

Observed evidence	
$w_{d,i}$	ith token in document d
V	vocabulary size
N_d	number of tokens in document d
y_d	response to document d
$\mathcal{A}$	the set of authors
$\boldsymbol{a}_d$	set of authors of document d ($\subseteq \mathcal{A}$)
T	number of timesteps
$\mathcal{D}_t$	the set of documents from timestep t
$\mathcal{D}$	the set of all documents ($= \bigcup_{t=1}^{T} \mathcal{D}_t$)
Latent variables	
$\boldsymbol{\beta}^{(t)}$	response coefficients at time t ($\in \mathbb{R}^K$)
$\boldsymbol{\eta}_a^{(t)}$	author a's topic preferences at time t ($\in \mathbb{R}^K$)
$\boldsymbol{\kappa}_a^{(t)}$	author a's tradeoff coefficient at time t ($\in \mathbb{R}_{\geq 0}$)
$\boldsymbol{\theta}_d$	document d topic associations ($\in \mathbb{R}^K$)
$c_{d,a}$	author a contrtibution to document d ($\sum_{a \in \boldsymbol{a}_d} c_{d,a} = 1$)
$\boldsymbol{\phi}_k$	distribution over terms for topic k
$z_{d,i}$	topic assignment of $w_{d,i}$
Constants and hyperparameters	
K	number of topics
ρ	symmetric Dirichlet hyperparameter for $\boldsymbol{\phi}_k$
σ_c^2	variance hyperparameter for author contributions $\boldsymbol{c}_d$
$\{\lambda^{(\beta)}, \alpha^{(\beta)}\}$, $\{\lambda^{(\eta)}, \alpha^{(\eta)}\}$, $\{\lambda^{(\kappa)}, \alpha^{(\kappa)}\}$	hyperparameters for priors of $\boldsymbol{\beta}, \boldsymbol{\eta}$, and $\log \kappa$ respectively

Table 1: Table of notation.

3.5 Full Model

Table 1 summarizes all of the notation. The log-likelihood of our model is:

$$\begin{aligned}\mathcal{L} &= \log p(\boldsymbol{\beta}) + \sum_{d \in \mathcal{D}} \log p(\boldsymbol{c}_d) \\ &+ \sum_{d \in \mathcal{D}} \log p(y_d \mid \boldsymbol{\theta}_d, \boldsymbol{\beta}) + \log p(\boldsymbol{w}_d \mid \boldsymbol{\theta}_d) \\ &+ \sum_{a \in \mathcal{A}} \log p(\boldsymbol{\eta}_a) + \log p(\kappa_a) \\ &+ \sum_{d \in \mathcal{D}} \sum_{a \in \boldsymbol{a}_d} \log p(\boldsymbol{\theta}_d \mid \boldsymbol{\beta}, \boldsymbol{\eta}_a, \kappa_a, c_{d,a}) \qquad (8)\end{aligned}$$

We adopt a Bayesian approach to parameter estimation. The generative story, including all priors, is as follows. Recall that $\mathcal{T}(\cdot, \cdot)$ denotes the time series prior discussed in §3.4. See also the plate diagram for the graphical model in Fig. 1.

1. For each topic $k \in \{1, \ldots, K\}$:
   (a) Draw response coefficients $\boldsymbol{\beta}_k^{(\cdot)} \sim \mathcal{T}(\lambda^{(\beta)}, \alpha^{(\beta)})$ and term distribution $\boldsymbol{\phi}_k \sim \text{Dirichlet}(\rho)$.
   (b) For each author $a \in \mathcal{A}$, draw preference strengths for topic k over time, $\langle \eta_{a,k}^{(1)}, \ldots, \eta_{a,k}^{(t)} \rangle \sim \mathcal{T}(\lambda^{(\eta)}, \alpha^{(\eta)})$.
2. For each author $a \in \mathcal{A}$, draw (transformed) tradeoff parameters $\langle \log \kappa_a^{(1)}, \ldots, \log \kappa_a^{(T)} \rangle \sim \mathcal{T}(\lambda^{(\kappa)}, \alpha^{(\kappa)})$.
3. For each timestep $t \in \{1, \ldots, T\}$, and each document $d \in \mathcal{D}_t$:
   (a) Draw author contributions $\boldsymbol{c}_d \sim \text{Softmax}(\mathcal{N}(\mathbf{0}, \sigma_c^2 \mathbf{I}))$. This is known as a logistic normal distribution (Aitchison, 1986).
   (b) Draw d's topic distributions (this distribution is discussed further below):

$$\boldsymbol{\theta}_d \sim \mathcal{N}\left(\sum_{a \in \boldsymbol{a}_d} \kappa_a^{(t)} \boldsymbol{\beta}^{(t)} + c_{d,a} \boldsymbol{\eta}_a^{(t)}, \|\boldsymbol{c}_d\|_2^2 \mathbf{I} \right) \qquad (9)$$

   (c) For each token $i \in \{1, \ldots, N_d\}$, draw topic $z_{d,i} \sim \text{Categorical}(\text{Softmax}(\boldsymbol{\theta}_d))$ and term $w_{d,i} \sim \text{Categorical}(\boldsymbol{\phi}_{z_{d,i}})$.
   (d) Draw response $y_d \sim \mathcal{N}\left(\boldsymbol{\beta}^{(t_d)\top} \boldsymbol{\theta}_d, 1\right)$; note that it collapses out ξ_d, which is drawn from a standard normal.

Eq. 9 captures the choice by authors $\boldsymbol{a}_d$ of a distribution over topics $\boldsymbol{\theta}_d$. Assuming that the $\boldsymbol{\epsilon}_{d,a}$s are i.i.d. and Gaussian, from Eq. 7, we get

$$\boldsymbol{\theta}_d = \sum_{a \in \boldsymbol{a}_d} \kappa_a \boldsymbol{\beta} + c_{d,a} \boldsymbol{\eta}_a + c_{d,a} \boldsymbol{\epsilon}_{d,a},$$

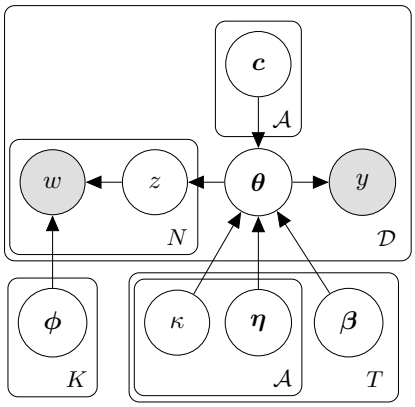


Figure 1: Plate diagram for author utility model. Hyperparameters and edges between consecutive time steps of $\boldsymbol{\beta}, \boldsymbol{\eta}$ and κ are omitted for clarity.

and the linear additive property of Gaussians gives us

$$\boldsymbol{\theta}_d \sim \mathcal{N}\left(\sum_{a \in \boldsymbol{a}_d} \kappa_a \boldsymbol{\beta} + c_{d,a} \boldsymbol{\eta}_a, \|\boldsymbol{c}_d\|_2^2 \mathbf{I} \right)$$

In §3.1 we described a utility function for each author. The model we are estimating is similar to those estimated in discrete choice econometrics (McFadden, 1974). We assumed that authors are utility maximizing (optimizing) and that their optimal topic distribution satisfies the first-order conditions (Eq. 7). However, we cannot see the idiosyncratic component, $\boldsymbol{\epsilon}_{d,a}$, which is assumed to be Gaussian; as noted, this is known as a random utility model. Together, these assumptions give the structure of the distribution over topics in terms of (estimated) utility, which allows us to naturally incorporate the utility function into our probabilistic model in a familiar way (Sim et al., 2015).

4 Learning and Inference

Exact inference in our model is intractable, so we resort to an approximate inference technique based on Monte Carlo EM (Wei and Tanner, 1990). During the E-step, we perform Bayesian inference over latent parameters $(\boldsymbol{\eta}, \boldsymbol{\kappa}, \boldsymbol{z}, \boldsymbol{\theta}, \boldsymbol{c}, \boldsymbol{\phi})$ using a Metropolis-Hastings within Gibbs algorithm (Tierney, 1994), and in the M-step, we compute maximum *a posteriori* estimates of $\boldsymbol{\beta}$ by directly optimizing the log-likelihood function. Since we are using conjugate priors for $\boldsymbol{\phi}$, we can integrate it out. We did not perform Bayesian posterior inference over $\boldsymbol{\beta}$ because the coupling of $\boldsymbol{\beta}$

would slow mixing of the MCMC chain.

E-step. We sample each $\boldsymbol{\eta}_a^{(t_d)}$, $\boldsymbol{c}_d$, $\log \boldsymbol{\kappa}_a^{(\cdot)}$, and $\boldsymbol{\theta}_d$ blockwise using the Metropolis-Hastings algorithm with a multivariate Gaussian proposal distribution, tuning the diagonal covariance matrix to a target acceptance rate of 15-45% (see appendix §A for sampling equations).

For $\boldsymbol{z}$, we integrate out $\boldsymbol{\phi}$ and sample each $z_{d,i}$ directly from

$$p(z_{d,i} = k \mid \boldsymbol{\theta}_d, \boldsymbol{\phi}_k) \propto \exp(\theta_{d,k}) \frac{C_{k,w_{d,i}}^{-d,i} + \rho}{C_{k,\cdot}^{-d,i} + V\rho}$$

where $C_{k,w}^{-d,i}$ and $C_{k,\cdot}^{-d,i}$ are the number of times w is associated with topic k, and the number of tokens associated with topic k respectively.

We run the E-step Gibbs sampler to collect 3,500 samples, discarding the first 500 samples for burn-in and only saving samples at every third iteration.

M-step. We approximate the expectations of our latent variables using the samples collected during the E-step, and directly optimize $\boldsymbol{\beta}^{(t)}$ using L-BFGS (Liu and Nocedal, 1989),[2] which requires a gradient. The gradient of the log-likelihood with respect to $\beta_j^{(t)}$ is

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \beta_j^{(t)}} &= -2\lambda^{(\beta)} \beta_j^{(t)} \\
&\quad - 2\lambda^{(\beta)} \alpha^{(\beta)} \mathbf{1}\{t > 1\} (\beta_j^{(t)} - \beta_j^{(t-1)}) \\
&\quad - 2\lambda^{(\beta)} \alpha^{(\beta)} \mathbf{1}\{t < T\} (\beta_j^{(t)} - \beta_j^{(t+1)}) \\
&\quad + 2 \sum_{d \in \mathcal{D}_t} \theta_{d,j} (y_d - \beta_j^{(t)} \theta_{d,j}) \\
&\quad + 2 \sum_{d \in \mathcal{D}_t} \kappa_d^{(t)} \left(\theta_{d,j} - \kappa_d^{(t)} \beta_j^{(t)} - \sum_{a \in \boldsymbol{a}_d} \frac{\eta_{a,j}^{(t)}}{|\boldsymbol{a}_d|} \right)
\end{aligned} \tag{10}$$

where $\kappa_d^{(t)} = \frac{1}{|\boldsymbol{a}_d|} \sum_{a \in \boldsymbol{a}_d} \kappa_a^{(t)}$.

We ran L-BFGS until convergence[3] and slice sampled the hyperparameters $\lambda^{(\eta)}, \alpha^{(\eta)}, \lambda^{(\kappa)}, \alpha^{(\kappa)}$ (with vague priors) at the end of the M-step. We fix the symmetric Dirichlet hyperparameter $\rho = 1/V$, and tuned $\lambda^{(\beta)}, \alpha^{(\beta)}$ on a held-out developement dataset by grid search over $\{0.01, 0.1, 1, 10\}$.

[2]We used libLBFGS, an open source C++ implementation (`https://github.com/chokkan/liblbfgs`).

[3]Relative tolerance of 10^{-4}.

During initialization, we randomly set the topic assignments, while the other latent parameters are set to 0. We ran the model for 10 EM iterations.

Inference. During inference, we fix the model parameters and only sample $(\boldsymbol{\theta}, \boldsymbol{z})$ for each document. As in the E-step, we discard the first 500 samples, and save samples at every third iteration, until we have 500 posterior samples. In our experiments, we found the posterior samples to be reasonably stable after the initial burn in.

5 Experiments

Data. The ACL Anthology Network Corpus contains 21,212 papers published in the field of computational linguistics between 1965 and 2013 and written by 17,792 authors. Additionally, the corpus provides metadata such as authors, venue and in-community citation networks. For our experiments, we focused on conference papers published between 1980 and 2010.[4] We tokenized the texts, tagged the tokens using the Stanford POS tagger (Toutanova et al., 2003), and extracted n-grams with tags that follow the simple (but effective) pattern of `(Adj|Noun)* Noun` (Justeson and Katz, 1995), representing the dth document as a *bag of phrases* ($\boldsymbol{w}_d$). Note that phrases can also be unigrams. We pruned phrases that appear in $< 1\%$ or $> 95\%$ of the documents, obtaining a vocabulary of V = 6,868 types. The pruned corpus contains 5,498 documents and 2,643,946 phrase tokens written by 5,575 authors. We let responses

$$y_d = \log(1 + \#\text{ of incoming citations in 3 years})$$

For our experiments, we used 3 different random splits of our data (70% train, 20% test, and 10% development) and averaged quantities of interest. Furthermore, we remove an author from a paper in the development or test set if we have not seen him before in the training data.

5.1 Examples of Authors and Topics

Table 2 illustrates ten manually selected topics (out of 64) learned by the author utility model. Each topic is labeled with the top 10 words most likely to be generated conditioned on the topic

[4]The conferences we included are: ACL, CoNLL, EACL, EMNLP, HLT, and NAACL. We ignored journal papers, as well as workshop papers, since they are characteristically different from conference papers.

(ϕ_k). For each topic, we compute an author's topic preference score:

$$\text{TPS}(a, k) = \eta_{a,k}^{(t_d)} \sum_{d \in D_a} [\text{Softmax}(\theta_d)]_k \times y_d$$

where $\text{Softmax}(\boldsymbol{x}) = \frac{\exp(\boldsymbol{x})}{\sum_i \exp(x_i)}$. The TPS scales the author's η preferences by the relative number of citations that the author received for the topic. This way, we can account for different ηs over time, and reduce variance due to authors who publish less frequently.[5] For each topic, the five authors with the highest TPS are displayed in the rightmost column of Table 2. These topics were among the roughly one third (out of 64) that seemed to coherently map to research topics within NLP. Some others corresponded to parts of a paper (e.g., explaining notation and formulae, experiments) or to stylistic groups (e.g., "rational words" including *rather*, *fact*, *clearly*, *argue*, *clear*, *perhaps*). Others were not interpretable to us.

5.2 Predicting Responses

We compare against two baselines for predicting in-community citations. Yogatama et al. (2011) is a strong baseline for predicting responses; they incorporated n-gram features and metadata features in a generalized linear model with the time series prior discussed in §3.4.[6] We also compare against a version of our model without the author utility component. This equates to replacing Yogatama et al.'s features with LDA topic mixtures, and performing joint learning of the topics and citations; we therefore call it "TimeLDA." Without the time series component, TimeLDA would instantiate supervised LDA (McAuliffe and Blei, 2008). Figure 2 shows the mean absolute error (MAE) for the three models.

With sufficiently many topics ($K \geq 16$), topic representations achieve lower error than surface features. Removing the author utility component from our model leads to better predictive performance. This is unsurprising, since our model forces β to explain both the responses (what is evaluated here) and the divergence between author preferences η_a and what is actually written. The utility model is nonetheless competitive with the Yogatama et al. baseline.

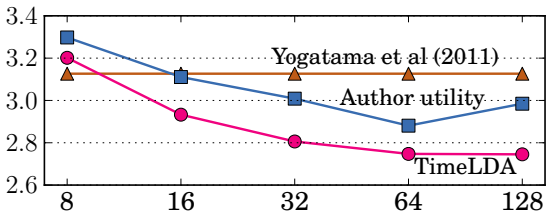


Figure 2: Mean absolute error (in citation counts) for predicted citation counts (y-axis) against the number of topics K (x-axis). Errors are in actual citation counts, while the models are trained with log counts. TimeLDA significantly outperforms Yogatama et al. (2011) for $K \geq 64$ (paired t-test, $p < 0.01$), while the differences between Yogatama et al. (2011) and author utility are not significant. The MAE is calculated over 3 random splits of the data with 809, 812, and 811 documents in the test set respectively.

5.3 Predicting Words

"Given a set of authors, what are they likely to write?" — we use perplexity as a proxy to measure the content predictive ability of our model. Perplexity on a test set is commonly used to quantify the generalization ability of probabilistic models and make comparisons among models over the same observation space. For a document $\boldsymbol{w}_d$ written by authors $\boldsymbol{a}_d$, perplexity is defined as

$$\text{perplexity}(\boldsymbol{w}_d \mid \boldsymbol{a}_d) = \exp\left(-\frac{\log p(\boldsymbol{w}_d \mid \boldsymbol{a}_d)}{N_d}\right)$$

and a lower perplexity indicates better generalization performance. Using S samples from the inference step, we can compute

$$p(\boldsymbol{w}_d \mid \boldsymbol{a}_d) = \frac{1}{S} \sum_{s=1}^{S} \prod_{i=1}^{N_d} \frac{1}{|\boldsymbol{a}_d|} \sum_{a \in \boldsymbol{a}_d, k} \theta_{d,k}^s \phi_{k,w_{di}}^s$$

where θ^s is the sth sample of θ, and ϕ^s is the topic-word distribution estimated from the sth sample of $\boldsymbol{z}$.

We compared the Author-Topic model of Rosen-Zvi et al. (2004). The AT model is similar to setting $\kappa_a = 0$ for all authors, $\boldsymbol{c}_d = \frac{1}{|\boldsymbol{a}_d|}$, and using a Dirichlet prior instead of logistic normal on η_a. Figure 3 present the perplexity of these

[5] The TPS is only a measure of an author's propensity to write papers in a specific topic area and is not meant to be a measure of an author's reputation in a particular research sub-field.

[6] For the ACL dataset, Yogatama et al. (2011)'s model predicts whether a paper will receive at least 1 citation within three years, while here, we train it to predict $\log(1 + \#\text{citations})$ instead.

Topic	Top words	Authors
"MT"	alignment, translation, align, decode, phrase, och, bleu, ney, bleu score, target language	Philipp Koehn, Chris Dyer, Qun Liu, Hermann Ney, David Chiang
"Empirical methods"	model, parameter, learn, iteration, maximize, prior, initialize, distribution, weight, crf	Noah Smith, Dan Klein, Percy Liang, John DeNero, Andrew McCallum
"Parsing"	parse, sentence, parser, accuracy, collins, dependency, tree, parse tree, head, charniak	Michael Collins, Joakim Nivre, Jens Nilsson, Dan Klein, Ryan McDonald
"Dialogue systems"	speak, speech, utterance, user, speaker, dialogue system, turn, act, recognition, transcription	Diane Litman, Marilyn Walker, Julia Hirschberg, Oliver Lemon, Amanda Stent
"NER"	name, entity, identify, person, location, list, organization, system, entity recognition, mention	Jenny Rose Finkel, Satoshi Sekine, Rion Snow, Christopher Manning, Abraham Ittycheriah
"Semantics"	argument, verb, predicate, syntactic, relation, semantic role, annotated, frame, assign	Martha Palmer, Alessandro Moschitti, Daniel Jurafsky, Sanda Harabagiu, Mirella Lapata
"Lexical semantics"	wordnet, noun, sense, concept, context, sens, relation, meaning, pair, disambiguate	Rion Snow, Rob Koeling, Eneko Agirre, Ido Dagan, Patrick Pantel
"Tagging & chunking"	method, sentence, propose, japanese, noun phrase, extract, table, analyze, precision, technology	Yuji Matsumoto, Hitoshi Isahara, Junichi Tsujii, Sadao Kurohashi, Kentaro Torisawa
"Coreference"	mention, instance, create, approach, report, due, text, pair, exist, system	Vincent Ng, Aria Haghighi, Xiaofeng Yang, Claire Cardie, Pascal Denis
"Sentiment classification"	classify, label, accuracy, positive, classification, annotated, annotator, classifier, review, negative	Janyce Wiebe, Soo Min Kim, Eduard Hovy, Carmen Banea, Ryan McDonald

Table 2: Top words from selected topics and authors with preferences in those topics. We manually labeled each of these topics.

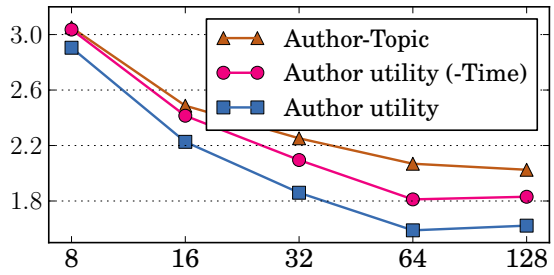


Figure 3: Held-out perplexity ($\times 10^3$, y-axis) with varying number of topics K (x-axis). The differences are significant between all models at $K \geq 64$ (paired t-test, $p < 0.01$). There are 523,381, 529,397, 533,792 phrase tokens in the random test sets.

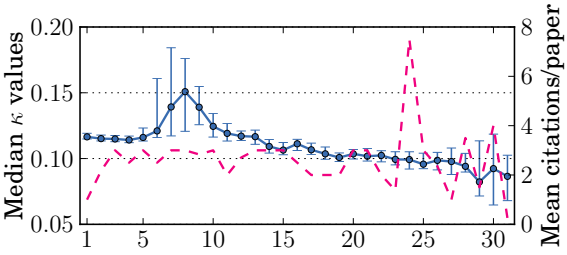


Figure 4: Plot of authors' median κ (blue, solid) and mean citation counts (magenta, dashed) against their academic age in this dataset (see text for explanation).

models at different values of K. We include a version of our author utility model that ignores temporal information ("–time"), i.e., setting $T = 1$ and collapsing all timesteps. We find that perplexity improves with the addition of the utility model as well as the temporal dynamics.

5.4 Exploration: Tradeoffs and Seniority

Recall that κ_a encodes author a's tradeoff between increasing citations (high κ_a) and writing papers on topics a prefers (low κ_a). We do not claim that individual κ_a values consistently represent authors' tradeoffs between citations and writing about preferred topics. We have noted a number of potentially confounding factors that affect authors' choices, for which our data do not allow us to control.

However, in aggregate, κ_a values can be explored in relation to other quantities. Given our model's posterior, one question we can ask is: do an author's tradeoffs tend to change over the course of her career? In Figure 4, we plot the median of κ (and 95% credible intervals) for authors at different "ages." Here, "age" is defined as the number of years since an author's first publication in this dataset.[7]

A general trend over the long term is observed: researchers appear to move from higher to lower κ_a. Statistically, there is significant dependence between κ of an author and her age; the Spearman's rank correlation coefficient is $\rho = -0.870$ with p-value $< 10^{-5}$. This finding is consis-

[7]This means that larger ages correspond to seniority, but smaller ages are a blend of junior researchers and researchers of any seniority new to this publication community.

tent with the idea that greater seniority brings increased and more stable resources and greater freedom to pursue idiosyncratic interests with less concern about extrinsic payoff. It is also consistent with decreased flexibility or openness to shifting topics over time.

To illustrate the importance of our model in making these observations, we also plot the mean number of citations per paper published (across all authors) against their academic age (magenta lines). There is no clear statistical trend between the two variables ($\rho = -0.017$). This suggests that through κ, our model is able to pick up evidence of author's optimizing behaviors, which is not possible using simple citation counts.

There is a noticeable effect during years 5–10, in which κ tends to rise by around 40% and then fall back. (Note that the model maintains considerable uncertainty—wider intervals—about this effect.) Recall that, for a researcher trained within the field and whose primary publication venue is in the ACL community, our measure of age corresponds roughly to academic age. Years 5–10 would correspond to the later part of a Ph.D. program and early postgraduate life, when many researchers begin faculty careers. Insofar as it reflects a true effect, this rise and fall suggests a stage during which a researcher focuses more on writing papers that will attract citations. However, more in-depth study based on data that is not merely observational is required to quantify this effect and, if it persists under scrutiny, determine its cause.

The effect in year 24 of mean citations per paper (magenta line) can be attributed to well cited papers co-authored by senior researchers in the field who published very few papers in their 24th year. Since there are relatively few authors in the dataset at that academic age, there is more variance in mean citations counts.

6 Related Work

Previous work on modeling author interests mostly focused on characterizing authors by their style (Holmes and Forsyth, 1995, *inter alia*),[8] through latent topic mixtures of documents they have co-authored (Rosen-Zvi et al., 2004) and their collaboration networks (Johri et al., 2011). Like our paper, the latter two are based on topic models, which have been popular for modeling the content of scientific articles. For instance, Gerrish and Blei (2010) measured scholarly impact using dynamic topic models, while Hall et al. (2008) analyzed the output of topic models to study the "history of ideas."

Predicting responses to scientific articles was explored in two shared tasks at KDD Cup 2003 (Brank and Leskovec, 2003; McGovern et al., 2003) and by Yogatama et al. (2011), which served as a baseline for our experiments and whose time-series prior we used in our model. Furthermore, there has been considerable research using topic models to predict (or recommend) citations (instead of aggregate counts), such as modeling link probabilities within the LDA framework (Cohn and Hofmann, 2000; Erosheva et al., 2004; Nallapati and Cohen, 2008; Kataria et al., 2010; Zhu et al., 2013) and augmenting topics with discriminative author features (Liu et al., 2009; Tanner and Charniak, 2015).

We modeled both interests of authors and responses to their articles jointly, by assuming authors' text production is an expected utility-maximizing decision. This approach is similar to our earlier work (Sim et al., 2015), where authors are rational agents writing texts to maximize the chance of a favorable decision by a judicial court. In that study, we did not consider the unique preferences of each decision making agent, nor the extrinsic-intrinsic reward tradeoffs that these agents face when authoring a document.

Our utility model can also be viewed as a form of natural language generator, where we take into account the context of an author (i.e., his preferences, the tradeoff coefficient, and what is popular) to generate his document. This is related to natural language pragmatics, where text is influenced by context.[9] Hovy (1990) approached the problem of generating text under pragmatic circumstances from a planning and goal-orientation perspective, while Vogel et al. (2013) used multi-agent decision-theoretic models to show cooperative pragmatic behavior. Vogel et al.'s models suggest an interesting extension of ours for future work: modeling cooperation among co-authors and, perhaps, in the larger scientific discourse.

[8] A closely related problem is that of authorship attribution. There has been extensive research on authorship attribution focusing mainly on learning "stylometric" features of authors; see Stamatatos (2009) for a detailed review.

[9] The $\boldsymbol{\beta}$ vectors can be seen as a naïve representation of world knowledge that motivates an author to select content that reflects his behavioral preferences and intentions.

7 Conclusions

We presented a model of scientific authorship in which authors trade off between seeking citation by others and staying true to their individual preferences among research topics. We find that topic modeling improves over state-of-the-art text regression models for predicting citation counts, and that the author utility model generalizes better than simpler models when predicting what a particular group of authors will write. Inspecting our model suggests interesting patterns in behavior across a researcher's career.

Acknowledgements

The authors thank the anonymous reviewers for their thoughtful feedback and members of the ARK group at CMU for their valuable comments. This research was supported in part by an A*STAR fellowship to Y. Sim, by a Google research award, and by computing resources from the Pittsburgh Supercomputing Center; it was completed while NAS was at CMU.

A Appendix: Sampling equations

We sample each $\boldsymbol{\eta}_{a,j}$, for $j = 1 \dots K$, and $\boldsymbol{\kappa}_a$ blockwise across time steps using Metropolis-Hastings algorithm with a multivariate Gaussian proposal distribution and likelihood:

$$p(\boldsymbol{\eta}_{a,j} \mid \boldsymbol{\eta}_{-(a,j)}, \boldsymbol{\theta}, \boldsymbol{c}, \boldsymbol{\kappa}, \boldsymbol{\beta}, \Lambda^{(\eta)}) \propto \exp\left(-\frac{1}{2}\boldsymbol{\eta}_{a,j}\Lambda^{(\eta)}\boldsymbol{\eta}_{a,j}^{\top} - \sum_{\substack{t \in T \\ d \in D_t}} \frac{\left(\theta_{d,j} - \sum_{a' \in \boldsymbol{a}_d} \kappa_{a'}^{(t)}\beta_j^{(t)} + c_{d,a'}\eta_{a',j}^{(t)}\right)^2}{2\|\boldsymbol{c}_d\|_2^2} \right)$$

$$p(\boldsymbol{\kappa}_a \mid \boldsymbol{\kappa}_{-(a)}, \boldsymbol{\theta}, \boldsymbol{c}, \boldsymbol{\eta}, \boldsymbol{\beta}, \Lambda^{(\kappa)}) \propto \exp\left(-\frac{1}{2}\log(\boldsymbol{\kappa}_a)\Lambda^{(\kappa)}\log(\boldsymbol{\kappa}_a^{\top}) - \sum_{\substack{t \in T \\ d \in D_t}} \frac{\|\boldsymbol{\theta}_d - \sum_{a' \in \boldsymbol{a}_d} \kappa_{a'}^{(t)}\boldsymbol{\beta}^{(t)} + c_{d,a'}\boldsymbol{\eta}_{a'}^{(t)}\|_2^2}{2\|\boldsymbol{c}_d\|_2^2} \right)$$

$\Lambda^{(\eta)}$ and $\Lambda^{(\kappa)}$ are shorthands for the precision matrices $\Lambda(\lambda^{(\eta)}, \alpha^{(\eta)})$ and $\Lambda(\lambda^{(\kappa)}, \alpha^{(\kappa)})$ respectively. Likewise, $\boldsymbol{\theta}_d$ is sampled blockwise for each document with a multivariate Gaussian distribution and likelihood:

$$p(\boldsymbol{\theta}_d \mid \boldsymbol{c}_d, \boldsymbol{\eta}, \boldsymbol{\kappa}, \boldsymbol{\beta}) \propto \exp\left(-\frac{(y_d - \boldsymbol{\beta}^{(t_d)\top}\boldsymbol{\theta}_d)^2}{2} - \frac{\|\boldsymbol{\theta}_d - \sum_{a \in \boldsymbol{a}_d} \kappa_a^{(t_d)}\boldsymbol{\beta}^{(t_d)} + c_{d,a}\boldsymbol{\eta}_a^{(t_d)}\|_2^2}{2\|\boldsymbol{c}_d\|_2^2} \right)$$

For $\boldsymbol{c}_d$, we first sampled each $\boldsymbol{c}_d$ from a multivariate Gaussian distribution, and applied a logistic transformation to map it onto the simplex. The likelihood for $\boldsymbol{c}_d$ is:

$$p(\boldsymbol{c}_d \mid \boldsymbol{\theta}_d, \boldsymbol{\eta}, \boldsymbol{\kappa}, \boldsymbol{\beta}) \propto \exp\left(-\frac{1}{2\sigma_c^2}\left\|\log\left(\frac{\boldsymbol{c}_d}{c_{d,|\boldsymbol{a}_d|}}\right)\right\|_2^2 - \frac{\|\boldsymbol{\theta}_d - \sum_{a \in \boldsymbol{a}_d} \kappa_a^{(t_d)}\boldsymbol{\beta}^{(t_d)} + c_{d,a}\boldsymbol{\eta}_a^{(t_d)}\|_2^2}{2\|\boldsymbol{c}_d\|_2^2} \right)$$

References

John Aitchison. 1986. *The Statistical Analysis of Compositional Data*. Chapman & Hall.

Katharine A. Anderson. 2012. Specialists and generalists: Equilibrium skill acquisition decisions in problem-solving populations. *Journal of Economic Behavior & Organization*, 84(1):463–473.

David M. Blei and John D. Lafferty. 2007. A correlated topic model of science. *The Annals of Applied Statistics*, pages 17–35.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, March.

Janez Brank and Jure Leskovec. 2003. The download estimation task on KDD Cup 2003. *SIGKDD Explorations Newsletter*, 5(2):160–162, December.

David A. Cohn and Thomas Hofmann. 2000. The missing link – a probabilistic model of document content and hypertext connectivity. In *NIPS*.

Elena Erosheva, Stephen Fienberg, and John Lafferty. 2004. Mixed-membership models of scientific publications. *Proceedings of the National Academy of Sciences*, 101(suppl. 1):5220–5227.

Sean Gerrish and David M. Blei. 2010. A language-based approach to measuring scholarly impact. In *Proc. of ICML*.

David Hall, Daniel Jurafsky, and Christopher D. Manning. 2008. Studying the history of ideas using topic models. In *Proc. of EMNLP*.

Thomas Hofmann. 1999. Probabilistic latent semantic indexing. In *Proc. of SIGIR*.

D. I. Holmes and R. S. Forsyth. 1995. The federalist revisited: New directions in authorship attribution. *Literary and Linguistic Computing*, 10(2):111–127.

Eduard H. Hovy. 1990. Pragmatics and natural language generation. *Artificial Intelligence*, 43(2):153–197, May.

Nikhil Johri, Daniel Ramage, Daniel A. McFarland, and Daniel Jurafsky. 2011. A study of academic collaboration in computational linguistics with latent mixtures of authors. In *Proc. of the Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities*.

John S. Justeson and Slava M. Katz. 1995. Technical terminology: Some linguistic properties and an algorithm for identification in text. *Natural Language Engineering*, 1:9–27, March.

Saurabh Kataria, Prasenjit Mitra, and Sumit Bhatia. 2010. Utilizing context in generative Bayesian models for linked corpus. In *Proc. of AAAI*.

Dong C. Liu and Jorge Nocedal. 1989. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45(1-3):503–528.

Yan Liu, Alexandru Niculescu-Mizil, and Wojciech Gryc. 2009. Topic-link LDA: Joint models of topic and author community. In *Proc. of ICML*.

Jon D. McAuliffe and David M. Blei. 2008. Supervised topic models. In *NIPS*.

Daniel McFadden. 1974. Conditional logit analysis of qualitative choice behavior. In *Frontiers in Econometrics*, pages 105–142. Academic Press.

Amy McGovern, Lisa Friedland, Michael Hay, Brian Gallagher, Andrew Fast, Jennifer Neville, and David Jensen. 2003. Exploiting relational structure to understand publication patterns in high-energy physics. *SIGKDD Exploration Newsletter*, 5(2):165–172, December.

Ramesh Nallapati and William W. Cohen. 2008. Link-PLSA-LDA: A new unsupervised model for topics and influence of blogs. In *Proc. of ICWSM*.

Dragomir R. Radev, Pradeep Muthukrishnan, Vahed Qazvinian, and Amjad Abu-Jbara. 2013. The ACL anthology network corpus. *Language Resources and Evaluation*, pages 1–26. Data available at `http://clair.eecs.umich.edu/aan/`.

Michal Rosen-Zvi, Thomas Griffiths, Mark Steyvers, and Padhraic Smyth. 2004. The author-topic model for authors and documents. In *Proc. of UAI*.

Yanchuan Sim, Bryan Routledge, and Noah A. Smith. 2015. The utility of text: The case of amicus briefs and the Supreme Court. In *Proc. of AAAI*.

Efstathios Stamatatos. 2009. A survey of modern authorship attribution methods. *Journal of the American Society for Information Science and Technology*, 60(3):538–556.

Chris Tanner and Eugene Charniak. 2015. A hybrid generative/discriminative approach to citation prediction. In *Proc. of NAACL*.

Luke Tierney. 1994. Markov chains for exploring posterior distributions. *The Annals of Statistics*, 22(4):pp. 1701–1728.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proc. of NAACL*.

Adam Vogel, Max Bodoia, Christopher Potts, and Daniel Jurafsky. 2013. Emergence of Gricean maxims from multi-agent decision theory. In *Proc. of NAACL*.

Greg C. G. Wei and Martin A. Tanner. 1990. A Monte Carlo implementation of the EM algorithm and the poor man's data augmentation algorithms. *Journal of the American Statistical Association*, 85(411):pp. 699–704.

Dani Yogatama, Michael Heilman, Brendan O'Connor, Chris Dyer, Bryan R. Routledge, and Noah A. Smith. 2011. Predicting a scientific community's response to an article. In *Proc. of EMNLP*.

Yaojia Zhu, Xiaoran Yan, Lise Getoor, and Cristopher Moore. 2013. Scalable text and link analysis with mixed-topic link models. In *Proc. of KDD*.

Finding Function in Form: Compositional Character Models for Open Vocabulary Word Representation

Wang Ling Tiago Luís Luís Marujo Ramón Fernandez Astudillo
Silvio Amir Chris Dyer Alan W Black Isabel Trancoso

L^{2}F Spoken Systems Lab, INESC-ID, Lisbon, Portugal
Language Technologies Institute, Carnegie Mellon University, Pittsburgh, PA, USA
Instituto Superior Técnico, Lisbon, Portugal
{lingwang,lmarujo,cdyer,awb}@cs.cmu.edu
{ramon.astudillo,samir,tmcl,isabel.trancoso}@inesc-id.pt

Abstract

We introduce a model for constructing vector representations of words by composing characters using bidirectional LSTMs. Relative to traditional word representation models that have independent vectors for each word type, our model requires only a single vector per character type and a fixed set of parameters for the compositional model. Despite the compactness of this model and, more importantly, the arbitrary nature of the form–function relationship in language, our "composed" word representations yield state-of-the-art results in language modeling and part-of-speech tagging. Benefits over traditional baselines are particularly pronounced in morphologically rich languages (e.g., Turkish).

1 Introduction

Good representations of words are important for good generalization in natural language processing applications. Of central importance are vector space models that capture functional (i.e., semantic and syntactic) similarity in terms of geometric locality. However, when word vectors are learned—a practice that is becoming increasingly common—most models assume that each word type has its own vector representation that can vary independently of other model components. This paper argues that this independence assumption is inherently problematic, in particular in morphologically rich languages (e.g., Turkish). In such languages, a more reasonable assumption would be that orthographic (formal) similarity is evidence for functional similarity.

However, it is manifestly clear that similarity in form is neither a necessary nor sufficient condition for similarity in function: small orthographic differences may correspond to large semantic or syntactic differences (*butter* vs. *batter*), and large orthographic differences may obscure nearly perfect functional correspondence (*rich* vs. *affluent*). Thus, any orthographically aware model must be able to capture *non-compositional* effects in addition to more regular effects due to, e.g., morphological processes. To model the complex form–function relationship, we turn to long short-term memories (LSTMs), which are designed to be able to capture complex non-linear and non-local dynamics in sequences (Hochreiter and Schmidhuber, 1997). We use bidirectional LSTMs to "read" the character sequences that constitute each word and combine them into a vector representation of the word. This model assumes that each character type is associated with a vector, and the LSTM parameters encode both idiosyncratic lexical and regular morphological knowledge.

To evaluate our model, we use a vector-based model for part-of-speech (POS) tagging and for language modeling, and we report experiments on these tasks in several languages comparing to baselines that use more traditional, orthographically-unaware parameterizations. These experiments show: (i) our character-based model is able to generate similar representations for words that are semantically and syntactically similar, even for words are orthographically distant (e.g., *October* and *January*); our model achieves improvements over word lookup tables using only a fraction of the number of parameters in two tasks; (iii) our model obtains state-of-the-art performance on POS tagging (including establishing a new best performance in English); and

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1520–1530,
Lisbon, Portugal, 17-21 September 2015.

(iv) performance improvements are especially dramatic in morphologically rich languages.

The paper is organized as follows: Section 2 presents our character-based model to generate word embeddings. Experiments on Language Modeling and POS tagging are described in Sections 4 and 5. We present related work in Section 6; and we conclude in Section 7.

2 Word Vectors and Wordless Word Vectors

It is commonplace to represent words as vectors. In contrast to naïve models in which all word types in a vocabulary V are equally different from each other, vector space models capture the intuition that words may be different or similar along a variety of dimensions. Learning vector representations of words by treating them as optimizable parameters in various kinds of language models has been found to be a remarkably effective means for generating vector representations that perform well in other tasks (Collobert et al., 2011; Kalchbrenner and Blunsom, 2013; Liu et al., 2014; Chen and Manning, 2014). Formally, such models define a matrix $\mathbf{P} \in \mathbb{R}^{d \times |V|}$, which contains d parameters for each word in the vocabulary V. For a given word type $w \in V$, a column is selected by right-multiplying $\mathbf{P}$ by a one-hot vector of length $|V|$, which we write $\mathbf{1}_w$, that is zero in every dimension except for the element corresponding to w. Thus, $\mathbf{P}$ is often referred to as word lookup table and we shall denote by $\mathbf{e}_w^W \in \mathbb{R}^d$ the embedding obtained from a word lookup table for w as $\mathbf{e}_w^W = \mathbf{P} \cdot \mathbf{1}_w$. This allows tasks with low amounts of annotated data to be trained jointly with other tasks with large amounts of data and leverage the similarities in these tasks. A common practice to this end is to initialize the word lookup table with the parameters trained on an unsupervised task. Some examples of these include the skip-n-gram and CBOW models of Mikolov et al. (2013).

2.1 Problem: Independent Parameters

There are two practical problems with word lookup tables. Firstly, while they can be pretrained with large amounts of data to learn semantic and syntactic similarities between words, each vector is independent. That is, even though models based on word lookup tables are often observed to learn that *cats*, *kings* and *queens* exist in roughly the same linear correspondences to each other as *cat*, *king* and *queen* do, the model does not represent the fact that adding an *s* at the end of the word is evidence for this transformation. This means that word lookup tables cannot generate representations for previously unseen words, such as *Frenchification*, even if the components, *French* and *-ification*, are observed in other contexts.

Second, even if copious data is available, it is impractical to actually store vectors for all word types. As each word type gets a set of parameters d, the total number of parameters is $d \times |V|$, where $|V|$ is the size of the vocabulary. Even in relatively morphological poor English, the number of word types tends to scale to the order of hundreds of thousands, and in noisier domains, such as online data, the number of word types raises considerably. For instance, in the English wikipedia dump with 60 million sentences, there are approximately 20 million different lowercased and tokenized word types, each of which would need its own vector. Intuitively, it is not sensible to use the same number of parameters for each word type.

Finally, it is important to remark that it is uncontroversial among cognitive scientists that our lexicon is structured into related forms—i.e., their parameters are not independent. The well-known "past tense debate" between connectionists and proponents of symbolic accounts concerns disagreements about how humans represent knowledge of inflectional processes (e.g., the formation of the English past tense), not whether such knowledge exists (Marslen-Wilson and Tyler, 1998).

2.2 Solution: Compositional Models

Our solution to these problems is to construct a vector representation of a word by composing smaller pieces into a representation of the larger form. This idea has been explored in prior work by composing *morphemes* into representations of words (Luong et al., 2013; Botha and Blunsom, 2014; Soricut and Och, 2015). Morphemes are an ideal primitive for such a model since they are—by definition—the minimal meaning-bearing (or syntax-bearing) units of language. The drawback to such approaches is they depend on a morphological analyzer.

In contrast, we would like to compose representations of *characters* into representations of words. However, the relationship between words

forms and their meanings is non-trivial (de Saussure, 1916). While some compositional relationships exist, e.g., morphological processes such as adding *-ing* or *-ly* to a stem have relatively regular effects, many words with lexical similarities convey different meanings, such as, the word pairs *lesson* $\Longleftrightarrow$ *lessen* and *coarse* $\Longleftrightarrow$ *course*.

3 C2W Model

Our compositional character to word (C2W) model is based on bidirectional LSTMs (Graves and Schmidhuber, 2005), which are able to learn complex non-local dependencies in sequence models. An illustration is shown in Figure 1. The input of the C2W model (illustrated on bottom) is a single word type w, and we wish to obtain is a d-dimensional vector used to represent w. This model shares the same input and output of a word lookup table (illustrated on top), allowing it to easily replace then in any network.

As input, we define an alphabet of characters C. For English, this vocabulary would contain an entry for each uppercase and lowercase letter as well as numbers and punctuation. The input word w is decomposed into a sequence of characters $c_1, \ldots, c_m$, where m is the length of w. Each c_i is defined as a one hot vector $\mathbf{1}_{c_i}$, with one on the index of c_i in vocabulary M. We define a projection layer $\mathbf{P}_C \in \mathbb{R}^{d_C \times |C|}$, where d_C is the number of parameters for each character in the character set C. This of course just a character lookup table, and is used to capture similarities between characters in a language (e.g., vowels *vs.* consonants). Thus, we write the projection of each input character c_i as $\mathbf{e}_{c_i} = \mathbf{P}_C \cdot \mathbf{1}_{c_i}$.

Given the input vectors $\mathbf{x}_1, \ldots, \mathbf{x}_m$, a LSTM computes the state sequence $\mathbf{h}_1, \ldots, \mathbf{h}_{m+1}$ by iteratively applying the following updates:

$$
\begin{aligned}
\mathbf{i}_t &= \sigma(\mathbf{W}_{ix}\mathbf{x}_t + \mathbf{W}_{ih}\mathbf{h}_{t-1} + \mathbf{W}_{ic}\mathbf{c}_{t-1} + \mathbf{b}_i) \\
\mathbf{f}_t &= \sigma(\mathbf{W}_{fx}\mathbf{x}_t + \mathbf{W}_{fh}\mathbf{h}_{t-1} + \mathbf{W}_{fc}\mathbf{c}_{t-1} + \mathbf{b}_f) \\
\mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \\
&\qquad \mathbf{i}_t \odot \tanh(\mathbf{W}_{cx}\mathbf{x}_t + \mathbf{W}_{ch}\mathbf{h}_{t-1} + \mathbf{b}_c) \\
\mathbf{o}_t &= \sigma(\mathbf{W}_{ox}\mathbf{x}_t + \mathbf{W}_{oh}\mathbf{h}_{t-1} + \mathbf{W}_{oc}\mathbf{c}_t + \mathbf{b}_o) \\
\mathbf{h}_t &= \mathbf{o}_t \odot \tanh(\mathbf{c}_t),
\end{aligned}
$$

where σ is the component-wise logistic sigmoid function, and $\odot$ is the component-wise (Hadamard) product. LSTMs define an extra cell memory $\mathbf{c}_t$, which is combined linearly at each timestamp t. The information that is propagated from $\mathbf{c}_{t-1}$ to $\mathbf{c}_t$ is controlled by the three gates $\mathbf{i}_t$, $\mathbf{f}_t$, and $\mathbf{o}_t$, which determine the what to include from the input $\mathbf{x}_t$, the what to forget from $\mathbf{c}_{t-1}$ and what is relevant to the current state $\mathbf{h}_t$. We write $\mathcal{W}$ to refer to all parameters the LSTM ($\mathbf{W}_{ix}$, $\mathbf{W}_{fx}$, $\mathbf{b}_f$, ...). Thus, given a sequence of character representations $\mathbf{e}^C_{c_1}, \ldots, \mathbf{e}^C_{c_m}$ as input, the forward LSTM, yields the state sequence $\mathbf{s}^f_0, \ldots, \mathbf{s}^f_m$, while the backward LSTM receives as input the reverse sequence, and yields states $\mathbf{s}^b_m, \ldots, \mathbf{s}^b_0$. Both LSTMs use a different set of parameters $\mathcal{W}^f$ and $\mathcal{W}^b$. The representation of the word w is obtained by combining the forward and backward states:

$$\mathbf{e}^C_w = \mathbf{D}^f\mathbf{s}^f_m + \mathbf{D}^b\mathbf{s}^b_0 + \mathbf{b}_d,$$

where $\mathbf{D}^f$, $\mathbf{D}^b$ and $\mathbf{b}_d$ are parameters that deter-

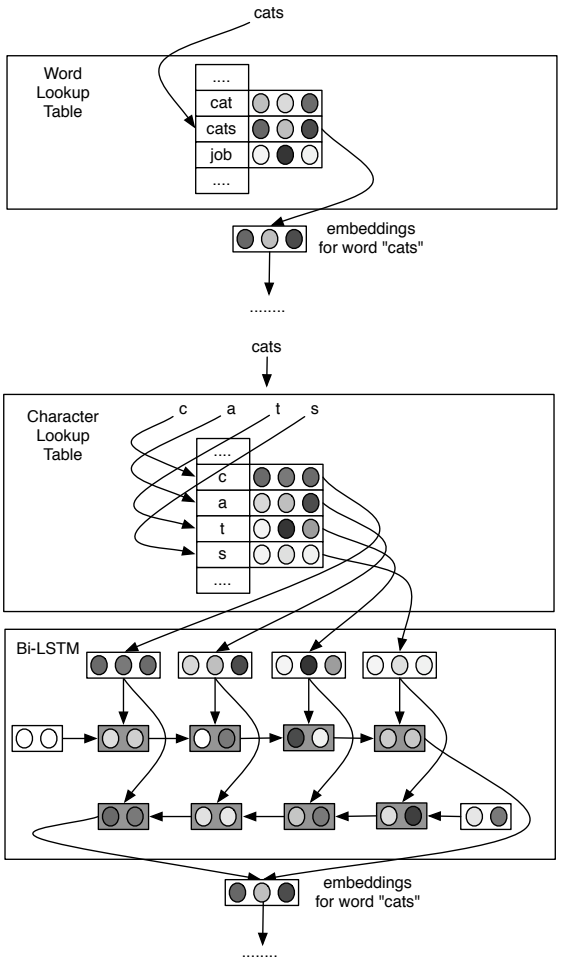


Figure 1: Illustration of the word lookup tables (top) and the lexical Composition Model (bottom). Square boxes represent vectors of neuron activations. Shaded boxes indicate that a non-linearity.

mine how the states are combined.

Caching for Efficiency. Relative to $\mathbf{e}_w^W$, computing $\mathbf{e}_w^C$ is computational expensive, as it requires two LSTMs traversals of length m. However, $\mathbf{e}_w^C$ only depends on the character sequence of that word, which means that unless the parameters are updated, it is possible to cache the value of $\mathbf{e}_w^C$ for each different w's that will be used repeatedly. Thus, the model can keep a list of the most frequently occurring word types in memory and run the compositional model only for rare words. Obviously, caching all words would yield the same performance as using a word lookup table $\mathbf{e}_w^W$, but also using the same amount of memory. Consequently, the number of word types used in cache can be adjusted to satisfy memory vs. performance requirements of a particular application.

At training time, when parameters are changing, repeated words within the same batch only need to be computed once, and the gradient at the output can be accumulated within the batch so that only one update needs to be done per word type. For this reason, it is preferable to define larger batches.

4 Experiments: Language Modeling

Our proposed model is similar to models used to compute composed representations of sentences from words (Cho et al., 2014; Li et al., 2015). However, the relationship between the meanings of individual words and the composite meaning of a phrase or sentence is arguably more regular than the relationship of representations of characters and the meaning of a word. Is our model capable of learning such an irregular relationship? We now explore this question empirically.

Language modeling is a task with many applications in NLP. An effective LM requires syntactic aspects of language to be modeled, such as word orderings (e.g., "John is smart" *vs.* "John smart is"), but also semantic aspects (e.g., "John ate fish" *vs.* "fish ate John"). Thus, if our C2W model only captures regular aspects of words, such as, prefixes and suffixes, the model will yield worse results compared to word lookup tables.

4.1 Language Model

Language modeling amounts to learning a function that computes the log probability, $\log p(\boldsymbol{w})$, of a sentence $\boldsymbol{w} = (w_1, \ldots, w_n)$. This quantity can be decomposed according to the chain rule into the sum of the conditional log probabilities $\sum_{i=1}^{n} \log p(w_i \mid w_1, \ldots, w_{i-1})$. Our language model computes $\log p(w_i \mid w_1, \ldots, w_{i-1})$ by composing representations of words $w_1, \ldots, w_{i-1}$ using an recurrent LSTM model (Mikolov et al., 2010; Sundermeyer et al., 2012).

The model is illustrated in Figure 2, where we observe on the first level that each word w_i is projected into their word representations. This can be done by using word lookup tables $\mathbf{e}_{w_i}^W$, in which case, we will have a regular recurrent language model. To use our C2W model, we can simply replace the word lookup table with the model $f(w_i) = \mathbf{e}_{w_i}^C$. Each LSTM block $\mathbf{s}_i$, is used to predict word w_{i+1}. This is performed by projecting the s_i into a vector of size of the vocabulary V and performing a softmax.

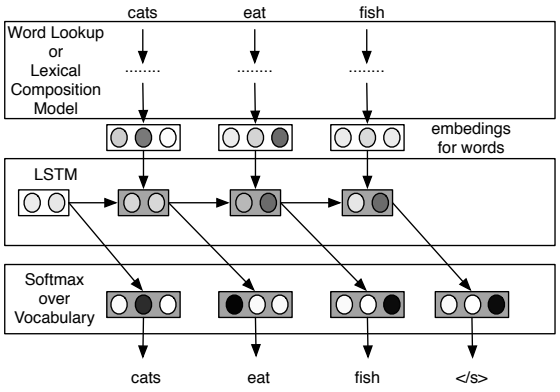


Figure 2: Illustration of our neural network for Language Modeling.

The softmax is still simply a $d \times V$ table, which encodes the likelihood of every word type in a given context, which is a closed-vocabulary model. Thus, at test time out-of-vocabulary (OOV) words cannot be addressed. A strategy that is generally applied is to prune the vocabulary V by replacing word types with lower frequencies as an OOV token. At test time, the probability of words not in vocabulary is estimated as the OOV token. Thus, depending on the number of word types that are pruned, the global perplexities may decrease, since there are fewer outcomes in the softmax, which makes the absolute value of perplexity not informative when comparing models of different vocabulary sizes. Yet, the relative perplexity between different models indicates which models can better predict words based on their contexts.

To address OOV words in the baseline setup, these are replaced by an unknown token, and also associated with a set of embeddings. During training, word types that occur once are replaced with the unknown token stochastically with 0.5 probability. The same process is applied at the character level for the C2W model.

4.2 Experiments

Datasets We look at the language model performance on English, Portuguese, Catalan, German and Turkish, which have a broad range of morphological typologies. While all these languages contain inflections, in agglutinative languages affixes tend to be unchanged, while in fusional languages they are not. For each language, Wikipedia articles were randomly extracted until 1 million words are obtained and these were used for training. For development and testing, we extracted an additional set of 20,000 words.

Setup We define the size of the word representation d to 50. In the C2W model requires setting the dimensionality of characters d_C and current states d_{CS}. We set $d_C = 50$ and $d_{CS} = 150$. Each LSTM state used in the language model sequence s_i is set to 150 for both states and cell memories. Training is performed with mini-batch gradient descent with 100 sentences. The learning rate and momentum were set to 0.2 and 0.95. The softmax over words is always performed on lowercased words. We restrict the output vocabulary to the most frequent 5000 words. Remaining word types will be replaced by an unknown token, which must also be predicted. The word representation layer is still performed over all word types (i.e., completely open vocabulary). When using word lookup tables, the input words are also lowercased, as this setup produces the best results. In the C2W, case information is preserved.

Evaluation is performed by computing the perplexities over the test data, and the parameters that yield the highest perplexity over the development data are used.

Perplexities Perplexities over the testset are reported on Table 4. From these results, we can see that in general, it is clear that C2W always outperforms word lookup tables (row "Word"), and that improvements are especially pronounced in Turkish, which is a highly morphological language, where word meanings differ radically depending on the suffixes used (*evde → in the house vs. evden → from the house*).

	Fusional			Agglutinative	
Perplexity	EN	PT	CA	DE	TR
5-gram KN	70.72	58.73	39.83	59.07	52.87
Word	59.38	46.17	35.34	43.02	44.01
C2W	**57.39**	**40.92**	**34.92**	**41.94**	**32.88**
#Parameters					
Word	4.3M	4.2M	4.3M	6.3M	5.7M
C2W	**180K**	**178K**	**182K**	**183K**	**174K**

Table 1: Language Modeling Results

Number of Parameters As for the number of parameters (illustrated for block "#Parameters"), the number of parameters in word lookup tables is $V \times d$. If a language contains 80,000 word types (a conservative estimate in morphologically rich languages), 4 million parameters would be necessary. On the other hand, the compositional model consists of 8 matrices of dimensions $d_{CS} \times d_C + 2d_{CS}$. Additionally, there is also the matrix that combines the forward and backward states of size $d \times 2d_{CS}$. Thus, the number of parameters is roughly 150,000 parameters—substantially fewer. This model also needs a character lookup table with d_C parameters for each entry. For English, there are 618 characters, for an additional 30,900 parameters. So the total number of parameters for English is roughly 180,000 parameters (2 to 3 parameters per word type), which is an order of magnitude lower than word lookup tables.

Performance As for efficiency, both representations can label sentences at a rate of approximately 300 words per second during training. While this is surprising, due to the fact that the C2W model requires a composition over characters, the main bottleneck of the system is the softmax over the vocabulary. Furthermore, caching is used to avoid composing the same word type twice in the same batch. This shows that the C2W model, is relatively fast compared operations such as a softmax.

Representations of (nonce) words While is is promising that the model is not simply learning lexical features, what is most interesting is that the model can propose embeddings for nonce words, in stark contrast to the situation observed with lookup table models. We show the 5-most-similar in-vocabulary words (measured with cosine similarity) as computed by our character model on two

increased	*John*	*Noahshire*	*phding*
reduced	Richard	Nottinghamshire	mixing
improved	George	Bucharest	modelling
expected	James	Saxony	styling
decreased	Robert	Johannesburg	blaming
targeted	Edward	Gloucestershire	christening

Table 2: Most-similar in-vocabular words under the C2W model; the two query words on the left are in the training vocabulary, those on the right are nonce (invented) words.

in-vocabulary words and two nonce words[1].This makes our model generalize significantly better than lookup tables that generally use unknown tokens for OOV words. Furthermore, this ability to generalize is much more similar to that of human beings, who are able to infer meanings for new words based on its form.

5 Experiments: Part-of-speech Tagging

As a second illustration of the utility of our model, we turn to POS tagging. As morphology is a strong indicator for syntax in many languages, a much effort has been spent engineering features (Nakagawa et al., 2001; Mueller et al., 2013). We now show that some of these features can be learnt automatically using our model.

5.1 Bi-LSTM Tagging Model

Our tagging model is likewise novel, but very straightforward. It builds a Bi-LSTM over words as illustrated in Figure 3. The input of the model is a sequence of features $f(w_1), \ldots, f(w_n)$. Once again, word vectors can either be generated using the C2W model $f(w_i) = \mathbf{e}^C_{w_i}$, or word lookup tables $f(w_i) = \mathbf{e}^W_{w_i}$. We also test the usage of hand-engineered features, in which case $f_1(w_i), \ldots, f_n(w_i)$. Then, the sequential features $f(w_1), \ldots, f(w_n)$ are fed into a bidirectional LSTM model, obtaining the forward states $\mathbf{s}^f_0, \ldots, \mathbf{s}^f_n$ and the backward states $\mathbf{s}^b_{N+1}, \ldots, \mathbf{s}^b_0$. Thus, state $\mathbf{s}^f_i$ contains the information of all words from 0 to i and $\mathbf{s}^b_i$ from n to i. The forward and backward states are combined, for each index from 1 to n, as follows:

$$\mathbf{l}_i = \tanh(\mathbf{L}^f \mathbf{s}^f_i + \mathbf{L}^b \mathbf{s}^b_i + \mathbf{b}_l),$$

where $\mathbf{L}^f$, $\mathbf{L}^b$ and $\mathbf{b}_l$ are parameters defining how the forward and backward states are combined.

[1] software submitted as supplementary material

The size of the forward $\mathbf{s}^f$ and backward states $\mathbf{s}^b$ and the combined state $\mathbf{l}$ are hyperparameters of the model, denoted as d^f_{WS}, d^b_{WS} and d_{WS}, respectively. Finally, the output labels for index i are obtained as a softmax over the POS tagset, by projecting the combined state $\mathbf{l}_i$.

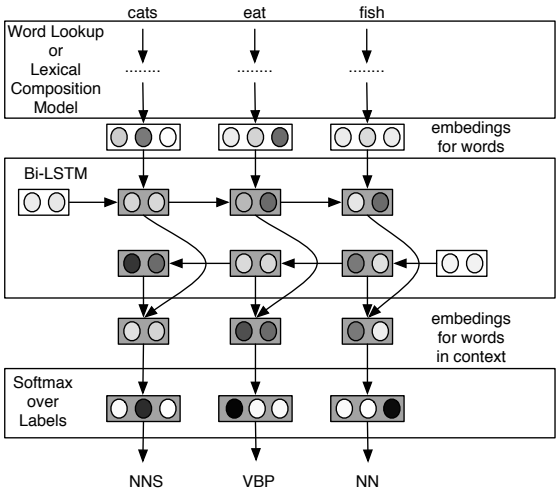


Figure 3: Illustration of our neural network for POS tagging.

5.2 Experiments

Datasets For English, we conduct experiments on the Wall Street Journal of the Penn Treebank dataset (Marcus et al., 1993), using the standard splits (sections 1–18 for train, 19–21 for tuning and 22–24 for testing). We also perform tests on 4 other languages, which we obtained from the CoNLL shared tasks (Martí et al., 2007; Brants et al., 2002; Afonso et al., 2002; Atalay et al., 2003). While the PTB dataset provides standard train, tuning and test splits, there are no tuning sets in the datasets in other languages, so we withdraw the last 100 sentences from the training dataset and use them for tuning.

Setup The POS model requires two sets of hyperparameters. Firstly, words must be converted into continuous representations and the same hyperparametrization as in language modeling (Section 4) is used. Additionally, we also compare to the convolutional model of Santos and Zadrozny (2014), which also requires the dimensionality for characters and the word representation size, which are set to 50 and 150, respectively. Secondly, words representations are combined to en-

code context. Our POS tagger has three hyperparameters d^f_{WS}, d^b_{WS} and d_{WS}, which correspond to the sizes of LSTM states, and are all set to 50. As for the learning algorithm, use the same setup (learning rate, momentum and mini-batch sizes) as used in language modeling.

Once again, we replace OOV words with an unknown token, in the setup that uses word lookup tables, and the same with OOV characters in the C2W model. In setups using pre-trained word embeddings, we consider a word an OOV if it was not seen in the labelled training data as well as in the unlabeled data used for pre-training.

Compositional Model Comparison A comparison of different recurrent neural networks for the C2W model is presented in Table 3. We used our proposed tagger tagger in all experiments and results are reported for the English Penn Treebank. Results on label accuracy test set is shown in the column "acc". The number of parameters in the word composition model is shown in the column "parameters". Finally, the number of words processed at test time per second are shown in column "words/sec".

We observe that approaches using RNN yield worse results than their LSTM counterparts with a difference of approximately 2%. This suggests that while regular RNNs can learn shorter character sequence dependencies, they are not ideal to learn longer dependencies. LSTMs, on the other hand, seem to effectively obtain relatively higher results, on par with using word look up tables (row "Word Lookup"), even when using forward (row "Forward LSTM") and backward (row "Backward LSTM") LSTMs individually. The best results are obtained using the bidirectional LSTM (row "Bi-LSTM"), which achieves an accuracy of 97.29% on the test set, surpassing the word lookup table. The convolution model (Santos and Zadrozny, 2014) obtained slightly lower results (row "Convolutional (S&Z)"), we think this is because the convolutional model uses a max-pooling layer over series of window convolutions. As order is only perserved within windows, longer distance dependences are unobserved.

There are approximately 40k lowercased word types in the training data in the PTB dataset. Thus, a word lookup table with 50 dimensions per type contains approximately 2 million parameters. In the C2W models, the number of characters types (including uppercase and lowercase) is approximately 80. Thus, the character look up table consists of only 4k parameters, which is negligible compared to the number of parameters in the compositional model, which is once again 150k parameters. One could argue that results in the Bi-LSTM model are higher than those achieved by other models as it contains more parameters, so we set the state size $d_{CS} = 50$ (row "Bi-LSTM $d_{CS} = 50$") and obtained similar results.

	acc	parameters	words/sec
Word Lookup	96.97	2000k	6K
Convolutional (S&Z)	96.80	42.5k	4K
Forward RNN	95.66	17.5k	4K
Backward RNN	95.52	17.5k	4K
Bi-RNN	95.93	40k	3K
Forward LSTM	97.12	80k	3K
Backward LSTM	97.08	80k	3K
Bi-LSTM $d_{CS} = 50$	97.22	70k	3K
Bi-LSTM	**97.36**	150k	2K

Table 3: POS accuracy results for the English PTB using word representation models.

In terms of computational speed, we can observe that there is a more significant slowdown when applying the C2W models compared to language modeling. This is because there is no longer a softmax over the whole word vocabulary as the main bottleneck of the network. However, we can observe that while the Bi-LSTM system is 3 times slower, it is does not significantly hurt the performance of the system.

Results on Multiple Languages Results on 5 languages are shown in Table 4. In general, we can observe that the model using word lookup tables (row "Word") performs consistently worse than the C2W model (row "C2W"). We also compare our results with Stanford's POS tagger, with the default set of features, found in Table 4. Results using these tagger are comparable or better than state-of-the-art systems. We can observe that in most cases we can slightly outperform the scores obtained using their tagger. This is a promising result, considering that we use the same training data and do not handcraft any features. Furthermore, we can observe that for Turkish, our results are significantly higher (>4%).

Comparison with Benchmarks Most state-of-the-art POS tagging systems are obtained by either learning or handcrafting good lexical features (Manning, 2011; Sun, 2014) or using ad-

System	Fusional			Agglutinative	
	EN	PT	CA	DE	TR
Word	96.97	95.67	98.09	97.51	83.43
C2W	**97.36**	97.47	**98.92**	**98.08**	**91.59**
Stanford	97.32	**97.54**	98.76	97.92	87.31

Table 4: POS accuracies on different languages

ditional raw data to learn features in an unsupervised fashion. Generally, optimal results are obtained by performing both. Table 5 shows the current Benchmarks in this task for the English PTB. Accuracies on the test set is reported on column "acc". Columns "+feat" and "+data" define whether hand-crafted features are used and whether additional data was used. We can see that even without feature engineering or unsupervised pretraining, our C2W model (row "C2W") is on par with the current state-of-the-art system (row "structReg"). However, if we add hand-crafted features, we can obtain further improvements on this dataset (row "C2W + features").

However, there are many words that do not contain morphological cues to their part-of-speech. For instance, the word *snake* does not contain any morphological cues that determine its tag. In these cases, if they are not found labelled in the training data, the model would be dependent on context to determine their tags, which could lead to errors in ambiguous contexts. Unsupervised training methods such as the Skip-n-gram model (Mikolov et al., 2013) can be used to pretrain the word representations on unannotated corpora. If such pretraining places *cat*, *dog* and *snake* near each other in vector space, and the supervised POS data contains evidence that *cat* and *dog* are nouns, our model will be likely to label *snake* with the same tag.

We train embeddings using English wikipedia with the dataset used in (Ling et al., 2015), and the Structured Skip-n-gram model. Results using pre-trained word lookup tables and the C2W with the pre-trained word lookup tables as additional parameters are shown in rows "word(sskip)" and "C2W + word(sskip)". We can observe that both systems can obtain improvements over their random initializations (rows "word" and (C2W)).

Finally, we also found that when using the C2W model in conjunction pre-trained word embeddings, that adding a non-linearity to the representations extracted from the C2W model $\mathbf{e}_w^C$ improves the results over using a simple linear transformation (row "C2W(tanh)+word (sskip)"). This setup, obtains 0.28 points over the current state-of-the-art system(row "SCCN").

	+feat	+data	acc
word	no	no	96.70
C2W	no	no	**97.36**
word+features	yes	no	97.34
C2W+features	yes	no	**97.57**
Stanford 2.0 (Manning, 2011)	yes	no	97.32
structReg (Sun, 2014)	yes	no	97.36
word (sskip)	no	yes	97.42
C2W+word (sskip)	no	yes	97.54
C2W(tanh)+word (sskip)	no	yes	**97.78**
Morče (Spoustová et al., 2009)	yes	yes	97.44
SCCN (Søgaard, 2011)	yes	yes	97.50

Table 5: POS accuracy result comparison with state-of-the-art systems for the English PTB.

5.3 Discussion

It is important to refer here that these results do not imply that our model always outperforms existing benchmarks, in fact in most experiments, results are typically fairly similar to existing systems. Even in Turkish, using morphological analysers in order to extract additional features could also accomplish similar results. The goal of our work is not to overcome existing benchmarks, but show that much of the feature engineering done in the benchmarks can be learnt automatically from the task specific data. More importantly, we wish to show large dimensionality word look tables can be compacted into a lookup table using characters and a compositional model allowing the model scale better with the size of the training data. This is a desirable property of the model as data becomes more abundant in many NLP tasks.

6 Related Work

Our work, which learns representations without relying on word lookup tables has not been explored to our knowledge. In essence, our model attempts to learn lexical features automatically while compacting the model by reducing the redundancy found in word lookup tables. Individually, these problems have been the focus of research in many areas.

Lexical information has been used to augment word lookup tables. Word representation learning can be thought of as a process that takes a string as input representing a word and outputs a set of values that represent a word in vector

space. Using word lookup tables is one possible approach to accomplish this. Many methods have been used to augment this model to learn lexical features with an additional model that is jointly maximized with the word lookup table. This is generally accomplished by either performing a component-wise addition of the embeddings produced by word lookup tables (Chen et al., 2015), and that generated by the additional lexical model, or simply concatenating both representations (Santos and Zadrozny, 2014). Many models have been proposed, the work in (Collobert et al., 2011) refers that additional features sets F_i can be added to the one-hot representation and multiple lookup tables $\mathbf{I}_{F_i}$ can be learnt to project each of the feature sets to the same low-dimensional vector $\mathbf{e}_w^W$. For instance, the work in (Botha and Blunsom, 2014) shows that using morphological analyzers to generate morphological features, such as stems, prefixes and suffixes can be used to learn better representations for words. A problem with this approach is the fact that the model can only learn from what has been defined as feature sets. The models proposed in (Santos and Zadrozny, 2014; Chen et al., 2015) allow the model to arbitrary extract meaningful lexical features from words by defining compositional models over characters. The work in (Chen et al., 2015) defines a simple compositional model by summing over all characters in a given word, while the work in (Santos and Zadrozny, 2014) defines a convolutional network, which combines windows of characters and a max-pooling layer to find important morphological features. The main drawback of these methods is that character order is often neglected, that is, when summing over all character embeddings, words such as *dog* and *god* would have the same representation according to the lexical model. Convolutional model are less susceptible to these problems as they combine windows of characters at each convolution, where the order within the window is preserved. However, the order between extracted windows is not, so the problem still persists for longer words, such as those found in agglutinative languages. Yet, these approaches work in conjunction with a word lookup table, as they compensate for this inability. Aside from neural approaches, character-based models have been applied to address multiple lexically oriented tasks, such as transliteration (Kang and Choi, 2000) and twitter normalization (Xu et al., 2013; Ling et al., 2013).

Compacting models has been a focus of research in tasks, such as language modeling and machine translation, as extremely large models can be built with the large amounts of training data that are available in these tasks. In language modeling, it is frequent to prune higher order n-grams that do not encode any additional information (Seymore and Rosenfeld, 1996; Stolcke, 1998; Moore and Quirk, 2009). The same be applied in machine translation (Ling et al., 2012; Zens et al., 2012) by removing longer translation pairs that can be replicated using smaller ones. In essence our model learns regularities at the subword level that can be leveraged for building more compact word representations.

Finally, our work has been applied to dependency parsing and found similar improvements over word models in morphologically rich languages (Ballesteros et al., 2015).

7 Conclusion

We propose a C2W model that builds word embeddings for words without an explicit word lookup table. Thus, it benefits from being sensitive to lexical aspects within words, as it takes characters as atomic units to derive the embeddings for the word. On POS tagging, our models using characters alone can still achieve comparable or better results than state-of-the-art systems, without the need to manually engineer such lexical features. Although both language modeling and POS tagging both benefit strongly from morphological cues, the success of our models in languages with impoverished morphological cues shows that it is able to learn non-compositional aspects of how letters fit together.

The code for the C2W model and our language model and POS tagger implementations is available from `https://github.com/wlin12/JNN`.

Acknowledgements

The PhD thesis of Wang Ling is supported by FCT grant SFRH/BD/51157/2010. This research was supported in part by the U.S. Army Research Laboratory, the U.S. Army Research Office under contract/grant number W911NF-10-1-0533 and NSF IIS-1054319 and FCT through the pluri-anual contract UID/CEC/50021/2013 and grant number SFRH/BPD/68428/2010.

References

Susana Afonso, Eckhard Bick, Renato Haber, and Diana Santos. 2002. "Floresta sintá(c)tica": a treebank for Portuguese. In *Proc. LREC*.

Nart B. Atalay, Kemal Oflazer, and Bilge Say. 2003. The annotation process in the Turkish treebank. In *In Proc. the 4th International Workshop on Linguistically Interpreted Corpora (LINC)*.

Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2015. Improved transition-based parsing by modeling characters instead of words with LSTMs. In *Proc. EMNLP*.

Jan A. Botha and Phil Blunsom. 2014. Compositional morphology for word representations and language modelling. In *Proc. ICML*.

Sabine Brants, Stefanie Dipper, Silvia Hansen, Wolfgang Lezius, and George Smith. 2002. The TIGER treebank.

Danqi Chen and Christopher D. Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proc. EMNLP*.

Xinxiong Chen, Lei Xu, Zhiyuan Liu, Maosong Sun, and Huanbo Luan. 2015. Joint learning of character and word embeddings.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proc. EMNLP*.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *JMLR*.

Ferdinand de Saussure. 1916. *Course in General Linguistics*.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8).

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *Proc. EMNLP*.

Byung-Ju Kang and Key-Sun Choi. 2000. Automatic transliteration and back-transliteration by decision tree learning. In *LREC*.

Jiwei Li, Dan Jurafsky, and Eduard H. Hovy. 2015. When are tree structures necessary for deep learning of representations? *CoRR*, abs/1503.00185.

Wang Ling, João Graça, Isabel Trancoso, and Alan Black. 2012. Entropy-based pruning for phrase-based machine translation. In *Proc. EMNLP*.

Wang Ling, Chris Dyer, Alan W Black, and Isabel Trancoso. 2013. Paraphrasing 4 microblog normalization. In *Proc. EMNLP*.

Wang Ling, Chris Dyer, Alan Black, and Isabel Trancoso. 2015. Two/too simple adaptations of word2vec for syntax problems. In *Proc. NAACL*.

Shujie Liu, Nan Yang, Mu Li, and Ming Zhou. 2014. A recursive recurrent neural network for statistical machine translation. In *Proc. ACL*.

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

Christopher D. Manning. 2011. Part-of-speech tagging from 97% to 100%: Is it time for some linguistics? In *Proc. CICLing*.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of English: The Penn treebank. *Comput. Linguist.*

William Marslen-Wilson and Lorraine K. Tyler. 1998. Rules, representations, and the English past tense. *Trends in Cognitive Science*, 2(11).

M. Antonia Martí, Mariona Taulé, Lluís Márquez, and Manuel Bertran. 2007. CESS-ECE: A multilingual and multilevel annotated corpus. In *Proc. LREC*.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Proc. Interspeech*.

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

Robert C. Moore and Chris Quirk. 2009. Less is more: significance-based n-gram selection for smaller, better language models. In *Proc. EMNLP*.

Thomas Mueller, Helmut Schmid, and Hinrich Schütze. 2013. Efficient higher-order CRFs for morphological tagging. In *Proc. EMNLP*.

Tetsuji Nakagawa, Taku Kudoh, and Yuji Matsumoto. 2001. Unknown word guessing and part-of-speech tagging using support vector machines. In *In Proc. the Sixth Natural Language Processing Pacific Rim Symposium*.

Cicero D. Santos and Bianca Zadrozny. 2014. Learning character-level representations for part-of-speech tagging. In *Proc. ICML*.

Kristie Seymore and Ronald Rosenfeld. 1996. Scalable backoff language models. In *Proc. ICSLP*.

Anders Søgaard. 2011. Semisupervised condensed nearest neighbor for part-of-speech tagging. In *Proc. ACL*.

Radu Soricut and Franz Och. 2015. Unsupervised morphology induction using word embeddings. In *Proc. NAACL*.

Drahomíra Spoustová, Jan Hajič, Jan Raab, and Miroslav Spousta. 2009. Semi-supervised training for the averaged perceptron POS tagger. In *Proc. EACL*.

Andreas Stolcke. 1998. Entropy-based pruning of backoff language models. In *In Proc. DARPA Broadcast News Transcription and Understanding Workshop*.

Xu Sun. 2014. Structure regularization for structured prediction: Theories and experiments. *CoRR*, abs/1411.6243.

Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. 2012. LSTM neural networks for language modeling. In *Proc. Interspeech*.

Wei Xu, Alan Ritter, and Ralph Grishman. 2013. Gathering and generating paraphrases from twitter with application to normalization. In *Proceedings of the Sixth Workshop on Building and Using Comparable Corpora*.

Richard Zens, Daisy Stanton, and Peng Xu. 2012. A systematic comparison of phrase table pruning techniques. In *Proc. EMNLP*.

Syntax-Aware Multi-Sense Word Embeddings for Deep Compositional Models of Meaning

Jianpeng Cheng
University of Oxford
Department of
Computer Science
jianpeng.cheng@stcatz.oxon.org

Dimitri Kartsaklis
Queen Mary University of London
School of Electronic Engineering
and Computer Science
d.kartsaklis@qmul.ac.uk

Abstract

Deep compositional models of meaning acting on distributional representations of words in order to produce vectors of larger text constituents are evolving to a popular area of NLP research. We detail a compositional distributional framework based on a rich form of word embeddings that aims at facilitating the interactions between words in the context of a sentence. Embeddings and composition layers are jointly learned against a generic objective that enhances the vectors with syntactic information from the surrounding context. Furthermore, each word is associated with a number of senses, the most plausible of which is selected dynamically during the composition process. We evaluate the produced vectors qualitatively and quantitatively with positive results. At the sentence level, the effectiveness of the framework is demonstrated on the MSRPar task, for which we report results within the state-of-the-art range.

1 Introduction

Representing the meaning of words by using their distributional behaviour in a large text corpus is a well-established technique in NLP research that has been proved useful in numerous tasks. In a distributional model of meaning, the semantic representation of a word is given as a vector in some high dimensional vector space, obtained either by explicitly collecting co-occurrence statistics of the target word with words belonging to a representative subset of the vocabulary, or by directly optimizing the word vectors against an objective function in some neural-network based architecture (Collobert and Weston, 2008; Mikolov et al., 2013).

Regardless their method of construction, distributional models of meaning do not scale up to larger text constituents such as phrases or sentences, since the uniqueness of multi-word expressions would inevitably lead to data sparsity problems, thus to unreliable vectorial representations. The problem is usually addressed by the provision of a compositional function, the purpose of which is to prepare a vectorial representation for a phrase or sentence by combining the vectors of the words therein. While the nature and complexity of these compositional models may vary, approaches based on deep-learning architectures have been shown to be especially successful in modelling the meaning of sentences for a variety of tasks (Socher et al., 2012; Kalchbrenner et al., 2014).

The mutual interaction of distributional word vectors by a means of a compositional model provides many opportunities for interesting research, the majority of which still remains to be explored. One such direction is to investigate in what way lexical ambiguity affects the compositional process. In fact, recent work has shown that shallow multi-linear compositional models that explicitly handle extreme cases of lexical ambiguity in a step prior to composition present consistently better performance than their "ambiguous" counterparts (Kartsaklis and Sadrzadeh, 2013; Kartsaklis et al., 2014). A first attempt to test these observations in a deep compositional setting has been presented by Cheng et al. (2014) with promising results.

Furthermore, a second important question relates to the very nature of the word embeddings used in the context of a compositional model. In a setting of this form, word vectors are not any more just a means for discriminating words based on their underlying semantic relationships; the main goal of a word vector is to contribute to a bigger whole—a task in which syntax, along with semantics, also plays a very important role. It is a central point of this paper, therefore, that in a compositional distributional model of meaning word vectors should be injected with information that reflects their syntactical roles in the training corpus.

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1531–1542,
Lisbon, Portugal, 17-21 September 2015.

The purpose of this work is to improve the current practice in deep compositional models of meaning in relation to both the compositional process itself and the quality of the word embeddings used therein. We propose an architecture for jointly training a compositional model and a set of word embeddings, in a way that imposes dynamic word sense induction for each word during the learning process. Note that this is in contrast with recent work in multi-sense neural word embeddings (Neelakantan et al., 2014), in which the word senses are learned without any compositional considerations in mind.

Furthermore, we make the word embeddings syntax-aware by introducing a variation of the hinge loss objective function of Collobert and Weston (2008), in which the goal is not only to predict the occurrence of a target word in a context, but to also predict the *position* of the word within that context. A qualitative analysis shows that our vectors reflect both semantic and syntactic features in a concise way.

In all current deep compositional distributional settings, the word embeddings are internal parameters of the model with no use for any other purpose than the task for which they were specifically trained. In this work, one of our main considerations is that the joint training step should be generic enough to not be tied in any particular task. In this way the word embeddings and the derived compositional model can be learned on data much more diverse than any task-specific dataset, reflecting a wider range of linguistic features. Indeed, experimental evaluation shows that the produced word embeddings can serve as a high quality general-purpose semantic word space, presenting performance on the Stanford Contextual Word Similarity (SCWS) dataset of Huang et al. (2012) competitive to and even better of the performance of well-established neural word embeddings sets.

Finally, we propose a dynamic disambiguation framework for a number of existing deep compositional models of meaning, in which the multi-sense word embeddings and the compositional model of the original training step are further refined according to the purposes of a specific task at hand. In the context of paraphrase detection, we achieve a result very close to the current state-of-the-art on the Microsoft Research Paraphrase Corpus (Dolan and Brockett, 2005). An interesting aspect at the sideline of the paraphrase detection experiment is that, in contrast to mainstream approaches that mainly rely on simple forms of classifiers, we approach the problem by following a siamese architecture (Bromley et al., 1993).

2 Background and related work

2.1 Distributional models of meaning

Distributional models of meaning follow the *distributional hypothesis* (Harris, 1954), which states that two words that occur in similar contexts have similar meanings. Traditional approaches for constructing a word space rely on simple counting: a word is represented by a vector of numbers (usually smoothed by the application of some function such as point-wise mutual information) which show how frequently this word co-occurs with other possible context words in a corpus of text.

In contrast to these methods, a recent class of distributional models treat word representations as parameters directly optimized on a word prediction task (Bengio et al., 2003; Collobert and Weston, 2008; Mikolov et al., 2013; Pennington et al., 2014). Instead of relying on observed co-occurrence counts, these models aim to maximize the objective function of a neural net-based architecture; Mikolov et al. (2013), for example, compute the conditional probability of observing words in a context around a target word (an approach known as the *skip-gram model*). Recent studies have shown that, compared to their co-occurrence counterparts, neural word vectors reflect better the semantic relationships between words (Baroni et al., 2014) and are more effective in compositional settings (Milajevs et al., 2014).

2.2 Syntactic awareness

Since the main purpose of distributional models until now was to measure the semantic relatedness of words, relatively little effort has been put into making word vectors aware of information regarding the syntactic role under which a word occurs in a sentence. In some cases the vectors are POS-tag specific, so that 'book' as noun and 'book' as verb are represented by different vectors (Kartsaklis and Sadrzadeh, 2013). Furthermore, word spaces in which the context of a target word is determined by means of grammatical dependencies (Padó and Lapata, 2007) are more effective in capturing syntactic relations than approaches based on simple word proximity.

For word embeddings trained in neural settings, syntactic information is not usually taken explicitly into account, with some notable exceptions. At the lexical level, Levy and Goldberg (2014) propose an extension of the skip-gram model

based on grammatical dependencies. Following a different approach, Mnih and Kavukcuoglu (2013) weight the vector of each context word depending on its distance from the target word. With regard to compositional settings (discussed in the next section), Hashimoto et al. (2014) use dependency-based word embeddings by employing a hinge loss objective, while Hermann and Blunsom (2013) condition their objectives on the CCG types of the involved words.

As we will see in Section 3, the current paper offers an appealing alternative to those approaches that does not depend on grammatical relations or types of any form.

2.3 Compositionality in distributional models

The methods that aim to equip distributional models of meaning with compositional abilities come in many different levels of sophistication, from simple element-wise vector operators such as addition and multiplication (Mitchell and Lapata, 2008) to category theory (Coecke et al., 2010). In this latter work relational words (such as verbs or adjectives) are represented as multi-linear maps acting on vectors representing their arguments (nouns and noun phrases). In general, the above models are shallow in the sense that they do not have functional parameters and the output is produced by the direct interaction of the inputs; yet they have been shown to capture the compositional meaning of sentences to an adequate degree.

The idea of using neural networks for compositionality in language appeared 25 years ago in a seminal paper by Pollack (1990), and has been recently re-popularized by Socher and colleagues (Socher et al., 2011a; Socher et al., 2012). The compositional architecture used in these works is that of a *recursive neural network* (RecNN) (Socher et al., 2011b), where the words get composed by following a parse tree. A particular variant of the RecNN is the *recurrent neural network* (RNN), in which a sentence is assumed to be generated by aggregating words in sequence (Mikolov et al., 2010). Furthermore, some recent work (Kalchbrenner et al., 2014) models the meaning of sentences by utilizing the concept of a convolutional neural network (LeCun et al., 1998), the main characteristic of which is that it acts on small overlapping parts of the input vectors. In all the above models, the word embeddings and the weights of the compositional layers are optimized against a task-specific objective function.

In Section 3 we will show how to remove the restriction of a supervised setting, introducing a generic objective that can be trained on any general-purpose text corpus. While we focus on recursive and recurrent neural network architectures, the general ideas we will discuss are in principle model-independent.

2.4 Disambiguation in composition

Regardless of the way they address composition, all the models of Section 2.3 rely on ambiguous word spaces, in which every meaning of a polysemous word is merged into a single vector. Especially for cases of homonymy (such as 'bank', 'organ' and so on), where the same word is used to describe two or more completely unrelated concepts, this approach is problematic: the semantic representation of the word becomes the average of all senses, inadequate to express any of them in a reliable way.

To address this problem, a prior disambiguation step on the word vectors is often introduced, the purpose of which is to find the word representations that best fit to the given context, before composition takes place (Reddy et al., 2011; Kartsaklis et al., 2013; Kartsaklis and Sadrzadeh, 2013; Kartsaklis et al., 2014). This idea has been tested on algebraic and tensor-based compositional functions with very positive results. Furthermore, it has been also found to provide minimal benefits for a RecNN compositional architecture in a number of phrase and sentence similarity tasks (Cheng et al., 2014). This latter work clearly suggests that explicitly dealing with lexical ambiguity in a deep compositional setting is an idea that is worth to be further explored. While treating disambiguation as only a preprocessing step is a strategy less than optimal for a neural setting, one would expect that the benefits should be greater for an architecture in which the disambiguation takes place in a dynamic fashion during training.

We are now ready to start detailing a compositional model that takes into account the above considerations. The issue of lexical ambiguity is covered in Section 4; Section 3 below deals with generic training and syntactic awareness.

3 Syntax-based generic training

We propose a novel architecture for learning word embeddings and a compositional model to use them in a single step. The learning takes places in the context of a RecNN (or an RNN), and both word embeddings and parameters of the compositional layer are optimized against a generic objective function that uses a hinge loss function.

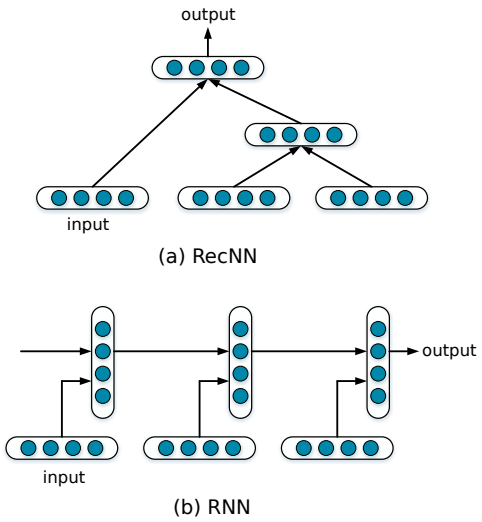


Figure 1: Recursive (a) and recurrent (b) neural networks.

Figure 1 shows the general form of recursive and recurrent neural networks. In architectures of this form, a compositional layer is applied on each pair of inputs $\mathbf{x_1}$ and $\mathbf{x_2}$ in the following way:

$$\mathbf{p} = g(\mathbf{W}\mathbf{x}_{[\mathbf{1:2}]} + \mathbf{b}) \quad (1)$$

where $\mathbf{x}_{[\mathbf{1:2}]}$ denotes the concatenation of the two vectors, g is a non-linear function, and $\mathbf{W}, \mathbf{b}$ are the parameters of the model. In the RecNN case, the compositional process continues recursively by following a parse tree until a vector for the whole sentence or phrase is produced; on the other hand, an RNN assumes that a sentence is generated in a left-to-right fashion, taking into consideration no dependencies other than word adjacency.

We amend the above setting by introducing a novel layer on the top of the compositional one, which scores the linguistic plausibility of the composed sentence or phrase vector with regard to *both* syntax and semantics. Following Collobert and Weston (2008), we convert the unsupervised learning problem to a supervised one by corrupting training sentences. Specifically, for each sentence s we create two sets of negative examples. In the first set, S', the target word within a given context is replaced by a random word; as in the original C&W paper, this set is used to enforce semantic coherence in the word vectors. Syntactic coherence is enforced by a second set of negative examples, S'', in which the words of the context have been randomly shuffled. The objective function is defined in terms of the following hinge losses:

$$\sum_{s \in S} \sum_{s' \in S'} \max(0, m - f(s) + f(s')) \quad (2)$$

$$\sum_{s \in S} \sum_{s'' \in S''} \max(0, m - f(s) + f(s'')) \quad (3)$$

where S is the set of sentences, f the compositional layer, and m a margin we wish to retain between the scores of the positive training examples and the negative ones. During training, all parameters in the scoring layer, the compositional layers and word representations are jointly updated by error back-propagation. As output, we get both general-purpose syntax-aware word representations and weights for the corresponding compositional model.

4 From words to senses

We now extend our model to address lexical ambiguity. We achieve that by applying a gated architecture, similar to the one used in the multi-sense model of Neelakantan et al. (2014), but advancing the main idea to the compositional setting detailed in Section 3.

We assume a fixed number of n senses per word.[1] Each word is associated with a main vector (obtained for example by using an existing vector set, or by simply applying the process of Section 3 in a separate step), as well as with n vectors denoting cluster centroids and an equal number of sense vectors. Both cluster centroids and sense vectors are randomly initialized in the beginning of the process. For each word w_t in a training sentence, we prepare a context vector by averaging the main vectors of all other words in the same context. This context vector is compared with the cluster centroids of w_t by cosine similarity, and the sense corresponding to the closest cluster is selected as the most representative of w_t in the current context. The selected cluster centroid is updated by the addition of the context vector, and the associated sense vector is passed as input to the compositional layer. The selected sense vectors for each word in the sentence are updated by back-propagation, based on the objectives of Equations 2 and 3. The overall architecture of our model, as described in this and the previous section, is illustrated in Figure 2.

5 Task-specific dynamic disambiguation

The model of Figure 2 decouples the training of word vectors and compositional parameters from

[1] Note that in principle the fixed number of senses assumption is not necessary; Neelakantan et al. (2014), for example, present a version of their model in which new senses are added dynamically when appropriate.

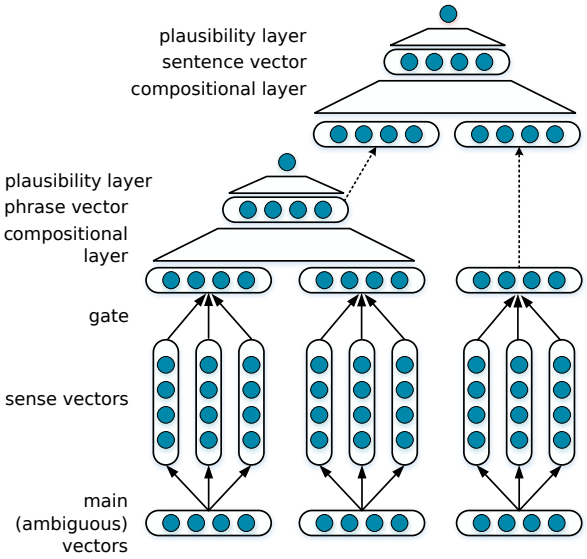


Figure 2: Training of syntax-aware multi-sense embeddings in the context of a RecNN.

a specific task, and as a consequence from any task-specific training dataset. However, note that by replacing the plausibility layer with a classifier trained for some task at hand, you get a task-specific network that transparently trains multi-sense word embeddings and applies dynamic disambiguation on the fly. While this idea of a single-step direct training seems appealing, one consideration is that the task-specific dataset used for the training will not probably reflect the linguistic variety that is required to exploit the expressiveness of the setting in its full. Additionally, in many cases the size of datasets tied to specific tasks is prohibiting for training a deep architecture.

It is a merit of this proposal that, in cases like these, it is possible for one to train the generic model of Figure 2 on any large corpus of text, and then use the produced word vectors and compositional weights to initialize the parameters of a more specific version of the architecture. As a result, the trained parameters will be further refined according to the task-specific objective. Figure 3 illustrates the generic case of a compositional framework applying dynamic disambiguation. Note that here sense selection takes place by a soft-max layer, which can be directly optimized on the task objective.

6 A siamese network for paraphrase detection

We will test the dynamic disambiguation framework of Section 5 in a paraphrase detection task. A *paraphrase* is a restatement of the meaning of a sentence using different words and/or syntax. The goal of a paraphrase detection model, thus, is to examine two sentences and decide if they express the same meaning.

While the usual way to approach this problem is to utilize a classifier that acts (for example) on the concatenation of the two sentence vectors, in this work we follow a novel perspective: specifically, we apply a *siamese architecture* (Bromley et al., 1993), a concept that has been extensively used in computer vision (Hadsell et al., 2006; Sun et al., 2014). While siamese networks have been also used in the past for NLP purposes (for example, by Yih et al. (2011)), to the best of our knowledge this is the first time that such a setting is applied for paraphrase detection.

In our model, two networks sharing the same parameters are used to compute the vectorial representations of two sentences, the paraphrase relation of which we wish to detect; this is achieved by employing a cost function that compares the two vectors. There are two commonly used cost functions: the first is based on the L_2 norm (Hadsell et al., 2006; Sun et al., 2014), while the second on the cosine similarity (Nair and Hinton, 2010; Sun et al., 2014). The L_2 norm variation is capable of handling differences in the magnitude of the vectors. Formally, the cost function is defined as:

$$E_f = \begin{cases} \frac{1}{2}\left\|f(s_1) - f(s_2)\right\|_2^2, & \text{if } y = 1 \\ \frac{1}{2}\max(0, m - \left\|f(s_1) - f(s_2)\right\|_2)^2, & o.w. \end{cases}$$

where s_1, s_2 are the input sentences, f the compositional layer (so $f(s_1)$ and $f(s_2)$ refer to sentence vectors), and $y = 1$ denotes a paraphrase relationship between the sentences; m stands for the margin, a hyper-parameter chosen in advance. On

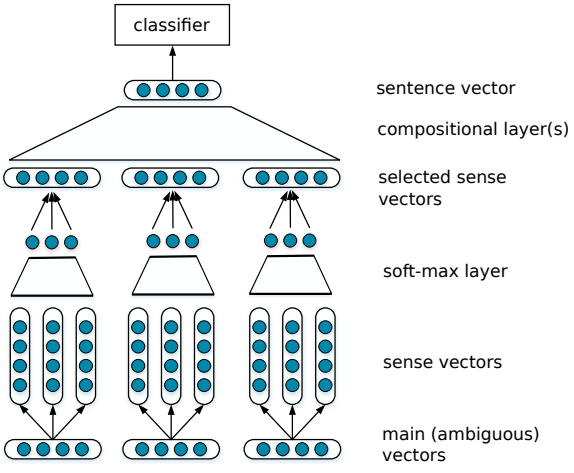


Figure 3: Dynamic disambiguation in a generic compositional deep net.

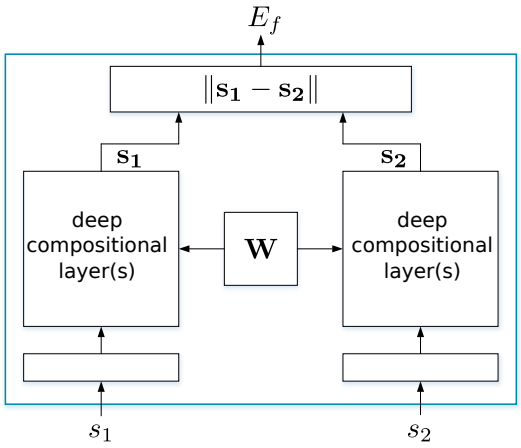


Figure 4: A siamese network for paraphrase detection.

the other hand, the cost function based on cosine similarity handles only directional differences, as follows:

$$E_f = \frac{1}{2}(y - \sigma(wd + b))^2 \quad (4)$$

where $d = \frac{f(s_1) \cdot f(s_2)}{\|f(s_1)\|_2 \|f(s_2)\|_2}$ is the cosine similarity of the two sentence vectors, w and b are the scaling and shifting parameters to be optimized, σ is the sigmoid function and y is the label. In the experiments that will follow in Section 7.4, both of these cost functions are evaluated. The overall architecture is shown in Figure 4.

In Section 7.4 we will use the pre-trained vectors and compositional weights for deriving sentence representations that will be subsequently fed to the siamese network. When the dynamic disambiguation framework is used, the sense vectors of the words are updated during training so that the sense selection process is gradually refined.

7 Experiments

We evaluate the quality of the compositional word vectors and the proposed deep compositional framework in the tasks of word similarity and paraphrase detection, respectively.

7.1 Model pre-training

In all experiments the word representations and compositional models are pre-trained on the British National Corpus (BNC), a general-purpose text corpus that contains 6 million sentences of written and spoken English. For comparison we train two sets of word vectors and compositional models, one ambiguous and one multi-sense (fixing 3 senses per word). The dimension of the embeddings is set to 300.

As our compositional architectures we use a RecNN and an RNN. In the RecNN case, the words are composed by following the result of an external parser, while for the RNN the composition takes place in sequence from left to right. To avoid the exploding or vanishing gradient problem (Bengio et al., 1994) for long sentences, we employ a *long short-term memory* (LSTM) network (Hochreiter and Schmidhuber, 1997). During the training of each model, we minimize the hinge loss in Equations 2 and 3. The plausibility layer is implemented as a 2-layer network, with 150 units at the hidden layer, and is applied at each individual node (as opposed to a single application at the sentence level). All parameters are updated with mini-batches by AdaDelta (Zeiler, 2012) gradient descent method ($\lambda = 0.03$, initial $\alpha = 0.05$).

7.2 Qualitative evaluation of the word vectors

As a first step, we qualitatively evaluate the trained word embeddings by examining the nearest neighbours lists of a few selected words. We compare the results with those produced by the skip-gram model (SG) of Mikolov et al. (2013) and the language model (CW) of Collobert and Weston (2008). We refer to our model as SAMS (Syntax-Aware Multi-Sense). The results in Table 1 show clearly that our model tends to group words that are both semantically and syntactically related; for example, and in contrast with the compared models which group words only at the semantic level, our model is able to retain tenses, numbers (singulars and plurals), and gerunds.

The observed behaviour is comparable to that of embedding models with objective functions conditioned on grammatical relations between words; Levy and Goldberg (2014), for example, present a similar table for their dependency-based extension of the skip-gram model. The advantage of our approach against such models is twofold: firstly, the word embeddings are accompanied by a generic compositional model that can be used for creating sentence representations independently of any specific task; and secondly, the training is quite forgiving to data sparsity problems that in general a dependency-based approach would intensify (since context words are paired with the grammatical relations they occur with the target word). As a result, a small corpus such as the BNC is sufficient for producing high quality syntax-aware word embeddings.

	SG	CW	SAMS
begged	beg, begging, cried	begging, pretended, beg	persuaded, asked, cried
refused	refusing, refuses, refusal	refusing , declined, refuse	declined, rejected, denied
interrupted	interrupting, punctuated, interrupt	interrupts, interrupt, interrupting	punctuated, preceded, disrupted
themes	thematic, theme, notions	theme, concepts, subtext	meanings, concepts, ideas
patiently	impatiently, waited, waits	impatiently, queue, expectantly	impatiently, siliently, anxiously
player	players, football, league	game, club, team	athlete, sportsman, team
prompting	prompted, prompt, sparking	prompt, amid, triggered	sparking, triggering, forcing
reproduce	reproducing, replicate, humans	reproducing, thrive, survive	replicate, produce, repopulate
predictions	prediction, predict, forecasts	predicting, assumption, predicted	expectations, projections, forecasts

Table 1: Nearest neighbours for a number of words with various embedding models.

7.3 Word similarity

We now proceed to a quantitative evaluation of our embeddings on the Stanford Contextual Word Similarity (SCWS) dataset of Huang et al. (2012). The dataset contains 2,003 pairs of words and the contexts they occur in. We can therefore make use of the contextual information in order to select the most appropriate sense for each ambiguous word. Similarly to Neelakantan et al. (2014), we use three different metrics: *globalSim* measures the similarity between two ambiguous word vectors; *localSim* selects a single sense for each word based on the context and computes the similarity between the two sense vectors; *avgSim* represents each word as a weighted average of all senses in the given context and computes the similarity between the two weighted sense vectors.

We compute and report the Spearman's correlation between the embedding similarities and human judgments (Table 2). In addition to the skip-gram and Collobert and Weston models, we also compare against the CBOW model (Mikolov et al., 2013) and the multi-sense skip-gram (MSSG) model of Neelakantan et al. (2014).

Model	globalSim	localSim	avgSim
CBOW	59.5	–	–
SG	**61.8**	–	–
CW	55.3	–	–
MSSG	61.3	56.7	62.1
SAMS	59.9	**58.5**	**62.5**

Table 2: Results for the word similarity task (Spearman's $\rho \times 100$).

Among all methods, only the MSSG model and ours are capable of learning multi-prototype word representations. Our embeddings show top performance for *localSim* and *avgSim* measures, and performance competitive to that of MSSG and SG for *globalSim*, both of which use a hierarchical soft-max as their objective function. Compared to the original C&W model, our version presents an improvement of 4.6%—a clear indication for the effectiveness of the proposed learning method and the enhanced objective.

7.4 Paraphrase detection

In the last set of experiments, the proposed compositional distributional framework is evaluated on the Microsoft Research Paraphrase Corpus (MSRPC) (Dolan and Brockett, 2005), which contains 5,800 pairs of sentences. This is a binary classification task, with labels provided by human annotators. We apply the siamese network detailed in Section 6.

While MSRPC is one of the most used datasets for evaluating paraphrase detection models, its size is prohibitory for any attempt of training a deep architecture. Therefore, for our training we rely on a much larger external dataset, the Paraphrase Database (PPDB) (Ganitkevitch et al., 2013). The PPDB contains more than 220 million paraphrase pairs, of which 73 million are phrasal paraphrases and 140 million are paraphrase patterns that capture syntactic transformations of sentences. We use these phrase- and sentence-level paraphrase pairs as additional training contexts to fine-tune the generic compositional model parameters and word embeddings and to train the baseline models. The original training set of the MSRPC is used as validation set for deciding hyperparameters, such as the margin of the error function and the number of training epochs.

The evaluations were conducted on various aspects, and the models are gradually refined to demonstrate performance within the state-of-the-art range.

Comparison of the two error functions In the first evaluation, we compare the two error functions of the siamese network using only ambigu-

ous vectors. As we can see in Table 3, the cosine error function consistently outperforms the L_2 norm-based one for both compositional models, providing a yet another confirmation of the already well-established fact that similarity in semantic vector spaces is better reflected by length-invariant measures.

Model	L_2	Cosine
RecNN	73.8	**74.9**
RNN	73.0	**74.3**

Table 3: Results with different error functions for the paraphrase detection task (accuracy $\times$ 100).

Effectiveness of disambiguation We now proceed to compare the effectiveness of the two compositional models when using ambiguous vectors and multi-sense vectors, respectively. Our error function is set to cosine similarity, following the results of the previous evaluation. When dynamic disambiguation is applied, we test two methods of selecting sense vectors: in the hard case the vector of the most plausible sense is selected, while in the soft case a new vector is prepared as the weighted average of all sense vectors according to probabilities returned by the soft-max layer (see Figure 3). As a baseline we use a simple compositional model based on vector addition.

The dynamic disambiguation models and the additive baseline are compared with variations that use a simple prior disambiguation step applied on the word vectors. This is achieved by first selecting for each word the sense vector that is the closest to the average of all other word vectors in the same sentence, and then composing the selected sense vectors without further considerations regarding ambiguity. The baseline model and the prior disambiguation variants are trained as separate logistic regression classifiers. The results are shown in Table 4.

Model	Ambig.	Prior	Hard DD	Soft DD
Addition	69.9	71.3	–	–
RecNN	74.9	75.3	75.7	**76.0**
RNN	74.3	74.6	75.1	**75.2**

Table 4: Different disambiguation choices for the paraphrase detection task (accuracy $\times$ 100).

Overall, disambiguated vectors work better than the ambiguous ones, with the improvement to be more significant for the additive model; there, a simple prior disambiguation step produces 1.4% gains. For the deep compositional models, simple prior disambiguation is still helpful with small improvements, a result which is consistent with the findings of Cheng et al. (2014). The small gains of the prior disambiguation models over the ambiguous models clearly show that deep architectures are quite capable of performing this elementary form of sense selection intrinsically, as part of the learning process itself. However, the situation changes when the dynamic disambiguation framework is used, where the gains over the ambiguous version become more significant. Comparing the two ways of dynamic disambiguation (hard method and soft method), the numbers that the soft method gives are slightly higher, producing a total gain of 1.1% over the ambiguous version for the RecNN case.[2]

Note that, at this stage, the advantage of using the dynamic disambiguation framework over simple prior disambiguation is still small (0.7% for the case of RecNN). We seek the reason behind this in the recursive nature of our architecture, which tends to progressively "hide" local features of word vectors, thus diminishing the effect of the fine-tuned sense vectors produced by the dynamic disambiguation mechanism. The next section discusses the problem and provides a solution.

The role of pooling One of the problems of the recursive and recurrent compositional architectures, especially in grammars with strict branching structure such as in English, is that any given composition is usually the product of a terminal and a non-terminal; i.e. a single word can contribute to the meaning of a sentence to the same extent as the rest of a sentence on its whole, as below:

$[[\text{kids}]_{\text{NP}}\ [\text{play ball games in the park}]_{\text{VP}}]_{\text{S}}$

In the above case, the contribution of the words within the verb phrase to the final sentence representation will be faded out due to the recursive composition mechanism. Inspired by related work in computer vision (Sun et al., 2014), we attempt to alleviate this problem by introducing an average pooling layer at the sense vector level and adding the resulting vector to the sentence representation. By doing this we expect that the new sentence vector will reflect local features from all words in the sentence that can help in the classification in a more direct way. The results for the new deep architectures are shown in Table 5, where we see substantial improvements for both deep nets. More importantly, the effect of dynamic

[2]For all subsequent experiments, the reported results are based on the soft selection method.

disambiguation now becomes more significant, as expected by our analysis.

Table 5 also includes results for two models trained in a single step, with word and sense vectors randomly initialized at the beginning of the process. We see that, despite the large size of the training set, the results are much lower than the ones obtained when using the pre-training step. This demonstrates the importance of the initial training on a general-purpose corpus: the resulting vectors reflect linguistic information that, although not obtainable from the task-specific training, can make great difference in the result of the classification.

Model	Ambig.	Prior	Dynamic
RecNN+pooling	75.5	76.3	**77.6**
RNN+pooling	74.8	75.9	**76.6**
1-step RecNN+pooling	74.4	–	72.9
1-step RNN+pooling	73.6	–	73.1

Table 5: Results with average pooling for the paraphrase detection task (accuracy $\times$ 100).

Cross-model comparison In this section we propose a method to further improve the performance of our models, and we present an evaluation against some of the previously reported results.

We notice that using distributional properties alone cannot capture efficiently subtle aspects of a sentence, for example numbers or human names. However, even small differences on those aspects between two sentences can lead to a different classification result. Therefore, we train (using the MSPRC training data) an additional logistic regression classifier which is based not only on the embeddings similarity, but also on a few hand-engineered features. We then ensemble the new classifier (C1) with the original one. In terms of feature selection, we follow Socher et al. (2011a) and Blacoe and Lapata (2012) and add the following features: the difference in sentence length, the unigram overlap among the two sentences, features related to numbers (including the presence or absence of numbers from a sentence and whether or not the numbers in the two sentences are the same). In Table 6 we report results of the original model and the ensembled model, and we compare with the performance of other existing models.

In all of the implemented models (including the additive baseline), disambiguation is performed to guarantee the best performance. We see that by ensembling the original classifier with C1, we improve the result of the previous section by another 1%. This is the second best result reported so far for the specific task, with a 0.6 difference in F-score from the first (Ji and Eisenstein, 2013).[3]

	Model	Acc.	F1
BL	All positive	66.5	79.9
	Addition (disamb.)	71.3	81.1
Dynamic Dis.	RecNN	76.0	84.0
	RecNN+Pooling	77.6	84.7
	RecNN+Pooling+C1	**78.6**	**85.3**
	RNN	75.2	83.6
	RNN+Pooling	76.6	84.3
	RNN+Pooling+C1	77.5	84.6
Published results	Mihalcea et al. (2006)	70.3	81.3
	Rus et al. (2008)	70.6	80.5
	Qiu et al. (2006)	72.0	81.6
	Islam and Inkpen (2009)	72.6	81.3
	Fernando and Stevenson (2008)	74.1	82.4
	Wan et al. (2006)	75.6	83.0
	Das and Smith (2009)	76.1	82.7
	Socher et al. (2011a)	76.8	83.6
	Madnani et al. (2012)	77.4	84.1
	Ji and Eisenstein (2013)	**80.4**	**85.9**

Table 6: Cross-model comparison in the paraphrase detection task.

8 Conclusion and future work

The main contribution of this paper is a deep compositional distributional model acting on linguistically motivated word embeddings.[4] The effectiveness of the syntax-aware, multi-sense word vectors and the dynamic compositional disambiguation framework in which they are used was demonstrated by appropriate tasks at the lexical and sentence level, respectively, with very positive results. As an aside, we also demonstrated the benefits of a siamese architecture in the context of a paraphrase detection task. While the architectures tested in this work were limited to a RecNN and an RNN, the ideas we presented are in principle directly applicable to any kind of deep network. As a future step, we aim to test the proposed models on a convolutional compositional architecture, similar to that of Kalchbrenner et al. (2014).

Acknowledgments

The authors would like to thank the three anonymous reviewers for their useful comments, as well as Nal Kalchbrenner and Ed Grefenstette for early discussions and suggestions on the paper, and Simon Šuster for comments on the final draft. Dimitri Kartsaklis gratefully acknowledges financial support by AFOSR.

[3] Source: ACL Wiki (`http://www.aclweb.org/aclwiki`), August 2015.

[4] Code in Python/Theano and the word embeddings can be found at `https://github.com/cheng6076`.

References

Marco Baroni, Georgiana Dinu, and Germán Kruszewski. 2014. Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, volume 1.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *Neural Networks, IEEE Transactions on*, 5(2):157–166.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *The Journal of Machine Learning Research*, 3:1137–1155.

William Blacoe and Mirella Lapata. 2012. A comparison of vector-based representations for semantic composition. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 546–556. Association for Computational Linguistics.

Jane Bromley, James W Bentz, Léon Bottou, Isabelle Guyon, Yann LeCun, Cliff Moore, Eduard Säckinger, and Roopak Shah. 1993. Signature verification using a siamese time delay neural network. *International Journal of Pattern Recognition and Artificial Intelligence*, 7(04):669–688.

Jianpeng Cheng, Dimitri Kartsaklis, and Edward Grefenstette. 2014. Investigating the role of prior disambiguation in deep-learning compositional models of meaning. In *2nd Workshop of Learning Semantics, NIPS 2014*, Montreal, Canada, December.

B Coecke, M Sadrzadeh, and S Clark. 2010. Mathematical foundations for a distributional compositional model of meaning. lambek festschrift. *Linguistic Analysis*, 36:345–384.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM.

Dipanjan Das and Noah A Smith. 2009. Paraphrase identification as probabilistic quasi-synchronous recognition. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pages 468–476. Association for Computational Linguistics.

W.B. Dolan and C. Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Third International Workshop on Paraphrasing (IWP2005)*.

Samuel Fernando and Mark Stevenson. 2008. A semantic similarity approach to paraphrase detection. In *Proceedings of the 11th Annual Research Colloquium of the UK Special Interest Group for Computational Linguistics*, pages 45–52. Citeseer.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. Ppdb: The paraphrase database. In *HLT-NAACL*, pages 758–764.

Raia Hadsell, Sumit Chopra, and Yann LeCun. 2006. Dimensionality reduction by learning an invariant mapping. In *Computer vision and pattern recognition, 2006 IEEE computer society conference on*, volume 2, pages 1735–1742. IEEE.

Zellig S Harris. 1954. Distributional structure. *Word*.

Kazuma Hashimoto, Pontus Stenetorp, Makoto Miwa, and Yoshimasa Tsuruoka. 2014. Jointly learning word representations and composition functions using predicate-argument structures. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1544–1555, Doha, Qatar, October. Association for Computational Linguistics.

Karl Moritz Hermann and Phil Blunsom. 2013. The role of syntax in vector space models of compositional semantics. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 894–904, Sofia, Bulgaria, August. Association for Computational Linguistics.

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

Eric H Huang, Richard Socher, Christopher D Manning, and Andrew Y Ng. 2012. Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 873–882. Association for Computational Linguistics.

Aminul Islam and Diana Inkpen. 2009. Semantic similarity of short texts. *Recent Advances in Natural Language Processing V*, 309:227–236.

Yangfeng Ji and Jacob Eisenstein. 2013. Discriminative improvements to distributional sentence similarity. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 891–896, Seattle, Washington, USA, October. Association for Computational Linguistics.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, June.

Dimitri Kartsaklis and Mehrnoosh Sadrzadeh. 2013. Prior disambiguation of word tensors for constructing sentence vectors. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1590–1601, Seattle, Washington, USA, October. Association for Computational Linguistics.

Dimitri Kartsaklis, Mehrnoosh Sadrzadeh, and Stephen Pulman. 2013. Separating disambiguation from composition in distributional semantics. In *Proceedings of 17th Conference on Natural Language Learning (CoNLL)*, pages 114–123, Sofia, Bulgaria, August.

Dimitri Kartsaklis, Nal Kalchbrenner, and Mehrnoosh Sadrzadeh. 2014. Resolving lexical ambiguity in tensor regression models of meaning. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Vol. 2: Short Papers)*, pages 212–217, Baltimore, USA, June. Association for Computational Linguistics.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Omer Levy and Yoav Goldberg. 2014. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 302–308, Baltimore, Maryland, June. Association for Computational Linguistics.

Nitin Madnani, Joel Tetreault, and Martin Chodorow. 2012. Re-examining machine translation metrics for paraphrase identification. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 182–190. Association for Computational Linguistics.

Rada Mihalcea, Courtney Corley, and Carlo Strapparava. 2006. Corpus-based and knowledge-based measures of text semantic similarity. In *AAAI*, volume 6, pages 775–780.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010*, pages 1045–1048.

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

Dmitrijs Milajevs, Dimitri Kartsaklis, Mehrnoosh Sadrzadeh, and Matthew Purver. 2014. Evaluating neural word representations in tensor-based compositional settings. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 708–719, Doha, Qatar, October. Association for Computational Linguistics.

Jeff Mitchell and Mirella Lapata. 2008. Vector-based models of semantic composition. In *Proceedings of ACL-08: HLT*, pages 236–244, Columbus, Ohio, June. Association for Computational Linguistics.

Andriy Mnih and Koray Kavukcuoglu. 2013. Learning word embeddings efficiently with noise-contrastive estimation. In *Advances in Neural Information Processing Systems*, pages 2265–2273.

Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 807–814.

Arvind Neelakantan, Jeevan Shankar, Alexandre Passos, and Andrew McCallum. 2014. Efficient non-parametric estimation of multiple embeddings per word in vector space. In *Proceedings of EMNLP*.

S. Padó and M. Lapata. 2007. Dependency-based Construction of Semantic Space Models. *Computational Linguistics*, 33(2):161–199.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. *Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014)*, 12.

Jordan B Pollack. 1990. Recursive distributed representations. *Artificial Intelligence*, 46(1):77–105.

Long Qiu, Min-Yen Kan, and Tat-Seng Chua. 2006. Paraphrase recognition via dissimilarity significance classification. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 18–26. Association for Computational Linguistics.

Siva Reddy, Ioannis P Klapaftis, Diana McCarthy, and Suresh Manandhar. 2011. Dynamic and static prototype vectors for semantic composition. In *IJCNLP*, pages 705–713.

Vasile Rus, Philip M McCarthy, Mihai C Lintean, Danielle S McNamara, and Arthur C Graesser. 2008. Paraphrase identification with lexico-syntactic graph subsumption. In *FLAIRS conference*, pages 201–206.

Richard Socher, Eric H Huang, Jeffrey Pennin, Christopher D Manning, and Andrew Y Ng. 2011a. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Advances in Neural Information Processing Systems*, pages 801–809.

Richard Socher, Cliff C Lin, Chris Manning, and Andrew Y Ng. 2011b. Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 129–136.

R. Socher, B. Huval, C. Manning, and Ng. A. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Conference on Empirical Methods in Natural Language Processing 2012*.

Yi Sun, Yuheng Chen, Xiaogang Wang, and Xiaoou Tang. 2014. Deep learning face representation by joint identification-verification. In *Advances in Neural Information Processing Systems*, pages 1988–1996.

Stephen Wan, Mark Dras, Robert Dale, and Cécile Paris. 2006. Using dependency-based features to take the para-farce out of paraphrase. In *Proceedings of the Australasian Language Technology Workshop*, volume 2006.

Wen-tau Yih, Kristina Toutanova, John C. Platt, and Christopher Meek. 2011. Learning discriminative projections for text similarity measures. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning*, pages 247–256, Portland, Oregon, USA, June. Association for Computational Linguistics.

Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Conversation Trees: A Grammar Model for Topic Structure in Forums

Annie Louis and Shay B. Cohen
School of Informatics
University of Edinburgh
Edinburgh, EH8 9AB, UK
{alouis,scohen}@inf.ed.ac.uk

Abstract

Online forum discussions proceed differently from face-to-face conversations and any single thread on an online forum contains posts on different subtopics. This work aims to characterize the content of a forum thread as a *conversation tree* of topics. We present models that jointly perform two tasks: segment a thread into subparts, and assign a topic to each part. Our core idea is a definition of topic structure using probabilistic grammars. By leveraging the flexibility of two grammar formalisms, Context-Free Grammars and Linear Context-Free Rewriting Systems, our models create desirable structures for forum threads: our topic segmentation is hierarchical, links non-adjacent segments on the same topic, and jointly labels the topic during segmentation. We show that our models outperform a number of tree generation baselines.

1 Introduction

Online forums are commonplace today and used for various purposes: product support and troubleshooting, opining about events and people, and student interaction on online course platforms. Threads in these forums become long, involve posts from multiple users, and the chronological order of the posts in a thread does not represent a continuous flow of dialog. Adding structure to these threads is important for tasks such as information extraction, search, and summarization.

One such aspect of structure is *topic*. Figure 1 shows a computer-troubleshooting related thread with six posts. The first post is the troubleshooting question and the remaining posts can be seen as focusing on either of two topics, the *driver software* (posts p_1, p_2, p_5) or the *speaker hardware* (p_3, p_4). By categorizing posts into such topics, we can provide a useful division of content in a thread and even across multiple threads. Note that the *driver* topic is not a contiguous sequence but present in non-adjacent parts, (p_1, p_2) and (p_5).

p_0 Bob:	When I play a recorded video on my camera, it looks and sounds fine. On my computer, it plays at a really fast rate and sounds like Alvin and the Chipmunks!
p_1 Kate:	I'd find and install the latest audio driver.
p_2 Mary:	The motherboard supplies the clocks for audio feedback. So update the audio and motherboard drivers.
p_3 Chris:	Another fine mess in audio is volume and speaker settings. You checked these?
p_4 Jane:	Yes, under speaker settings, look for hardware acceleration. Turning it off worked for me.
p_5 Matt:	Audio drivers are at this link. Rather than just audio drivers, I would also just do all drivers.

Table 1: Example forum thread conversation

We tackle the problem of joint topic segmentation and topic labeling of forum threads. Given a thread's posts in chronological order (the order in which they were posted), we create a phrase structure tree indicating how the posts are grouped hierarchically into subtopics and super-topics. In these *conversation trees*, leaves span entire posts. Each non-terminal identifies the topic characterizing the posts in its span. Topics are concepts or themes which summarize the content of a group of posts. Specifically, a topic is a set of words which frequently co-occur in posts which are similar in content and other conversation regularities.

Our key insight in this work is to formalize topic structure using probabilistic grammars. We define a base grammar for topic structure of forum threads and refine it to represent finer topics and subtrees. We learn to predict trees under our grammar based on two formalisms: Probabilistic Context-Free Grammars (PCFG) and Probabilistic Linear Context-Free Rewriting Systems (PLCFRS). In the PCFG model, a non-terminal spans a contiguous sequence of posts. In the

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1543–1553,
Lisbon, Portugal, 17-21 September 2015.

PLCFRS model, non-terminals are allowed to span discontinuous segments of posts. We leverage algorithms from probabilistic parsing of natural language sentences and modify them for our domain. We show that our model performs well and sidesteps a number of limitations of prior topic segmentation approaches. In particular:

• Our models perform joint topic segmentation and topic labeling while most existing models identify unlabeled segments. Labeling topics on segments creates richer annotation, and links non-adjacent segments on the same topic.

• Our grammar-based probabilistic models have two key benefits. They naturally create tree structures which are considered linguistically suitable for topic segmentation but were difficult to create under previous approaches. Second, the flexibility of grammars such as PLCFRS allow our models to seamlessly learn to produce trees where non-adjacent segments on the same topic are explicitly linked, an issue that was not addressed before.

We present large-scale experiments on a collection of forum threads from the computer-troubleshooting domain.[1] We show that our grammar models achieve a good balance between identifying when posts should be in the same topic versus a different topic. These grammar models outperform other tree generation baselines by a significant margin especially on short threads.

2 Related work

The ideas in this paper are related to three areas of prior research.

Forum thread analysis. Finding structure in forum threads has been previously addressed in two ways. The first is reply structure prediction where a parent post is linked to its children (replies) which were posted later in time (Wang et al., 2008; Cong et al., 2008). Reply links are sometimes augmented with a dialog act label indicating whether the child post is a question, answer, or confirmation to the parent post (Kim et al., 2010; Wang et al., 2011). The second set of methods partition sentences in emails or blog comments into topical clusters and then show salient words per cluster as topic tags (Joty et al., 2013).

We focus on producing rich hierarchical segmentation going beyond clusters which do not contain any cluster-internal structure. We also believe that topic structure is complementary to dialog act and reply link annotations. Tasks on forum data such as user expertise (Lui and Baldwin, 2009) and post quality prediction (Agichtein et al., 2008), and automatic summarization (Nenkova and Bagga, 2003) can be carried out on a fine-grained level using topic information.

Conversation disentanglement. A related problem of clustering utterances is defined specifically for Internet Relay Chat (IRC) and speech conversations. In this case, multiple conversations, on different topics, are mixed in and systems extract threads which separate out individual conversations (Shen et al., 2006; Adams and Martell, 2008; Elsner and Charniak, 2010).

Disentanglement is typically applied for the coarse-level problem of identifying a coherent conversation. In addition, these methods do not create any structure upon the clustered utterances in contrast to the focus of this work.

Topic Segmentation. is a task which directly focuses on the topic aspect of text and speech. This task is of greater importance for speech which lacks explicit structure such as paragraphs and sections. Many approaches perform linear segmentation where boundaries are inserted in the text or speech to divide it into a flat set of topic segments (Hearst, 1994; Utiyama and Isahara, 2001; Galley et al., 2003; Malioutov and Barzilay, 2006; Eisenstein and Barzilay, 2008). Very few methods recursively combine smaller segments into larger ones. Such hierarchical models (Eisenstein, 2009) have been applied at a coarse level for segmenting very long texts such as books into sections. Other work has focused on linear segmentation for documents within the same domain and having a regular structure (Chen et al., 2009; Jeong and Titov, 2010; Du et al., 2015). These latter approaches rely on three assumptions: that the documents contain a regular set of topics, that these topics are discussed in a fairly regular consensus order, and that the same topic does not recur in the same document.

Our models address two deficiencies in these approaches. First, text is commonly understood to have a hierarchical structure (Grosz and Sidner, 1986) and our grammar model is an ideal framework for this goal. Tree structures also have other advantages, for example, we do not predefine the number of expected topic segments in a conversation tree, a requirement posed by many prior seg-

[1] Our corpus is available from `http://kinloch.inf.ed.ac.uk/public/CTREES/ConversationTrees.html`

mentation algorithms. The second limitation of prior studies is assuming that topics do not recur in the same document. But linguistic theories allow for non-adjacent utterances to belong to the same topic segment (Grosz and Sidner, 1986) and this fact is empirically true in chat and forum conversations (Elsner and Charniak, 2010; Wang et al., 2011). Our models can flexibly handle and link recurring topics within and across threads.

As a final note, because of the annotations required, most prior work on forums or IRC chats have typically used few hundred threads. We present a heuristically derived large corpus of topic structure on which we evaluate our models.

3 Background

Our topic discovery methods are based on two constituency grammar formalisms.

3.1 Probabilistic Context-Free Grammars

A PCFG is defined by a 5-tuple G_C =(N, T, P, S, D) where N is a set of non-terminal symbols, T a set of terminal symbols, and S is the start symbol. P is a set of production rules of the form $A \rightarrow \beta$ where A is a non-terminal and $\beta \in \{N \cup T\}^*$. D is a function that associates each production rule with a conditional probability of the form $p(A \rightarrow \beta|A)$. This probability indicates how often the non-terminal A expands into β. The probabilities of all the rules conditioned on a particular non-terminal should sum to 1.

The joint probability of a tree T with yield Y, $P(T, Y)$, is the product of the probabilities of all the productions used to construct T. The parsing problem is to find the tree $\hat{T}$ which is most likely given the yield Y. $\hat{T} = \arg\max_T P(T|Y) = \arg\max_T P(T, Y)$.

Given training trees, we can enumerate the productions and compute their probabilities using maximum likelihood estimates (MLE):

$$p(A \rightarrow \beta|A) = \frac{\text{count}(A \rightarrow \beta)}{\text{count}(A)}$$

which is the fraction of the times the non-terminal A expands into β.

The most likely parse tree can be found using a number of algorithms. In this work, we use the CYK algorithm for PCFGs in Chomsky Normal Form. This algorithm has complexity $\mathcal{O}(n^3)$ where n is the length of the yield.

PCFGs do not capture a frequently occurring property of forum threads, discontinuous segments on the same topic. Indirectly however, a PCFG may assign the same non-terminal for each of these segments. To model these discontinuities more directly, we present a second model based on PLCFRS where non-terminals are allowed to span discontinuous yield strings.

3.2 Probabilistic Linear Context-Free Rewriting Systems

LCFRS grammars (Vijay-Shanker et al., 1987) generalize CFGs, where non-terminals can span discontinuous constituents. Formally, the span of an LCFRS non-terminal is a tuple, with size $k \geq 1$, of strings, where k is the non-terminal "fan-out". As such, the fan-out of a CFG non-terminal is 1.

An LCFRS G_L =(N, T, P, S, V) where N is the set of non-terminals, T the terminals and S is the start symbol. A function $f : N \rightarrow \mathbb{N}$ gives the fan-out of each non-terminal. P is the set of productions or otherwise called rewriting rules of the LCFRS. V is a set of variables used to indicate the spans of each non-terminal in these rules. A rewriting rule has the form:

$$A(\alpha_1, \alpha_2, \ldots, \alpha_{f(A)}) \rightarrow$$
$$A_1(x_1^1 \ldots, x_{f(A_1)}^1), \ldots, A_m(x_1^m, \ldots, x_{f(A_m)}^m)$$

Here A,A_1,...,$A_m \in N$. Since there are m non-terminals on the RHS, this rule has rank m. $x_j^i \in V$ for $1 \leq i \leq m$ and $1 \leq j \leq f(A_i)$ indicate the $f(A_i)$ discontinuous spans dominated by A_i. $\alpha_i \in (T \cup V)^*$, $1 \leq i \leq f(A)$ are the spans of the LHS non-terminal A.

A rewriting rule explains how the left-hand side (LHS) non-terminal's span can be composed from the yields of the right-hand side (RHS) non-terminals. For example, in the rule $A(x_1x_2, x_3) \rightarrow B(x_1)C(x_2, x_3)$, A and C have fan-out 2, B has fan-out 1. The two spans of A, x_1x_2 and x_3, are composed from the spans of B and C. For comparison, the productions of a CFG take the single spans of each non-terminal on the RHS and concatenate them in the same order to yield a single span of the LHS non-terminal.

A Probabilistic LCFRS (PLCFRS) (Levy, 2005) also contains D, a function which assigns conditional probabilities p($A(\vec{x}) \rightarrow \vec{\phi}|A(\vec{x})$) to the rules. The probabilities conditioned on a particular non-terminal and span configuration, $A(\vec{x})$ should sum to 1. Given a training corpus, LCFRS rules can be read out and probabilities computed similar to CFG rules.

To find the most likely parse tree, we use the parsing algorithm proposed by Kallmeyer and Maier (2013) for binary PLCFRS. The approach uses weighted deduction rules (Shieber et al., 1995; Nederhof, 2003), which specify how to compute a new item from other existing items. Each item is of the form $[A, \vec{\rho}]$ where A is a non-terminal and $\vec{\rho}$ is a vector indicating the spans dominated by A. A weight w is attached to each item which gives the $|\log|$ of the Viterbi inside probability of the subtree under that item. A set of goal items specify the form of complete parse trees. By using the Knuth's generalization (Knuth, 1977) of the shortest paths algorithm, the most likely tree can be found without exhaustive parsing as in Viterbi parsing of CFGs. The complexity of parsing is $\mathcal{O}(n^{3k})$ where k is the fan-out of the grammar (the maximum fan-out of its rules).

4 Problem Formulation

Given a thread consisting of a sequence of posts $(p_1, p_2, \ldots, p_n)$ in chronological order, the task is to produce a constituency tree with yield $(p_1, p_2 \ldots p_n)$. A leaf in this tree spans an entire post. Non-terminals identify the topic of the posts within their span. Non-terminals at higher levels of the tree represent coarser topics in the conversation (the span covered by these nodes contain more posts) than those lower in the tree. The root topic node indicates the overall topic of the thread.

Below we define a Context-Free Grammar (CFG) for such trees.

4.1 A Grammar for Conversation Trees

G_B is our *base grammar* which is context-free and has four non-terminals $\{S, X, T, C\}$. Each post p in the corpus is a terminal symbol (i.e. a terminal symbol is a bag of words). The productions in G_B are: $S \rightarrow T\ X^*$, $X \rightarrow T\ X^*$ and $T \rightarrow p$.

G_B generates trees with the following structure. A root-level topic S characterizes the content of the entire thread. *Thread-starting* rules are of the form, $S \rightarrow T\ X^*$, where X^* indicates a sequence of zero or more X non-terminals. T nodes are pre-terminals analogous to part-of-speech tags in the case of syntactic parsing. In our grammar, the $T \rightarrow p$ rule generates a post in the thread. In the *thread-starting* rules, T generates the first post of the thread which poses the query or comment that elicits the rest of the conversation. The X^* sequence denotes *topic branches*, the subtree under each X is assumed to correspond to a different topic. These X's characterize all but the first post of the thread. The *continuation rules*, $X \rightarrow T\ X^*$, recursively subdivide the X subtrees into topics spanning fewer posts. In each case, the T node on the right-hand side of these rules generates the first post (in terms of posting time) in that subtree. Therefore posts made earlier in time always dominate (in the tree structure) those which come later in the thread. We define the head of a non-terminal as the first post as per the chronological order of posts in the span of the non-terminal.

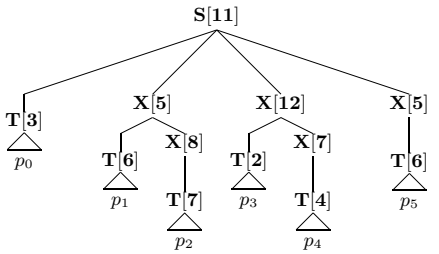


Figure 1: Example conversation tree for the thread in Table 1

This grammar does not generate binary trees. We binarize the tree with C nodes to obtain an equivalent grammar in Chomsky Normal Form (CNF) (CNF yields parsing algorithms with lower complexity)[2]: $S \rightarrow T\ C \mid T$, $X \rightarrow T\ C \mid T$, $T \rightarrow p$ and $C \rightarrow X \mid X\ X \mid X\ C$. The C nodes can be collapsed and its daughters attached to the parent of C to revert back to the non-binary tree.

While this CFG defines the structure of conversation trees, by itself this grammar is insufficient for our task. In particular, it contains a single non-terminal of each type (S, X, T, C) and so does not distinguish between topics. We extend this grammar to create G_E which has a *set* of non-terminals corresponding to each non-terminal in G_B, these fine-grained non-terminals correspond to different topics. G_E is created using latent annotations (Matsuzaki et al., 2005) on the X, T, S and C non-terminals from G_B. The resulting non-terminals for G_E are $S[i]$, $X[j]$, $T[k]$ and $C[l]$, such that $1 \leq i \leq N_S$, $1 \leq j \leq N_X$, $1 \leq k \leq N_T$, $1 \leq l \leq N_C$. i, j, k and l identify specific topics attached to a particular node type.

Our output trees are created with G_E to depict the topic segmentation of the thread and are non-binary. The binary trees produced by our algorithms are converted by collapsing the C. As a result, conversation trees have $S[i]$, $X[j]$ and $T[k]$

[2]Any context-free grammar can be converted to an equivalent CNF grammar. Our algorithms support unary rules.

nodes but no $C[l]$ nodes.

An example conversation tree for the thread in Table 1 is shown in Figure 1. At level 1, $T[3]$ describes the topic of the first post while the remaining posts are under $X[5]$ which may indicate a *driver* topic, and $X[12]$, a *speaker hardware* topic. Note how $X[5]$ may re-occur in the conversation to accommodate post p_5 on the *driver* topic.

4.2 Supervised learning framework

We use a supervised framework for learning the models. We assume that we have training trees according to the base grammar, G_B. The following section describes our data and how we obtain these G_B-based trees. In Section 6, we present a method for creating G_E-type trees with non-terminal refinements. Estimates of rule probabilities from this augmented training corpus are used to develop the parsers for topic segmentation.

5 Data

We collected 13,352 computer-troubleshooting related threads from `http://forums.cnet.com/`. The number of posts per thread varies greatly between 1 and 394, and the average is around 5 posts. We divide these threads into training, development and test sets. The most frequent 100 words from the training set are used as stopwords. After filtering stopwords, a post contains 39 tokens on average and the vocabulary size of our corpus is 81,707. For development and testing, we only keep threads with a minimum of 3 posts (so that the problem is non-trivial) and a maximum of 50 posts (due to complexity of parsing). We have 9,243 training threads, 2,014 for development, and 2,071 for testing.

A particular feature of the forums on `cnet.com` is the explicit reply structure present in the threads. The forum interface elicits these reply relationships as users develop a thread. When a user replies in a particular thread, she has to choose (only) one of the earlier posts in the thread (including the question post) to attach her reply to. In this way, each post is linked to a unique post earlier in time in the same thread. This reply structure forms a dependency tree. Figure 2 (a) is a possible reply tree for the thread in Table 1.

5.1 Deriving conversation trees

Next we convert these reply-link trees into phrase-structure conversation trees. We developed a deterministic conversion method that uses the generative process defined by the base grammar G_B. The key idea is to track when the conversation branches into sub-topics and when the replies are proceeding within the same topic.

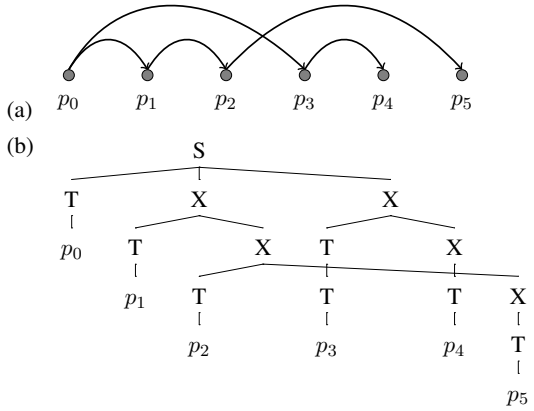


Figure 2: (a) A reply structure tree for the thread in Table 1 and (b) the derived conversation tree

The algorithm traverses the nodes of the dependency tree D in breadth-first order, starting at the root (first) post. We create a root S node in the phrase structure tree H. Then the *thread-starting* rule from G_B, $S \rightarrow T\ X^*$, is used to create one T and k X nodes as children of S. The first post p_0 is attached as a child of the T node. k is equal to the number of replies to p_0 (children of p_0 in D). For each of these k X nodes, we instantiate a $X \rightarrow T\ X^*$ rule in H. The k replies of p_0 are attached one each as a child of the T nodes in these rules. Any set of children are always instantiated in chronological order. So the span of a non-terminal in H always contains posts in non-decreasing time order. We continue the procedure with the next post from D in the traversal order.

This procedure converts the reply tree of Figure 2 (a) into the conversation tree (b). Note that (a) is a possible reply structure for our example thread in Table 1. The conversation tree (b) derived according to this reply structure has a non-projective structure where p_1, p_2 and p_5 are linked under one X node (at level 1). Such a tree can be produced by our LCFRS model. The ideal PCFG tree will repeat the topic branch as in Figure 1.

The derived trees at this stage follow G_B and contain only the S, X, T non-terminals (without any latent annotations). This tree is converted into Chomsky Normal Form using C nodes.

5.2 Discontinuous topic segments

As in our example above, non-projective edges in the *reply structure* are rather frequent. Of the

total threads in our corpus 14.5% contain a non-projective edge. A thread should have a minimum of four posts to have the possibility of non-projective edges. Among the 7,691 threads with at least four posts, the percentage of non-projective trees is even higher, 25%. This finding suggests that in any thread of reasonable size which we wish to summarize or categorize, non-projective edges will be common. Hence a direct approach for addressing discontinuous segments such as our PLCFRS model is important for this domain.

6 Parsers for Conversation Trees

The training data are conversation trees with rules from G_B. We refine the non-terminals to create G_E, extract PCFG or PLCFRS rules from the training trees, and build a CYK parser that predicts the most likely tree according to G_E.

6.1 Refining the non-terminals

We use a clustering approach, akin to the spectral algorithm of Cohen et al. (2013) and Narayan and Cohen (2015),[3] to create finer grained categories corresponding to G_B's non-terminals: S, X, C and T. Each node in each tree in the training data is associated with a feature vector, which is a function of the tree and the anchor node. These vectors are clustered (for each of the non-terminals separately) and then, each node is annotated with the corresponding cluster. This process gives us the non-terminals $S[i]$, $X[j]$, $T[k]$ and $C[l]$ of G_E.

The features for a node n_l are: depth of n_l in the tree, root is at depth 0; maximum depth of the subtree under n_l; number of siblings of n_l; number of children of n_l; number of posts in the span of n_l; average length (in terms of tokens) of the posts in the span of n_l; average similarity of the span of n_l with the span of n_l's siblings[4]; similarity of n_l's span with the span of its left-most sibling; elapsed time between the first and last posts in n_l's span.

We use CLUTO toolkit (Karypis, 2002) to perform clustering. The algorithm maximizes the pairwise cosine similarity between the feature vectors of nodes within the same cluster. The best number of clusters for the four non-terminal node types are tuned jointly to give the best performance on our final topic segmentation task.

6.2 Learning rule probabilities

As mentioned previously, each terminal in our grammar is an entire post's text. For the pre-terminal to terminal productions in our grammar $T[j] \rightarrow p_i$, we compute $p(T[j] \rightarrow p_i|T[j])$ as the probability under a unigram language model L_j which is trained on the collection of the posts from the training corpus which are dominated by $T[j]$ nodes. $p(T[j] \rightarrow p_i|T[j]) = \prod_{k=1}^{N_{p_i}} L_j(w_k^i)$ where $w_1^i, w_2^i ... w_{N_{p_i}}^i$ are the tokens in post p_i.

The rest of the production probabilities are learned using MLE on the training trees. In the case of LCFRS rules, the gap information is also obtained during the extraction.

6.3 CYK parsing

For both PCFG and LCFRS we use CYK style algorithms, as outlined in §3, to obtain the most likely tree. For the more computationally complex LCFRS model, we make a number of additions to improve speed. First, we restrict the fan-out of the grammar to 2, i.e. any non-terminal can only span a maximum of two discontinuous segments. 97% of the productions in fact have only non-terminals with fan-out ≤ 2. Second, we use A^* search (Maier et al., 2012) to prioritize our agenda. Last, we reduce the number of items added to the agenda. An item has the form [A, $\vec{\rho}$], A is a non-terminal and $\vec{\rho}$ is the spans covered by A. For every span, we only keep the top 5 non-terminal items according to the score. In addition, we only allow spans with a gap of at most 2 since 77% of all gaps (dominated by fan-out $\leq$ 2) non-terminals are ≤ 2 posts. Moreover, after a certain number of items (10,000) are added to the chart, we only allow the creation of new items which have a contiguous span.

7 Systems for comparison

We compare our models to two types of systems.

7.1 STRUCTURE ONLY

The first type generate tree structures without considering the content of the threads.

Right-branching tree (RBT). produces a strictly right branching tree where each post is dominated by the immediately previous (according to time) post in the thread. It uses the grammar

[3] The main difference between our algorithm and the algorithm by Narayan and Cohen (2015) is that we do not decompose the trees into "inside" trees and "outside" trees, or use a singular value decomposition step before clustering the features.

[4] The span of n_l and that of a sibling are each represented by binary vectors indicating the presence and absence of a term in the span. The similarity value is computing using cosine overlap between the vectors and the average across all siblings is recorded.

with the rules $\{S \rightarrow TX,\ X \rightarrow TX,\ X \rightarrow T,\ T \rightarrow p\}$. This method does not perform useful topic segmentation as it produces only a single topic branch containing all the posts.

Attach-to-root tree (ART). attaches each post to the root of the tree. The grammar rules are $\{S \rightarrow TX_1...X_n,\ X \rightarrow T,\ T \rightarrow p\}$, where n is the number of posts in the thread. This approach assumes each post belongs to a different topic in the thread. In contrast to RBT, ART contains too many topic branches, one per post in the thread.

Random tree (RAND). mixes decisions to create a new topic branch or continue in the same branch. The generation process is top down, at each step, the algorithm chooses a certain number of topic branches (Xs) to create ($\leq$ number of posts left to add to the tree). Then, the number of posts under each branch is sampled (such that each branch has at least one post). This process is then recursively done at the new topic branches.

7.2 STRUCTURE AND CONTENT

These approaches produce tree structures informed by content. We build these parsers by modifying prior models for chat disentanglement and linear topic segmentation of documents.

Similarity tree (SIM). produces trees by attaching each post as a child of the most similar of the previous (by time) posts (Wang et al., 2008). We use cosine similarity between vector representations of two posts in order to compute similarity. When the similarity exceeds a threshold value, the post is added under the topic branch of the prior post. Otherwise the post is under a new topic branch attached to the root of the tree. A threshold of 0.15 was chosen after tuning on the development data.

Cluster tree (CLUS). uses an approach related to chat disentanglement (Elsner and Charniak, 2010). The posts within *each* thread are clustered *separately* into k_l clusters where $k_l = l/h$ depends on the number of posts in the thread, l. $h = 6$ was chosen by tuning. The posts in each cluster are ordered by time and a right branching tree is created over them. These k_l cluster-level trees are then attached as children of a new node to create a thread-level tree. The cluster-trees are ordered left to right in the thread-tree according to the time of the earliest post in each cluster.

Linear segmentation tree (LSEG). is based on postprocessing the output of a Bayesian linear topic segmentation model (Eisenstein and Barzilay, 2008). Each post's content is treated as a sentence and a document is created for *each* thread by appending its posts in their time order. The model is then used to group consecutive sentences into k_l segments. For each thread of length l, $k_l = l/h$, $h = 6$ was chosen by tuning. For each segment, a right branching tree is created and these segment-level trees are made siblings in a thread-level tree. The segments are added left to right in the thread-tree as per their order in the text.

All STRUCTURE trees contain thread structure but no topic labels. In other words, they have coarse non-terminals (X, T and S) only. The STRUCTURE AND CONTENT trees, LSEG and CLUS contain topics or groups but only at one top level, and further the number and labels of these topics are different per thread. Hence there is no linking across threads. Within a thread, the SIM and CLUS tree can link non-adjacent posts under the same topic. These links are also not available from a LSEG tree.

8 Evaluation metrics

To evaluate the topic segmentation, we develop a node-governance based measure. Our score compares two conversation trees g and h, where g is the gold-standard tree and h is the hypothesized one. We assume that g and h are in dependency format, reversing the transformation from §5.1.

We break g (and h) into a set of pairs, for each pair of nodes in the tree (each node is a post in the thread). For each such pair, p and q, we find their least common ancestor, $\ell(p, q|g)$ (or $\ell(p, q|h)$. If these nodes are in a governing relation (p dominates q or vice versa), then $\ell(p, q)$ is the dominating node. We then define the following sets and quantities for $\cdot \in \{g, h\}$:

- $S_1(\cdot) = \{(p, q, \ell(p, q)) \mid \ell(p, q|\cdot) \in \{p, q\}\}$.
- $S_2(\cdot) = \{(p, q, \ell(p, q)) \mid \ell(p, q|\cdot) \notin \{p, q\}\}$.
- $n_1(g, h) = |S_1(g) \cap S_1(h)|$.
- $n_2(g, h) = |S_2(g) \cap S_2(h)|$.

$s_1(\cdot)$ and $s_2(\cdot)$ are defined as the size of $S_1(\cdot)$ and $S_2(\cdot)$, repsectively. Let $g_1, \ldots, g_n$ and $h_1, \ldots, h_n$ be a corpus of gold-standard conversation trees and their corresponding hypothesized conversation trees. Then the evaluation metric we compute is the harmonic mean (Fscore) of the micro-average of the precision for governing (G-p) and non-governing (NG-p) pairs, and recall for governing (G-r) and non-governing (NG-r) pairs. For example, G-p is calculated as

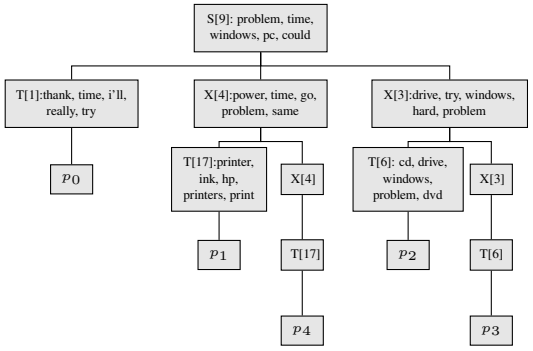
$$\text{G-p} = \frac{\sum_{i=1}^{n} n_1(g_i, h_i)}{\sum_{i=1}^{n} s_1(h_i)}.$$

Traditional parsing evaluation measures such as constituency bracketting and dependency attachment scores were too local for our purpose. For example, if a long chain of posts is placed in a different topic but their local dependencies are maintained, we only penalize one constituent and one node's parent in the constituency and dependency scores respectively. But the topic segmentation created by this change has several posts placed in the wrong topic branch. Our scores overcome this problem by considering the relationship between all pairs of posts and also dividing the relationship in the pair as governing or non-governing.

9 Results and discussion

We tune the number of latent topic annotations for the non-terminals using grid search on the development set. The best settings are 40 S, 100 X, 20 C, 80 T clusters for PCFG and 10 S, 5 X, 15 C, 40 T for LCFRS.

Below we show an example non-projective tree created by our LCFRS parser. The topics are indicated with the most frequent 5 words in the matching cluster.



Here post p_4 though later in posting time is predicted to be on the same topic as p_1.

The non-terminals in our trees enable useful topic segmentation and we found that performance is extremely sensitive to the number of non-terminals of each type S, X, C and T. Currently, we do not have a direct method to evaluate the non-terminals in our tree but we plan to use the information in other applications as an evaluation.

Table 2 and 3 shows the segmentation performance of the models (as percentages). The performance varied greatly depending on the length of the threads and hence we show the results separately for threads with up to 15 posts (SHORT) and those with 16 to 50 posts (LONG). The results are divided into sections based on the subset of test data on which the evaluation is performed. The first section (R1.) is performance on all threads, (R2.) only on the projective threads in the test data, and (R3.) only on the non-projective threads.

Among the baselines, the Right-branching trees (RBT) or Attaching to the root (ART) have some advantages: the RBT receives 100% recall of the governing pairs and the ART tree has high recall of the non-governing pairs. However, their Fscores are 0. Recall that the RBT contains a single topic branch and hence no useful segmentation is done; ART is the other extreme where every post is put in a separate topic branch. RAND is the average performance of 3 randomly generated trees for each thread. This method has a better balance between branching and depth leading to 33.4 Fscore for SHORT and 21.5 for LONG threads.

The PCFG and the LCFRS models clearly outperform these baselines. The Fscore improves up to 15% over RAND on SHORT and LONG threads. The grammar models also consistently outperform SIM systems.

With regard to CLUS and LSEG, there is a difference in performance between SHORT and LONG threads and based on whether the desired structure was projective or non-projective. On SHORT threads, the grammar models outperform LSEG and CLUS particularly on the projective threads (the LCFRS model has a 22% higher Fscore). On the longer threads however, the CLUS and LSEG models perform best overall and for non-projective threads. CLUS and LSEG directly model the content similarity of posts while the grammar models make many decisions at level of topic nodes. Remember that the clustering is done per thread in CLUS and LSEG compared to using a common set of topics across all threads. Making such fine-grained similarity comparison appears to be helpful especially for longer threads and even though LSEG does not make non-projective decisions, its accuracy is high on the attachments it makes leading to good performance on non-projective threads too. In future work, we plan to explore how we can combine the advantages of direct similarity with the grammar models.

Between the two grammar models, the LCFRS model is better than PCFG, even on projective threads, and can produce non-projective trees. Part of this improvement on projective trees could be due to more data being available in the LCFRS model since all the data can be used for training it.

Model	Ex	G-p	G-r	NG-p	NG-r	F
	R1. On all gold threads (1,971 threads, 24,620 post pairs)					
RBT	20.4	50.8	100.0	100.0	0.0	0.0
ART	5.6	100.0	0.0	42.3	86.0	0.0
RAND	5.2	55.5	19.4	39.2	65.5	33.4
SIM	5.7	68.2	13.3	43.1	79.0	27.9
CLUS	20.2	52.9	85.5	47.5	17.2	42.8
LSEG	20.2	53.0	88.2	52.2	16.5	42.8
PCFG	9.7	52.7	60.4	41.0	34.9	**48.3**
LCFRS	11.4	53.3	62.5	43.6	35.9	**49.9**
	R2. On projective gold threads only					
RBT	24.4	59.8	100.0	100.0	0.0	0.0
ART	6.7	100.0	0.0	35.1	87.4	0.0
RAND	6.2	62.0	22.0	32.4	63.5	35.3
SIM	5.9	73.8	13.9	36.0	79.6	28.4
CLUS	24.2	59.8	90.6	31.3	7.3	26.8
LSEG	24.2	60.1	91.9	35.6	7.6	27.9
PCFG	11.7	61.2	60.5	35.8	36.4	**49.5**
LCFRS	13.5	62.0	64.5	37.6	35.3	**50.8**
	R3. On non-projective gold threads only					
RBT	0.0	39.1	100.0	100.0	0.0	0.0
ART	0.0	100.0	0.0	51.7	84.8	0.0
RAND	0.0	42.0	14.3	47.3	67.3	26.9
SIM	4.3	58.1	12.1	52.1	78.5	26.0
CLUS	0.0	41.2	75.1	54.4	25.8	45.4
LSEG	0.0	41.8	80.7	59.8	24.1	**45.9**
PCFG	0.0	41.1	60.1	47.6	33.5	45.2
LCFRS	0.3	40.8	58.5	50.4	36.4	**46.0**

Table 2: Results on threads with up to 15 posts: for the grammar models (PCFG and LCFRS) and comparison systems (See Section 7). 'Ex' is percentage of fully correct trees and other scores are from Section 8. Top two Fscores are in bold.

Model	Ex	G-p	G-r	NG-p	NG-r	F
	R1. On all gold threads (100 threads, 27,590 post pairs)					
RBT	0.0	21.2	100.0	100.0	0.0	0.0
ART	0.0	100.0	0.0	69.8	88.6	0.0
RAND	0.0	38.9	10.0	65.4	78.4	21.5
SIM	0.0	37.0	8.9	67.2	80.9	19.6
CLUS	0.0	32.6	37.3	73.3	70.5	**42.0**
LSEG	0.0	35.4	50.4	76.8	68.0	**48.4**
PCFG	0.0	24.6	54.1	54.3	36.8	36.6
LCFRS	0.0	22.8	71.4	68.7	29.4	36.5
	R2. On projective gold threads only					
RBT	0.0	36.7	100.0	100.0	0.0	0.0
ART	0.0	100.0	0.0	57.1	90.3	0.0
RAND	0.0	59.9	11.0	56.0	82.6	24.3
SIM	0.0	45.9	8.3	54.4	80.2	19.1
CLUS	0.0	42.0	38.1	60.0	63.2	45.3
LSEG	0.0	51.3	55.0	68.0	65.1	**56.9**
PCFG	0.0	42.2	66.6	34.5	22.9	39.9
LCFRS	0.0	49.0	65.3	51.6	41.7	**52.3**
	R3. On non-projective gold threads only					
RBT	0.0	19.6	100.0	100.0	0.0	0.0
ART	0.0	100.0	0.0	71.1	88.5	0.0
RAND	0.0	36.1	9.9	66.3	78.1	20.9
SIM	0.0	35.8	9.0	68.5	81.0	19.7
CLUS	0.0	31.2	37.1	74.6	71.1	**41.3**
LSEG	0.0	33.2	49.6	77.6	68.2	**46.8**
PCFG	0.0	22.3	51.6	55.8	37.9	34.8
LCFRS	0.0	20.9	72.6	71.6	28.4	34.8

Table 3: Results on threads with > 15 posts

For the PCFG model, only the projective data can be used for training.

Overall, the LCFRS model is powerful on projective threads and SHORT non-projective threads. Compared to PCFG, the LCFRS model has a number of advantages: we can use more data, can predict non-projective trees. Some of the constraints we imposed on the LCFRS parser, such as restricting the gap degree are likely to have limited the ability of the model to generate more flexible non-projective edges. We believe that as we figure out how to make these parsers faster, we will see even more improvements from the LCFRS models.

10 Conclusions

This work represents a first approach to learn discourse structure of forum threads within an explicit grammar framework. We show that a coarse grammar for structure can be refined using latent annotations to indicate the finer topic differences. Our trees have good segmentation performance and provide useful summaries of the thread content at the non-terminal nodes. A main goal for future work is to incorporate further domain specific constraints on the models to improve parsing speed and at the same time allow more flexible trees. We also plan to evaluate the usefulness of conversation trees in tasks such as predicting if a thread is resolved, and user expertise.

Acknowledgements

We thank the anonymous reviewers for their suggestions. We also thank Bonnie Webber, Adam Lopez and other members of the Probabilistic Models of Language reading group at the University of Edinburgh for helpful discussions. The first author was supported by a Newton International Fellowship (NF120479) from the Royal Society and the British Academy.

References

P. H. Adams and C. H. Martell. 2008. Topic detection and extraction in chat. In *Proceedings of the IEEE International Conference on Semantic Computing*, pages 581–588.

E. Agichtein, C. Castillo, D. Donato, A. Gionis, and G. Mishne. 2008. Finding high-quality content in social media. In *Proceedings of WSDM*, pages 183–194.

H. Chen, S. R. K. Branavan, R. Barzilay, and D. R. Karger. 2009. Content modeling using latent permutations. *Journal of Artificial Intelligence Research*, 36(1):129–163.

S. B. Cohen, K. Stratos, M. Collins, D. P. Foster, and L. Ungar. 2013. Experiments with spectral learning of latent-variable PCFGs. In *Proceedings of NAACL*.

G. Cong, L. Wang, C. Lin, Y. Song, and Y. Sun. 2008. Finding question-answer pairs from online forums. In *Proceedings of SIGIR*, pages 467–474.

L. Du, J. K. Pate, and M. Johnson. 2015. Topic segmentation with an ordering-based topic model. In *Proceedings of AAAI*, pages 2232–2238.

J. Eisenstein and R. Barzilay. 2008. Bayesian unsupervised topic segmentation. In *Proceedings of EMNLP*, pages 334–343.

J. Eisenstein. 2009. Hierarchical text segmentation from multi-scale lexical cohesion. In *Proceedings of HLT:NAACL*, pages 353–361.

M. Elsner and E. Charniak. 2010. Disentangling chat. *Computational Linguistics*, 36(3):389–409.

M. Galley, K. McKeown, E. Fosler-Lussier, and H. Jing. 2003. Discourse segmentation of multi-party conversation. In *Proceedings of ACL*, pages 562–569.

B. J. Grosz and C. L. Sidner. 1986. Attention, intentions, and the structure of discourse. *Computational Linguistics*, 12(3):175–204, July.

M.A. Hearst. 1994. Multi-paragraph segmentation of expository text. In *Proceedings of ACL*, pages 9–16.

M. Jeong and I. Titov. 2010. Unsupervised discourse segmentation of documents with inherently parallel structure. In *Proceedings of ACL: Short papers*, pages 151–155.

S. Joty, G. Carenini, and R. T. Ng. 2013. Topic segmentation and labeling in asynchronous conversations. *Journal of Artificial Intelligence Research*, 47(1):521–573.

L. Kallmeyer and W. Maier. 2013. Data-driven parsing using probabilistic linear context-free rewriting systems. *Computational Linguistics*, 39(1):87–119.

G. Karypis. 2002. Cluto - a clustering toolkit. Technical Report TR-02-017, Dept. of Computer Science, University of Minnesota.

S. Kim, L. Cavedon, and T. Baldwin. 2010. Classifying dialogue acts in one-on-one live chats. In *Proceedings of EMNLP*, pages 862–871.

D. E. Knuth. 1977. A generalization of dijkstra's algorithm. *Information Processing Letters*, 6(1):1–5.

R. Levy. 2005. *Probabilistic models of word order and syntactic discontinuity*. Ph.D. thesis, Stanford University.

M. Lui and T. Baldwin. 2009. Classifying user forum participants: Separating the gurus from the hacks, and other tales of the internet. In *Proceedings of the 2010 Australasian Language Technology Workshop*, pages 49–57.

W. Maier, M. Kaeshammer, and L. Kallmeyer. 2012. Plcfrs parsing revisited: Restricting the fan-out to two. In *Proceedings of the 11th International Workshop on Tree Adjoining Grammar and Related Formalisms*.

I. Malioutov and R. Barzilay. 2006. Minimum cut model for spoken lecture segmentation. In *Proceedings of COLING-ACL*, pages 25–32.

T. Matsuzaki, Y. Miyao, and J. Tsujii. 2005. Probabilistic cfg with latent annotations. In *Proceedings of ACL*, pages 75–82.

S. Narayan and S. B. Cohen. 2015. Diversity in spectral learning for natural language parsing. In *Proceedings of EMNLP*.

M. Nederhof. 2003. Weighted deductive parsing and knuth's algorithm. *Computational Linguistics*, 29(1):135–143.

A. Nenkova and A. Bagga. 2003. Facilitating email thread access by extractive summary generation. In *Proceedings of RANLP*.

D. Shen, Q. Yang, J. Sun, and Z. Chen. 2006. Thread detection in dynamic text message streams. In *Proceedings of SIGIR*, pages 35–42.

S. M. Shieber, Y. Schabes, and F. C. N. Pereira. 1995. Principles and implementation of deductive parsing. *Journal of Logic Programming*, 24(1&2):3–36.

M. Utiyama and H. Isahara. 2001. A statistical model for domain-independent text segmentation. In *Proceedings of ACL*, pages 499–506.

K. Vijay-Shanker, D. J. Weir, and A. K. Joshi. 1987. Characterizing structural descriptions produced by various grammatical formalisms. In *Proceedings of ACL*, pages 104–111.

Y. Wang, M. Joshi, W. Cohen, and C. P. Rosé. 2008. Recovering implicit thread structure in newsgroup style conversations. In *Proceedings of ICWSM*.

L. Wang, M. Lui, S. Kim, J. Nivre, and T. Baldwin. 2011. Predicting thread discourse structure over technical web forums. In *Proceedings of EMNLP*, pages 13–25.

Fast, Flexible Models for Discovering Topic Correlation across Weakly-Related Collections

Jingwei Zhang[1], **Aaron Gerow**[2], **Jaan Altosaar**[3], **James Evans**[2,4], **Richard Jean So**[5]

[1]Department of Computer Science, Columbia University
jz2541@columbia.edu
[2] Computation Institute, University of Chicago
{gerow,jevans}@uchicago.edu
[3] Department of Physics, Princeton University
altosaar@princeton.edu
[4] Department of Sociology, University of Chicago
[5] Department of English Language and Literature, University of Chicago
richardjeanso@uchicago.edu

Abstract

Weak topic correlation across document collections with different numbers of topics in individual collections presents challenges for existing cross-collection topic models. This paper introduces two probabilistic topic models, *Correlated LDA* (C-LDA) and *Correlated HDP* (C-HDP). These address problems that can arise when analyzing large, asymmetric, and potentially weakly-related collections. Topic correlations in weakly-related collections typically lie in the tail of the topic distribution, where they would be overlooked by models unable to fit large numbers of topics. To efficiently model this long tail for large-scale analysis, our models implement a parallel sampling algorithm based on the Metropolis-Hastings and alias methods (Yuan et al., 2015). The models are first evaluated on synthetic data, generated to simulate various collection-level asymmetries. We then present a case study of modeling over 300k documents in collections of sciences and humanities research from JSTOR.

1 Introduction

Comparing large text collections is a critical task for the curation and analysis of human cultural history. Achievements of research and scholarship are most accessible through textual artifacts, which are increasingly available in digital archives. Text-based research, often undertaken by humanists, historians, lexicographers, and corpus linguists, explores patterns of words in documents across time-periods and distinct collections of text. Here, we introduce two new topic models designed to compare large collections, Correlated LDA (C-LDA) and Correlated HDP (C-HDP), which are sensitive to document-topic asymmetry (where collections have different topic distributions) and topic-word asymmetry (where a single topic has different word distributions in each collection). These models seek to address terminological questions, such as how a topic on physics is articulated distinctively in scientific compared to humanistic research. Accommodating potential collection-level asymmetries is particularly important when researchers seek to analyze collections with little prior knowledge about shared or collection-specific topic structure. Our models extend existing cross-collection approaches to accommodate these asymmetries and implement an efficient parallel sampling algorithm enabling users to examine the long tail of topics in particularly large collections.

Using topic models for comparative text mining was introduced by Zhai et al. (2004), who developed the ccMix model which extended pLSA (Hofmann, 1999). Later work by Paul and Girju (2009) developed ccLDA, which adopted the hierarchical Bayes framework of Latent Dirichlet Allocation or LDA (Blei et al., 2003). These models account for topic-word asymmetry by assuming variation in the vocabularies of topics is due to collection-level differences. Nevertheless, they require the same topics to be present in each collection. These models are useful for comparing collections under specific assumptions, but cannot accommodate collection-topic asymmetry (which

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1554–1564,
Lisbon, Portugal, 17-21 September 2015.

arises in collections that do not share every topic or that have different numbers of topics). In situations where collections do not share all topics, the results often include junk, mixed, or sparse topics, making them difficult to interpret (Paul and Girju, 2009). Such asymmetries make it difficult to use models like ccLDA and ccMix when little is known about collections in advance. This motivates our efforts to model variation in the long tail of topic distributions, where correlations are more likely to appear when collections are weakly related.

C-LDA and C-HDP extend ccLDA (Paul and Girju, 2009) to accommodate collection-topic level asymmetries, particularly by allowing non-common topics to appear in each collection. This added flexibility allows our models to discover topic correlations across arbitrary collections with different numbers of topics, even when there are few (or unknown) numbers of common topics. To demonstrate the effectiveness of our models, we evaluate them on synthetic data and show that they outperform related models such as ccLDA and differential topic models (Chen et al., 2014). We then fit C-LDA to two large collections of humanities and sciences documents from JSTOR. Such historical analyses of text would be intractable without an efficient sampler. An optimized sampler is required in such situations because common topics in weakly-correlated collections are usually found in the tail of the document-topic distribution of a sufficiently large set of topics. To make this feasible on large datasets such as JSTOR, we employ a parallelized Metropolis-Hastings (Kronmal and Peterson Jr, 1979) and alias-table sampling framework, adapted from LightLDA (Yuan et al., 2015). These optimizations, which achieve $\mathcal{O}(1)$ amortized sampling time per token, allow our models to be fit to large corpora with up to thousands of topics in a matter of hours — an order of magnitude speed-up from ccLDA.

After reviewing work related to topic modeling across collections, section 3 describes C-LDA and C-HDP, and then details their technical relationship to existing models. Section 5 introduces the synthetic data and part of the JSTOR corpus used in our evaluations. We then compare our models' performances to other models in terms of held-out perplexity and a measure of distinguishability. The final results section exemplifies the use of C-LDA in a qualitative analysis of humanities and sciences research. We conclude with a brief discussion of the strengths of C-LDA and C-HDP, and outline directions for future work and applications.

2 Related Work

Our models seek to enable users to compare large collections that may only be weakly correlated and that may contain different numbers of topics. While topic models could be fit to separate collections to make post-hoc comparisons (Denny et al., 2014; Yang et al., 2011), our goal is to account for both document-topic asymmetry and topic-word asymmetry "in-model". In short, we seek to model the correlation between *arbitrary* collections. Prioritizing in-model solutions for document-topic asymmetry has been explored elsewhere, such as in hierarchical Dirichlet processes (HDP), which use an additional level to account for collection variations in document-topic distributions (Teh et al., 2006).

One method designed to model topic-word asymmetry is ccMix (Zhai et al., 2004), which models the generative probability of a word in topic z from collection c as a mixture of shared and collection-specific distributions θ_z:

$$p(w) = \lambda_c \, p(w|\theta_z) + (1 - \lambda_c) \, p(w|\theta_{z,c})$$

where $\theta_{z,c}$ is collection-specific and λ_c controls the mixing between shared and collection-specific topics. ccLDA extends ccMix to the LDA framework and adds a beta prior over λ_c that reduces sensitivity to input parameters (Paul and Girju, 2009). Another approach, differential topic models (Chen et al., 2014), is based on hierarchical Bayesian models over topic-word distributions. This method uses the transformed Pitman-Yor process (TPYP) to model topic-word distributions in each collection, with shared common base measures. As (Paul and Girju, 2009) note, ccLDA cannot accommodate a topic if it is not common across collections — an assumption made by ccMix, ccLDA and the TPYP. In a situation where a topic is found in only one collection, it would either dominate the shared topic portion (resulting in a noisy, collection-specific portion), or it would appear as a mixed topic, revealing two sets of unrelated words (Newman et al., 2010b). C-LDA ameliorates this situation by allowing the number of common and non-common topics to be specified separately and by efficiently sampling the tail

of the document-topic distribution, allowing users to examine less prominent regions of the topic space. C-HDP also grants collections document-topic independence using a hierarchical structure to model the differences between collections.

Due to increased demand for scalable topic model implementations, there has been a proliferation of optimized methods for efficient inference, such as SparseLDA (Yao et al., 2009) and AliasLDA (Li et al., 2014). AliasLDA achieves $\mathcal{O}(K_d)$ complexity by using the Metropolis-Hastings-Walker algorithm and an alias table to sample topic-word distributions in $\mathcal{O}(1)$ time. Although this strategy introduces temporal staleness in the updates of sufficient statistics, the lag is overcome by more iterations, and converges significantly faster. A similar technique by Yuan et al. (2015), LightLDA, employs cycle-based Metropolis Hastings mixing with alias tables for both document-topic and topic-word distributions. Despite introducing lag in the sufficient statistics, this method achieves $\mathcal{O}(1)$ amortized sampling complexity and results in even faster convergence than AliasLDA. In addition to being fully parallelized, C-LDA adopts this sampling framework to make comparing large collections more tractable for large numbers of topics. Our models' efficient sampling methods allow users to fit large numbers of topics to big datasets where variation might not be observed in sub-sampled datasets or models with fewer topics.

3 The Models

3.1 Correlated LDA

In ccLDA (and ccMix), each topic has shared and collection-specific components for each collection. C-LDA extends ccLDA to make it more robust with respect to topic asymmetries between collections (Figure 1a). The crucial extension is that by allowing each collection to define a set of non-common topics in addition to common topics, the model removes an assumption imposed by ccLDA and other inter-collection models, namely that collections have the same number of topics. As a result, C-LDA is suitable for collections without a large proportion of common topics, and can also reduce noise (discussed in Section 2). To achieve this, C-LDA assumes document d in collection c has a multinomial document-topic distribution θ with an asymmetric Dirichlet prior for K_c topics, where the first $K^{\emptyset}$ are common across collections. It is also possible to introduce a tree structure into the model that uses a binomial distribution to decide whether a word was drawn from common or non-common topics. This yields collection-specific background topics by using a binomial distribution instead of a multinomial. However, we prefer the simpler, non-tree version because background topics are unnecessary when using an asymmetric α prior (Wallach et al., 2009a).

The generative process for C-LDA is as follows:

1. Sample a distribution ϕ_k (shared component) from Dir(β) and a distribution σ_k from Beta(δ_1, δ_2) for each common topic $k \in \{1, \ldots, K^{\emptyset}\}$;
2. For each collection c, sample a distribution ϕ_k^c (collection-specific component) from Dir(β) for each common topic $k \in \{1, \ldots, K^{\emptyset}\}$ and non-common topic $k \in \{K^{\emptyset} + 1, \ldots, K_c\}$;
3. For each document d in c, sample a distribution θ from Dir(α_c);
4. For each word w_i in d:
   (a) Sample a topic $z_i \in \{1, \ldots, K_c\}$ from Multi(θ);
   (b) If $z_i \leq K^{\emptyset}$, sample y_i from Binomial(σ_{z_i});
   (c) Sample w_i from Multi($\phi_{z_i}^{\xi}$), where
   $$\xi = \begin{cases} \texttt{null} & , z_i \leq K^{\emptyset} \text{ and } y_i = 0; \\ c & , \text{otherwise.} \end{cases}$$

Note that to capture common topics, $K^{\emptyset}$ should be set such that $\exists\ c$ where $K_c = K^{\emptyset}$. Otherwise, words sampled as a non-common topic will not have information about non-common topics in other collections. Then a "common-topic word" is found among non-common topics in all collections (a local minima) and it will take a long time to stabilize as a common topic. To avoid this, when determining the number of topics for sampling, the number of non-common topics for the collection with the smallest number of total topics should be zero. After inference, to distinguish common and non-common topics in this collection, we model σ independently by assuming collections have the same mixing ratio for common topics. With this reasonable assumption and an asymmetric α, common topics become sparse enough that some σ distributions reduce nearly to 0, distinguishing them as non-common topics. Although this may seem counterintuitive, it does not negatively affect results.

Three kinds of collection-level imbalance can confound inter-collection topic models: 1) in the numbers of topics between collections, 2) in the numbers of documents between collections, and 3) in the document-topic distributions. Each of

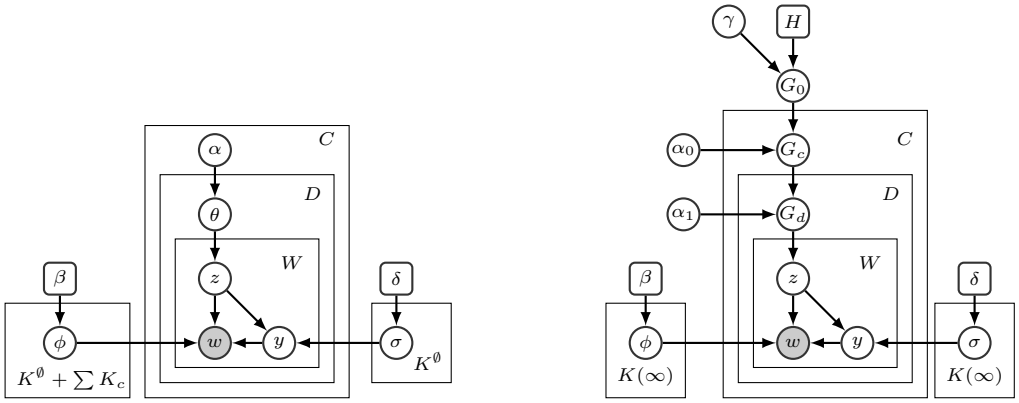


Figure 1: Graphical models of C-LDA (a; left) and C-HDP (b; right).

these can cause topics in different collections to have significantly different numbers of words assigned to the same topic. In this way, a topic can be dominated by the collection comprising most of its words. C-LDA addresses imbalances in the document-topic distributions between collections by estimating α. For imbalance in the number of topics and documents, C-LDA mimics document over-sampling in the Gibbs sampler using a different unit-value in the word count table for each collection. Specifically, a unit η_c is chosen for each collection such that the average equivalent number of assigned words per-topic ($\sum_{d \in c} \eta_c N_d / K_c$, where N_d is the length of document d) is equal. This process both increases the topic quality (in terms of collection balance) in the resulting held-out perplexity of the model.

3.2 Correlated HDP

To alleviate C-LDA's requirement that $\exists\ c$ such that $K_c = K^{\emptyset}$, we introduce a variant of the model, the correlated hierarchical Dirichlet process (C-HDP), that uses a 3-level hierarchical Dirichlet process (Teh et al., 2006). The generative process for C-HDP is the same as C-LDA shown above, except that here we assume a word's topic, z, is generated by a hierarchical Dirichlet process:

$$\begin{aligned} G_0|\gamma, H &\sim \text{DP}(\gamma, H) \\ G_c|\alpha_0, G_0 &\sim \text{DP}(\alpha_0, G_0) \\ G_d|\alpha_1, G_c &\sim \text{DP}(\alpha_1, G_c) \\ z|G_d &\sim G_d \end{aligned}$$

where G_0 is a base measure for each collection-level Dirichlet process, and G_c are base measures of document-level Dirichlet processes in each collection (Figure 1b). Thus, documents from the same collection will have similar topic distributions compared to those from other collections, and collections are allowed to have distinct sets of topics due to the use of HDP.

4 Inference

4.1 Posterior Inference in C-LDA

C-LDA can be trained using collapsed Gibbs sampling with ϕ, θ, and σ integrated out. Given the status assignments of other words, the sampling distribution for word w_i is given by:

$$\begin{aligned} & p(y_i, z_i | \boldsymbol{w}, \boldsymbol{y}_{-i}, \boldsymbol{z}_{-i}, \delta, \alpha, \beta) \\ \propto\ & \underbrace{(N(d, z_i) + \alpha_{c,z_i})}_{q_d} \\ & \times \underbrace{\begin{cases} \frac{N(y_i, z_i) + \delta_{y_i}}{N(z_i) + \sum_k \delta_k} \times \frac{N(w_i, y_i, z_i, \zeta) + \beta}{N(y_i, z_i, \zeta) + V\beta} & z_i \le K^{\emptyset} \\ \frac{N(w_i, z_i, c) + \beta}{N(z_i, c) + V\beta} & z_i > K^{\emptyset} \end{cases}}_{q_w} \end{aligned} \tag{1}$$

where $\zeta = \begin{cases} * & y_i = 0 \\ c & y_i = 1 \end{cases}$, $N(\cdots)$ is the number of status assignments for $(\cdots)$, not including w_i.

Inference in C-LDA employs two optimizations: a parallelized sampler and an efficient sampling algorithm (Algorithm 1). We use the parallel schema in (Smola and Narayanamurthy, 2010; Lu et al., 2013) which applies atomic updates to the sufficient statistics to avoid race conditions. The key idea behind the optimized sampler is the combination of alias tables and the Metropolis-Hastings method (MH), adapted from (Yuan et al., 2015; Li et al., 2014). Metropolis-Hastings is a Markov chain Monte Carlo method that uses a proposal distribution to approximate the true distribu-

Algorithm 1 Sampling in C-LDA

repeat
  for all documents $\{d\}$ in parallel **do**
    for words $\{w\}$ in d **do**
      $z \leftarrow \text{CYCLEMH}(p, q_w, q_d, z)$
      sample y given z
    Atomic update sufficient statics
  Estimate α
until convergence

procedure $\text{CYCLEMH}(p, q_w, q_d, z)$
  for $i = 1$ **to** N **do**
    if i is even **then**
      proposal $q \leftarrow q_w$
    else
      proposal $q \leftarrow q_d$
    sample $z' \sim \text{ALIASTABLE}(q)$
    if RandUnif(1) $< \min(1, \frac{p(z')q(z)}{p(z)q(z')})$ **then**
      $z \leftarrow z'$
  return z

tion when exact sampling is difficult. In a complimentary way, Walker's alias method (2004) allows one to effectively sample from a discrete distribution by using an alias table, constructed in $\mathcal{O}(K)$ time, from which we can sample in $\mathcal{O}(1)$. Thus, reusing the sampler K times as the proposal distribution for Metropolis-Hastings yields $\mathcal{O}(1)$ amortized sampling time per-token.

Notice that in Eq. 1, the sampling distribution is the product of a single document-dependent term q_d and a single word-dependent term q_w. After burn-in, both terms will be sparse (without the smoothing factor). It is therefore reasonable to use q_d and q_w as cycle proposals (Yuan et al., 2015), alternating them in each Metropolis-Hastings step. Our experiments show that the primary drawback of this method — stale sufficient statistics — does not empirically affect convergence. Our implementation uses proposal distributions q_w and q_d, with y marginalized out. After the Metropolis-Hastings steps, y is sampled to update z, to reduce the size of the alias tables, yielding even faster convergence.

Lastly, the use of an asymmetric α allows C-LDA to discover correlations between less dominant topics across collections (Wallach et al., 2009a). We use Minka's fixed-point method, with a gamma hyper-prior to optimize α_c for each collection separately (Wallach, 2008). All other hyperparameters were fixed during inference.

4.2 Posterior Inference in C-HDP

C-HDP uses the block sampling algorithm described in (Chen et al., 2011), which is based on the Chinese restaurant process metaphor. Here, rather than tracking all assignments (as the samplers given in (Teh et al., 2006)), table indicators are used to track only the start of new tables, which allows us to adopt the same sampling framework as C-LDA. In the Chinese restaurant process, each Dirichlet process in the hierarchical structure is represented as a restaurant with an infinite number of tables, each serving the same dish. New customers can either join a table with existing customers, or start a new table. If a new table is chosen, a proxy customer will be sent to the parent restaurant to determine the dish served to that table.

In the block sampler, indicators are used to denote a customer creating a table (or tables) up to level u (0 as the root, 1 for collection level, and 2 for the document level), and $u = \emptyset$ indicates no table has been created. For example, when a customer creates a table at the collection level, and the proxy customer in the collection level creates a table at the root level, u is 0. With this metaphor, let n_{lz} be the number of customers (including their proxies) served dish z at restaurant l, and let t_{lz} be the number of tables serving dish z at restaurant l ($l = 0$ for root, $l = c$ for collection level or $l = d$ for document level), with $N_0 = \sum_z n_{0z}$ and $N_c = \sum_z n_{cz}$. By the chain rule, the conditional probability of the state assignments for w_i, given all others, is

$$
\begin{aligned}
&p(y_i, z_i, u_i | \boldsymbol{w}, \boldsymbol{y}_{-i}, \boldsymbol{z}_{-i}, \boldsymbol{u}_{-i}, \ldots) \\
\propto &\frac{N(y,z) + \delta_y}{N(z) + \sum_k \delta_k} \times \frac{N(w,y,z,\zeta) + \beta}{N(y,z,\zeta) + V\beta} \\
\times &\begin{cases}
\frac{\gamma\alpha_0}{\gamma + N_0} & u = 0 \\
\frac{\alpha_0}{\gamma+N_0} \frac{S^{n_{cz}+1}_{t_{cz}+1}}{S^{n_{cz}}_{t_{cz}}} \frac{S^{n_{dz}+1}_{t_{dz}+1}}{S^{n_{dz}}_{t_{dz}}} \frac{n^2_{0z}(t_{cz}+1)(t_{dz}+1)}{(n_{0z}+1)(n_{cz}+1)(n_{dz}+1)} & u = 1 \\
\frac{S^{n_{cz}+1}_{t_{cz}}}{S^{n_{cz}}_{t_{cz}}} \frac{S^{n_{dz}+1}_{t_{dz}+1}}{S^{n_{dz}}_{t_{dz}}} \frac{(t_{dz}+1)(n_{cz}-t_{cz}+1)}{(n_{cz}+1)(n_{dz}+1)} & u = 2 \\
\frac{\alpha_0+N_1}{\alpha_1} \frac{S^{n_{dz}+1}_{t_{dz}}}{S^{n_{dz}}_{t_{dz}}} \frac{n_{dz}-t_{dz}+1}{n_{dz}+1} & u = \emptyset
\end{cases}
\end{aligned}
$$

Here, S^n_t is the Stirling number, the ratios of which can be efficiently precomputed (Buntine and Hutter, 2010). The concentration parameters γ, α_0, and α_1 can be sampled using the auxiliary variable method (Teh et al., 2006).

Note that because conditional probability has the same separability as C-LDA (to give term q_w and q_d), the same sampling framework can be used with two alterations: 1) when a new topic is created or removed at the root, collection, or document level, the related alias tables must be reset, which makes the sampling slightly slower

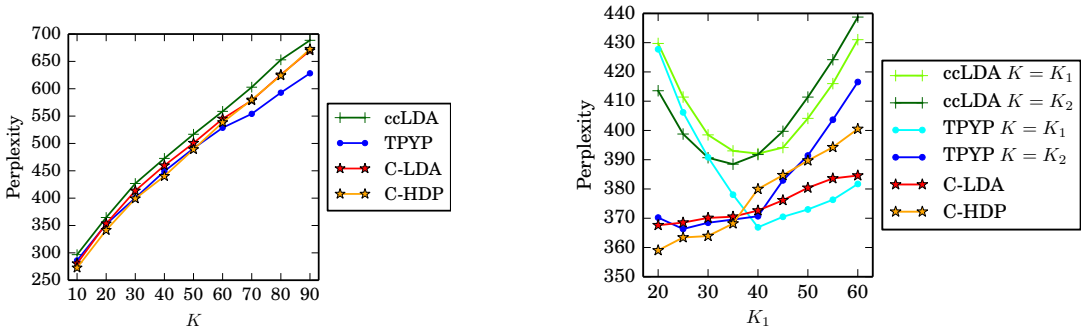


Figure 2: Held-out perplexity of C-LDA, C-HDP, ccLDA and TPYP fit to synthetic data, where $K_1 = K_2 = K$ (a; left) and data with an asymmetric number of topics (b; right).

than $\mathcal{O}(1)$, and 2) while the document alias table samples z and u simultaneously, after sampling z from the word alias table u must be sampled using t_{lc}/n_{lz} (Chen et al., 2011). Parallelizing C-HDP requires an additional empirical method of merging new topics between threads (Newman et al., 2009), which is outside of the scope of this work. Our implementation of both models, C-LDA and C-HDP, are open-sourced online [1].

5 Experiments

5.1 Model Comparison

We use perplexity on held-out documents to evaluate the performance of C-LDA and C-HDP. In all experiments, the gamma prior for α in C-LDA was set to $(1, 1)$, and $(5, 0.1)$, $(5, 0.1)$, $(0.1, 0.1)$ for γ, α_0, α_1 respectively in C-HDP. In the hold-out procedure, 20% of documents were randomly selected as test data. LDA, C-LDA and ccLDA were run for 1,000 iterations and C-HDP and the TII-variant of TPYP for 1,500 iterations (unless otherwise noted), all of which converged to a state where change in perplexity was less than 1% for ten consecutive iterations.

Perplexity was calculated from the marginal likelihood of a held-out document $p(\mathbf{w}|\mathbf{\Phi}, \alpha)$, estimated using the "left-to-right" method (Wallach et al., 2009b). Because it is difficult to validate real-world data that exhibits different kinds of asymmetry, we use synthetic data generated specifically for our evaluation tasks (AlSumait et al., 2009; Wallach et al., 2009b; Kucukelbir and Blei, 2014).

5.1.1 Topic Correlation

C-LDA is unique in the amount of freedom it allows when setting the number of topics for collections. To assess the models' performances with various topic correlations in a fair setting, we generated two collections of synthetic data by following the generative process (varying the number of topics) and measured the models' perplexities against the ground truth parameters. In each experiment, two collections were generated, each with 1,000 documents containing 50 words each, over a vocabulary of 3,000. β and δ were fixed at 0.01 and 1.0 respectively, and α was asymmetrically defined as $1/(i + \sqrt{K_c})$ for $i \in [0, K_c - 1]$.

Completely shared topics The assumptions imposed by ccLDA and TPYP effectively make them a special case of our model where $K^{\emptyset} = K_1 = K_2 = \ldots$. To compare results, data was generated such that all numbers of topics were equal to $K \in [10, 90]$. Additionally, all models were configured to use this ground truth parameter when training. Not surprisingly, ccLDA, C-LDA, and C-HDP have almost the same perplexity with respect to K because their structure is the same when all topics are shared (Figure 2a).

Asymmetric numbers of topics To explore the effect of asymmetry in the number of topics, data was generated such that one collection had $K_1 \in [20, 60]$ topics while a second had a fixed $K_2 = 40$ topics. The number of shared topics was set to $K^{\emptyset} = 20$. The parameters for C-LDA and C-HDP (initial values) were set to ground truths, and, to retain a fair comparison, versions of ccLDA and TPYP were fit with both $K = K_1$ and $K = K_2$.

We find that ccLDA performs nearly as well as C-LDA and C-HDP when there is more symmetry between collection, namely when $K_1 \approx K_2$ (Figure 2b). TPYP, on the other hand, performs well with more topics ($2 \times \max(K_1, K_2)$ where the ground truth is K_1 & K_2). In contrast, C-LDA

[1] https://github.com/iceboal/correlated-lda

and C-HDP perform more consistently than other models across varying degrees of asymmetry.

Partially-shared topics When collections have the same number of topics, C-LDA, C-HDP and ccLDA exhibit adequate flexibility, resulting in similar perplexities. When collections have increasingly few common topics, however, common and non-common topics from ccLDA are considerably less distinguishable than those from C-LDA. To evaluate the models' abilities in such situations, data was generated for two collections having $K_1 = K_2 = 50$ topics, but with the shared number of topics $K^{\emptyset} \in [5, 45]$. We also set $\delta^{(0)} = \delta^{(1)} = 5$, and for comparison to ccLDA we used $K = 50$.

To measure this distinguishability, we examine the inferred σ. Recall that σ indicates what percentage of a common topic is shared. When a topic is actually non-common, the value of σ should be small. We sort σ_k for $k \in [1, K]$ in reverse and use

$$\begin{aligned} \bar{\sigma}_{\text{common}} &= \frac{1}{K^{\emptyset}} \sum_{k=1}^{K^{\emptyset}} \sigma_k \\ \bar{\sigma}_{\text{non-common}} &= \frac{1}{K - K^{\emptyset}} \sum_{k=K^{\emptyset}+1}^{K} \sigma_k \end{aligned} \tag{2}$$

as measures of how well common and non-common topics were learned[2]. $\bar{\sigma}_{\text{common}}$ is the average of the $K^{\emptyset}$ largest σ values, and $\bar{\sigma}_{\text{non-common}}$ is the average of the rest. When $\delta^{(0)} = \delta^{(1)}$ in the synthetic data, σ in the common portion should be 0.5, whereas it should be 0 in the non-common part. Figure 3 shows that C-LDA better distinguishes between common and non-common topics, especially when $K^{\emptyset}$ is small. This allows non-common topics to be separated from the results by examining the value of σ. C-HDP has similar performance but larger σ values. In ccLDA, all topics are shared between collections which means that common and non-common topics are mixed. As expected, ccLDA performs similarly when all topics are common across collections.

5.2 Semantic Coherence

Semantic coherence is a corpus-based metric of the quality of a topic, defined as the average pairwise similarity of the top n words (Newman et al., 2010a; Mimno et al., 2011). A PMI-based form of coherence, which has been found to be the best

[2]TPYP is not comparable using this metric, but its hierarchical structure will cause topics to mix naturally.

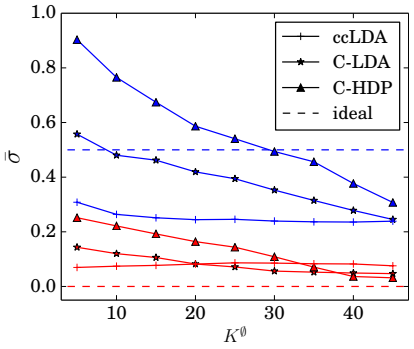


Figure 3: Distinguishability (Eq. 2) of topics fit with C-LDA,C-HDP and ccLDA. Blues lines denote $\bar{\sigma}_{\text{common}}$ and red denote $\bar{\sigma}_{\text{non-common}}$.

proxy human judgements of topic quality, is defined for a topic k as:

$$\text{C}(k) = \frac{2}{n(n-1)} \sum_{\substack{(w_i, w_j) \in k \\ i<j}}^{n} \log \frac{D(w_i, w_j) + 1}{D(w_i) D(w_j)}$$

where $D(\cdot)$ computes the document co-occurrence. To accommodate coherence with common topics in C-LDA that have shared and collection-specific components we define *mutual coherence*, $\text{MC}(k)$, as

$$\text{MC}(k) = \frac{1}{n^2} \sum_{\substack{w_i \in \text{shared}, \\ w_j \in \text{collection-specific}}}^{n} \log \frac{D(w_i, w_j) + 1}{D(w_i) D(w_j)}$$

so that for each collection, $\text{C}(k)$ ($2n$ words) is equal to $\text{C}(k, \text{shared})$ + $\text{C}(k, \text{collection-specific})$ + $\text{MC}(k)$. Table 1 shows the semantic coherence of topics fit with ccLDA and C-LDA. We used a 10% sample of JSTOR due to the limited speed of ccLDA, using 50 (common) topics for ccLDA / C-LDA, and 250 non-common humanities topics for C-LDA. Although these settings are different for the models, the science topics are still comparable because they both have 50 topics. We found that C-LDA provides improved coherence in nearly all situations.

5.2.1 Inference Efficiency

To compare the model efficiency, we timed runs on a sample of 5,036 documents from JSTOR (introduced in the next section) with a 20% held-out and set $K = K_1 = K_2 = 200$ run on a commodity computer with four cores and 16GB of memory. Figure 4a shows the perplexity over

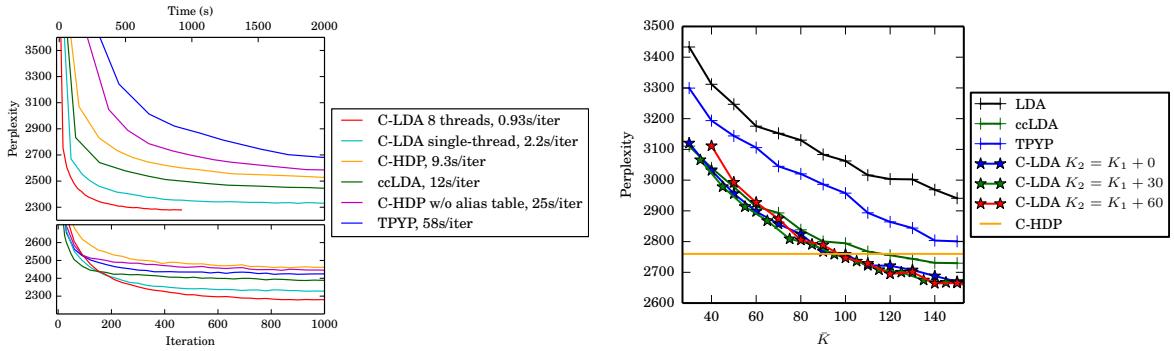


Figure 4: Using JSTOR: perplexity vs. runtime and iterations (a; left) and perplexity vs. K (b; right).

	Coherence					**Mutual Coherence**	
	shared component			collection-specific		shared & collection-specific	
	all documents	science	humanities	science	humanities	science	humanities
C-LDA	-8.83	-7.73	-8.04	-8.38	-8.14	-8.54	-8.37
ccLDA	-9.04	-8.22	-8.27	-8.38	-8.15	-8.69	-8.40
C-LDA	-7.22	-3.68	-6.11	-8.25	-8.09	-7.75	-7.97
ccLDA	-8.11	-5.68	-7.12	-8.24	-7.88	-8.22	-7.95

Table 1: Average semantic coherence of the 50 common topics from JSTOR (top) and the average of the 10 best common topics judged by the mean value of different types of coherence (bottom).

time and iterations. The inference algorithm introduces some staleness, which yields slower convergence in the first 200 iterations. This effect, however, is outweighed in both C-LDA and C-HDP by the increased sampling speed. With 8 threads, C-LDA not only converges faster, but yields lower perplexity, likely due to threads introducing additional stochasticity.

5.3 Performance on JSTOR

To compare our models against slower models, we sampled 2,465 documents from JSTOR, withholding 20% as testing set. We fit a model with 100 common and 50 non-common initial topics using C-HDP, which produced 272 root topics after 2,000 iterations.The perplexity scores are roughly the same when C-LDA uses the same average number of topics per collection (Figure 4b), except when numbers of topics are very asymmetric. Our model begins to outperform ccLDA after 80 topics. C-HDP did not, however, out-perform C-LDA despite the original HDP outperforming LDA. This could be do to the fact that the hierarchical structure of C-HDP is considerably different than the typical 2-level HDP. Held-out perplexity on real data provides a quantitative evaluation of our models' performance in a real-world setting. However, the goal of our models is to enable a deeper analysis of large, weakly-related corpora, which we next discuss.

5.4 Qualitative Analysis

Our models are designed to enable researchers to compare collections of text in a way that is scalable and sensitive to collection-level asymmetries. To demonstrate that C-LDA can fill this role, we fit a model to the entire JSTOR sciences and humanities collections with 100 science topics and 1000 humanities topics (to reveal the less popular science-related topics in the humanities), and $\beta = 0.01, \delta = 1.0$. JSTOR includes books and journal publications in over 9 million documents across nearly 3 thousand journals. We used the journal *Science* to represent a collection of scientific research and 76 humanist journals to represent humanities research[3]. Words were lemmatized, and the most and least frequent words discarded. The final humanities collection contained 149,734 documents and the sciences collection had 160,680 documents, with a combined vocabulary of 21,513 unique words. Together, these collections typify a real-world situation where there is likely some, but not overwhelming correlation.

The results indicate that the sciences and humanities share several topics. Both exhibit an interest in a "non-human" theme (common topic #2; Table 2). This topic is quite similar in both collections (*pig* and *monkey* for science documents; *bird* and *gorilla* for humanities documents), while their shared component forms a cohesive topic (*animal*,

[3]The list is available at http://j.mp/humanities-txt.

Topic 2			Topic 21			Topic 23		
shared	**science**	**humanities**	**shared**	**science**	**humanities**	**shared**	**science**	**humanities**
animal	pig	beast	economic	cost	rural	particle	energy	universe
specie	fly	creature	government	industry	local	physic	electron	quantum
dog	monkey	nonhuman	economy	company	community	physicist	ray	physic
wild	guinea	natural	trade	price	village	energy	ion	technical
wolf	primate	humanity	major	market	region	experiment	atom	scientific
monkey	worm	bird	growth	product	urban	event	particle	relativity
horse	dog	living	capital	income	country	measurement	mass	physical
sheep	cat	gorilla	industry	industrial	area	atom	neutron	mechanic
lion	mammal	brute	institution	business	regional	interaction	proton	law
cat	cattle	ape	support	private	population	atomic	nucleus	reality

Table 2: Three topics from the JSTOR collections with their top words in shared and specific components. Complete results available at http://j.mp/jstor-html.

specie, and *monkey*). This kind of correlation is also evident in topic #23, about physics. While the science documents clearly represent research in particle physics, it is interesting to find the topic is also represented by humanist research focused on cultural representations of science. This reflects a growing interest in science and technology studies that has gained recent traction in the humanities. Despite their differences, both collections engage with a similar theme, seen in the shared component with words like particle, *energy* and *atom*.

The results also indicate that while sciences and humanities documents can share themes, they often diverge in how they are discussed. For example, common topic #21 could be identified as *economic* or *capitalist*, but in the collection-specific components, the two disciplines differ in their articulatation. Science uses terms like *price* and *market*, indicating an acceptance of free-market capitalism (especially as it affects the practice of science), while the humanities, which has long been critical of free-market capitalism, uses terms like *rural* and *community*, highlighting cultural facets of modern economics. These results provide evidence about how ideas move between the sciences and humanities — a phenomenon that constitutes a growing area of research for historians (Galison, 2003; Canales, 2015). C-LDA provides empirical, measurable, and reproducible evidence of the shared research between these disciplines, as well as how concepts are articulated.

6 Discussion

Our models provide a robust way to explore large and potentially weakly-related text collections without imposing assumptions about the data. Like ccLDA and TPYP, our models account for topic-word variation at the collection level. The models accommodate asymmetry in the numbers of topics (set in C-LDA, fit in C-HDP) and provide an efficient inference method which allows them to fit data with large values for K, which can help find correlations in less prevalent topics. Our primary contribution is our models' ability to accommodate asymmetries between arbitrary collections. JSTOR, the world's largest digital collection of humanities research, was an ideal application setting given the size, asymmetry, and comprehensiveness of the humanities collection. As we show, humanities and science research exhibit asymmetries with regard to vocabulary and topic structure — asymmetries that would be systematically overlooked using existing models. By characterizing common topics as mixtures of shared and collection-specific components, we can capture a kind of topic-level homophily, where similar themes are articulated in different ways due to word-, document-, and collection-level variation. Future work on these models could explore methods to fit non-common topics for both collections. In general, C-LDA and C-HDP can be used whenever documents are sampled from ostensibly different populations, where the nature of the difference is unknown.

Acknowledgements

Thanks to David Blei for advice on applications of the model. This work contains analysis of private, or otherwise restricted data, made available to James Evans and Eamon Duede by ITHAKA (JSTOR), the opinions of whom are not represented in this paper. Jaan Altosaar acknowledges support from the Natural Sciences and Engineering Research Council of Canada. This work was supported by a grant from the Templeton Foundation to the Metaknowledge Research Network and by grant #1158803 from the National Science Foundation.

References

Loulwah AlSumait, Daniel Barbará, James Gentle, and Carlotta Domeniconi. 2009. Topic significance ranking of LDA generative models. In *Machine Learning and Knowledge Discovery in Databases*, pages 67–82. Springer.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Wray Buntine and Marcus Hutter. 2010. A Bayesian view of the Poisson-Dirichlet process. *arXiv preprint arXiv:1007.0296*.

Jimena Canales. 2015. *The Physicist and the Philosopher: Einstein, Bergson, and the Debate that Changed our Understanding of Time*. Princeton University Press, Princeton, NJ.

Changyou Chen, Lan Du, and Wray Buntine. 2011. Sampling table configurations for the hierarchical Poisson-Dirichlet process. In *Machine Learning and Knowledge Discovery in Databases*, pages 296–311. Springer.

Changyou Chen, Wray Buntine, Nan Ding, Lexing Xie, and Lan Du. 2014. Differential topic models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

M. Denny, J. ben Aaron, H. Wallach, and B. Desmarais. 2014. Modeling email network content and structure. In *the 72nd Annual Midwest Political Science Association Conference, 2014; the Northeast Political Methodology Meeting, 2014; the 7th Annual Political Networks Conference, the Society for Political Methodology 31st Annual Summer Meeting*.

Peter Galison. 2003. *Poincar's Maps*. Norton, New York, NY.

Thomas Hofmann. 1999. Probabilistic latent semantic indexing. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 50–57.

Richard A Kronmal and Arthur V Peterson Jr. 1979. On the alias method for generating random variables from a discrete distribution. *The American Statistician*, 33(4):214–218.

Alp Kucukelbir and David M Blei. 2014. Profile predictive inference. *arXiv preprint arXiv:1411.0292*.

Aaron Q Li, Amr Ahmed, Sujith Ravi, and Alexander J Smola. 2014. Reducing the sampling complexity of topic models. In *Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 891–900.

Mian Lu, Ge Bai, Qiong Luo, Jie Tang, and Jiuxin Zhao. 2013. Accelerating topic model training on a single machine. In *Web Technologies and Applications*, pages 184–195. Springer.

George Marsaglia, Wai Wan Tsang, and Jingbo Wang. 2004. Fast generation of discrete random variables. *Journal of Statistical Software*, 11:1–8.

David Mimno, Hanna M Wallach, Edmund Talley, Miriam Leenders, and Andrew McCallum. 2011. Optimizing semantic coherence in topic models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 262–272.

David Newman, Arthur Asuncion, Padhraic Smyth, and Max Welling. 2009. Distributed algorithms for topic models. *The Journal of Machine Learning Research*, 10:1801–1828.

David Newman, Jey Han Lau, Karl Grieser, and Timothy Baldwin. 2010a. Automatic evaluation of topic coherence. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 100–108.

David Newman, Youn Noh, Edmund Talley, Sarvnaz Karimi, and Timothy Baldwin. 2010b. Evaluating topic models for digital libraries. In *Proceedings of the 10th Annual Joint Conference on Digital Libraries*, JCDL '10, pages 215–224. ACM.

Michael Paul and Roxana Girju. 2009. Cross-cultural analysis of blogs and forums with mixed-collection topic models. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 3*, pages 1408–1417.

Alexander Smola and Shravan Narayanamurthy. 2010. An architecture for parallel topic models. *Proceedings of the VLDB Endowment*, 3(1-2):703–710.

Yee Whye Teh, Michael I Jordan, Matthew J Beal, and David M Blei. 2006. Hierarchical Dirichlet processes. *Journal of the american statistical association*, 101(476).

Hanna M Wallach, David Mimno, and Andrew Mccallum. 2009a. Rethinking LDA: Why priors matter. In *Advances in Neural Information Processing Systems*, pages 1973–1981.

Hanna M. Wallach, Iain Murray, Ruslan Salakhutdinov, and David Mimno. 2009b. Evaluation methods for topic models. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 1105–1112, New York, NY, USA. ACM.

Hanna M Wallach. 2008. Structured topic models for language. *Unpublished doctoral dissertation, Univ. of Cambridge*.

Tze-I Yang, Andrew J Torget, and Rada Mihalcea. 2011. Topic modeling on historical newspapers. In *Proceedings of the 5th ACL-HLT Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities*, pages 96–104.

Limin Yao, David Mimno, and Andrew McCallum. 2009. Efficient methods for topic model inference on streaming document collections. In *Proceedings of the 15th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 937–946.

Jinhui Yuan, Fei Gao, Qirong Ho, Wei Dai, Jinliang Wei, Xun Zheng, Eric Po Xing, Tie-Yan Liu, and Wei-Ying Ma. 2015. Lightlda: Big topic models on modest computer clusters. In *Proceedings of the 24th International Conference on World Wide Web*, pages 1351–1361. International World Wide Web Conferences Steering Committee.

ChengXiang Zhai, Atulya Velivelli, and Bei Yu. 2004. A cross-collection mixture model for comparative text mining. In *Proceedings of the tenth ACM SIGKDD international conference on knowledge discovery and data mining*, pages 743–748.

Molding CNNs for text: non-linear, non-consecutive convolutions

Tao Lei, Regina Barzilay, and Tommi Jaakkola
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
{taolei, regina, tommi}@csail.mit.edu

Abstract

The success of deep learning often derives from well-chosen operational building blocks. In this work, we revise the temporal convolution operation in CNNs to better adapt it to text processing. Instead of concatenating word representations, we appeal to tensor algebra and use low-rank n-gram tensors to directly exploit interactions between words already at the convolution stage. Moreover, we extend the n-gram convolution to non-consecutive words to recognize patterns with intervening words. Through a combination of low-rank tensors, and pattern weighting, we can efficiently evaluate the resulting convolution operation via dynamic programming. We test the resulting architecture on standard sentiment classification and news categorization tasks. Our model achieves state-of-the-art performance both in terms of accuracy and training speed. For instance, we obtain 51.2% accuracy on the fine-grained sentiment classification task.[1]

1 Introduction

Deep learning methods and convolutional neural networks (CNNs) among them have become de facto top performing techniques across a range of NLP tasks such as sentiment classification, question-answering, and semantic parsing. As methods, they require only limited domain knowledge to reach respectable performance with increasing data and computation, yet permit easy architectural and operational variations so as to fine tune them to specific applications to reach top performance. Indeed, their success is often contingent on specific architectural and operational choices.

[1] Our code and data are available at https://github.com/taolei87/text_convnet

CNNs for text applications make use of temporal convolution operators or filters. Similar to image processing, they are applied at multiple resolutions, interspersed with non-linearities and pooling. The convolution operation itself is a *linear* mapping over "n-gram vectors" obtained by concatenating *consecutive* word (or character) representations. We argue that this basic building block can be improved in two important respects. First, the power of n-grams derives precisely from multi-way interactions and these are clearly missed (initially) with linear operations on stacked n-gram vectors. Non-linear interactions within a local context have been shown to improve empirical performance in various tasks (Mitchell and Lapata, 2008; Kartsaklis et al., 2012; Socher et al., 2013). Second, many useful patterns are expressed as *non-consecutive* phrases, such as semantically close multi-word expressions (e.g.,"not that good", "not nearly as good"). In typical CNNs, such expressions would have to come together and emerge as useful patterns after several layers of processing.

We propose to use a feature mapping operation based on *tensor products* instead of linear operations on stacked vectors. This enables us to directly tap into non-linear interactions between adjacent word feature vectors (Socher et al., 2013; Lei et al., 2014). To offset the accompanying parametric explosion we maintain a low-rank representation of the tensor parameters. Moreover, we show that this feature mapping can be applied to all possible *non-consecutive* n-grams in the sequence with an exponentially decaying weight depending on the length of the span. Owing to the low rank representation of the tensor, this operation can be performed efficiently in linear time with respect to the sequence length via dynamic programming. Similar to traditional convolution operations, our non-linear feature mapping can be applied successively at multiple levels.

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1565–1575,
Lisbon, Portugal, 17-21 September 2015.

We evaluate the proposed architecture in the context of sentence sentiment classification and news categorization. On the Stanford Sentiment Treebank dataset, our model obtains state-of-the-art performance among a variety of neural networks in terms of both accuracy and training cost. Our model achieves 51.2% accuracy on fine-grained classification and 88.6% on binary classification, outperforming the best published numbers obtained by a deep recursive model (Tai et al., 2015) and a convolutional model (Kim, 2014). On the Chinese news categorization task, our model achieves 80.0% accuracy, while the closest baseline achieves 79.2%.

2 Related Work

Deep neural networks have recently brought about significant advancements in various natural language processing tasks, such as language modeling (Bengio et al., 2003; Mikolov et al., 2010), sentiment analysis (Socher et al., 2013; Iyyer et al., 2015; Le and Zuidema, 2015), syntactic parsing (Collobert and Weston, 2008; Socher et al., 2011a; Chen and Manning, 2014) and machine translation (Bahdanau et al., 2014; Devlin et al., 2014; Sutskever et al., 2014). Models applied in these tasks exhibit significant architectural differences, ranging from recurrent neural networks (Mikolov et al., 2010; Kalchbrenner and Blunsom, 2013) to recursive models (Pollack, 1990; Küchler and Goller, 1996), and including convolutional neural nets (Collobert and Weston, 2008; Collobert et al., 2011; Yih et al., 2014; Shen et al., 2014; Kalchbrenner et al., 2014; Zhang and LeCun, 2015).

Our model most closely relates to the latter. Since these models have originally been developed for computer vision (LeCun et al., 1998), their application to NLP tasks introduced a number of modifications. For instance, Collobert et al. (2011) use the max-over-time pooling operation to aggregate the features over the input sequence. This variant has been successfully applied to semantic parsing (Yih et al., 2014) and information retrieval (Shen et al., 2014; Gao et al., 2014). Kalchbrenner et al. (2014) instead propose (dynamic) k-max pooling operation for modeling sentences. In addition, Kim (2014) combines CNNs of different filter widths and either static or fine-tuned word vectors. In contrast to the traditional CNN models, our method considers non-consecutive n-grams thereby expanding the representation capacity of the model. Moreover, our model captures non-linear interactions within n-gram snippets through the use of tensors, moving beyond direct linear projection operator used in standard CNNs. As our experiments demonstrate these advancements result in improved performance.

3 Background

Let $\mathbf{x} \in \mathbb{R}^{L \times d}$ be the input sequence such as a document or sentence. Here L is the length of the sequence and each $\mathbf{x}_i \in \mathbb{R}^d$ is a vector representing the i^{th} word. The (consecutive) *n-gram* vector ending at position j is obtained by simply concatenating the corresponding word vectors

$$v_j \;=\; [\mathbf{x}_{j-n+1}; \mathbf{x}_{j-n+2}; \cdots; \mathbf{x}_j]$$

Out-of-index words are simply set to all zeros.

The traditional convolution operator is parameterized by filter matrix $\mathbf{m} \in \mathbb{R}^{nd \times h}$ which can be thought of as n smaller filter matrices applied to each $\mathbf{x}_i$ in vector v_j. The operator maps each n-gram vector v_j in the input sequence to $\mathbf{m}^\top v_j \in \mathbb{R}^h$ so that the input sequence $\mathbf{x}$ is transformed into a sequence of feature representations,

$$\left[\mathbf{m}^\top v_1, \cdots, \mathbf{m}^\top v_L\right] \quad \in \quad \mathbb{R}^{L \times h}$$

The resulting feature values are often passed through non-linearities such as the hyper-tangent (element-wise) as well as aggregated or reduced by "sum-over" or "max-pooling" operations for later (similar stages) of processing.

The overall architecture can be easily modified by replacing the basic n-gram vectors and the convolution operation with other feature mappings. Indeed, we appeal to tensor algebra to introduce a non-linear feature mapping that operates on non-consecutive n-grams.

4 Model

N-gram tensor Typical $n-$gram feature mappings where concatenated word vectors are mapped linearly to feature coordinates may be insufficient to directly capture relevant information in the $n-$gram. As a remedy, we replace concatenation with a tensor product. Consider a 3-gram $(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3)$ and the corresponding tensor product $\mathbf{x}_1 \otimes \mathbf{x}_2 \otimes \mathbf{x}_3$. The tensor product is a 3-way array of coordinate interactions such that each ijk

entry of the tensor is given by the product of the corresponding coordinates of the word vectors

$$(\mathbf{x}_1 \otimes \mathbf{x}_2 \otimes \mathbf{x}_3)_{ijk} = \mathbf{x}_{1i} \cdot \mathbf{x}_{2j} \cdot \mathbf{x}_{3k}$$

Here $\otimes$ denotes the tensor product operator. The tensor product of a 2-gram analogously gives a two-way array or matrix $\mathbf{x}_1 \otimes \mathbf{x}_2 \in \mathbb{R}^{d \times d}$. The n-gram tensor can be seen as a direct generalization of the typical concatenated vector[2].

Tensor-based feature mapping Since each n-gram in the sequence is now expanded into a high-dimensional tensor using tensor products, the set of filters are analogously maintained as high-order tensors. In other words, our filters are linear mappings over the higher dimensional interaction terms rather than the original word coordinates.

Consider again mapping a 3-gram $(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3)$ into a feature representation. Each filter is a 3-way tensor with dimensions $d \times d \times d$. The set of h filters, denoted as T, is a 4-way tensor of dimension $d \times d \times d \times h$, where each d^3 slice of T represents a single filter and h is the number of such filters, i.e., the feature dimension. The resulting $h-$dimensional feature representation $z \in \mathbb{R}^h$ for the 3-gram $(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3)$ is obtained by multiplying the filter $\mathbf{T}$ and the 3-gram tensor as follows. The l^{th} coordinate of z is given by

$$\begin{aligned} z_l &= \sum_{ijk} \mathbf{T}_{ijkl} \cdot (\mathbf{x}_1 \otimes \mathbf{x}_2 \otimes \mathbf{x}_3)_{ijk} \\ &= \sum_{ijk} \mathbf{T}_{ijkl} \cdot \mathbf{x}_{1i} \cdot \mathbf{x}_{2j} \cdot \mathbf{x}_{3k} \end{aligned} \quad (1)$$

The formula is equivalent to summing over all the third-order polynomial interaction terms where tensor $\mathbf{T}$ stores the coefficients.

Directly maintaining the filters as full tensors leads to parametric explosion. Indeed, the size of the tensor $\mathbf{T}$ (i.e. $h \times d^n$) would be too large even for typical low-dimensional word vectors where, e.g., $d = 300$. To this end, we assume a *low-rank factorization* of the tensor $\mathbf{T}$, represented in the Kruskal form. Specifically, $\mathbf{T}$ is decomposed into a sum of h rank-1 tensors

$$\mathbf{T} = \sum_{i=1}^{h} \mathbf{P}_i \otimes \mathbf{Q}_i \otimes \mathbf{R}_i \otimes \mathbf{O}_i$$

[2]To see this, consider word vectors with a "bias" term $\mathbf{x}_i{}' = [\mathbf{x}_i; 1]$. The tensor product of n such vectors includes the concatenated vector as a subset of tensor entries but, in addition, contains all up to n^{th}-order interaction terms.

where $\mathbf{P}, \mathbf{Q}, \mathbf{R} \in \mathbb{R}^{h \times d}$ and $\mathbf{O} \in \mathbb{R}^{h \times h}$ are four smaller parameter matrices. $\mathbf{P}_i$ (similarly $\mathbf{Q}_i$, $\mathbf{R}_i$ and $\mathbf{O}_i$) denotes the i^{th} row of the matrix. Note that, for simplicity, we have assumed that the number of rank-1 components in the decomposition is equal to the feature dimension h. Plugging the low-rank factorization into Eq.(1), the feature-mapping can be rewritten in a vector form as

$$z = \mathbf{O}^\top (\mathbf{P}\mathbf{x}_1 \odot \mathbf{Q}\mathbf{x}_2 \odot \mathbf{R}\mathbf{x}_3) \quad (2)$$

where $\odot$ is the element-wise product such that, e.g., $(a \odot b)_k = a_k \times b_k$ for $a, b \in \mathbb{R}^m$. Note that while $\mathbf{P}\mathbf{x}_1$ (similarly $\mathbf{Q}\mathbf{x}_2$ and $\mathbf{R}\mathbf{x}_3$) is a linear mapping from each word $\mathbf{x}_1$ (similarly $\mathbf{x}_2$ and $\mathbf{x}_3$) into a h-dimensional feature space, higher order terms arise from the element-wise products.

Non-consecutive n-gram features Traditional convolution uses consecutive n-grams in the feature map. Non-consecutive n-grams may nevertheless be helpful since phrases such as "not good", "not so good" and "not nearly as good" express similar sentiments but involve variable spacings between the key words. Variable spacings are not effectively captured by fixed n-grams.

We apply the feature-mapping in a weighted manner to all n-grams thereby gaining access to patterns such as "not ... good". Let $z[i, j, k] \in \mathbb{R}^h$ denote the feature representation corresponding to a 3-gram $(\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k)$ of words in positions i, j, and k along the sequence. This vector is calculated analogously to Eq.(2),

$$z[i, j, k] = \mathbf{O}^\top (\mathbf{P}\mathbf{x}_i \odot \mathbf{Q}\mathbf{x}_j \odot \mathbf{R}\mathbf{x}_k)$$

We will aggregate these vectors into an $h-$dimensional feature representation at each position in the sequence. The idea is similar to neural bag-of-words models where the feature representation for a document or sentence is obtained by averaging (or summing) of all the word vectors. In our case, we define the aggregate representation $z_3[k]$ in position k as the weighted sum of all 3-gram feature representations ending at position k, i.e.,

$$\begin{aligned} z_3[k] &= \sum_{i<j<k} z[i, j, k] \cdot \lambda^{(k-j-1)+(j-i-1)} \\ &= \sum_{i<j<k} z[i, j, k] \cdot \lambda^{k-i-2} \end{aligned} \quad (3)$$

where $\lambda \in [0, 1)$ is a decay factor that down-weights 3-grams with longer spans (i.e., 3-grams

that skip more in-between words). As $\lambda \to 0$ all non-consecutive 3-grams are omitted, $z_3[k] = z[k-2, k-1, k]$, and the model acts like a traditional model with only consecutive n-grams. When $\lambda > 0$, however, $z_3[k]$ is a weighted average of many 3-grams with variable spans.

Aggregating features via dynamic programming Directly calculating $z_3[\cdot]$ according to Eq.(3) by enumerating all 3-grams would require $O(L^3)$ feature-mapping operations. We can, however, evaluate the features more efficiently by relying on the associative and distributive properties of the feature operation in Eq.(2).

Let $f_3[k]$ be a dynamic programming table representing the sum of 3-gram feature representations before multiplying with matrix $\mathbf{O}$. That is, $z_3[k] = \mathbf{O}^\top f_3[k]$ or, equivalently,

$$f_3[k] = \sum_{i<j<k} \lambda^{k-i-2} \cdot (\mathbf{P}\mathbf{x}_i \odot \mathbf{Q}\mathbf{x}_j \odot \mathbf{R}\mathbf{x}_k)$$

We can analogously define $f_1[i]$ and $f_2[j]$ for 1-grams and 2-grams,

$$f_1[i] = \mathbf{P}\mathbf{x}_i$$
$$f_2[j] = \sum_{i<j} \lambda^{j-i-1} \cdot (\mathbf{P}\mathbf{x}_i \odot \mathbf{Q}\mathbf{x}_j)$$

These dynamic programming tables can be calculated recursively according to the following formulas:

$$\begin{aligned} f_1[i] &= \mathbf{P}\mathbf{x}_i \\ s_1[i] &= \lambda \cdot s_1[i-1] + f_1[i] \end{aligned}$$

$$\begin{aligned} f_2[j] &= s_1[j-1] \odot \mathbf{Q}\mathbf{x}_j \\ s_2[j] &= \lambda \cdot s_2[j-1] + f_2[j] \end{aligned}$$

$$f_3[k] = s_2[k-1] \odot \mathbf{R}\mathbf{x}_k$$

$$z[k] = \mathbf{O}^\top (f_1[k] + f_2[k] + f_3[k])$$

where $s_1[\cdot]$ and $s_2[\cdot]$ are two auxiliary tables. The resulting $z[\cdot]$ is the sum of 1, 2, and 3-gram features. We found that aggregating the 1,2 and 3-gram features in this manner works better than using 3-gram features alone. Overall, the n-gram feature aggregation can be performed in $O(Ln)$ matrix multiplication/addition operations, and remains linear in the sequence length.

The overall architecture The dynamic programming algorithm described above maps the original input sequence to a sequence of feature representations $\mathbf{z} = z[1:L] \in \mathbb{R}^{L \times h}$. As in standard convolutional architectures, the resulting sequence can be used in multiple ways. One can directly aggregate it to a classifier or expose it to non-linear element-wise transformations and use it as an input to another sequence-to-sequence feature mapping.

The simplest strategy (adopted in neural bag-of-words models) would be to average the feature representations and pass the resulting averaged vector directly to a softmax output unit

$$\bar{z} = \frac{1}{L} \sum_{i=1}^{L} z[i]$$
$$\tilde{y} = \text{softmax}\left(\mathbf{W}^\top \bar{z}\right)$$

Our architecture, as illustrated in Figure 1, includes two additional refinements. First, we add a non-linear activation function after each feature representation, i.e. $\mathbf{z}' = \text{ReLU}(\mathbf{z} + b)$, where b is a bias vector and ReLU is the rectified linear unit function. Second, we stack multiple tensor-based feature mapping layers. That is, the input sequence $\mathbf{x}$ is first processed into a feature sequence and passed through the non-linear transformation to obtain $\mathbf{z}^{(1)}$. The resulting feature sequence $\mathbf{z}^{(1)}$ is then analogously processed by another layer, parameterized by a different set of feature-mapping matrices $\mathbf{P}, \cdots, \mathbf{O}$, to obtain a higher-level feature sequence $\mathbf{z}^{(2)}$, and so on. The output feature representations of all these layers are averaged within each layer and concatenated as shown in Figure 1. The final prediction is therefore obtained on the basis of features across the levels.

5 Learning the Model

Following standard practices, we train our model by minimizing the cross-entropy error on a given training set. For a single training sequence $\mathbf{x}$ and the corresponding gold label $y \in [0,1]^m$, the error is defined as,

$$\text{loss}(\mathbf{x}, y) = \sum_{l=1}^{m} y_l \log(\tilde{y}_l)$$

where m is the number of possible output label.

The set of model parameters (e.g. $\mathbf{P}, \cdots, \mathbf{O}$ in each layer) are updated via stochastic gradient

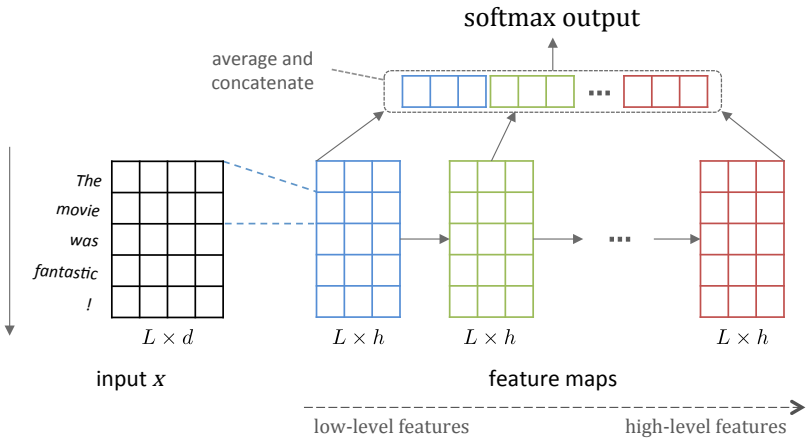


Figure 1: Illustration of the model architecture. The input is represented as a matrix where each row is a d-dimensional word vector. Several feature map layers (as described in Section 4) are stacked, mapping the input into different levels of feature representations. The features are averaged within each layer and then concatenated. Finally a softmax layer is applied to obtain the prediction output.

descent using AdaGrad algorithm (Duchi et al., 2011).

Initialization We initialize matrices $\mathbf{P}, \mathbf{Q}, \mathbf{R}$ from uniform distribution $\left[-\sqrt{3/d}, \sqrt{3/d}\right]$ and similarly $\mathbf{O} \sim \mathrm{U}\left[-\sqrt{3/h}, \sqrt{3/h}\right]$. In this way, each row of the matrices is an unit vector in expectation, and each rank-1 filter slice has unit variance as well,

$$\mathbb{E}\left[\|\mathbf{P}_i \otimes \mathbf{Q}_i \otimes \mathbf{R}_i \otimes \mathbf{O}_i\|^2\right] = 1$$

In addition, the parameter matrix $\mathbf{W}$ in the softmax output layer is initialized as zeros, and the bias vectors b for ReLU activation units are initialized to a small positive constant 0.01.

Regularization We apply two common techniques to avoid overfitting during training. First, we add L2 regularization to all parameter values with the same regularization weight. In addition, we randomly dropout (Hinton et al., 2012) units on the output feature representations $\mathbf{z}^{(i)}$ at each level.

6 Experimental Setup

Datasets We evaluate our model on sentence sentiment classification task and news categorization task. For sentiment classification, we use the Stanford Sentiment Treebank benchmark (Socher et al., 2013). The dataset consists of 11855 parsed English sentences annotated at both the root (i.e. sentence) level and the phrase level using 5-class fine-grained labels. We use the standard 8544/1101/2210 split for training, development and testing respectively. Following previous work, we also evaluate our model on the binary classification variant of this benchmark, ignoring all neutral sentences. The binary version has 6920/872/1821 sentences for training, development and testing.

For the news categorization task, we evaluate on Sogou Chinese news corpora.[3] The dataset contains 10 different news categories in total, including Finance, Sports, Technology and Automobile etc. We use 79520 documents for training, 9940 for development and 9940 for testing. To obtain Chinese word boundaries, we use LTP-Cloud[4], an open-source Chinese NLP platform.

Baselines We implement the standard **SVM** method and the neural bag-of-words model **NBoW** as baseline methods in both tasks. To assess the proposed tensor-based feature map, we also implement a convolutional neural network model **CNN** by replacing our filter with traditional linear filter. The rest of the framework (such as feature averaging and concatenation) remains the same.

In addition, we compare our model with a wide range of top-performing models on the sentence sentiment classification task. Most of these models fall into either the category of recursive neural networks (RNNs) or the category of convolutional neural networks (CNNs). The recursive neural

[3] http://www.sogou.com/labs/dl/c.html
[4] http://www.ltp-cloud.com/intro/en/
https://github.com/HIT-SCIR/ltp

Model	Fine-grained		Binary		Time (in seconds)	
	Dev	Test	Dev	Test	per epoch	per 10k samples
RNN		43.2		82.4	-	-
RNTN		45.7		85.4	1657	1939
DRNN		49.8		86.8	431	504
RLSTM		51.0		88.0	140	164
DCNN		48.5		86.9	-	-
CNN-MC		47.4		88.1	2452	156
CNN	48.8	47.2	85.7	86.2	32	37
PVEC		48.7		87.8	-	-
DAN		48.2		86.8	73	5
SVM	40.1	38.3	78.6	81.3	-	-
NBoW	45.1	44.5	80.7	82.0	1	1
Ours	49.5	50.6	87.0	87.0	28	33
+ phrase labels	53.4	**51.2**	88.9	**88.6**	445	28

Table 1: Comparison between our model and other baseline methods on Stanford Sentiment Treebank. The top block lists recursive neural network models, the second block are convolutional network models and the third block contains other baseline methods, including the paragraph-vector model (Le and Mikolov, 2014), the deep averaging network model (Iyyer et al., 2015) and our implementation of neural bag-of-words. The training time of baseline methods is taken from (Iyyer et al., 2015) or directly from the authors. For our implementations, timings were performed on a single core of a 2.6GHz Intel i7 processor.

network baselines include standard **RNN** (Socher et al., 2011b), **RNTN** with a small core tensor in the composition function (Socher et al., 2013), the deep recursive model **DRNN** (Irsoy and Cardie, 2014) and the most recent recursive model using long-short-term-memory units **RLSTM** (Tai et al., 2015). These recursive models assume the input sentences are represented as parse trees. As a benefit, they can readily utilize annotations at the phrase level. In contrast, convolutional neural networks are trained on sequence-level, taking the original sequence and its label as training input. Such convolutional baselines include the dynamic CNN with k-max pooling **DCNN** (Kalchbrenner et al., 2014) and the convolutional model with multi-channel **CNN-MC** by Kim (2014). To leverage the phrase-level annotations in the Stanford Sentiment Treebank, all phrases and the corresponding labels are added as separate instances when training the sequence models. We follow this strategy and report results with and without phrase annotations.

Word vectors The word vectors are pre-trained on much larger unannotated corpora to achieve better generalization given limited amount of training data (Turian et al., 2010). In particular, for the English sentiment classification task, we use the publicly available 300-dimensional GloVe word vectors trained on the Common Crawl with 840B tokens (Pennington et al., 2014). This choice of word vectors follows most recent work, such as **DAN** (Iyyer et al., 2015) and **RLSTM** (Tai et al., 2015). For Chinese news categorization, there is no widely-used publicly available word vectors. Therefore, we run `word2vec` (Mikolov et al., 2013) to train 200-dimensional word vectors on the 1.6 million Chinese news articles. Both word vectors are normalized to unit norm (i.e. $\|w\|_2^2 = 1$) and are fixed in the experiments without fine-tuning.

Hyperparameter setting We perform an extensive search on the hyperparameters of our full model, our implementation of the **CNN** model (with linear filters), and the **SVM** baseline. For our model and the **CNN** model, the initial learning rate of AdaGrad is fixed to 0.01 for sentiment classification and 0.1 for news categorization, and the L2 regularization weight is fixed to $1e-5$ and $1e-6$ respectively based on preliminary runs. The rest of the hyperparameters are randomly chosen as follows: number of feature-mapping layers $\in \{1, 2, 3\}$, n-gram order $n \in \{2, 3\}$, hidden feature dimension $h \in \{50, 100, 200\}$, dropout probability $\in \{0.0, 0.1, 0.3, 0.5\}$, and length de-

cay $\lambda \in \{0.0, 0.3, 0.5\}$. We run each configuration 3 times to explore different random initializations. For the **SVM** baseline, we tune L2 regularization weight $C \in \{0.01, 0.1, 1.0, 10.0\}$, word cut-off frequency $\in \{1, 2, 3, 5\}$ (i.e. pruning words appearing less than this times) and n-gram feature order $n \in \{1, 2, 3\}$.

Implementation details The source code is implemented in Python using the Theano library (Bergstra et al., 2010), a flexible linear algebra compiler that can optimize user-specified computations (models) with efficient automatic low-level implementations, including (back-propagated) gradient calculation.

7 Results

7.1 Overall Performance

Table 1 presents the performance of our model and other baseline methods on Stanford Sentiment Treebank benchmark. Our full model obtains the highest accuracy on both the development and test sets. Specifically, it achieves 51.2% and 88.6% test accuracies on fine-grained and binary tasks respectively[5]. As shown in Table 2, our model performance is relatively stable – it remains high accuracies with around 0.5% standard deviation under different initializations and dropout rates.

Our full model is also several times faster than other top-performing models. For example, the convolutional model with multi-channel (**CNN-MC**) runs over 2400 seconds per training epoch. In contrast, our full model (with 3 feature layers) runs on average 28 seconds with only root labels and on average 445 seconds with all labels.

Our results also show that the **CNN** model, where our feature map is replaced with traditional linear map, performs worse than our full model. This observation confirms the importance of the proposed non-linear, tensor-based feature mapping. The **CNN** model also lags behind the **DCNN** and **CNN-MC** baselines, since the latter two propose several advancements over standard CNN.

Table 3 reports the results of **SVM**, **NBoW** and our model on the news categorization task. Since the dataset is much larger compared to the sentiment dataset (80K documents vs. 8.5K sentences), the **SVM** method is a competitive baseline. It achieves 78.5% accuracy compared to 74.4% and 79.2% obtained by the neural bag-of-words model and CNN model. In contrast, our model obtains 80.0% accuracy on both the development and test sets, outperforming the three baselines by a 0.8% absolute margin. The best hyperparameter configuration in this task uses less feature layers and lower n-gram order (specifically, 2 layers and $n = 2$) compared to the sentiment classification task. We hypothesize that the difference is due to the nature of the two tasks: the document classification task requires to handle less compositions or context interactions than sentiment analysis.

[5]Best hyperparameter configuration based on dev accuracy: 3 layers, 3-gram tensors (n=3), feature dimension $d = 200$ and length decay $\lambda = 0.5$

	Dataset	Accuracy
Fine-grained	Dev	52.5 (±0.5) %
	Test	51.4 (±0.6) %
Binary	Dev	88.4 (±0.3) %
	Test	88.4 (±0.5) %

Table 2: Analysis of average accuracy and standard deviation of our model on sentiment classification task.

Model	Dev Acc.	Test Acc.
SVM (1-gram)	77.5	77.4
SVM (2-gram)	78.2	78.0
SVM (3-gram)	78.2	78.5
NBoW	74.4	74.4
CNN	79.5	79.2
Ours	80.0	80.0

Table 3: Performance of various methods on Chinese news categorization task. Our model obtains better results than the SVM, NBoW and traditional CNN baselines.

7.2 Hyperparameter Analysis

We next investigate the impact of hyperparameters in our model performance. We use the models trained on fine-grained sentiment classification task with only root labels.

Number of layers We plot the fine-grained sentiment classification accuracies obtained during hyperparameter grid search. Figure 2 illustrates how the number of feature layers impacts the model performance. As shown in the figure, adding higher-level features clearly improves the classification accuracy across various hyperparameter settings and initializations.

Non-consecutive n-gram features We also analyze the effect of modeling non-consecutive n-

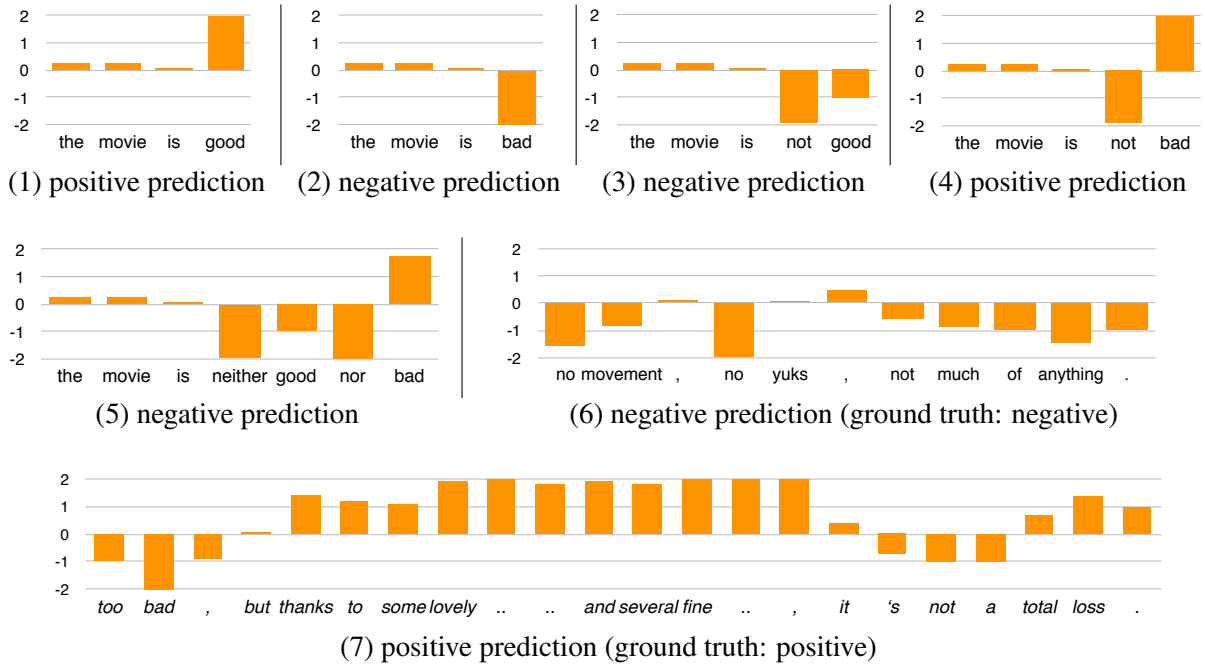


Figure 5: Example sentences and their sentiments predicted by our model trained with root labels. The predicted sentiment scores at each word position are plotted. Examples (1)-(5) are synthetic inputs, (6) and (7) are two real inputs from the test set. Our model successfully identifies negation, double negation and phrases with different sentiment in one sentence.

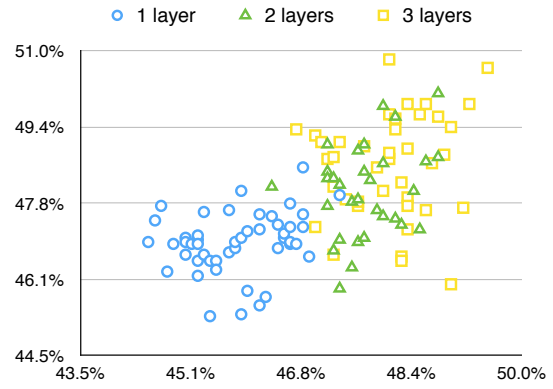


Figure 2: Dev accuracy (x-axis) and test accuracy (y-axis) of independent runs of our model on fine-grained sentiment classification task. Deeper architectures achieve better accuracies.

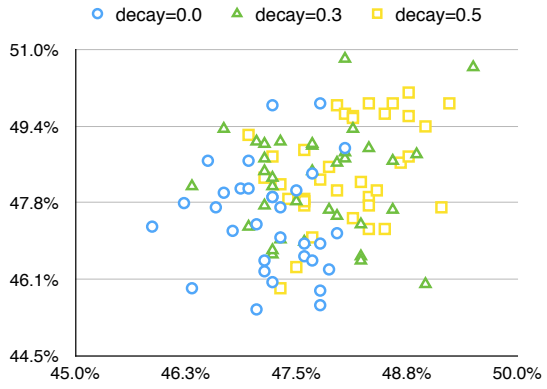


Figure 3: Comparison of our model variations in sentiment classification task when considering consecutive n-grams only (decaying factor $\lambda = 0$) and when considering non-consecutive n-grams ($\lambda > 0$). Modeling non-consecutive n-gram features leads to better performance.

grams. Figure 3 splits the model accuracies according to the choice of span decaying factor λ. Note when $\lambda = 0$, the model applies feature extractions to consecutive n-grams only. As shown in Figure 3, this setting leads to consistent performance drop. This result confirms the importance of handling non-consecutive n-gram patterns.

Non-linear activation Finally, we verify the effectiveness of rectified linear unit activation function (ReLU) by comparing it with no activation (or identity activation $f(x) = x$). As shown in Figure 4, our model with ReLU activation generally outperforms its variant without ReLU. The observation is consistent with previous work on convolutional neural networks and other neural network models.

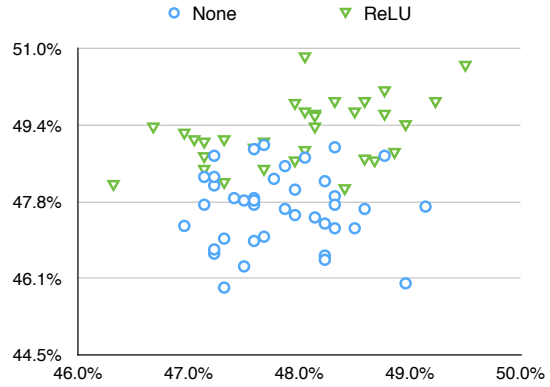


Figure 4: Applying ReLU activation on top of tensor-based feature mapping leads to better performance in sentiment classification task. Runs with no activation are plotted as blue circles.

7.3 Example Predictions

Figure 5 gives examples of input sentences and the corresponding predictions of our model in fine-grained sentiment classification. To see how our model captures the sentiment at different local context, we apply the learned softmax activation to the extracted features at each position without taking the average. That is, for each index i, we obtain the local sentiment $p = \text{softmax}\left(\mathbf{W}^\top \left(z^{(1)}[i] \oplus z^{(2)}[i] \oplus z^{(3)}[i]\right)\right)$. We plot the expected sentiment scores $\sum_{s=-2}^{2} s \cdot p(s)$, where a score of 2 means "very positive", 0 means "neutral" and -2 means "very negative". As shown in the figure, our model successfully learns negation and double negation. The model also identifies positive and negative segments appearing in the sentence.

8 Conclusion

We proposed a feature mapping operator for convolutional neural networks by modeling n-gram interactions based on tensor product and evaluating all non-consecutive n-gram vectors. The associated parameters are maintained as a low-rank tensor, which leads to efficient feature extraction via dynamic programming. The model achieves top performance on standard sentiment classification and document categorization tasks.

Acknowledgments

We thank Kai Sheng Tai, Mohit Iyyer and Jordan Boyd-Graber for answering questions about their paper. We also thank Yoon Kim, the MIT NLP group and the reviewers for their comments. We acknowledge the support of the U.S. Army Research Office under grant number W911NF-10-1-0533. The work is developed in collaboration with the Arabic Language Technologies (ALT) group at Qatar Computing Research Institute (QCRI). Any opinions, findings, conclusions, or recommendations expressed in this paper are those of the authors, and do not necessarily reflect the views of the funding organizations.

References

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

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *The Journal of Machine Learning Research*, 3:1137–1155.

James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. 2010. Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*.

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

R. Collobert and J. Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *International Conference on Machine Learning, ICML*.

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

Jacob Devlin, Rabih Zbib, Zhongqiang Huang, Thomas Lamar, Richard Schwartz, and John Makhoul. 2014. Fast and robust neural network joint models for statistical machine translation. In *52nd Annual Meeting of the Association for Computational Linguistics*.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159.

Jianfeng Gao, Patrick Pantel, Michael Gamon, Xiaodong He, Li Deng, and Yelong Shen. 2014. Modeling interestingness with deep neural networks. In

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing.

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

Ozan Irsoy and Claire Cardie. 2014. Deep recursive neural networks for compositionality in language. In *Advances in Neural Information Processing Systems*.

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daume III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *Association for Computational Linguistics*.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP 2013)*, pages 1700–1709.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of the 52th Annual Meeting of the Association for Computational Linguistics*.

Dimitri Kartsaklis, Mehrnoosh Sadrzadeh, and Stephen Pulman. 2012. A unified sentence space for categorical distributional-compositional semantics: Theory and experiments. In *In Proceedings of COLING: Posters*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014)*.

Andreas Küchler and Christoph Goller. 1996. Inductive learning in symbolic domains using structure-driven recurrent neural networks. In *KI-96: Advances in Artificial Intelligence*, pages 183–197.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1188–1196.

Phong Le and Willem Zuidema. 2015. Compositional distributional semantics with long short term memory. In *Proceedings of Joint Conference on Lexical and Computational Semantics (*SEM)*.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, November.

Tao Lei, Yu Xin, Yuan Zhang, Regina Barzilay, and Tommi Jaakkola. 2014. Low-rank tensors for scoring dependency structures. In *Proceedings of the 52th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010*, pages 1045–1048.

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

Jeff Mitchell and Mirella Lapata. 2008. Vector-based models of semantic composition. In *ACL*, pages 236–244.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. volume 12.

Jordan B Pollack. 1990. Recursive distributed representations. *Artificial Intelligence*, 46:77–105.

Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. 2014. Learning semantic representations using convolutional neural networks for web search. In *Proceedings of the companion publication of the 23rd international conference on World wide web companion*, pages 373–374. International World Wide Web Conferences Steering Committee.

Richard Socher, Cliff C. Lin, Andrew Y. Ng, and Christopher D. Manning. 2011a. Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of the 26th International Conference on Machine Learning (ICML)*.

Richard Socher, Jeffrey Pennington, Eric H Huang, Andrew Y Ng, and Christopher D Manning. 2011b. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 151–161. Association for Computational Linguistics.

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

Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53th Annual Meeting of the Association for Computational Linguistics*.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10. Association for Computational Linguistics.

Wen-tau Yih, Xiaodong He, and Christopher Meek. 2014. Semantic parsing for single-relation question answering. In *Proceedings of ACL*.

Xiang Zhang and Yann LeCun. 2015. Text understanding from scratch. *arXiv preprint arXiv:1502.01710*.

Multi-Perspective Sentence Similarity Modeling with Convolutional Neural Networks

Hua He,[1] **Kevin Gimpel,**[2] **and Jimmy Lin**[3]

[1] Department of Computer Science, University of Maryland, College Park
[2] Toyota Technological Institute at Chicago
[3] David R. Cheriton School of Computer Science, University of Waterloo

huah@cs.umd.edu, kgimpel@ttic.edu, jimmylin@uwaterloo.ca

Abstract

Modeling sentence similarity is complicated by the ambiguity and variability of linguistic expression. To cope with these challenges, we propose a model for comparing sentences that uses a multiplicity of perspectives. We first model each sentence using a convolutional neural network that extracts features at multiple levels of granularity and uses multiple types of pooling. We then compare our sentence representations at several granularities using multiple similarity metrics. We apply our model to three tasks, including the Microsoft Research paraphrase identification task and two SemEval semantic textual similarity tasks. We obtain strong performance on all tasks, rivaling or exceeding the state of the art without using external resources such as WordNet or parsers.

1 Introduction

Measuring the semantic relatedness of two pieces of text is a fundamental problem in language processing tasks like plagiarism detection, query ranking, and question answering. In this paper, we address the sentence similarity measurement problem: given a query sentence S_1 and a comparison sentence S_2, the task is to compute their similarity in terms of a score $sim(S_1, S_2)$. This similarity score can be used within a system that determines whether two sentences are paraphrases, e.g., by comparing it to a threshold.

Measuring sentence similarity is challenging because of the variability of linguistic expression and the limited amount of annotated training data. This makes it difficult to use sparse, hand-crafted features as in conventional approaches in NLP. Recent successes in sentence similarity have been obtained by using neural networks (Tai et al., 2015; Yin and Schütze, 2015). Our approach is also based on neural networks: we propose a modular functional architecture with two components, sentence modeling and similarity measurement.

For sentence modeling, we use a convolutional neural network featuring convolution filters with multiple granularities and window sizes, followed by multiple types of pooling. We experiment with two types of word embeddings as well as part-of-speech tag embeddings (Sec. 4). For similarity measurement, we compare pairs of local regions of the sentence representations, using multiple distance functions: cosine distance, Euclidean distance, and element-wise difference (Sec. 5).

We demonstrate state-of-the-art performance on two SemEval semantic relatedness tasks (Agirre et al., 2012; Marelli et al., 2014), and highly competitive performance on the Microsoft Research paraphrase (MSRP) identification task (Dolan et al., 2004). On the SemEval-2014 task, we match the state-of-the-art dependency tree Long Short-Term Memory (LSTM) neural networks of Tai et al. (2015) without using parsers or part-of-speech taggers. On the MSRP task, we outperform the recently-proposed convolutional neural network model of Yin and Schütze (2015) without any pretraining. In addition, we perform ablation experiments to show the contribution of our modeling decisions for all three datasets, demonstrating clear benefits from our use of multiple perspectives both in sentence modeling and structured similarity measurement.

2 Related Work

Most previous work on modeling sentence similarity has focused on feature engineering. Several types of sparse features have been found useful, including: (1) string-based, including n-gram overlap features on both the word and character levels (Wan et al., 2006) and features based on machine translation evaluation metrics (Madnani

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1576–1586,
Lisbon, Portugal, 17-21 September 2015.

et al., 2012); (2) knowledge-based, using external lexical resources such as WordNet (Fellbaum, 1998; Fern and Stevenson, 2008); (3) syntax-based, e.g., modeling divergence of dependency syntax between the two sentences (Das and Smith, 2009); (4) corpus-based, using distributional models such as latent semantic analysis to obtain features (Hassan, 2011; Guo and Diab, 2012).

Several strongly-performing approaches used system combination (Das and Smith, 2009; Madnani et al., 2012) or multi-task learning. Xu et al. (2014) developed a feature-rich multi-instance learning model that jointly learns paraphrase relations between word and sentence pairs.

Recent work has moved away from handcrafted features and towards modeling with distributed representations and neural network architectures. Collobert and Weston (2008) used convolutional neural networks in a multitask setting, where their model is trained jointly for multiple NLP tasks with shared weights. Kalchbrenner et al. (2014) introduced a convolutional neural network for sentence modeling that uses dynamic k-max pooling to better model inputs of varying sizes. Kim (2014) proposed several modifications to the convolutional neural network architecture of Collobert and Weston (2008), including the use of both fixed and learned word vectors and varying window sizes of the convolution filters.

For the MSRP task, Socher et al. (2011) used a recursive neural network to model each sentence, recursively computing the representation for the sentence from the representations of its constituents in a binarized constituent parse. Ji and Eisenstein (2013) used matrix factorization techniques to obtain sentence representations, and combined them with fine-tuned sparse features using an SVM classifier for similarity prediction. Both Socher et al. and Ji and Eisenstein incorporated sparse features to improve performance, which we do not use in this work.

Hu et al. (2014) used convolutional neural networks that combine hierarchical sentence modeling with layer-by-layer composition and pooling. While they performed comparisons directly over entire sentence representations, we instead develop a structured similarity measurement layer to compare local regions. A variety of other neural network models have been proposed for similarity tasks (Weston et al., 2011; Huang et al., 2013; Andrew et al., 2013; Bromley et al., 1993).

Most recently, Tai et al. (2015) and Zhu et al. (2015) concurrently proposed a tree-based LSTM neural network architecture for sentence modeling. Unlike them, we do not use syntactic parsers, yet our performance matches Tai et al. (2015) on the similarity task. This result is appealing because high-quality parsers are difficult to obtain for low-resource languages or specialized domains. Yin and Schütze (2015) concurrently developed a convolutional neural network architecture for paraphrase identification, which we compare to in our experiments. Their best results rely on an unsupervised pretraining step, which we do not need to match their performance.

Our model architecture differs from previous work in several ways. We exploit multiple perspectives of input sentences in order to maximize information utilization and perform structured comparisons over particular regions of the sentence representations. We now proceed to describe our model in detail, and we compare to the above related work in our experimental evaluation.

3 Model Overview

Modeling textual similarity is complicated by the ambiguity and variability of linguistic expression. We designed a model with these phenomena in mind, exploiting multiple types of input which are processed by multiple types of convolution and pooling. Our similarity architecture likewise uses multiple similarity functions.

To summarize, our model (shown in Figure 1) consists of two main components:

1. A **sentence model** for converting a sentence into a representation for similarity measurement; we use a convolutional neural network architecture with multiple types of convolution and pooling in order to capture different granularities of information in the inputs.
2. A **similarity measurement layer** using multiple similarity measurements, which compare local regions of the sentence representations from the sentence model.

Our model has a "Siamese" structure (Bromley et al., 1993) with two subnetworks each processing a sentence in parallel. The subnetworks share all of their weights, and are joined by the similarity measurement layer, then followed by a fully connected layer for similarity score output.

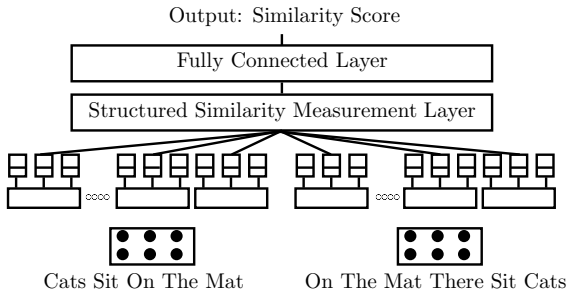


Figure 1: Model overview. Two input sentences (on the bottom) are processed in parallel by identical neural networks, outputting sentence representations. The sentence representations are compared by the structured similarity measurement layer. The similarity features are then passed to a fully-connected layer for computing the similarity score (top).

Importantly, we do not require resources like WordNet or syntactic parsers for the language of interest; we only use optional part-of-speech tags and pretrained word embeddings. The main difference from prior work lies in our use of multiple types of convolution, pooling, and structured similarity measurement over local regions. We show later in our experiments that the bulk of our performance comes from this use of multiple "perspectives" of the input sentences.

We describe our sentence model in Section 4 and our similarity measurement layer in Section 5.

4 Sentence Modeling

In this section we describe our convolutional neural network for modeling each sentence. We use two types of convolution filters defined on different perspectives of the input (Sec. 4.1), and also use multiple types of pooling (Sec. 4.2).

Our inputs are streams of tokens, which can be interpreted as a temporal sequence where nearby words are likely to be correlated. Let $sent \in \mathbf{R}^{len \times Dim}$ be a sequence of len input words represented by Dim-dimensional word embeddings, where $sent_i \in \mathbf{R}^{Dim}$ is the embedding of the i-th word in the sequence and $sent_{i:j}$ represents the concatenation of embeddings from word i up to and including word j. We denote the k-th dimension of the i-th word vector by $sent_i^{[k]}$ and we denote the vector containing the k-th dimension of words i to j by $sent_{i:j}^{[k]}$.

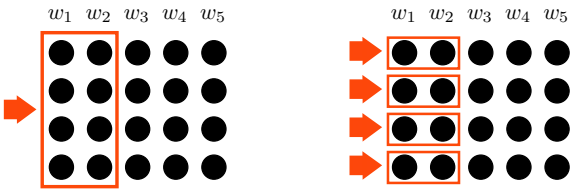


Figure 2: Left: a holistic filter matches entire word vectors (here, $ws = 2$). Right: per-dimension filters match against each dimension of the word embeddings independently.

4.1 Convolution on Multiple Perspectives

We define a convolution filter F as a tuple $\langle ws, w_F, b_F, h_F \rangle$, where ws is the sliding window width, $w_F \in \mathbf{R}^{ws \times Dim}$ is the weight vector for the filter, $b_F \in \mathbf{R}$ is the bias, and h_F is the activation function (a nonlinear function such as $\tanh$). When filter F is applied to sequence $sent$, the inner product is computed between w_F and each possible window of word embeddings of length ws in $sent$, then the bias is added and the activation function is applied. This results in an output vector $out_F \in \mathbf{R}^{1+len-ws}$ where entry i equals

$$out_F[i] = h_F(w_F \cdot sent_{i:i+ws-1} + b_F) \quad (1)$$

where $i \in [1, 1 + len - ws]$. This filter can be viewed as performing "temporal" convolution, as it matches against regions of the word sequence. Since these filters consider the entirety of each word embedding at each position, we call them holistic filters; see the left half of Figure 2.

In addition, we target information at a finer granularity by constructing per-dimension filters $F^{[k]}$ for each dimension k of the word embeddings, where $w_{F^{[k]}} \in \mathbf{R}^{ws}$. See the right half of Figure 2. The per-dimension filters are similar to "spatial convolution" filters except that we limit each to a single, predefined dimension. We include separate per-dimension filters for each dimension of the input word embeddings.

Applying a per-dimension filter $F^{[k]} = \langle ws, w_{F^{[k]}}, b_{F^{[k]}}, h_{F^{[k]}} \rangle$ for dimension k results in an output vector $out_{F^{[k]}} \in \mathbf{R}^{1+len-ws}$ where entry i (for $i \in [1, 1 + len - ws]$) equals

$$out_{F^{[k]}}[i] = h_{F^{[k]}}(w_{F^{[k]}} \cdot sent_{i:i+ws-1}^{[k]} + b_{F^{[k]}})$$

Our use of word embeddings in both ways allows more information to be extracted for richer sentence modeling. While we typically do not expect individual dimensions of neural word embeddings

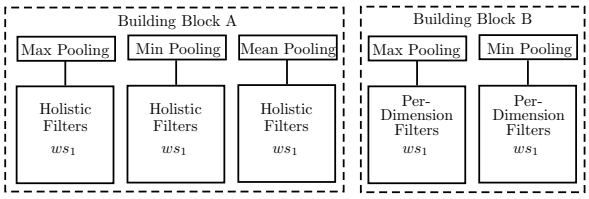


Figure 3: Each building block consists of multiple independent pooling layers and convolution layers with width ws_1. Left: $block_A$ operates on entire vectors of word embeddings. Right: $block_B$ operates on individual dimensions of word vectors to capture information of a finer granularity.

to be interpretable to humans, there may still be distinct information captured by the different dimensions that our model could exploit. Furthermore, if we update the word embeddings during learning, different dimensions could be encouraged further to capture distinct information.

We define a convolution layer as a set of convolution filters that share the same type (holistic or per-dimension), activation function, and width ws. The type, width, activation function, and number of filters $numFilter$ in the layer are chosen by the modeler and the weights of each filter (w_F and b_F) are learned.

4.2 Multiple Pooling Types

The output vector out_F of a convolution filter F is typically converted to a scalar for subsequent use by the model using some method of pooling. For example, "max-pooling" applies a $\max$ operation across the entries of out_F and returns the maximum value. In this paper, we experiment with two additional types of pooling: "min-pooling" and "mean-pooling".

A group, denoted $group(ws, pooling, sent)$, is an object that contains a convolution layer with width ws, uses pooling function $pooling$, and operates on sentence $sent$. We define a building block to be a set of groups. We use two types of building blocks, $block_A$ and $block_B$, as shown in Figure 3. We define $block_A$ as

$$\{group_A(ws_a, p, sent) : p \in \{\max, \min, \text{mean}\}\}.$$

That is, an instance of $block_A$ has three convolution layers, one corresponding to each of the three pooling functions; all have the same window size ws_a. An alternative choice would be to use the multiple types of pooling on the *same* filters (Rennie et al., 2014); we instead use independent sets of filters for the different pooling types.[1] We use blocks of type A for all holistic convolution layers.

We define $block_B$ as

$$\{group_B(ws_b, p, sent) : p \in \{\max, \min\}\}.$$

That is, $block_B$ contains two groups of convolution layers of width ws_b, one with max-pooling and one with min-pooling. Each $group_B(*)$ contains a convolution layer with Dim per-dimension convolution filters. That is, we use blocks of type B for convolution layers that operate on individual dimensions of word vectors.

We use these multiple types of pooling to extract different types of information from each type of filter. The design of each $group(*)$ allows a pooling function to interact with its own underlying convolution layers independently, so each convolution layer can learn to recognize distinct phenomena of the input for richer sentence modeling.

For a $group_A(ws_a, pooling_a, sent)$ with a convolution layer with $numFilter_A$ filters, we define the output oG_A as a vector of length $numFilter_A$ where entry j is

$$oG_A[j] = pooling_a(out_{F_j}) \quad (2)$$

where filters are indexed as F_j. That is, the output of $group_A(*)$ is a $numFilter_A$-length vector containing the output of applying the pooling function on each filter's vector of filter match outputs.[2]

A component $group_B(*)$ of $block_B$ contains Dim filters, each operating on a particular dimension of the word embeddings. We define the output oG_B of $group_B(ws_b, pooling_b, sent)$ as a $Dim \times numFilter_B$ matrix where entry $[k][j]$ is

$$oG_B[k][j] = pooling_b(out_{F_j^{[k]}})$$

where filter $F_j^{[k]}$ is filter j for dimension k.

4.3 Multiple Window Sizes

Similar to traditional n-gram-based models, we use multiple window sizes ws in our building blocks in order to learn features of different lengths. For example, in Figure 4 we use four building blocks, each with one window size $ws =$

[1] We note that $\max$ and $\min$ are not both strictly necessary when using certain activation functions, but they still may help us find a more felicitous local optimum.

[2] We note that there is no pooling across multiple filters in a layer/group, or across groups. Each pooling operation is performed independently on the matches of a single filter.

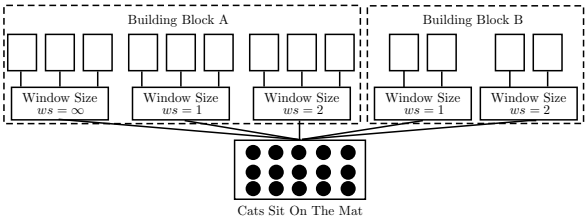


Figure 4: Example neural network architecture for a single sentence, containing 3 instances of $block_A$ (with 3 types of pooling) and 2 instances of $block_B$ (with 2 types) on varying window sizes $ws = 1, 2$ and $ws = \infty$; $block_A$ operates on entire word vectors while $block_B$ contains filters that operate on individual dimensions independently.

1 or 2 for its own convolution layers. In order to retain the original information in the sentences, we also include the entire matrix of word embeddings in the sentence, which essentially corresponds to $ws = \infty$.

The width ws represents how many words are matched by a filter, so using larger values of ws corresponds to matching longer n-grams in the input sentences. The ranges of ws values and the numbers of filters $numFilter$ of $block_A$ and $block_B$ are empirical choices tuned based on validation data.

5 Similarity Measurement Layer

In this section we describe the second part of our model, the similarity measurement layer.

Given two input sentences, the first part of our model computes sentence representations for each of them in parallel. One straightforward way to compare them is to flatten the sentence representations into two vectors, then use standard metrics like cosine similarity. However, this may not be optimal because different regions of the flattened sentence representations are from different underlying sources (e.g., groups of different widths, types of pooling, dimensions of word vectors, etc.). Flattening might discard useful compositional information for computing similarity. We therefore perform structured comparisons over particular regions of the sentence representations.

One important consideration is how to identify suitable local regions for comparison so that we can best utilize the compositional information in the sentence representations. There are many possible ways to group local comparison regions. In doing so, we consider the following four aspects: 1) whether from the same building block; 2) whether from convolutional layers with the same window size; 3) whether from the same pooling layer; 4) whether from the same filter of the underlying convolution layers.[3] We focus on comparing regions that share at least two of these conditions.

To concretize this, we provide two algorithms below to identify meaningful local regions. While there exist other sets of comparable regions that share the above conditions, we do not explore them all due to concerns about learning efficiency; we find that the subset we consider performs strongly in practice.

5.1 Similarity Comparison Units

We define two comparison units for comparing two local regions in the sentence representations:

$$comU_1(\overrightarrow{x}, \overrightarrow{y}) = \{\cos(\overrightarrow{x}, \overrightarrow{y}), L_2Euclid(\overrightarrow{x}, \overrightarrow{y}), |\overrightarrow{x} - \overrightarrow{y}|\} \quad (3)$$

$$comU_2(\overrightarrow{x}, \overrightarrow{y}) = \{\cos(\overrightarrow{x}, \overrightarrow{y}), L_2Euclid(\overrightarrow{x}, \overrightarrow{y})\} \quad (4)$$

Cosine distance (cos) measures the distance of two vectors according to the angle between them, while L_2 Euclidean distance ($L_2Euclid$) and element-wise absolute difference measure magnitude differences.

5.2 Comparison over Local Regions

Algorithms 1 and 2 show how the two sentence representations are compared in our model. Algorithm 1 works on the output of $block_A$ only, while Algorithm 2 deals with both $block_A$ and $block_B$, focusing on regions from the output of the same pooling type and same block type, but with different filters and window sizes of convolution layers.

Given two sentences S_1 and S_2, we set the maximum window size ws of $block_A$ and $block_B$ to be n, let $regM_*$ represent a $numFilter_A$ by $n+1$ matrix, and assume that each $group_*$ outputs its corresponding oG_*. The output features are accumulated in a final vector fea.

5.3 One Simplified Example

We provide a simplified working example to show how the two algorithms compare outputs of $block_A$ only. If we arrange the sentence representations into the shape of sentence matrices as in Figure 5,

[3] We note that since we apply the same network to both sentences, the same filters are used to match both sentences, so we can directly compare filter matches of individual filters across the two sentences.

Algorithm 1 Horizontal Comparison

1: **for** each pooling $p = \max, \min, \text{mean}$ **do**
2: **for** each width $ws_1 = 1...n, \infty$ **do**
3: $regM_1[*][ws_1] = group_A(ws_1, p, S_1)$
4: $regM_2[*][ws_1] = group_A(ws_1, p, S_2)$
5: **end for**
6: **for** each $i = 1...numFilter_A$ **do**
7: $fea_h = comU_2(regM_1[i], regM_2[i])$
8: accumulate fea_h for final layer
9: **end for**
10: **end for**

Algorithm 2 Vertical Comparison

1: **for** each pooling $p = \max, \min, \text{mean}$ **do**
2: **for** each width $ws_1 = 1...n, \infty$ **do**
3: $oG_{1A} = group_A(ws_1, p, S_1)$
4: **for** each width $ws_2 = 1...n, \infty$ **do**
5: $oG_{2A} = group_A(ws_2, p, S_2)$
6: $fea_a = comU_1(oG_{1A}, oG_{2A})$
7: accumulate fea_a for final layer
8: **end for**
9: **end for**
10: **for** each width $ws_1 = 1...n$ **do**
11: $oG_{1B} = group_B(ws_1, p, S_1)$
12: $oG_{2B} = group_B(ws_1, p, S_2)$
13: **for** each $i = 1...numFilter_B$ **do**
14: $fea_b = comU_1(oG_{1B}[*][i], oG_{2B}[*][i])$
15: accumulate fea_b for final layer
16: **end for**
17: **end for**
18: **end for**

then in Algorithms 1 and 2 we are essentially comparing local regions of the two matrices in two directions: along rows and columns.

In Figure 5, each column of the max/min/mean groups is compared with all columns of the same pooling group for the other sentence. This is shown in red dotted lines in the Figure and listed in lines 2 to 9 in Algorithm 2. Note that both ws_1 and ws_2 columns within each pooling group should be compared using red dotted lines, but we omit this from the figure for clarity.

In the horizontal direction, each equal-sized max/min/mean group is extracted as a vector and is compared to the corresponding one for the other sentence. This process is repeated for all rows and comparisons are shown in green solid lines, as performed by Algorithm 1.

5.4 Other Model Details

Output Fully-Connected Layer. On top of the similarity measurement layer (which outputs a vector containing all fea_*), we stack two linear layers with an activation layer in between, followed by a log-softmax layer as the final output layer, which outputs the similarity score.

Activation Layers. We used element-wise tanh as the activation function for all convolution filters and for the activation layer placed between the final two layers.

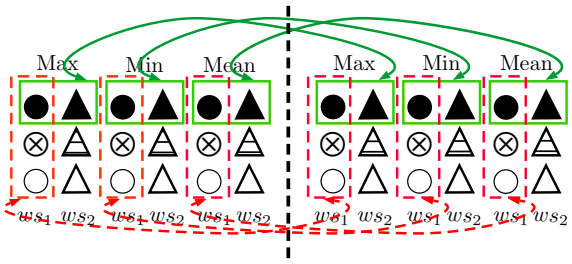


Figure 5: Simplified example of local region comparisons over two sentence representations that use $block_A$ only. The "horizontal comparison" (Algorithm 1) is shown with green solid lines and "vertical comparison" (Algorithm 2) with red dotted lines. Each sentence representation uses window sizes ws_1 and ws_2 with max/min/mean pooling and $numFilter_A = 3$ filters.

6 Experiments and Results

Everything necessary to replicate our experimental results can be found in our open-source code repository.[4]

6.1 Tasks and Datasets

We consider three sentence pair similarity tasks:

1. Microsoft Research Paraphrase Corpus (MSRP). This data was collected from news sources (Dolan et al., 2004) and contains 5,801 pairs of sentences, with 4,076 for training and the remaining 1,725 for testing. Each sentence pair is annotated with a binary label indicating whether the two sentences are paraphrases, so the task here is binary classification.

2. Sentences Involving Compositional Knowledge (SICK) dataset. This data was collected for the 2014 SemEval competition (Marelli et al., 2014) and consists of 9,927 sentence pairs, with 4,500 for training, 500 as a development set, and the remaining 4,927 in the test set. The sentences are drawn from image and video descriptions. Each sentence pair is annotated with a relatedness score $\in [1, 5]$, with higher scores indicating the two sentences are more closely-related.

[4] http://hohocode.github.io/textSimilarityConvNet/

3. Microsoft Video Paraphrase Corpus (MSRVID). This dataset was collected for the 2012 SemEval competition and consists of 1,500 pairs of short video descriptions which were then annotated (Agirre et al., 2012). Half of it is for training and the other half is for testing. Each sentence pair has a relatedness score $\in [0, 5]$, with higher scores indicating the two sentences are more closely-related.

6.2 Training

We use a hinge loss for the MSRP paraphrase identification task. This is simpler than log loss since it only penalizes misclassified cases. The training objective is to minimize the following loss (summed over examples $\langle x, y_{gold} \rangle$):

$$loss(\theta, x, y_{gold}) = \sum_{y' \neq y_{gold}} \max(0, 1 + f_\theta(x, y') - f_\theta(x, y_{gold})) \quad (5)$$

where y_{gold} is the ground truth label, input x is the pair of sentences $x = \{S_1, S_2\}$, θ is the model weight vector to be trained, and the function $f_\theta(x, y)$ is the output of our model.

We use regularized KL-divergence loss for the semantic relatedness tasks (SICK and MSRVID), since the goal is to predict the similarity of the two sentences. The training objective is to minimize the KL-divergence loss plus an L_2 regularizer:

$$loss(\theta) = \frac{1}{m} \sum_{k=1}^{m} KL\left(f^k \,||\, \widehat{f}_\theta^k\right) + \frac{\lambda}{2}||\theta||_2^2 \quad (6)$$

where $\widehat{f}_\theta$ is the predicted distribution with model weight vector θ, f is the ground truth, m is the number of training examples, and λ is the regularization parameter. Note that we use the same KL-loss function and same sparse target distribution technique as Tai et al. (2015).

6.3 Experiment Settings

We conduct experiments with ws values in the range $[1, 3]$ as well as $ws = \infty$ (no convolution).

We use multiple kinds of embeddings to represent each sentence, both on words and part-of-speech (POS) tags. We use the $Dim_g = 300$-dimensional GloVe word embeddings (Pennington et al., 2014) trained on 840 billion tokens. We use $Dim_k = 25$-dimensional PARAGRAM vectors (Wieting et al., 2015) only on the MSRP task since they were developed for paraphrase tasks, having been trained on word pairs from the Paraphrase Database (Ganitkevitch et al., 2013). For POS embeddings, we run the Stanford POS tagger (Manning et al., 2014) on the English side of the Xinhua machine translation parallel corpus, which consists of Xinhua news articles with approximately 25 million words. We then train $Dim_p = 200$-dimensional POS embeddings using the `word2vec` toolkit (Mikolov et al., 2013). Adding POS embeddings is expected to retain syntactic information which is reported to be effective for paraphrase identification (Das and Smith, 2009). We use POS embeddings only for the MSRP task.

Therefore for MSRP, we concatenate all word and POS embeddings and obtain $Dim = Dim_g + Dim_p + Dim_k = 525$-dimension vectors for each input word; for SICK and MSRVID we only use $Dim = 300$-dimension GloVe embeddings.

We use 5-fold cross validation on the MSRP *training* data for tuning, then largely re-use the same hyperparameters for the other two datasets. However, there are two changes: 1) for the MSRP task we update word embeddings during training but not so on SICK and MSRVID tasks; 2) we set the fully connected layer to contain 250 hidden units for MSRP, and 150 for SICK and MSRVID. These changes were done to speed up our experimental cycle on SICK and MSRVID; on SICK data they are the same experimental settings as used by Tai et al. (2015), which makes for a cleaner empirical comparison.

We set the number of holistic filters in $block_A$ to be the same as the input word embeddings, therefore $numFilter_A = 525$ for MSRP and $numFilter_A = 300$ for SICK and MSRVID. We set the number of per-dimension filters in $block_B$ to be $numFilter_B = 20$ per dimension for all three datasets, which corresponds to $20 * Dim$ filters in total.

We perform optimization using stochastic gradient descent (Bottou, 1998). The backpropagation algorithm is used to compute gradients for all parameters during training (Goller and Kuchler, 1996). We fix the learning rate to 0.01 and regularization parameter $\lambda = 10^{-4}$.

6.4 Results on Three Datasets

Results on MSRP Data. We report F1 scores and accuracies from prior work in Table 1. Ap-

Model	Acc.	F1
Hu et al. (2014) ARC-I	69.6%	80.3%
Hu et al. (2014) ARC-II	69.9%	80.9%
Blacoe and Lapata (2012)	73.0%	82.3%
Fern and Stevenson (2008)	74.1%	82.4%
Finch (2005)	75.0%	82.7%
Das and Smith (2009)	76.1%	82.7%
Wan et al. (2006)	75.6%	83.0%
Socher et al. (2011)	76.8%	83.6%
Madnani et al. (2012)	77.4%	84.1%
Ji and Eisenstein (2013)	**80.41**%	**85.96**%
Yin and Schütze (2015) (without pretraining)	72.5%	81.4%
Yin and Schütze (2015) (with pretraining)	78.1%	84.4%
Yin and Schütze (2015) (pretraining+sparse features)	78.4%	84.6%
This work	**78.60**%	**84.73**%

Table 1: Test set results on MSRP for paraphrase identification. Rows in grey are neural network-based approaches.

proaches shown in gray rows of the table are based on neural networks. The recent approach by Yin and Schütze (2015) includes a pretraining technique which significantly improves results, as shown in the table. We do not use any pretraining but still slightly outperform their best results which use both pretraining and additional sparse features from Madnani et al. (2012).

When comparing to their model without pretraining, we outperform them by 6% absolute in accuracy and 3% in F1. Our model is also superior to other recent neural network models (Hu et al., 2014; Socher et al., 2011) without requiring sparse features or unlabeled data as in (Yin and Schütze, 2015; Socher et al., 2011). The best result on MSRP is from Ji and Eisenstein (2013) which uses unsupervised learning on the MSRP test set and rich sparse features.

Results on SICK Data. Our results on the SICK task are summarized in Table 2, showing Pearson's r, Spearman's ρ, and mean squared error (MSE). We include results from the literature as reported by Tai et al. (2015), including prior work using recurrent neural networks (RNNs), the best submissions in the SemEval-2014 competition, and variants of LSTMs. When measured by Pearson's r, the previous state-of-the-art approach uses a tree-structured LSTM (Tai et al., 2015); note that their best results require a dependency parser.

On the contrary, our approach does not rely on parse trees, nor do we use POS/PARAGRAM embeddings for this task. The word embeddings,

Model	r	ρ	MSE
Socher et al. (2014) DT-RNN	0.7863	0.7305	0.3983
Socher et al. (2014) SDT-RNN	0.7886	0.7280	0.3859
Lai and Hockenmaier (2014)	0.7993	0.7538	0.3692
Jimenez et al. (2014)	0.8070	0.7489	0.3550
Bjerva et al. (2014)	0.8268	0.7721	0.3224
Zhao et al. (2014)	0.8414	-	-
LSTM	0.8477	0.7921	0.2949
Bi-LSTM	0.8522	0.7952	0.2850
2-layer LSTM	0.8411	0.7849	0.2980
2-layer Bidirectional LSTM	0.8488	0.7926	0.2893
Tai et al. (2015) Const. LSTM	0.8491	0.7873	0.2852
Tai et al. (2015) Dep. LSTM	0.8676	**0.8083**	**0.2532**
This work	**0.8686**	0.8047	0.2606

Table 2: Test set results on SICK, as reported by Tai et al. (2015), grouped as: (1) RNN variants; (2) SemEval 2014 systems; (3) sequential LSTM variants; (4) dependency and constituency tree LSTMs (Tai et al., 2015). Evaluation metrics are Pearson's r, Spearman's ρ, and mean squared error (MSE).

Model	Pearson's r
Rios et al. (2012)	0.7060
Wang and Cer (2012)	0.8037
Beltagy et al. (2014)	0.8300
Bär et al. (2012)	0.8730
Šarić et al. (2012)	0.8803
This work	**0.9090**

Table 3: Test set results on MSRVID data. The Bär et al. (2012) and Šarić et al. (2012) results were the top two submissions in the Semantic Textual Similarity task at the SemEval-2012 competition.

sparse distribution targets, and KL loss function are exactly the same as used by Tai et al. (2015), therefore representing comparable conditions.

Results on MSRVID Data. Our results on the MSRVID data are summarized in Table 3, which includes the top 2 submissions in the Semantic Textual Similarity (STS) task from SemEval-2012. We find that we outperform the top system from the task by nearly 3 points in Pearson's r.

6.5 Model Ablation Study

We report the results of an ablation study in Table 4. We identify nine major components of our approach, remove one at a time (if applicable), and perform re-training and re-testing for all three tasks. We use the same experimental settings in Sec. 6.3 and report differences (in accuracy for MSRP, Pearson's r for SICK/MSRVID) compared to our results in Tables 1–3.

Gp	ID	Ablation Component	MSRP Accuracy Diff.	MSRVID Pearson Diff.	SICK Pearson Diff.
1	1	Remove POS embeddings (Sec. 6.3)	-0.81	NA	NA
	2	Remove PARAGRAM embeddings (Sec. 6.3)	-1.33	NA	NA
2	3	Remove per-dimension embeddings, building block A only (Sec. 4.1)	-0.75	-0.0067	-0.0014
	4	Remove $\min$ and mean pooling, use $\max$ pooling only (Sec. 4.2)	-0.58	-0.0112	+0.0001
	5	Remove multiple widths, $ws = 1$ and $ws = \infty$ only (Sec. 4.3)	-2.14	-0.0048	-0.0012
3	6	Remove cosine and $L_2 Euclid$ distance in $comU_*$ (Sec. 5.1)	-2.31	-0.0188	-0.0309
4	7	Remove Horizontal Algorithm (Sec. 5.2)	-0.92	-0.0097	-0.0117
	8	Remove Vertical Algorithm (Sec. 5.2)	-2.15	-0.0063	-0.0027
	9	Remove similarity layer (completely flatten) (Sec. 5)	-1.90	-0.0121	-0.0288

Table 4: Ablation study over test sets of all three datasets. Nine components are divided into four groups. We remove components one at a time and show differences.

The nine components can be divided into four groups: (1) input embeddings (components 1–2); (2) sentence modeling (components 3–5); (3) similarity measurement metrics (component 6); (4) similarity measurement layer (components 7–9). For MSRP, we use all nine components. For SICK and MSRVID, we use components 3–9 (as described in Sec. 6.3).

From Table 4 we find drops in performance for all components, with the largest differences appearing when removing components of the similarity measurement layer. For example, conducting comparisons over flattened sentence representations (removing component 9) leads to large drops across tasks, because this ignores structured information within sentence representations. Groups (1) and (2) are also useful, particularly for the MSRP task, demonstrating the extra benefit obtained from our multi-perspective approach in sentence modeling.

We see consistent drops when ablating the Vertical/Horizontal algorithms that target particular regions for comparison. Also, removing group (3) hinders both the Horizontal and Vertical algorithms (as described in Section 5.1), so its removal similarly causes large drops in performance. Though convolutional neural networks already perform strongly when followed by flattened vector comparison, we are able to leverage the full richness of the sentence models by performing structured similarity modeling on their outputs.

7 Discussion and Conclusion

On the SICK dataset, the dependency tree LSTM (Tai et al., 2015) and our model achieve comparable performance despite taking very different approaches. Tai et al. use syntactic parse trees and gating mechanisms to convert each sentence into a vector, while we use large sets of flexible feature extractors in the form of convolution filters, then compare particular subsets of features in our similarity measurement layer.

Our model architecture, with its many paths of information flow, is admittedly complex. Though we have removed hand engineering of features, we have added a substantial amount of functional architecture engineering. This may be necessary when using the small training sets provided for the tasks we consider here. We conjecture that a simpler, deeper neural network architecture may outperform our model when given large amounts of training data, but we leave an investigation of this direction to future work.

In summary, we developed a novel model for sentence similarity based on convolutional neural networks. We improved both sentence modeling and similarity measurement. Our model achieves highly competitive performance on three datasets. Ablation experiments show that the performance improvement comes from our use of multiple perspectives in both sentence modeling and structured similarity measurement over local regions of sentence representations. Future work could extend this model to related tasks including question answering and information retrieval.

Acknowledgments

This work was supported by the U.S. National Science Foundation under awards IIS-1218043 and CNS-1405688. Any opinions, findings, conclusions, or recommendations expressed are those of the authors and do not necessarily reflect the views of the sponsor. We would like to thank the anonymous reviewers for their feedback and CLIP labmates for their support.

References

Eneko Agirre, Mona Diab, Daniel Cer, and Aitor Gonzalez-Agirre. 2012. SemEval-2012 task 6: a pilot on semantic textual similarity. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics*, pages 385–393.

Galen Andrew, Raman Arora, Jeff Bilmes, and Karen Livescu. 2013. Deep canonical correlation analysis. In *Proceedings of the 30th International Conference on Machine Learning*, pages 1247–1255.

Daniel Bär, Chris Biemann, Iryna Gurevych, and Torsten Zesch. 2012. UKP: computing semantic textual similarity by combining multiple content similarity measures. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics*, pages 435–440.

Islam Beltagy, Katrin Erk, and Raymond Mooney. 2014. Probabilistic soft logic for semantic textual similarity. *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics*, pages 1210–1219.

Johannes Bjerva, Johan Bos, Rob van der Goot, and Malvina Nissim. 2014. The meaning factory: formal semantics for recognizing textual entailment and determining semantic similarity. *International Workshop on Semantic Evaluation*.

William Blacoe and Mirella Lapata. 2012. A comparison of vector-based representations for semantic composition. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 546–556.

Léon Bottou. 1998. Online learning and stochastic approximations. *On-line learning in neural networks*, 17(9):142.

Jane Bromley, James W Bentz, Léon Bottou, Isabelle Guyon, Yann LeCun, Cliff Moore, Eduard Säckinger, and Roopak Shah. 1993. Signature verification using a "siamese" time delay neural network. *International Journal of Pattern Recognition and Artificial Intelligence*, 7(4):669–688.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine learning*, pages 160–167.

Dipanjan Das and Noah A. Smith. 2009. Paraphrase identification as probabilistic quasi-synchronous recognition. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 468–476.

Bill Dolan, Chris Quirk, and Chris Brockett. 2004. Unsupervised construction of large paraphrase corpora: exploiting massively parallel news sources. In *Proceedings of the 20th International Conference on Computational Linguistics*.

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

Samuel Fern and Mark Stevenson. 2008. A semantic similarity approach to paraphrase detection. In *Computational Linguistics UK 11th Annual Research Colloquium*.

Andrew Finch. 2005. Using machine translation evaluation techniques to determine sentence-level semantic equivalence. In *Proceedings of the International Workshop on Paraphrasing*.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: the Paraphrase Database. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Christoph Goller and Andreas Kuchler. 1996. Learning task-dependent distributed representations by backpropagation through structure. In *Proceedings of the International Conference on Neural Networks*, pages 347–352.

Weiwei Guo and Mona Diab. 2012. Modeling sentences in the latent space. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, pages 864–872.

Samer Hassan. 2011. *Measuring Semantic Relatedness Using Salient Encyclopedic Concepts*. Ph.D. thesis, University of North Texas, Denton, Texas, USA.

Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. In *Advances in Neural Information Processing Systems*, pages 2042–2050.

Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM International Conference on Information & Knowledge Management*, pages 2333–2338.

Yangfeng Ji and Jacob Eisenstein. 2013. Discriminative improvements to distributional sentence similarity. In *Proceedings of the 2013 Conference on Empirical Methods for Natural Language Processing*, pages 891–896.

Sergio Jimenez, George Duenas, Julia Baquero, Alexander Gelbukh, Av Juan Dios Bátiz, and Av Mendizábal. 2014. UNAL-NLP: combining soft cardinality features for semantic textual similarity, relatedness and entailment. *International Workshop on Semantic Evaluation*.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods for Natural Language Processing*.

Alice Lai and Julia Hockenmaier. 2014. Illinois-LH: a denotational and distributional approach to semantics. *International Workshop on Semantic Evaluation*.

Nitin Madnani, Joel Tetreault, and Martin Chodorow. 2012. Re-examining machine translation metrics for paraphrase identification. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 182–190.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60.

Marco Marelli, Luisa Bentivogli, Marco Baroni, Raffaella Bernardi, Stefano Menini, and Roberto Zamparelli. 2014. SemEval-2014 task 1: evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. *International Workshop on Semantic Evaluation*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *Proceedings of Workshop at International Conference on Learning Representations*.

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

Steven Rennie, Vaibhava Goel, and Samuel Thomas. 2014. Deep order statistic networks. In *Proceedings of the IEEE Workshop on Spoken Language Technology*.

Miguel Rios, Wilker Aziz, and Lucia Specia. 2012. UOW: semantically informed text similarity. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics*, pages 673–678.

Frane Šarić, Goran Glavaš, Mladen Karan, Jan Šnajder, and Bojana Dalbelo Bašić. 2012. TakeLab: systems for measuring semantic text similarity. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics*, pages 441–448.

Richard Socher, Eric H. Huang, Jeffrey Pennington, Andrew Y. Ng, and Christopher D. Manning. 2011. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Advances in Neural Information Processing Systems*.

Richard Socher, Andrej Karpathy, Quoc V. Le, Christopher D. Manning, and Andrew Y. Ng. 2014. Grounded compositional semantics for finding and describing images with sentences. *Transactions of the Association for Computational Linguistics*, 2:207–218.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*.

Stephen Wan, Mark Dras, Robert Dale, and Cecile Paris. 2006. Using Dependency-based Features to Take the "Para-farce" out of Paraphrase. In *Australasian Language Technology Workshop*, pages 131–138.

Mengqiu Wang and Daniel Cer. 2012. Probabilistic edit distance metrics for STS. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics*, pages 648–654.

Jason Weston, Samy Bengio, and Nicolas Usunier. 2011. Wsabie: scaling up to large vocabulary image annotation. In *International Joint Conference on Artificial Intelligence*, pages 2764–2770.

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

Wei Xu, Alan Ritter, Chris Callison-Burch, William B. Dolan, and Yangfeng Ji. 2014. Extracting lexically divergent paraphrases from Twitter. *Transactions of the Association for Computational Linguistics*, 2:435–448.

Wenpeng Yin and Hinrich Schütze. 2015. Convolutional neural network for paraphrase identification. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 901–911.

Jiang Zhao, Tian Tian Zhu, and Man Lan. 2014. ECNU: one stone two birds: ensemble of heterogenous measures for semantic relatedness and textual entailment. *International Workshop on Semantic Evaluation*.

Xiaodan Zhu, Parinaz Sobhani, and Hongyu Guo. 2015. Long short-term memory over recursive structures. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 1604–1612.

Posterior calibration and exploratory analysis for natural language processing models

Khanh Nguyen
Department of Computer Science
University of Maryland, College Park
College Park, MD 20742
kxnguyen@cs.umd.edu

Brendan O'Connor
College of Information and Computer Sciences
University of Massachusetts, Amherst
Amherst, MA, 01003
brenocon@cs.umass.edu

Abstract

Many models in natural language processing define probabilistic distributions over linguistic structures. We argue that (1) the quality of a model's posterior distribution can and should be directly evaluated, as to whether probabilities correspond to empirical frequencies; and (2) NLP uncertainty can be projected not only to pipeline components, but also to exploratory data analysis, telling a user when to trust and not trust the NLP analysis. We present a method to analyze calibration, and apply it to compare the miscalibration of several commonly used models. We also contribute a coreference sampling algorithm that can create confidence intervals for a political event extraction task.[1]

1 Introduction

Natural language processing systems are imperfect. Decades of research have yielded analyzers that mis-identify named entities, mis-attach syntactic relations, and mis-recognize noun phrase coreference anywhere from 10-40% of the time. But these systems are accurate enough so that their outputs can be used as soft, if noisy, indicators of language meaning for use in downstream analysis, such as systems that perform question answering, machine translation, event extraction, and narrative analysis (McCord et al., 2012; Gimpel and Smith, 2008; Miwa et al., 2010; Bamman et al., 2013).

To understand the performance of an analyzer, researchers and practitioners typically measure the accuracy of individual labels or edges among a single predicted output structure $\mathbf{y}$, such as a most-probable tagging or entity clustering $\arg\max_{\mathbf{y}} P(\mathbf{y}|x)$ (conditional on text data x).

[1]See the extended version of this paper for software, appendix, and acknowledgments sections:
http://brenocon.com/nlpcalib/
http://arxiv.org/abs/1508.05154

But a probabilistic model gives a probability distribution over many other output structures that have smaller predicted probabilities; a line of work has sought to control cascading pipeline errors by passing on multiple structures from earlier stages of analysis, by propagating prediction uncertainty through multiple samples (Finkel et al., 2006), K-best lists (Venugopal et al., 2008; Toutanova et al., 2008), or explicitly diverse lists (Gimpel et al., 2013); often the goal is to marginalize over structures to calculate and minimize an expected loss function, as in minimum Bayes risk decoding (Goodman, 1996; Kumar and Byrne, 2004), or to perform joint inference between early and later stages of NLP analysis (e.g. Singh et al., 2013; Durrett and Klein, 2014).

These approaches should work better when the posterior probabilities of the predicted linguistic structures reflect actual probabilities of the structures or aspects of the structures. For example, say a model is overconfident: it places too much probability mass in the top prediction, and not enough in the rest. Then there will be little benefit to using the lower probability structures, since in the training or inference objectives they will be incorrectly outweighed by the top prediction (or in a sampling approach, they will be systematically undersampled and thus have too-low frequencies). If we only evaluate models based on their top predictions or on downstream tasks, it is difficult to diagnose this issue.

Instead, we propose to directly evaluate the *calibration* of a model's posterior prediction distribution. A perfectly calibrated model knows how often it's right or wrong; when it predicts an event with 80% confidence, the event empirically turns out to be true 80% of the time. While perfect accuracy for NLP models remains an unsolved challenge, perfect calibration is a more achievable goal, since a model that has imperfect accuracy could, in principle, be perfectly calibrated. In this paper, we develop a method to empirically analyze calibration that is appropriate for NLP models (§3)

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1587–1598,
Lisbon, Portugal, 17-21 September 2015.

and use it to analyze common generative and discriminative models for tagging and classification (§4).

Furthermore, if a model's probabilities are meaningful, that would justify using its probability distributions for any downstream purpose, including exploratory analysis on unlabeled data. In §6 we introduce a representative corpus exploration problem, identifying temporal event trends in international politics, with a method that is dependent on coreference resolution. We develop a coreference sampling algorithm (§5.2) which projects uncertainty into the event extraction, inducing a posterior distribution over event frequencies. Sometimes the event trends have very high posterior variance (large confidence intervals),[2] reflecting when the NLP system genuinely does not know the correct semantic extraction. This highlights an important use of a calibrated model: being able to tell a user when the model's predictions are likely to be incorrect, or at least, not giving a user a false sense of certainty from an erroneous NLP analysis.

2 Definition of calibration

Consider a binary probabilistic prediction problem, which consists of binary labels and probabilistic predictions for them. Each instance has a *ground-truth label* $y \in \{0, 1\}$, which is used for evaluation. The prediction problem is to generate a *predicted probability* or *prediction strength* $q \in [0, 1]$. Typically, we use some form of a probabilistic model to accomplish this task, where q represents the model's posterior probability[3] of the instance having a positive label ($y = 1$).

Let $S = \{(q_1, y_1), (q_2, y_2), \cdots (q_N, y_N)\}$ be the set of prediction-label pairs produced by the model. Many metrics assess the overall quality of how well the predicted probabilities match the data, such as the familiar cross entropy (negative average log-likelihood),

$$L_\ell(\vec{y}, \vec{q}) = \frac{1}{N} \sum_i y_i \log \frac{1}{q_i} + (1 - y_i) \log \frac{1}{1 - q_i}$$

or mean squared error, also known as the *Brier score* when y is binary (Brier, 1950),

$$L_2(\vec{y}, \vec{q}) = \frac{1}{N} \sum_i (y_i - q_i)^2$$

Both tend to attain better (lower) values when q is near 1 when $y = 1$, and near 0 when $y = 0$; and they achieve a perfect value of 0 when all $q_i = y_i$.[4]

Let $\mathbb{P}(y, q)$ be the joint empirical distribution over labels and predictions. Under this notation, $L_2 = \mathbb{E}_{q,y}[y - q]^2$. Consider the factorization

$$\mathbb{P}(y, q) = \mathbb{P}(y \mid q)\, \mathbb{P}(q)$$

where $\mathbb{P}(y \mid q)$ denotes the label empirical frequency, conditional on a prediction strength (Murphy and Winkler, 1987).[5] Applying this factorization to the Brier score leads to the calibration-refinement decomposition (DeGroot and Fienberg, 1983), in terms of expectations with respect to the prediction strength distribution $\mathbb{P}(q)$:

$$L_2 = \underbrace{\mathbb{E}_q[q - p_q]^2}_{\text{Calibration MSE}} + \underbrace{\mathbb{E}_q[p_q(1 - p_q)]}_{\text{Refinement}} \quad (1)$$

where we denote $p_q \equiv \mathbb{P}(y = 1 \mid q)$ for brevity.

Here, *calibration* measures to what extent a model's probabilistic predictions match their corresponding empirical frequencies. Perfect calibration is achieved when $\mathbb{P}(y = 1 \mid q) = q$ for all q; intuitively, if you aggregate all instances where a model predicted q, they should have $y = 1$ at q percent of the time. We define the magnitude of miscalibration using root mean squared error:

Definition 1 (RMS calibration error)**.**

$$CalibErr = \sqrt{\mathbb{E}_q[q - \mathbb{P}(y = 1 \mid q)]^2}$$

The second term of Eq 1 refers to *refinement*, which reflects to what extent the model is able to separate different labels (in terms of the conditional Gini entropy $p_q(1 - p_q)$). If the prediction strengths tend to cluster around 0 or 1, the refinement score tends to be lower. The calibration-refinement breakdown offers a useful perspective on the accuracy of a model posterior. This paper focuses on calibration.

There are several other ways to break down squared error, log-likelihood, and other probabilistic scoring rules.[6] We use the Brier-based calibration error in this work, since unlike cross-entropy

[2] We use the terms *confidence interval* and *credible interval* interchangeably in this work; the latter term is debatably more correct, though less widely familiar.

[3] Whether q comes from a Bayesian posterior or not is irrelevant to the analysis in this section. All that matters is that predictions are numbers $q \in [0, 1]$.

[4] These two loss functions are instances of *proper scoring rules* (Gneiting and Raftery, 2007; Bröcker, 2009).

[5] We alternatively refer to this as *label frequency* or *empirical frequency*. The $\mathbb{P}$ probabilities can be thought of as frequencies from the hypothetical population the data and predictions are drawn from. $\mathbb{P}$ probabilities are, definitionally speaking, completely separate from a probabilistic model that might be used to generate q predictions.

[6] They all include a notion of calibration corresponding to a Bregman divergence (Bröcker, 2009); for example, cross-entropy can be broken down such that KL divergence is the measure of miscalibration.

Algorithm 1 Estimate calibration error using adaptive binning.

Input: A set of N prediction-label pairs $\{(q_1, y_1), (q_2, y_2), \cdots, (q_N, y_N)\}$.

Output: Calibration error.

Parameter: Target bin size β.

Step 1: Sort pairs by prediction values q_k in ascending order.

Step 2: For each, assign bin label $b_k = \left\lfloor \frac{k-1}{\beta} \right\rfloor + 1$.

Step 3: Define each bin B_i as the set of indices of pairs that have the same bin label. If the last bin has size less than β, merge it with the second-to-last bin (if one exists). Let $\{B_1, B_2, \cdots, B_T\}$ be the set of bins.

Step 4: Calculate empirical and predicted probabilities per bin:

$$\hat{p}_i = \frac{1}{|B_i|} \sum_{k \in B_i} y_k \qquad \text{and} \qquad \hat{q}_i = \frac{1}{|B_i|} \sum_{k \in B_i} q_k$$

Step 5: Calculate the calibration error as the root mean squared error per bin, weighted by bin size in case they are not uniformly sized:

$$CalibErr = \sqrt{\frac{1}{N} \sum_{i=1}^{T} |B_i| (\hat{q}_i - \hat{p}_i)^2}$$

it does not tend toward infinity when near probability 0; we hypothesize this could be an issue since both p and q are subject to estimation error.

3 Empirical calibration analysis

From a test set of labeled data, we can analyze model calibration both in terms of the calibration error, as well as visualizing the *calibration curve* of label frequency versus predicted strength. However, computing the label frequencies $\mathbb{P}(y = 1|q)$ requires an infinite amount of data. Thus approximation methods are required to perform calibration analysis.

3.1 Adaptive binning procedure

Previous studies that assess calibration in supervised machine learning models (Niculescu-Mizil and Caruana, 2005; Bennett, 2000) calculate label frequencies by dividing the prediction space into deciles or other evenly spaced bins—e.g. $q \in [0, 0.1)$, $q \in [0.1, 0.2)$, etc.—and then calculating the empirical label frequency in each bin. This procedure may be thought of as using a form of nonparametric regression (specifically, a regressogram; Tukey 1961) to estimate the function $f(q) = \mathbb{P}(y = 1 \mid q)$ from observed data points. But models in natural language processing give very skewed distributions of confidence scores q (many are near 0 or 1), so this procedure performs poorly, having much more variable estimates near the middle of the q distribution (Figure 1).

We propose adaptive binning as an alternative. Instead of dividing the interval $[0, 1]$ into fixed-width bins, adaptive binning defines the bins such that there are an equal number of points in each, after which the same averaging procedure is used. This method naturally gives wider bins to area with fewer data points (areas that require more smoothing), and ensures that these areas have roughly similar standard errors as those near the boundaries, since for a bin with β number of points and empirical frequency p, the standard error is estimated by $\sqrt{p(1-p)/\beta}$, which is bounded above by $0.5/\sqrt{\beta}$. Algorithm 1 describes the procedure for estimating calibration error using adaptive binning, which can be applied to any probabilistic model that predicts posterior probabilities.

Algorithm 2 Estimate calibration error's confidence interval by sampling.

Input: A set of N prediction-label pairs $\{(q_1, y_1), (q_2, y_2), \cdots, (q_N, y_N)\}$.

Output: Calibration error with a 95% confidence interval.

Parameter: Number of samples, S.

Step 1: Calculate $\{\hat{p}_1, \hat{p}_2, \cdots, \hat{p}_T\}$ from step 4 of Algorithm 1.

Step 2: Draw S samples. For each $s = 1..S$,

- For each bin $i = 1..T$, draw $\hat{p}_i^{(s)} \sim \mathcal{N}\left(\hat{p}_i, \hat{\sigma}_i^2\right)$, where $\hat{\sigma}_i^2 = \hat{p}_i(1 - \hat{p}_i)/|B_i|$. If necessary clip to $[0, 1]$: $\hat{p}_i^{(s)} := \min(1, \max(0, \hat{p}_i^{(s)}))$
- Calculate the sample's $CalibErr$ from using the pairs $(\hat{q}_i, \hat{p}_i^{(s)})$ as per Step 5 of Algorithm 1.

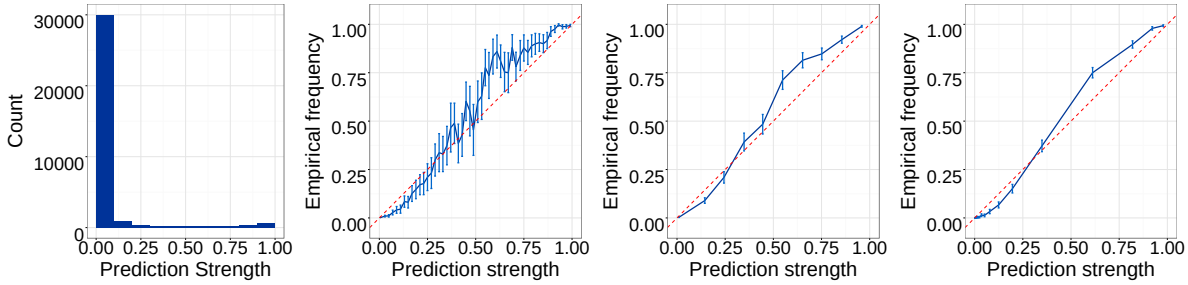
Step 3: Calculate the 95% confidence interval for the calibration error as:

$$CalibErr_{avg} \pm 1.96 \; \hat{s}_{error}$$

where $CalibErr_{avg}$ and $\hat{s}_{error}$ are the mean and the standard deviation, respectively, of the $CalibErr$s calculated from the samples.

3.2 Confidence interval estimation

Especially when the test set is small, estimating calibration error may be subject to error, due to uncertainty in the label frequency estimates. Since how to estimate confidence bands for nonparametric regression is an unsolved problem (Wasserman, 2006), we resort to a simple method based on the binning. We construct a binomial normal approximation for the label frequency estimate in each bin, and simulate from it; every simulation across all bins is used to construct a calibration error; these simulated calibration errors are collected to construct a normal approximation for the calibra-



(a) (b) (c) (d)

Figure 1: (a) A skewed distribution of predictions on whether a word has the NN tag (§4.2.2). Calibration curves produced by equally-spaced binning with bin width equal to 0.02 (b) and 0.1 (c) can have wide confidence intervals. Adaptive binning (with 1000 points in each bin) (d) gives small confidence intervals and also captures the prediction distribution. The confidence intervals are estimated as described in §3.1.

tion error estimate. Since we use bin sizes of at least $\beta \geq 200$ in our experiments, the central limit theorem justifies these approximations. We report all calibration errors along with their 95% confidence intervals calculated by Algorithm 2.[7]

3.3 Visualizing calibration

In order to better understand a model's calibration properties, we plot the pairs $(\hat{p}_1, \hat{q}_1), (\hat{p}_2, \hat{q}_2), \cdots, (\hat{p}_T, \hat{q}_T)$ obtained from the adaptive binning procedure to visualize the *calibration curve* of the model—this visualization is known as a *calibration* or *reliability plot*. It provides finer grained insight into the calibration behavior in different prediction ranges. A perfectly calibrated curve would coincide with the $y = x$ diagonal line. When the curve lies above the diagonal, the model is underconfident ($q < p_q$); and when it is below the diagonal, the model is overconfident ($q > p_q$).

An advantage of plotting a curve estimated from fixed-size bins, instead of fixed-width bins, is that the distribution of the points hints at the refinement aspect of the model's performance. If the points' positions tend to cluster in the bottom-left and top-right corners, that implies the model is making more refined predictions.

4 Calibration for classification and tagging models

Using the method described in §3, we assess the quality of posterior predictions of several classification and tagging models. In all of our experiments, we set the target bin size in Algorithm 1 to be 5,000 and the number of samples in Algorithm 2 to be 10,000.

4.1 Naive Bayes and logistic regression

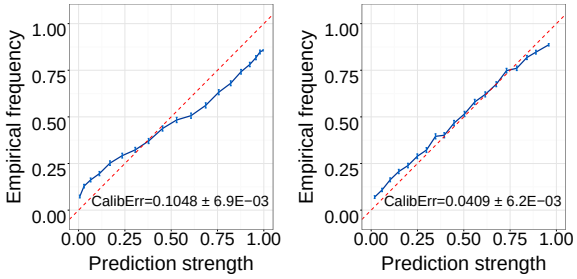
4.1.1 Introduction

Previous work on Naive Bayes has found its probabilities to have calibration issues, in part due to its incorrect conditional independence assumptions (Niculescu-Mizil and Caruana, 2005; Bennett, 2000; Domingos and Pazzani, 1997). Since logistic regression has the same log-linear representational capacity (Ng and Jordan, 2002) but does not suffer from the independence assumptions, we select it for comparison, hypothesizing it may have better calibration.

We analyze a binary classification task of Twitter sentiment analysis from emoticons. We collect a dataset consisting of tweets identified by the Twitter API as English, collected from 2014 to 2015, with the "emoticon trick" (Read, 2005; Lin and Kolcz, 2012) to label tweets that contain at least one occurrence of the smiley emoticon ":)" as "happy" ($y = 1$) and others as $y = 0$. The smiley emoticons are deleted in positive examples. We sampled three sets of tweets (subsampled from the Decahose/Gardenhose stream of public tweets) with Jan-Apr 2014 for training, May-Dec 2014 for development, and Jan-Apr 2015 for testing. Each set contains 10^5 tweets, split between an equal number of positive and negative instances. We use binary features based on unigrams extracted from the *twokenize.py*[8] tokenization. We use the *scikit-learn* (Pedregosa et al., 2011) implementations of Bernoulli Naive Bayes and L2-regularized logistic regression. The models' hyperparameters (Naive Bayes' smoothing paramter and logistic regression's regularization strength) are chosen to

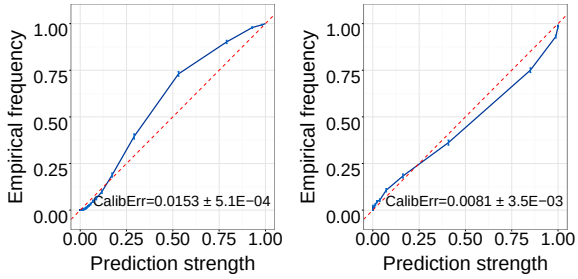
[7] A major unsolved issue is how to fairly select the bin size. If it is too large, the curve is oversmoothed and calibration looks better than it should be; if it is too small, calibration looks worse than it should be. Bandwidth selection and cross-validation techniques may better address this problem in future work. In the meantime, visualizations of calibration curves help inform the reader of the resolution of a particular analysis—if the bins are far apart, the data is sparse, and the specific details of the curve are not known in those regions.

[8] https://github.com/myleott/ark-twokenize-py



(a) (b)

Figure 2: Calibration curve of (a) Naive Bayes and (b) logistic regression on predicting whether a tweet is a "happy" tweet.



(a) (b)

Figure 3: Calibration curves of (a) HMM, and (b) CRF, on predictions over all POS tags.

maximize the F-1 score on the development set.

4.1.2 Results

Naive Bayes attains a slightly higher F-1 score (NB 73.8% vs. LR 72.9%), but logistic regression has much lower calibration error: less than half as much RMSE (NB 0.105 vs. LR 0.041; Figure 2). Both models have a tendency to be underconfident in the lower prediction range and overconfident in the higher range, but the tendency is more pronounced for Naive Bayes.

4.2 Hidden Markov models and conditional random fields

4.2.1 Introduction

Hidden Markov models (HMM) and linear chain conditional random fields (CRF) are another commonly used pair of analogous generative and discriminative models. They both define a posterior over tag sequences $P(\mathbf{y}|x)$, which we apply to part-of-speech tagging.

We can analyze these models in the binary calibration framework (§2-3) by looking at marginal distribution of binary-valued outcomes of parts of the predicted structures. Specifically, we examine calibration of predicted probabilities of individual tokens' tags (§4.2.2), and of pairs of consecutive tags (§4.2.3). These quantities are calculated with the forward-backward algorithm.

To prepare a POS tagging dataset, we extract *Wall Street Journal* articles from the English CoNLL-2011 coreference shared task dataset from Ontonotes (Pradhan et al., 2011), using the CoNLL-2011 splits for training, development and testing. This results in 11,772 sentences for training, 1,632 for development, and 1,382 for testing, over a set of 47 possible tags.

We train an HMM with Dirichlet MAP using one pseudocount for every transition and word emission. For the CRF, we use the L_2-regularized L-BFGS algorithm implemented in *CRFsuite* (Okazaki, 2007). We compare an HMM to a CRF that only uses basic transition (tag-tag) and emission (tag-word) features, so that it does not have an advantage due to more features. In order to compare models with similar task performance, we train the CRF with only 3000 sentences from the training set, which yields the same accuracy as the HMM (about 88.7% on the test set). In each case, the model's hyperparameters (the CRF's L_2 regularizer, the HMM's pseudocount) are selected by maximizing accuracy on the development set.

4.2.2 Predicting single-word tags

In this experiment, we measure miscalibration of the two models on predicting tags of single words. First, for each tag type, we produce a set of 33,306 prediction-label pairs (for every token); we then concatenate them across the tags for calibration analysis. Figure 3 shows that the two models exhibit distinct calibration patterns. The HMM tends to be very underconfident whereas the CRF is overconfident, and the CRF has a lower (better) overall calibration error.

We also examine the calibration errors of the individual POS tags (Figure 4(a)). We find that CRF is significantly better calibrated than HMM in most but not all categories (39 out of 47). For example, they are about equally calibrated on predicting the NN tag. The calibration gap between the two models also differs among the tags.

4.2.3 Predicting two-consecutive-word tags

There is no reason to restrict ourselves to model predictions of single words; these models define marginal distributions over larger textual units. Next we examine the calibration of posterior predictions of tag pairs on two consecutive words in the test set. The same analysis may be important for, say, phrase extraction or other chunking/parsing tasks.

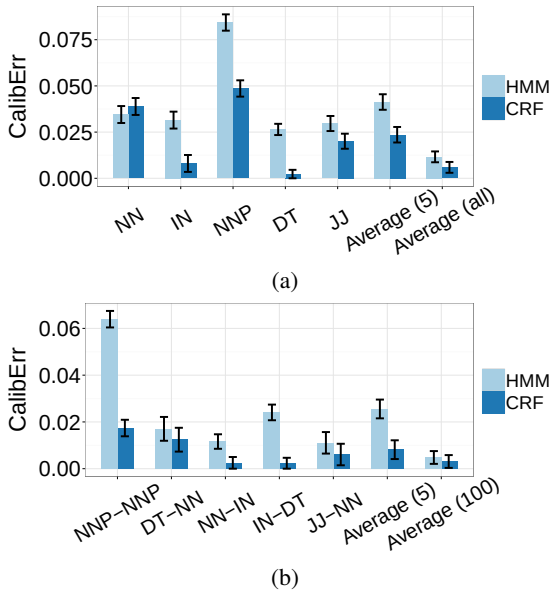


Figure 4: Calibration errors of HMM and CRF on predicting (a) single-word tags and (b) two-consecutive-word tags. Lower errors are better. The last two columns in each graph are the average calibration errors over the most common labels.

We report results for the top 5 and 100 most frequent tag pairs (Figure 4(b)). We observe a similar pattern as seen from the experiment on single tags: the CRF is generally better calibrated than the HMM, but the HMM does achieve better calibration errors in 29 out of 100 categories.

These tagging experiments illustrate that, depending on the application, different models can exhibit different levels of calibration.

5 Coreference resolution

We examine a third model, a probabilistic model for within-document noun phrase coreference, which has an efficient sampling-based inference procedure. In this section we introduce it and analyze its calibration, in preparation for the next section where we use it for exploratory data analysis.

5.1 Antecedent selection model

We use the Berkeley coreference resolution system (Durrett and Klein, 2013), which was originally presented as a CRF; we give it an equivalent a series of independent logistic regressions (see appendix for details). The primary component of this model is a locally-normalized log-linear distribution over clusterings of noun phrases, each cluster denoting an entity. The model takes a fixed input of N mentions (noun phrases), indexed by i in their positional order in the document. It posits that every mention i has a latent antecedent selection decision, $a_i \in \{1, \ldots, i-1, \text{NEW}\}$, denoting which previous mention it attaches to, or NEW if it is starting a new entity that has not yet been seen at a previous position in the text. Such a mention-mention attachment indicates coreference, while the final entity clustering includes more links implied through transitivity. The model's generative process is:

Definition 2 (Antecedent coreference model and sampling algorithm)**.**

- For $i = 1..N$, sample $a_i \sim \frac{1}{Z_i} \exp(\mathbf{w}^\top \mathbf{f}(i, a_i, x))$
- Calculate the entity clusters as $\mathbf{e} := CC(\mathbf{a})$, the connected components of the antecedent graph having edges (i, a_i) for i where $a_i \neq$ NEW.

Here x denotes all information in the document that is conditioned on for log-linear features $\mathbf{f}$. $\mathbf{e} = \{e_1, ...e_M\}$ denotes the entity clusters, where each element is a set of mentions. There are M entity clusters corresponding to the number of connected components in $\mathbf{a}$. The model defines a joint distribution over antecedent decisions $P(\mathbf{a}|x) = \prod_i P(a_i|x)$; it also defines a joint distribution over entity clusterings $P(\mathbf{e}|x)$, where the probability of an $\mathbf{e}$ is the sum of the probabilities of all $\mathbf{a}$ vectors that could give rise to it. In a manner similar to a distance-dependent Chinese restaurant process (Blei and Frazier, 2011), it is non-parametric in the sense that the number of clusters M is not fixed in advance.

5.2 Sampling-based inference

For both calibration analysis and exploratory applications, we need to analyze the posterior distribution over entity clusterings. This distribution is a complex mathematical object; an attractive approach to analyze it is to draw samples from this distribution, then analyze the samples.

This antecedent-based model admits a very straightforward procedure to draw independent $\mathbf{e}$ samples, by stepping through Def. 2: independently sample each a_i then calculate the connected components of the resulting antecedent graph. By construction, this procedure samples from the joint distribution of $\mathbf{e}$ (even though we never compute the probability of any single clustering $\mathbf{e}$).

Unlike approximate sampling approaches, such as Markov chain Monte Carlo methods used in other coreference work to sample $\mathbf{e}$ (Haghighi and Klein, 2007), here there are no questions about burn-in or autocorrelation (Kass et al., 1998). Every sample is independent and very fast to

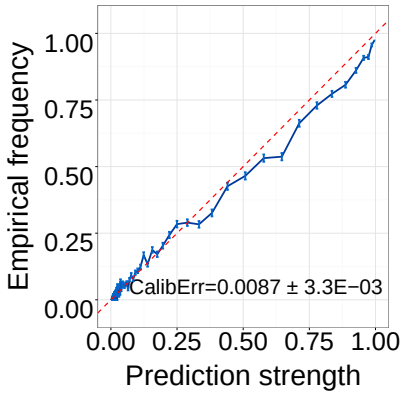


Figure 5: Coreference calibration curve for predicting whether two mentions belong to the same entity cluster.

compute—only slightly slower than calculating the MAP assignment (due to the exp and normalization for each a_i). We implement this algorithm by modifying the publicly available implementation from Durrett and Klein.[9]

5.3 Calibration analysis

We consider the following inference query: for a randomly chosen pair of mentions, are they coreferent? Even if the model's accuracy is comparatively low, it may be the case that it is correctly calibrated—if it thinks there should be great variability in entity clusterings, it may be uncertain whether a pair of mentions should belong together.

Let ℓ_{ij} be 1 if the mentions i and j are predicted to be coreferent, and 0 otherwise. Annotated data defines a gold-standard $\ell_{ij}^{(g)}$ value for every pair i, j. Any probability distribution over $\mathbf{e}$ defines a marginal Bernoulli distribution for every proposition ℓ_{ij}, marginalizing out $\mathbf{e}$:

$$P(\ell_{ij} = 1 \mid x) = \sum_{\mathbf{e}} 1\{(i,j) \in \mathbf{e}\} P(\mathbf{e} \mid x) \quad (2)$$

where $(i,j) \in \mathbf{e}$ is true iff there is an entity in $\mathbf{e}$ that contains both i and j.

In a traditional coreference evaluation of the best-prediction entity clustering, the model assigns 1 or 0 to every ℓ_{ij} and the pairwise precision and recall can be computed by comparing them to the corresponding $\ell_{ij}^{(g)}$. Here, we instead compare the $q_{ij} \equiv P(\ell_{ij} = 1 \mid x, \mathbf{e})$ prediction strengths against $\ell_{ij}^{(g)}$ empirical frequencies to assess pairwise calibration, with the same binary calibration analysis tools developed in §3 by aggregating pairs with similar q_{ij} values. Each q_{ij} is computed by averaging over 1,000 samples, simply taking the fraction of samples where the pair (i,j) is coreferent.

[9]Berkeley Coreference Resolution System, version 1.1: http://nlp.cs.berkeley.edu/projects/coref.shtml

We perform this analysis on the development section of the English CoNLL-2011 data (404 documents). Using the sampling inference method discussed in §5.2, we compute 4.3 millions prediction-label pairs and measure their calibration error. Our result shows that the model produces very well-calibrated predictions with less than 1% $CalibErr$ (Figure 5), though slightly overconfident on middle to high-valued predictions. The calibration error indicates that it is the most calibrated model we examine within this paper. This result suggests we might be able to trust its level of uncertainty.

6 Uncertainty in Entity-based Exploratory Analysis

6.1 Entity-syntactic event aggregation

We demonstrate one important use of calibration analysis: to ensure the usefulness of propagating uncertainty from coreference resolution into a system for exploring unannotated text. Accuracy cannot be calculated since there are no labels; but if the system is calibrated, we postulate that uncertainty information can help users understand the underlying reliability of aggregated extractions and isolate predictions that are more likely to contain errors.

We illustrate with an event analysis application to count the number of "country attack events": for a particular country of the world, how many news articles describe an entity affiliated with that country as the agent of an attack, and how does this number change over time? This is a simplified version of a problem where such systems have been built and used for political science analysis (Schrodt et al., 1994; Schrodt, 2012; Leetaru and Schrodt, 2013; Boschee et al., 2013; O'Connor et al., 2013). A coreference component can improve extraction coverage in cases such as "*Russian troops* were sighted . . . and *they attacked* . . . "

We use the coreference system examined in §5 for this analysis. To propagate coreference uncertainty, we re-run event extraction on multiple coreference samples generated from the algorithm described in §5.2, inducing a posterior distribution over the event counts. To isolate the effects of coreference, we use a very simple syntactic dependency system to identify affiliations and events. Assume the availability of dependency parses for a document d, a coreference resolution $\mathbf{e}$, and a lexicon of country names, which contains a small set of words $w(c)$ for each country c; for example, $w(\text{FRA}) = \{\text{france}, \text{french}\}$. The binary function

$f(c, e; x_d)$ assesses whether an entity e is affiliated with country c and is described as the agent of an attack, based on document text and parses x_d; f returns true iff both:[10]

- There exists a mention $i \in e$ described as country c: either its head word is in $w(c)$ (e.g. "Americans"), or its head word has an *nmod* or *amod* modifier in $w(c)$ (e.g. "American forces", "president of the U.S."); and there is only one unique country c among the mentions in the entity.
- There exists a mention $j \in e$ which is the *nsubj* or *agent* argument to the verb "attack" (e.g. "they attacked", "the forces attacked", "attacked by them").

For a given c, we first calculate a binary variable for whether there is at least one entity fulfilling f in a particular document,

$$a(d, c, \mathbf{e}_d) = \bigvee_{e \in \mathbf{e}_d} f(c, e; x_d) \quad (3)$$

and second, the number of such documents in $d(t)$, the set of *New York Times* articles published in a given time period t,

$$n(t, c, \mathbf{e}_{d(t)}) = \sum_{d \in d(t)} a(d, c, \mathbf{e}_d) \quad (4)$$

These quantities are both random variables, since they depend on $\mathbf{e}$; thus we are interested in the posterior distribution of n, marginalizing out $\mathbf{e}$,

$$P(n(t, c, \mathbf{e}_{d(t)}) \mid x_{d(t)}) \quad (5)$$

If our coreference model was highly certain (only one structure, or a small number of similar structures, had most of the probability mass in the space of all possible structures), each document would have an a posterior near either 0 or 1, and their sum in Eq. 5 would have a narrow distribution. But if the model is uncertain, the distribution will be wider. Because of the transitive closure, the probability of a is potentially more complex than the single antecedent linking probability between two mentions—the affiliation and attack information can propagate through a long coreference chain.

6.2 Results

We tag and parse a 193,403 article subset of the Annotated New York Times LDC corpus (Sandhaus, 2008), which includes articles about world news from the years 1987 to 2007 (details in appendix). For each article, we run the coreference system to predict 100 samples, and evaluate f on every entity in every sample.[11] The quantity of interest is the number of articles mentioning attacks in a 3-month period (quarter), for a given country. Figure 6 illustrates the mean and 95% posterior credible intervals for each quarter. The posterior mean m is calculated as the mean of the samples, and the interval is the normal approximation $m \pm 1.96\ s$, where s is the standard deviation among samples for that country and time period.

Uncertainty information helps us understand whether a difference between data points is real. In the plots of Figure 6, if we had used a 1-best coreference resolution, only a single line would be shown, with no assessment of uncertainty. This is problematic in cases when the model genuinely does not know the correct answer. For example, the 1993-1996 period of the USA plot (Figure 6, top) shows the posterior mean fluctuating from 1 to 5 documents; but when credible intervals are taken into consideration, we see that model does not know whether the differences are real, or were caused by coreference noise.

A similar case is highlighted at the bottom plot of Figure 6. Here we compare the event counts for Yugoslavia and NATO, which were engaged in a conflict in 1999. Did the *New York Times* devote more attention to the attacks by one particular side? To a 1-best system, the answer would be yes. But the posterior intervals for the two countries' event counts in mid-1999 heavily overlap, indicating that the coreference system introduces too much uncertainty to obtain a conclusive answer for this question. Note that calibration of the coreference model is important for the credible intervals to be useful; for example, if the model was badly calibrated by being overconfident (too much probability over a small set of similar structures), these intervals would be too narrow, leading to incorrect interpretations of the event dynamics.

Visualizing this uncertainty gives richer information for a potential user of an NLP-based system, compared to simply drawing a line based on a single 1-best prediction. It preserves the genuine uncertainty due to ambiguities the system was unable to resolve. This highlights an alternative use of Finkel et al. (2006)'s approach of sampling multiple NLP pipeline components, which in that work was used to perform joint inference. Instead

[10] Syntactic relations are Universal Dependencies (de Marneffe et al., 2014); more details for the extraction rules are in the appendix.

[11] We obtained similar results using only 10 samples. We also obtained similar results with a different query function, the total number of entities, across documents, that fulfill f.

of focusing on improving an NLP pipeline, we can pass uncertainty on to exploratory purposes, and try to highlight to a user where the NLP system may be wrong, or where it can only imprecisely specify a quantity of interest.

Finally, calibration can help error analysis. For a calibrated model, the more uncertain a prediction is, the more likely it is to be erroneous. While coreference errors comprise only one part of event extraction errors (alongside issues in parse quality, factivity, semantic roles, etc.), we can look at highly uncertain event predictions to understand the nature of coreference errors relative to our task. We manually analyzed documents with a 50% probability to contain an "attack"ing country-affiliated entity, and found difficult coreference cases.

In one article from late 1990, an "attack" event for IRQ is extracted from the sentence "But some political leaders said that they feared that *Mr. Hussein might attack* Saudi Arabia". The mention "Mr. Hussein" is classified as IRQ only when it is coreferent with a previous mention "President Saddam Hussein of Iraq"; this occurs only 50% of the time, since in some posterior samples the coreference system split apart these two "Hussein" mentions. This particular document is additionally difficult, since it includes the names of more than 10 countries (e.g. United States, Saudi Arabia, Egypt), and some of the Hussein mentions are even clustered with presidents of other countries (such as "President Bush"), presumably because they share the "president" title. These types of errors are a major issue for a political analysis task; further analysis could assess their prevalence and how to address them in future work.

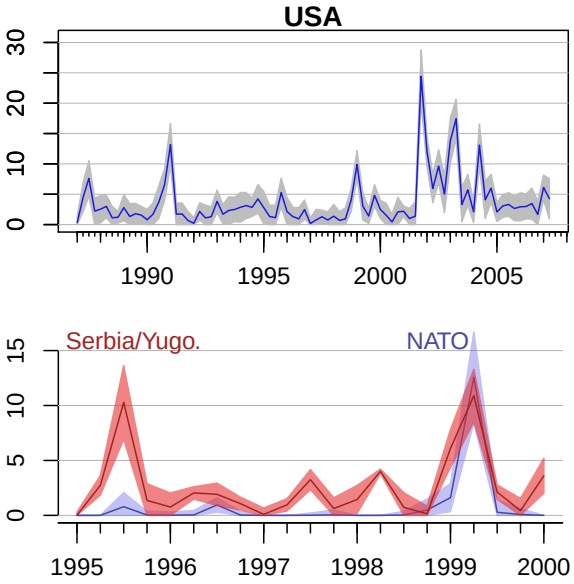


Figure 6: Number of documents with an "attack"ing country per 3-month period, and coreference posterior uncertainty for that quantity. The dark line is the posterior mean, and the shaded region is the 95% posterior credible interval. More examples in appendix.

7 Conclusion

In this work, we argue that the calibration of posterior predictions is a desirable property of probabilistic NLP models, and that it can be directly evaluated. We also demonstrate a use case of having calibrated uncertainty: its propagation into downstream exploratory analysis.

Our posterior simulation approach for exploratory and error analysis relates to *posterior predictive checking* (Gelman et al., 2013), which analyzes a posterior to test model assumptions; Mimno and Blei (2011) apply it to a topic model.

One avenue of future work is to investigate more effective nonparametric regression methods to better estimate and visualize calibration error, such as Gaussian processes or bootstrapped kernel density estimation.

Another important question is: what types of inferences are facilitated by correct calibration? Intuitively, we think that overconfidence will lead to overly narrow confidence intervals; but in what sense are confidence intervals "good" when calibration is perfect? Also, does calibration help joint inference in NLP pipelines? It may also assist calculations that rely on expectations, such as inference methods like minimum Bayes risk decoding, or learning methods like EM, since calibrated predictions imply that calculated expectations are statistically unbiased (though the implications of this fact may be subtle). Finally, it may be interesting to pursue recalibration methods, which readjust a non-calibrated model's predictions to be calibrated; recalibration methods have been developed for binary (Platt, 1999; Niculescu-Mizil and Caruana, 2005) and multiclass (Zadrozny and Elkan, 2002) classification settings, but we are unaware of methods appropriate for the highly structured outputs typical in linguistic analysis. Another approach might be to directly constrain $CalibErr = 0$ during training, or try to reduce it as a training-time risk minimization or cost objective (Smith and Eisner, 2006; Gimpel and Smith, 2010; Stoyanov et al., 2011; Brümmer and Doddington, 2013).

Calibration is an interesting and important property of NLP models. Further work is necessary to address these and many other questions.

References

David Bamman, Brendan O'Connor, and Noah A. Smith. Learning latent personas of film characters. In *Proceedings of ACL*, 2013.

Paul N. Bennett. Assessing the calibration of naive Bayes' posterior estimates. Technical report, Carnegie Mellon University, 2000.

David M. Blei and Peter I. Frazier. Distance dependent Chinese restaurant processes. *The Journal of Machine Learning Research*, 12: 2461–2488, 2011.

Elizabeth Boschee, Premkumar Natarajan, and Ralph Weischedel. Automatic extraction of events from open source text for predictive forecasting. *Handbook of Computational Approaches to Counterterrorism*, page 51, 2013.

Glenn W. Brier. Verification of forecasts expressed in terms of probability. *Monthly weather review*, 78(1):1–3, 1950.

Jochen Bröcker. Reliability, sufficiency, and the decomposition of proper scores. *Quarterly Journal of the Royal Meteorological Society*, 135(643):1512–1519, 2009.

Niko Brümmer and George Doddington. Likelihood-ratio calibration using prior-weighted proper scoring rules. *arXiv preprint arXiv:1307.7981*, 2013. Interspeech 2013.

Marie-Catherine de Marneffe, Timothy Dozat, Natalia Silveira, Katri Haverinen, Filip Ginter, Joakim Nivre, and Christopher D. Manning. Universal Stanford dependencies: A cross-linguistic typology. In *Proceedings of LREC*, 2014.

Morris H. DeGroot and Stephen E. Fienberg. The comparison and evaluation of forecasters. *The statistician*, pages 12–22, 1983.

Pedro Domingos and Michael Pazzani. On the optimality of the simple Bayesian classifier under zero-one loss. *Machine learning*, 29(2-3):103–130, 1997.

Greg Durrett and Dan Klein. Easy victories and uphill battles in coreference resolution. In *EMNLP*, pages 1971–1982, 2013.

Greg Durrett and Dan Klein. A joint model for entity analysis: Coreference, typing, and linking. *Transactions of the Association for Computational Linguistics*, 2:477–490, 2014.

Jenny Rose Finkel, Christopher D. Manning, and Andrew Y. Ng. Solving the problem of cascading errors: Approximate Bayesian inference for linguistic annotation pipelines. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 618–626. Association for Computational Linguistics, 2006.

Andrew Gelman, John B. Carlin, Hal S. Stern, David B. Dunson, Aki Vehtari, and Donald B. Rubin. *Bayesian data analysis*. Chapman and Hall/CRC, 3rd edition, 2013.

Kevin Gimpel and Noah A. Smith. Rich source-side context for statistical machine translation. In *Proceedings of the Third Workshop on Statistical Machine Translation*, pages 9–17, 2008.

Kevin Gimpel and Noah A. Smith. Softmax-margin CRFs: Training log-linear models with cost functions. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 733–736. Association for Computational Linguistics, 2010.

Kevin Gimpel, Dhruv Batra, Chris Dyer, and Gregory Shakhnarovich. A systematic exploration of diversity in machine translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1100–1111, Seattle, Washington, USA, October 2013. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/D13-1111.

Tilmann Gneiting and Adrian E. Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102(477):359–378, 2007.

Joshua Goodman. Parsing algorithms and metrics. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, pages 177–183, Santa Cruz, California, USA, June 1996. Association for Computational Linguistics. doi: 10.3115/981863.981887. URL http://www.aclweb.org/anthology/P96-1024.

Aria Haghighi and Dan Klein. Unsupervised coreference resolution in a nonparametric Bayesian model. In *Annual Meeting, Association for Computational Linguistics*, volume 45, page 848, 2007.

Robert E. Kass, Bradley P. Carlin, Andrew Gelman, and Radford M. Neal. Markov chain Monte Carlo in practice: a roundtable discussion. *The American Statistician*, 52(2):93–100, 1998.

Shankar Kumar and William Byrne. Minimum Bayes-risk decoding for statistical machine translation. In Daniel Marcu Susan Dumais and Salim Roukos, editors, *HLT-NAACL 2004: Main Proceedings*, pages 169–176, Boston, Massachusetts, USA, May 2 - May 7 2004. Association for Computational Linguistics.

Kalev Leetaru and Philip A. Schrodt. GDELT: Global data on events, location, and tone, 1979–2012. In *ISA Annual Convention*, volume 2, page 4, 2013.

Jimmy Lin and Alek Kolcz. Large-scale machine learning at Twitter. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, pages 793–804. ACM, 2012.

Michael C. McCord, J. William Murdock, and Branimir K. Boguraev. Deep parsing in Watson. *IBM Journal of Research and Development*, 56 (3.4):3–1, 2012.

David Mimno and David Blei. Bayesian checking for topic models. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 227–237, Edinburgh, Scotland, UK., July 2011. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/D11-1021.

Makoto Miwa, Sampo Pyysalo, Tadayoshi Hara, and Jun'ichi Tsujii. Evaluating dependency representations for event extraction. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 779–787, Beijing, China, August 2010. Coling 2010 Organizing Committee. URL http://www.aclweb.org/anthology/C10-1088.

Allan H. Murphy and Robert L. Winkler. A general framework for forecast verification. *Monthly Weather Review*, 115(7):1330–1338, 1987.

Andrew Ng and Michael Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive Bayes. *Advances in neural information processing systems*, 14: 841, 2002.

Alexandru Niculescu-Mizil and Rich Caruana. Predicting good probabilities with supervised learning. In *Proceedings of the 22nd International Conference on Machine Learning*, pages 625–632, 2005.

Brendan O'Connor, Brandon Stewart, and Noah A. Smith. Learning to extract international relations from political context. In *Proceedings of ACL*, 2013.

Naoaki Okazaki. Crfsuite: a fast implementation of conditional random fields (CRFs), 2007. URL http://www.chokkan.org/software/crfsuite/.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

John Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances in large margin classifiers*. MIT Press (2000), 1999. URL http://research.microsoft.com/pubs/69187/svmprob.ps.gz.

Sameer Pradhan, Lance Ramshaw, Mitchell Marcus, Martha Palmer, Ralph Weischedel, and Nianwen Xue. CoNLL-2011 shared task: Modeling unrestricted coreference in Ontonotes. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–27. Association for Computational Linguistics, 2011.

Jonathon Read. Using emoticons to reduce dependency in machine learning techniques for sentiment classification. In *Proceedings of the ACL Student Research Workshop*, pages 43–48. Association for Computational Linguistics, 2005.

Evan Sandhaus. The New York Times Annotated Corpus. *Linguistic Data Consortium*, LDC2008T19, 2008.

Philip A. Schrodt. Precedents, progress, and prospects in political event data. *International Interactions*, 38(4):546–569, 2012.

Philip A. Schrodt, Shannon G. Davis, and Judith L. Weddle. KEDS – a program for the machine coding of event data. *Social Science Computer Review*, 12(4):561 –587, December 1994. doi: 10.1177/089443939401200408. URL http://ssc.sagepub.com/content/12/4/561.abstract.

Sameer Singh, Sebastian Riedel, Brian Martin, Jiaping Zheng, and Andrew McCallum. Joint inference of entities, relations, and coreference.

In *Proceedings of the 2013 Workshop on Automated Knowledge Base Construction*, pages 1–6. ACM, 2013.

David A. Smith and Jason Eisner. Minimum risk annealing for training log-linear models. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, pages 787–794, Sydney, Australia, July 2006. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P06-2101.

Veselin Stoyanov, Alexander Ropson, and Jason Eisner. Empirical risk minimization of graphical model parameters given approximate inference, decoding, and model structure. In *International Conference on Artificial Intelligence and Statistics*, pages 725–733, 2011.

Kristina Toutanova, Aria Haghighi, and Christopher D. Manning. A global joint model for semantic role labeling. *Computational Linguistics*, 34(2):161–191, 2008.

John W. Tukey. Curves as parameters, and touch estimation. In *Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Contributions to the Theory of Statistics*, pages 681–694, Berkeley, Calif., 1961. University of California Press. URL http://projecteuclid.org/euclid.bsmsp/1200512189.

Ashish Venugopal, Andreas Zollmann, Noah A. Smith, and Stephan Vogel. Wider pipelines: N-best alignments and parses in MT training. In *Proceedings of AMTA*, 2008.

Larry Wasserman. *All of nonparametric statistics*. Springer Science & Business Media, 2006.

Bianca Zadrozny and Charles Elkan. Transforming classifier scores into accurate multiclass probability estimates. In *Proceedings of KDD*, pages 694–699. ACM, 2002.

A Generative Word Embedding Model and its Low Rank Positive Semidefinite Solution

Shaohua Li[1], **Jun Zhu**[2], **Chunyan Miao**[1]
[1]Joint NTU-UBC Research Centre of Excellence in Active Living for the Elderly (LILY),
Nanyang Technological University, Singapore
[2]Tsinghua University, P.R. China
lish0018@ntu.edu.sg, dcszj@tsinghua.edu.cn, ascymiao@ntu.edu.sg

Abstract

Most existing word embedding methods can be categorized into Neural Embedding Models and Matrix Factorization (MF)-based methods. However some models are opaque to probabilistic interpretation, and MF-based methods, typically solved using Singular Value Decomposition (SVD), may incur loss of corpus information. In addition, it is desirable to incorporate global latent factors, such as topics, sentiments or writing styles, into the word embedding model. Since generative models provide a principled way to incorporate latent factors, we propose a generative word embedding model, which is easy to interpret, and can serve as a basis of more sophisticated latent factor models. The model inference reduces to a low rank weighted positive semidefinite approximation problem. Its optimization is approached by eigendecomposition on a submatrix, followed by online blockwise regression, which is scalable and avoids the information loss in SVD. In experiments on 7 common benchmark datasets, our vectors are competitive to word2vec, and better than other MF-based methods.

1 Introduction

The task of word embedding is to model the distribution of a word and its context words using their corresponding vectors in a Euclidean space. Then by doing regression on the relevant statistics derived from a corpus, a set of vectors are recovered which best fit these statistics. These vectors, commonly referred to as the *embeddings*, capture semantic/syntactic regularities between the words.

The core of a word embedding method is the *link function* that connects the input — the embeddings, with the output — certain corpus statistics. Based on the link function, the objective function is developed. The reasonableness of the link function impacts the quality of the obtained embeddings, and different link functions are amenable to different optimization algorithms, with different scalability. Based on the forms of the link function and the optimization techniques, most methods can be divided into two classes: the traditional *neural embedding models*, and more recent *low rank matrix factorization methods*.

The neural embedding models use the **softmax** link function to model the conditional distribution of a word given its context (or vice versa) as a function of the embeddings. The normalizer in the softmax function brings intricacy to the optimization, which is usually tackled by gradient-based methods. The pioneering work was (Bengio et al., 2003). Later Mnih and Hinton (2007) propose three different link functions. However there are interaction matrices between the embeddings in all these models, which complicate and slow down the training, hindering them from being trained on huge corpora. Mikolov et al. (2013a) and Mikolov et al. (2013b) greatly simplify the conditional distribution, where the two embeddings interact directly. They implemented the well-known "word2vec", which can be trained efficiently on huge corpora. The obtained embeddings show excellent performance on various tasks.

Low-Rank Matrix Factorization (MF in short) methods include various link functions and optimization methods. The link functions are usually not softmax functions. MF methods aim to reconstruct certain corpus statistics matrix by the product of two low rank factor matrices. The objective is usually to minimize the reconstruction error, optionally with other constraints. In this line of research, Levy and Goldberg (2014b) find that "word2vec" is essentially doing stochastic weighted factorization of the word-context pointwise mutual information (PMI) matrix. They then

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1599–1609,
Lisbon, Portugal, 17-21 September 2015.

factorize this matrix directly as a new method. Pennington et al. (2014) propose a bilinear regression function of the conditional distribution, from which a weighted MF problem on the bigram log-frequency matrix is formulated. Gradient Descent is used to find the embeddings. Recently, based on the intuition that words can be organized in semantic hierarchies, Yogatama et al. (2015) add hierarchical sparse regularizers to the matrix reconstruction error. With similar techniques, Faruqui et al. (2015) reconstruct a set of pretrained embeddings using sparse vectors of greater dimensionality. Dhillon et al. (2015) apply Canonical Correlation Analysis (CCA) to the word matrix and the context matrix, and use the canonical correlation vectors between the two matrices as word embeddings. Stratos et al. (2014) and Stratos et al. (2015) assume a Brown language model, and prove that doing CCA on the bigram occurrences is equivalent to finding a transformed solution of the language model. Arora et al. (2015) assume there is a hidden discourse vector on a random walk, which determines the distribution of the current word. The slowly evolving discourse vector puts a constraint on the embeddings in a small text window. The maximum likelihood estimate of the embeddings within this text window approximately reduces to a squared norm objective.

There are two limitations in current word embedding methods. The first limitation is, all MF-based methods map words and their context words to two different sets of embeddings, and then employ Singular Value Decomposition (SVD) to obtain a low rank approximation of the word-context matrix $\boldsymbol{M}$. As SVD factorizes $\boldsymbol{M}^\top \boldsymbol{M}$, some information in $\boldsymbol{M}$ is lost, and the learned embeddings may not capture the most significant regularities in $\boldsymbol{M}$. Appendix A gives a toy example on which SVD does not work properly.

The second limitation is, a generative model for documents parametered by embeddings is absent in recent development. Although (Stratos et al., 2014; Stratos et al., 2015; Arora et al., 2015) are based on generative processes, the generative processes are only for deriving the local relationship between embeddings within a small text window, leaving the likelihood of a document undefined. In addition, the learning objectives of some models, e.g. (Mikolov et al., 2013b, Eq.1), even have no clear probabilistic interpretation. A generative word embedding model for documents is not only easier to interpret and analyze, but more importantly, provides a basis upon which document-level *global* latent factors, such as document topics (Wallach, 2006), sentiments (Lin and He, 2009), writing styles (Zhao et al., 2011b), can be incorporated in a principled manner, to better model the text distribution and extract relevant information.

Based on the above considerations, we propose to unify the embeddings of words and context words. Our link function factorizes into three parts: the interaction of two embeddings capturing linear correlations of two words, a residual capturing nonlinear or noisy correlations, and the unigram priors. To reduce overfitting, we put Gaussian priors on embeddings and residuals, and apply Jelinek-Mercer Smoothing to bigrams. Furthermore, to model the probability of a sequence of words, we assume that the contributions of more than one context word approximately add up. Thereby a generative model of documents is constructed, parameterized by embeddings and residuals. The learning objective is to maximize the corpus likelihood, which reduces to a weighted low-rank positive semidefinite (PSD) approximation problem of the PMI matrix. A Block Coordinate Descent algorithm is adopted to find an approximate solution. This algorithm is based on Eigendecomposition, which avoids information loss in SVD, but brings challenges to scalability. We then exploit the sparsity of the weight matrix and implement an efficient online blockwise regression algorithm. On seven benchmark datasets covering similarity and analogy tasks, our method achieves competitive and stable performance.

The source code of this method is provided at `https://github.com/askerlee/topicvec`.

2 Notations and Definitions

Throughout the paper, we always use a uppercase bold letter as $\boldsymbol{S}, \boldsymbol{V}$ to denote a matrix or set, a lowercase bold letter as $\boldsymbol{v}_{w_i}$ to denote a vector, a normal uppercase letter as N, W to denote a scalar constant, and a normal lowercase letter as s_i, w_i to denote a scalar variable.

Suppose a vocabulary $\boldsymbol{S} = \{s_1, \cdots, s_W\}$ consists of all the words, where W is the vocabulary size. We further suppose $s_1, \cdots, s_W$ are sorted in decending order of the frequency, i.e. s_1 is most frequent, and s_W is least frequent. A document d_i is a sequence of words $d_i = (w_{i1}, \cdots, w_{iL_i}), w_{ij} \in \boldsymbol{S}$. A corpus is a collec-

Name	Description
$\boldsymbol{S}$	Vocabulary $\{s_1, \cdots, s_W\}$
$\boldsymbol{V}$	Embedding matrix $(\boldsymbol{v}_{s_1}, \cdots, \boldsymbol{v}_{s_W})$
$\boldsymbol{D}$	Corpus $\{d_1, \cdots, d_M\}$
$\boldsymbol{v}_{s_i}$	Embedding of word s_i
$a_{s_i s_j}$	Bigram residual for s_i, s_j
$\tilde{P}(s_i,s_j)$	Empirical probability of s_i, s_j in the corpus
$\boldsymbol{u}$	Unigram probability vector $(P(s_1),\cdots,P(s_W))$
$\boldsymbol{A}$	Residual matrix $(a_{s_i s_j})$
$\boldsymbol{B}$	Conditional probability matrix $\Big(P(s_j\|s_i)\Big)$
$\boldsymbol{G}$	PMI matrix $\Big(\text{PMI}(s_i, s_j)\Big)$
$\boldsymbol{H}$	Bigram empirical probability matrix $\Big(\tilde{P}(s_i, s_j)\Big)$

Table 1: Notation Table

tion of M documents $\boldsymbol{D} = \{d_1, \cdots, d_M\}$. In the vocabulary, each word s_i is mapped to a vector $\boldsymbol{v}_{s_i}$ in N-dimensional Euclidean space.

In a document, a sequence of words is referred to as a *text window*, denoted by $w_i, \cdots, w_{i+l}$, or $w_i{:}w_{i+l}$ in shorthand. A text window of chosen size c before a word w_i defines the *context* of w_i as $w_{i-c}, \cdots, w_{i-1}$. Here w_i is referred to as the *focus word*. Each context word w_{i-j} and the focus word w_i comprise a bigram w_{i-j}, w_i.

The *Pointwise Mutual Information* between two words s_i, s_j is defined as

$$\text{PMI}(s_i, s_j) = \log \frac{P(s_i, s_j)}{P(s_i)P(s_j)}.$$

3 Link Function of Text

In this section, we formulate the probability of a sequence of words as a function of their embeddings. We start from the link function of bigrams, which is the building blocks of a long sequence. Then this link function is extended to a text window with c context words, as a first-order approximation of the actual probability.

3.1 Link Function of Bigrams

We generalize the link function of "word2vec" and "GloVe" to the following:

$$P(s_i, s_j) = \exp\Big\{\boldsymbol{v}_{s_j}^\top \boldsymbol{v}_{s_i} + a_{s_i s_j}\Big\} P(s_i)P(s_j) \quad (1)$$

The rationale for (1) originates from the idea of the *Product of Experts* in (Hinton, 2002). Suppose different types of semantic/syntactic regularities between s_i and s_j are encoded in different dimensions of $\boldsymbol{v}_{s_i}, \boldsymbol{v}_{s_j}$. As $\exp\{\boldsymbol{v}_{s_j}^\top \boldsymbol{v}_{s_i}\} = \prod_l \exp\{v_{s_i,l} \cdot v_{s_j,l}\}$, this means the effects of different regularities on the probability are combined by multiplying together. If s_i and s_j are independent, their joint probability should be $P(s_i)P(s_j)$. In the presence of correlations, the actual joint probability $P(s_i, s_j)$ would be a scaling of it. The scale factor reflects how much s_i and s_j are positively or negatively correlated. Within the scale factor, $\boldsymbol{v}_{s_j}^\top \boldsymbol{v}_{s_i}$ captures linear interactions between s_i and s_j, the residual $a_{s_i s_j}$ captures nonlinear or noisy interactions. In applications, only $\boldsymbol{v}_{s_j}^\top \boldsymbol{v}_{s_i}$ is of interest. Hence the bigger magnitude $\boldsymbol{v}_{s_j}^\top \boldsymbol{v}_{s_i}$ is of relative to $a_{s_i s_j}$, the better.

Note that we do *not* assume $a_{s_i s_j} = a_{s_j s_i}$. This provides the flexibility $P(s_i, s_j) \neq P(s_j, s_i)$, agreeing with the asymmetry of bigrams in natural languages. At the same time, $\boldsymbol{v}_{s_j}^\top \boldsymbol{v}_{s_i}$ imposes a symmetric part between $P(s_i, s_j)$ and $P(s_j, s_i)$.

(1) is equivalent to

$$P(s_j|s_i) = \exp\Big\{\boldsymbol{v}_{s_j}^\top \boldsymbol{v}_{s_i} + a_{s_i s_j} + \log P(s_j)\Big\}, \quad (2)$$

$$\log \frac{P(s_j|s_i)}{P(s_j)} = \boldsymbol{v}_{s_j}^\top \boldsymbol{v}_{s_i} + a_{s_i s_j}. \quad (3)$$

(3) of all bigrams is represented in matrix form:

$$\boldsymbol{V}^\top \boldsymbol{V} + \boldsymbol{A} = \boldsymbol{G}, \quad (4)$$

where $\boldsymbol{G}$ is the PMI matrix.

3.1.1 Gaussian Priors on Embeddings

When (1) is employed on the regression of empirical bigram probabilities, a practical issue arises: more and more bigrams have zero frequency as the constituting words become less frequent. A zero-frequency bigram does not necessarily imply negative correlation between the two constituting words; it could simply result from missing data. But in this case, even after smoothing, (1) will force $\boldsymbol{v}_{s_j}^\top \boldsymbol{v}_{s_i} + a_{s_i s_j}$ to be a big negative number, making $\boldsymbol{v}_{s_i}$ overly long. The increased magnitude of embeddings is a sign of overfitting.

To reduce overfitting of embeddings of infrequent words, we assign a Spherical Gaussian prior $\mathcal{N}(0, \frac{1}{2\mu_i}\boldsymbol{I})$ to $\boldsymbol{v}_{s_i}$:

$$P(\boldsymbol{v}_{s_i}) \sim \exp\{-\mu_i \|\boldsymbol{v}_{s_i}\|^2\},$$

where the hyperparameter μ_i increases as the frequency of s_i decreases.

3.1.2 Gaussian Priors on Residuals

We wish $\boldsymbol{v}_{s_j}^\top \boldsymbol{v}_{s_i}$ in (1) captures as much correlations between s_i and s_j as possible. Thus the smaller $a_{s_i s_j}$ is, the better. In addition, the more frequent s_i, s_j is in the corpus, the less noise there is in their empirical distribution, and thus the residual $a_{s_i s_j}$ should be more heavily penalized.

To this end, we penalize the residual $a_{s_i s_j}$ by $f(\tilde{P}(s_i, s_j))a_{s_i s_j}^2$, where $f(\cdot)$ is a nonnegative monotonic transformation, referred to as the *weighting function*. Let h_{ij} denote $\tilde{P}(s_i, s_j)$, then the total penalty of all residuals are the square of the *weighted Frobenius norm* of $\boldsymbol{A}$:

$$\sum_{s_i, s_j \in \boldsymbol{S}} f(h_{ij}) a_{s_i s_j}^2 = \|\boldsymbol{A}\|_{f(\boldsymbol{H})}^2. \quad (5)$$

By referring to "GloVe", we use the following weighting function, and find it performs well:

$$f(h_{ij}) = \begin{cases} \frac{\sqrt{h_{ij}}}{C_{\text{cut}}} & \sqrt{h_{ij}} < C_{\text{cut}}, i \neq j \\ 1 & \sqrt{h_{ij}} \geq C_{\text{cut}}, i \neq j \\ 0 & i = j \end{cases},$$

where C_{cut} is chosen to cut the most frequent 0.02% of the bigrams off at 1. When $s_i = s_j$, two identical words usually have much smaller probability to collocate. Hence $\tilde{P}(s_i, s_i)$ does not reflect the true correlation of a word to itself, and should not put constraints to the embeddings. We eliminate their effects by setting $f(h_{ii})$ to 0.

If the domain of $\boldsymbol{A}$ is the whole space $R^{W \times W}$, then this penalty is equivalent to a Gaussian prior $\mathcal{N}\left(0, \frac{1}{2f(h_{ij})}\right)$ on each $a_{s_i s_j}$. The variances of the Gaussians are determined by the bigram empirical probability matrix $\boldsymbol{H}$.

3.1.3 Jelinek-Mercer Smoothing of Bigrams

As another measure to reduce the impact of missing data, we apply the commonly used Jelinek-Mercer Smoothing (Zhai and Lafferty, 2004) to smooth the empirical conditional probability $\tilde{P}(s_j|s_i)$ by the unigram probability $\tilde{P}(s_j)$ as:

$$\tilde{P}_{\text{smoothed}}(s_j|s_i) = (1-\kappa)\tilde{P}(s_j|s_i) + \kappa P(s_j). \quad (6)$$

Accordingly, the smoothed bigram empirical joint probability is defined as

$$\tilde{P}(s_i, s_j) = (1-\kappa)\tilde{P}(s_i, s_j) + \kappa P(s_i)P(s_j). \quad (7)$$

In practice, we find $\kappa = 0.02$ yields good results. When $\kappa \geq 0.04$, the obtained embeddings begin to degrade with κ, indicating that smoothing distorts the true bigram distributions.

3.2 Link Function of a Text Window

In the previous subsection, a regression link function of bigram probabilities is established. In this section, we adopt a first-order approximation based on Information Theory, and extend the link function to a longer sequence $w_0, \cdots, w_{c-1}, w_c$.

Decomposing a distribution conditioned on n random variables as the conditional distributions on its subsets roots deeply in Information Theory. This is an intricate problem because there could be both (pointwise) *redundant information* and (pointwise) *synergistic information* among the conditioning variables (Williams and Beer, 2010). They are both functions of the PMI. Based on an analysis of the complementing roles of these two types of pointwise information, we assume they are approximately equal and cancel each other when computing the *pointwise interaction information*. See Appendix B for a detailed discussion.

Following the above assumption, we have $\text{PMI}(w_2; w_0, w_1) \approx \text{PMI}(w_2; w_0) + \text{PMI}(w_2; w_1)$:

$$\log \frac{P(w_0, w_1|w_2)}{P(w_0, w_1)} \approx \log \frac{P(w_0|w_2)}{P(w_0)} + \log \frac{P(w_1|w_2)}{P(w_1)}.$$

Plugging (1) and (3) into the above, we obtain

$$P(w_0, w_1, w_2) \approx \exp\left\{ \sum_{\substack{i,j=0 \\ i \neq j}}^{2} (\boldsymbol{v}_{w_i}^\top \boldsymbol{v}_{w_j} + a_{w_i w_j}) + \sum_{i=0}^{2} \log P(w_i) \right\}.$$

We extend the above assumption to that the pointwise interaction information is still close to 0 within a longer text window. Accordingly the above equation extends to a context of size $c > 2$:

$$P(w_0, \cdots, w_c) \approx \exp\left\{ \sum_{\substack{i,j=0 \\ i \neq j}}^{c} (\boldsymbol{v}_{w_i}^\top \boldsymbol{v}_{w_j} + a_{w_i w_j}) + \sum_{i=0}^{c} \log P(w_i) \right\}.$$

From it derives the conditional distribution of w_c, given its context $w_0, \cdots, w_{c-1}$:

$$P(w_c \mid w_0 : w_{c-1}) = \frac{P(w_0, \cdots, w_c)}{P(w_0, \cdots, w_{c-1})} \approx P(w_c) \exp\left\{ \boldsymbol{v}_{w_c}^\top \sum_{i=0}^{c-1} \boldsymbol{v}_{w_i} + \sum_{i=0}^{c-1} a_{w_i w_c} \right\}. \quad (8)$$

4 Generative Process and Likelihood

We proceed to assume the text is generated from a *Markov chain* of order c, i.e., a word only depends on words within its context of size c. Given the hyperparameter $\boldsymbol{\mu} = (\mu_1, \cdots, \mu_W)$, the generative process of the whole corpus is:

1. For each word s_i, draw the embedding $\boldsymbol{v}_{s_i}$ from $\mathcal{N}(0, \frac{1}{2\mu_i}\boldsymbol{I})$;
2. For each bigram s_i, s_j, draw the residual $a_{s_i s_j}$ from $\mathcal{N}\left(0, \frac{1}{2f(h_{ij})}\right)$;
3. For each document d_i, for the j-th word, draw word w_{ij} from $\boldsymbol{S}$ with probability $P(w_{ij} \mid w_{i,j-c} : w_{i,j-1})$ defined by (8).

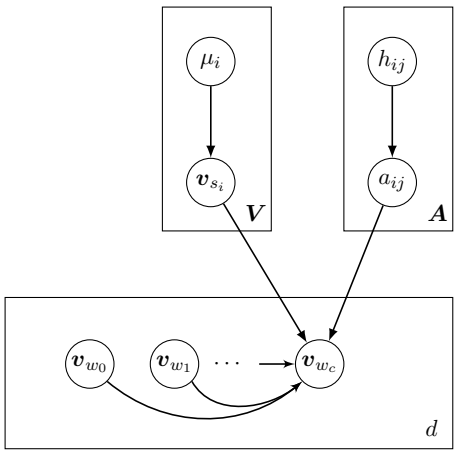


Figure 1: The Graphical Model of PSDVec

The above generative process for a document d is presented as a graphical model in Figure 1.

Based on this generative process, the probability of a document d_i can be derived as follows, given the embeddings and residuals $\boldsymbol{V}, \boldsymbol{A}$:

$$P(d_i|\boldsymbol{V},\boldsymbol{A}) = \prod_{j=1}^{L_i} P(w_{ij}) \exp\left\{ \boldsymbol{v}_{w_{ij}}^\top \sum_{k=j-c}^{j-1} \boldsymbol{v}_{w_{ik}} + \sum_{k=j-c}^{j-1} a_{w_{ik}w_{ij}} \right\}.$$

The complete-data likelihood of the corpus is:

$$\begin{aligned} & p(\boldsymbol{D},\boldsymbol{V},\boldsymbol{A}) \\ =& \prod_{i=1}^{W} \mathcal{N}(0, \frac{\boldsymbol{I}}{2\mu_i}) \prod_{i,j=1}^{W,W} \mathcal{N}\left(0, \frac{1}{2f(h_{ij})}\right) \prod_{i=1}^{M} p(d_i|\boldsymbol{V},\boldsymbol{A}) \\ =& \frac{1}{\mathcal{Z}(\boldsymbol{H},\boldsymbol{\mu})} \exp\Big\{ -\sum_{i,j=1}^{W,W} f(h_{i,j}) a_{s_i s_j}^2 - \sum_{i=1}^{W} \mu_i \|\boldsymbol{v}_{s_i}\|^2 \Big\} \\ & \cdot \prod_{i,j=1}^{M,L_i} P(w_{ij}) \exp\left\{ \boldsymbol{v}_{w_{ij}}^\top \sum_{k=j-c}^{j-1} \boldsymbol{v}_{w_{ik}} + \sum_{k=j-c}^{j-1} a_{w_{ik}w_{ij}} \right\}, \end{aligned}$$

where $\mathcal{Z}(\boldsymbol{H},\boldsymbol{\mu})$ is the normalizing constant.

Taking the logarithm of both sides of $p(\boldsymbol{D},\boldsymbol{A},\boldsymbol{V})$ yields

$$\begin{aligned} & \log p(\boldsymbol{D},\boldsymbol{V},\boldsymbol{A}) \\ =& C_0 - \log \mathcal{Z}(\boldsymbol{H},\boldsymbol{\mu}) - \|\boldsymbol{A}\|_{f(\boldsymbol{H})}^2 - \sum_{i=1}^{W} \mu_i \|\boldsymbol{v}_{s_i}\|^2 \\ & + \sum_{i,j=1}^{M,L_i} \left\{ \boldsymbol{v}_{w_{ij}}^\top \sum_{k=j-c}^{j-1} \boldsymbol{v}_{w_{ik}} + \sum_{k=j-c}^{j-1} a_{w_{ik}w_{ij}} \right\}, \quad (9) \end{aligned}$$

where $C_0 = \sum_{i,j=1}^{M,L_i} \log P(w_{ij})$ is constant.

5 Learning Algorithm

5.1 Learning Objective

The learning objective is to find the embeddings $\boldsymbol{V}$ that maximize the corpus log-likelihood (9).

Let x_{ij} denote the (smoothed) frequency of bigram s_i, s_j in the corpus. Then (9) is sorted as:

$$\begin{aligned} & \log p(\boldsymbol{D},\boldsymbol{V},\boldsymbol{A}) \\ =& C_0 - \log \mathcal{Z}(\boldsymbol{H},\boldsymbol{\mu}) - \|\boldsymbol{A}\|_{f(\boldsymbol{H})}^2 - \sum_{i=1}^{W} \mu_i \|\boldsymbol{v}_{s_i}\|^2 \\ & + \sum_{i,j=1}^{W,W} x_{ij} (\boldsymbol{v}_{s_i}^\top \boldsymbol{v}_{s_j} + a_{s_i s_j}). \quad (10) \end{aligned}$$

As the corpus size increases, $\sum_{i,j=1}^{W,W} x_{ij}(\boldsymbol{v}_{s_i}^\top \boldsymbol{v}_{s_j} + a_{s_i s_j})$ will dominate the parameter prior terms. Then we can ignore the prior terms when maximizing (10).

$$\begin{aligned} & \max \sum x_{ij} (\boldsymbol{v}_{s_i}^\top \boldsymbol{v}_{s_j} + a_{s_i s_j}) \\ =& \Big(\sum x_{ij} \Big) \cdot \max \sum \tilde{P}_{\text{smoothed}}(s_i, s_j) \log P(s_i, s_j). \end{aligned}$$

As both $\{\tilde{P}_{\text{smoothed}}(s_i, s_j)\}$ and $\{P(s_i, s_j)\}$ sum to 1, the above sum is maximized when $P(s_i, s_j) = \tilde{P}_{\text{smoothed}}(s_i, s_j)$.

The maximum likelihood estimator is then:

$$P(s_j|s_i) = \tilde{P}_{\text{smoothed}}(s_j|s_i),$$

$$\boldsymbol{v}_{s_i}^\top \boldsymbol{v}_{s_j} + a_{s_i s_j} = \log \frac{\tilde{P}_{\text{smoothed}}(s_j|s_i)}{P(s_j)}. \quad (11)$$

Writing (11) in matrix form:

$$\begin{aligned} \boldsymbol{B}^* &= \left(\tilde{P}_{\text{smoothed}}(s_j|s_i) \right)_{s_i, s_j \in \boldsymbol{S}} \\ \boldsymbol{G}^* &= \log \boldsymbol{B}^* - \log \boldsymbol{u} \otimes (1 \cdots 1), \quad (12) \end{aligned}$$

where "$\otimes$" is the outer product.

Now we fix the values of $\boldsymbol{v}_{s_i}^\top \boldsymbol{v}_{s_j} + a_{s_i s_j}$ at the above optimal. The corpus likelihood becomes

$$\begin{aligned} \log p(\boldsymbol{D},\boldsymbol{V},\boldsymbol{A}) =& C_1 - \|\boldsymbol{A}\|_{f(\boldsymbol{H})}^2 - \sum_{i=1}^{W} \mu_i \|\boldsymbol{v}_{s_i}\|^2, \\ & \text{subject to} \quad \boldsymbol{V}^\top \boldsymbol{V} + \boldsymbol{A} = \boldsymbol{G}^*, \quad (13) \end{aligned}$$

where $C_1 = C_0 + \sum x_{ij} \log \tilde{P}_{\text{smoothed}}(s_i, s_j) - \log \mathcal{Z}(\boldsymbol{H},\boldsymbol{\mu})$ is constant.

5.2 Learning V as Low Rank PSD Approximation

Once $\boldsymbol{G}^*$ has been estimated from the corpus using (12), we seek $\boldsymbol{V}$ that maximizes (13). This is to find the maximum a posteriori (MAP) estimates of $\boldsymbol{V}, \boldsymbol{A}$ that satisfy $\boldsymbol{V}^\top \boldsymbol{V} + \boldsymbol{A} = \boldsymbol{G}^*$. Applying this constraint to (13), we obtain

Algorithm 1 BCD algorithm for finding a unregularized rank-N weighted PSD approximant.

Input: matrix $\boldsymbol{G}^*$, weight matrix $\boldsymbol{W} = f(\boldsymbol{H})$, iteration number $\mathcal{T}$, rank N

Randomly initialize $\boldsymbol{X}^{(0)}$
for $t = 1, \cdots, \mathcal{T}$ **do**
  $\boldsymbol{G}_t = \boldsymbol{W} \circ \boldsymbol{G}^* + (1 - \boldsymbol{W}) \circ \boldsymbol{X}^{(t-1)}$
  $\boldsymbol{X}^{(t)} = \text{PSD\_Approximate}(\boldsymbol{G}_t, N)$
end for
$\boldsymbol{\lambda}, \boldsymbol{Q} = \text{Eigen\_Decomposition}(\boldsymbol{X}^{(\mathcal{T})})$
$\boldsymbol{V}^* = \text{diag}(\boldsymbol{\lambda}^{\frac{1}{2}}[1{:}N]) \cdot \boldsymbol{Q}^\top[1{:}N]$

Output: $\boldsymbol{V}^*$

$$\arg\max_{\boldsymbol{V}} \log p(\boldsymbol{D}, \boldsymbol{V}, \boldsymbol{A})$$
$$= \arg\min_{\boldsymbol{V}} \|\boldsymbol{G}^* - \boldsymbol{V}^\top \boldsymbol{V}\|_{f(\boldsymbol{H})} + \sum_{i=1}^{W} \mu_i \|\boldsymbol{v}_{s_i}\|^2. \quad (14)$$

Let $\boldsymbol{X} = \boldsymbol{V}^\top \boldsymbol{V}$. Then $\boldsymbol{X}$ is positive semidefinite of rank N. Finding $\boldsymbol{V}$ that minimizes (14) is equivalent to finding a rank-N *weighted positive semidefinite approximant* $\boldsymbol{X}$ of $\boldsymbol{G}^*$, subject to Tikhonov regularization. This problem does not admit an analytic solution, and can only be solved using local optimization methods.

First we consider a simpler case where all the words in the vocabulary are enough frequent, and thus Tikhonov regularization is unnecessary. In this case, we set $\forall \mu_i = 0$, and (14) becomes an unregularized optimization problem. We adopt the Block Coordinate Descent (BCD) algorithm[1] in (Srebro et al., 2003) to approach this problem. The original algorithm is to find a generic rank-N matrix for a weighted approximation problem, and we tailor it by constraining the matrix within the positive semidefinite manifold.

We summarize our learning algorithm in **Algorithm 1.** Here "$\circ$" is the entry-wise product. We suppose the eigenvalues $\boldsymbol{\lambda}$ returned by Eigen_Decomposition($\boldsymbol{X}$) are in descending order. $\boldsymbol{Q}^\top[1{:}N]$ extracts the 1 to N rows from $\boldsymbol{Q}^\top$.

One key issue is how to initialize $\boldsymbol{X}$. Srebro et al. (2003) suggest to set $\boldsymbol{X}^{(0)} = \boldsymbol{G}^*$, and point out that $\boldsymbol{X}^{(0)} = \boldsymbol{0}$ is far from a local optimum, thus requires more iterations. However we find $\boldsymbol{G}^*$ is also far from a local optimum, and this setting converges slowly too. Setting $\boldsymbol{X}^{(0)} = \boldsymbol{G}^*/2$ usually yields a satisfactory solution in a few iterations.

The subroutine PSD_Approximate() computes the *unweighted* nearest rank-N PSD approximation, measured in F-norm (Higham, 1988).

[1]It is referred to as an Expectation-Maximization algorithm by the original authors, but we think this is a misnomer.

5.3 Online Blockwise Regression of $\boldsymbol{V}$

In Algorithm 1, the essential subroutine PSD_Approximate() does eigendecomposition on $\boldsymbol{G}_t$, which is dense due to the logarithm transformation. Eigendecomposition on a $W \times W$ dense matrix requires $O(W^2)$ space and $O(W^3)$ time, difficult to scale up to a large vocabulary. In addition, the majority of words in the vocabulary are infrequent, and Tikhonov regularization is necessary for them.

It is observed that, as words become less frequent, fewer and fewer words appear around them to form bigrams. Remind that the vocabulary $\boldsymbol{S} = \{s_1, \cdots, s_W\}$ are sorted in decending order of the frequency, hence the lower-right blocks of $\boldsymbol{H}$ and $f(\boldsymbol{H})$ are very sparse, and cause these blocks in (14) to contribute much less penalty relative to other regions. Therefore these blocks could be ignored when doing regression, without sacrificing too much accuracy. This intuition leads to the following *online blockwise regression*.

The basic idea is to select a small set (e.g. 30,000) of the most frequent words as the *core words*, and partition the remaining *noncore words* into sets of moderate sizes. Bigrams consisting of two core words are referred to as *core bigrams*, which correspond to the top-left blocks of $\boldsymbol{G}$ and $f(\boldsymbol{H})$. The embeddings of core words are learned approximately using Algorithm 1, on the top-left blocks of $\boldsymbol{G}$ and $f(\boldsymbol{H})$. Then we fix the embeddings of core words, and find the embeddings of each set of noncore words in turn. After ignoring the lower-right regions of $\boldsymbol{G}$ and $f(\boldsymbol{H})$ which correspond to bigrams of two noncore words, the **quadratic terms** of noncore embeddings are ignored. Consequently, finding these embeddings becomes a *weighted ridge regression* problem, which can be solved efficiently in closed-form. Finally we combine all embeddings to get the embeddings of the whole vocabulary. The details are as follows:

1. Partition $\boldsymbol{S}$ into K consecutive groups $\boldsymbol{S}_1, \cdots, \boldsymbol{S}_k$. Take $K = 3$ as an example. The first group is core words;
2. Accordingly partition $\boldsymbol{G}$ into $K \times K$ blocks,

in this example as $\left(\begin{array}{c|cc} \boldsymbol{G}_{11} & \boldsymbol{G}_{12} & \boldsymbol{G}_{13} \\ \hline \boldsymbol{G}_{21} & \boldsymbol{G}_{22} & \boldsymbol{G}_{23} \\ \boldsymbol{G}_{31} & \boldsymbol{G}_{32} & \boldsymbol{G}_{33} \end{array}\right)$. Partition $f(\boldsymbol{H})$,$\boldsymbol{A}$ in the same way. $\boldsymbol{G}_{11}, f(\boldsymbol{H})_{11}, \boldsymbol{A}_{11}$ correspond to core bigrams. Partition $\boldsymbol{V}$ into $\underbrace{(\boldsymbol{V}_1}_{\boldsymbol{S}_1} \mid \underbrace{\boldsymbol{V}_2}_{\boldsymbol{S}_2} \; \underbrace{\boldsymbol{V}_3)}_{\boldsymbol{S}_3}$;

3. Solve $\boldsymbol{V}_1^\top \boldsymbol{V}_1 + \boldsymbol{A}_{11} = \boldsymbol{G}_{11}$ using Algorithm 1, and obtain core embeddings $\boldsymbol{V}_1^*$;
4. Set $\boldsymbol{V}_1 = \boldsymbol{V}_1^*$, and find $\boldsymbol{V}_2^*$ that minimizes the total penalty of the 12-th and 21-th blocks of residuals (the 22-th block is ignored due to its high sparsity):

$$\begin{aligned} &\arg\min_{\boldsymbol{V}_2} \|\boldsymbol{G}_{12} - \boldsymbol{V}_1^\top \boldsymbol{V}_2\|^2_{f(\boldsymbol{H})_{12}} \\ &\quad + \|\boldsymbol{G}_{21} - \boldsymbol{V}_2^\top \boldsymbol{V}_1\|^2_{f(\boldsymbol{H})_{21}} + \sum_{s_i \in \boldsymbol{S}_2} \mu_i \|\boldsymbol{v}_{s_i}\|^2 \\ =&\arg\min_{\boldsymbol{V}_2} \|\overline{\boldsymbol{G}}_{12} - \boldsymbol{V}_1^\top \boldsymbol{V}_2\|^2_{\bar{f}(\boldsymbol{H})_{12}} + \sum_{s_i \in \boldsymbol{S}_2} \mu_i \|\boldsymbol{v}_{s_i}\|^2, \end{aligned}$$

   where $\bar{f}(\boldsymbol{H})_{12} = f(\boldsymbol{H})_{12} + f(\boldsymbol{H})_{21}^\top$; $\overline{\boldsymbol{G}}_{12} = \left(\boldsymbol{G}_{12} \circ f(\boldsymbol{H})_{12} + \boldsymbol{G}_{21}^\top \circ f(\boldsymbol{H})_{21}^\top\right) / \left(f(\boldsymbol{H})_{12} + f(\boldsymbol{H})_{21}^\top\right)$ is the weighted average of $\boldsymbol{G}_{12}$ and $\boldsymbol{G}_{21}^\top$, "$\circ$" and "/" are element-wise product and division, respectively. The columns in $\boldsymbol{V}_2$ are independent, thus for each $\boldsymbol{v}_{s_i}$, it is a separate weighted ridge regression problem, whose solution is (Holland, 1973): $\boldsymbol{v}_{s_i}^* = (\boldsymbol{V}_1^\top \mathrm{diag}(\bar{\boldsymbol{f}}_i) \boldsymbol{V}_1 + \mu_i \boldsymbol{I})^{-1} \boldsymbol{V}_1^\top \mathrm{diag}(\bar{\boldsymbol{f}}_i) \bar{\boldsymbol{g}}_i$, where $\bar{\boldsymbol{f}}_i$ and $\bar{\boldsymbol{g}}_i$ are columns corresponding to s_i in $\bar{f}(\boldsymbol{H})_{12}$ and $\overline{\boldsymbol{G}}_{12}$, respectively;
5. For any other set of noncore words $\boldsymbol{S}_k$, find $\boldsymbol{V}_k^*$ that minimizes the total penalty of the $1k$-th and $k1$-th blocks, ignoring all other kj-th and jk-th blocks;
6. Combine all subsets of embeddings to form $\boldsymbol{V}^*$. Here $\boldsymbol{V}^* = (\boldsymbol{V}_1^*, \boldsymbol{V}_2^*, \boldsymbol{V}_3^*)$.

6 Experimental Results

We trained our model along with a few state-of-the-art competitors on Wikipedia, and evaluated the embeddings on 7 common benchmark sets.

6.1 Experimental Setup

Our own method is referred to as **PSD**. The competitors include:

- (Mikolov et al., 2013b): **word2vec**[2], or SGNS in some literature;
- (Levy and Goldberg, 2014b): the **PPMI** matrix without dimension reduction, and **SVD** of PPMI matrix, both yielded by hyperwords;
- (Pennington et al., 2014): **GloVe**[3];
- (Stratos et al., 2015): **Singular**[4], which does SVD-based CCA on the weighted bigram frequency matrix;
- (Faruqui et al., 2015): **Sparse**[5], which learns new sparse embeddings in a higher dimensional space from pretrained embeddings.

All models were trained on the English Wikipedia snapshot in March 2015. After removing non-textual elements and non-English words, 2.04 billion words were left. We used the default hyperparameters in Hyperwords when training PPMI and SVD. Word2vec, GloVe and Singular were trained with their own default hyperparameters.

The embedding sets PSD-Reg-180K and PSD-Unreg-180K were trained using our online block-wise regression. Both sets contain the embeddings of the most frequent 180,000 words, based on 25,000 core words. PSD-Unreg-180K was traind with all $\mu_i = 0$, i.e. disabling Tikhonov regularization. PSD-Reg-180K was trained with $\mu_i = \begin{cases} 2 & i \in [25001, 80000] \\ 4 & i \in [80001, 130000] \\ 8 & i \in [130001, 180000] \end{cases}$, i.e. increased regularization as the sparsity increases. To contrast with the batch learning performance, the performance of PSD-25K is listed, which contains the core embeddings only. PSD-25K took advantages that it contains much less false candidate words, and some test tuples (generally harder ones) were not evaluated due to missing words, thus its scores are not comparable to others.

Sparse was trained with PSD-180K-reg as the input embeddings, with default hyperparameters.

The benchmark sets are almost identical to those in (Levy et al., 2015), except that (Luong et al., 2013)'s Rare Words is not included, as many rare words are cut off at the frequency 100, making more than 1/3 of test pairs invalid.

Word Similarity There are 5 datasets: WordSim Similarity (**WS Sim**) and WordSim Relatedness (**WS Rel**) (Zesch et al., 2008; Agirre et al., 2009), partitioned from WordSim353 (Finkelstein et al., 2002); Bruni et al. (2012)'s **MEN** dataset;

[2] https://code.google.com/p/word2vec/

[3] http://nlp.stanford.edu/projects/glove/

[4] https://github.com/karlstratos/singular

[5] https://github.com/mfaruqui/sparse-coding

	Similarity Tasks					Analogy Tasks	
Method	WS Sim	WS Rel	MEN	Turk	SimLex	Google	MSR
word2vec	0.742	0.543	0.731	0.663	0.395	**0.734 / 0.742**	**0.650 / 0.674**
PPMI	0.735	0.678	0.717	0.659	0.308	0.476 / 0.524	0.183 / 0.217
SVD	0.687	0.608	0.711	0.524	0.270	0.230 / 0.240	0.123 / 0.113
GloVe	0.759	0.630	0.756	0.641	0.362	0.535 / 0.544	0.408 / 0.435
Singular	0.763	**0.684**	0.747	0.581	0.345	0.440 / 0.508	0.364 / 0.399
Sparse	0.739	0.585	0.725	0.625	0.355	0.240 / 0.282	0.253 / 0.274
PSD-Reg-180K	**0.792**	0.679	**0.764**	**0.676**	**0.398**	0.602 / 0.623	0.465 / 0.507
PSD-Unreg-180K	0.786	0.663	0.753	0.675	0.372	0.566 / 0.598	0.424 / 0.468
PSD-25K	*0.801*	*0.676*	*0.765*	*0.678*	*0.393*	*0.671 / 0.695*	*0.533 / 0.586*

Table 2: Performance of each method across different tasks.

Radinsky et al. (2011)'s Mechanical **Turk** dataset; and (Hill et al., 2014)'s **SimLex**-999 dataset. The embeddings were evaluated by the Spearman's rank correlation with the human ratings.

Word Analogy The two datasets are **MSR**'s analogy dataset (Mikolov et al., 2013c), with 8000 questions, and **Google**'s analogy dataset (Mikolov et al., 2013a), with 19544 questions. After filtering questions involving out-of-vocabulary words, i.e. words that appear less than 100 times in the corpus, 7054 instances in MSR and 19364 instances in Google were left. The analogy questions were answered using 3CosAdd as well as 3CosMul proposed by Levy and Goldberg (2014a).

6.2 Results

Table 2 shows the results on all tasks. Word2vec significantly outperformed other methods on analogy tasks. PPMI and SVD performed much worse on analogy tasks than reported in (Levy et al., 2015), probably due to sub-optimal hyperparameters. This suggests their performance is unstable. The new embeddings yielded by Sparse systematically *degraded* compared to the old embeddings, contradicting the claim in (Faruqui et al., 2015).

Our method PSD-Reg-180K performed well consistently, and is best in 4 similarity tasks. It performed worse than word2vec on analogy tasks, but still better than other MF-based methods. By comparing to PSD-Unreg-180K, we see Tikhonov regularization brings 1-4% performance boost across tasks. In addition, on similarity tasks, online blockwise regression only degrades slightly compared to batch factorization. Their performance gaps on analogy tasks were wider, but this might be explained by the fact that some hard cases were not counted in PSD-25K's evaluation, due to its limited vocabulary.

7 Conclusions and Future Work

In this paper, inspired by the link functions in previous works, with the support from Information Theory, we propose a new link function of a text window, parameterized by the embeddings of words and the residuals of bigrams. Based on the link function, we establish a generative model of documents. The learning objective is to find a set of embeddings maximizing their posterior likelihood given the corpus. This objective is reduced to weighted low-rank positive-semidefinite approximation, subject to Tikhonov regularization. Then we adopt a Block Coordinate Descent algorithm, jointly with an online blockwise regression algorithm to find an approximate solution. On seven benchmark sets, the learned embeddings show competitive and stable performance.

In the future work, we will incorporate global latent factors into this generative model, such as topics, sentiments, or writing styles, and develop more elaborate models of documents. Through learning such latent factors, important summary information of documents would be acquired, which are useful in various applications.

Acknowledgments

We thank Omer Levy, Thomas Mach, Peilin Zhao, Mingkui Tan, Zhiqiang Xu and Chunlin Wu for their helpful discussions and insights. This research is supported by the National Research Foundation, Prime Minister's Office, Singapore under its IDM Futures Funding Initiative and administered by the Interactive and Digital Media Programme Office.

Appendix A Possible Trap in SVD

Suppose $\boldsymbol{M}$ is the bigram matrix of interest. SVD embeddings are derived from the low rank approximation of $\boldsymbol{M}^\top \boldsymbol{M}$, by keeping the largest singular values/vectors. When some of these singular values correspond to negative eigenvalues, undesirable correlations might be captured. The following is an example of approximating a PMI matrix.

A vocabulary consists of 3 words s_1, s_2, s_3. Two corpora derive two PMI matrices:

$$\boldsymbol{M}^{(1)} = \begin{pmatrix} 1.4 & 0.8 & 0 \\ 0.8 & 2.6 & 0 \\ 0 & 0 & 2 \end{pmatrix}, \quad \boldsymbol{M}^{(2)} = \begin{pmatrix} 0.2 & -1.6 & 0 \\ -1.6 & -2.2 & 0 \\ 0 & 0 & 2 \end{pmatrix}.$$

They have identical left singular matrix and singular values $(3, 2, 1)$, but their eigenvalues are $(3, 2, 1)$ and $(-3, 2, 1)$, respectively.

In a rank-2 approximation, the largest two singular values/vectors are kept, and $\boldsymbol{M}^{(1)}$ and $\boldsymbol{M}^{(2)}$ yield identical SVD embeddings $\boldsymbol{V} = \begin{pmatrix} 0.45 & 0.89 & 0 \\ 0 & 0 & 1 \end{pmatrix}$ (the rows may be scaled depending on the algorithm, without affecting the validity of the following conclusion). The embeddings of s_1 and s_2 (columns 1 and 2 of $\boldsymbol{V}$) point at the same direction, suggesting they are positively correlated. However as $\boldsymbol{M}^{(2)}_{1,2} = \boldsymbol{M}^{(2)}_{2,1} = -1.6 < 0$, they are actually negatively correlated in the second corpus. This inconsistency is because the principal eigenvalue of $\boldsymbol{M}^{(2)}$ is negative, and yet the corresponding singular value/vector is kept.

When using eigendecomposition, the largest two positive eigenvalues/eigenvectors are kept. $\boldsymbol{M}^{(1)}$ yields the same embeddings $\boldsymbol{V}$. $\boldsymbol{M}^{(2)}$ yields $\boldsymbol{V}^{(2)} = \begin{pmatrix} -0.89 & 0.45 & 0 \\ 0 & 0 & 1.41 \end{pmatrix}$, which correctly preserves the negative correlation between s_1, s_2.

Appendix B Information Theory

Redundant information refers to the reduced uncertainty by knowing the value of any one of the conditioning variables (hence redundant). *Synergistic information* is the reduced uncertainty ascribed to knowing all the values of conditioning variables, that cannot be reduced by knowing the value of any variable alone (hence synergistic).

The mutual information $I(y; x_i)$ and the redundant information $\text{Rdn}(y; x_1, x_2)$ are defined as:

$$I(y; x_i) = E_{P(x_i,y)}[\log \frac{P(y|x_i)}{P(y)}]$$

$$\text{Rdn}(y; x_1, x_2) = E_{P(y)}\left[\min_{x_1,x_2} E_{P(x_i|y)}[\log \frac{P(y|x_i)}{P(y)}]\right]$$

The synergistic information $\text{Syn}(y; x_1, x_2)$ is defined as the PI-function in (Williams and Beer, 2010), skipped here.

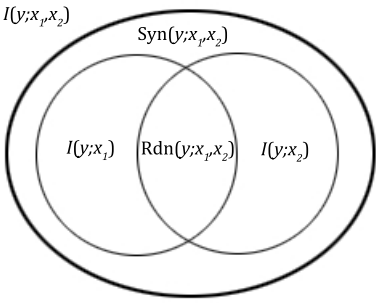


Figure 2: Different types of information among 3 random variables y, x_1, x_2. $I(y; x_1, x_2)$ is the mutual information between y and (x_1, x_2). $\text{Rdn}(y; x_1, x_2)$ and $\text{Syn}(y; x_1, x_2)$ are the redundant information and synergistic information between x_1, x_2, conditioning y, respectively.

The interaction information $\text{Int}(x_1, x_2, y)$ measures the relative strength of $\text{Rdn}(y; x_1, x_2)$ and $\text{Syn}(y; x_1, x_2)$ (Timme et al., 2014):

$$\begin{aligned}
&\text{Int}(x_1, x_2, y) \\
=&\text{Syn}(y; x_1, x_2) - \text{Rdn}(y; x_1, x_2) \\
=&I(y; x_1, x_2) - I(y; x_1) - I(y; x_2) \\
=&E_{P(x_1,x_2,y)}[\log \frac{P(x_1)P(x_2)P(y)P(x_1, x_2, y)}{P(x_1, x_2)P(x_1, y)P(x_2, y)}]
\end{aligned}$$

Figure 2 shows the relationship of different information among 3 random variables y, x_1, x_2 (based on Fig.1 in (Williams and Beer, 2010)).

PMI is the pointwise counterpart of mutual information I. Similarly, all the above concepts have their pointwise counterparts, obtained by dropping the expectation operator. Specifically, the *pointwise interaction information* is defined as $\text{PInt}(x_1, x_2, y) = \text{PMI}(y; x_1, x_2) - \text{PMI}(y; x_1) - \text{PMI}(y; x_2) = \log \frac{P(x_1)P(x_2)P(y)P(x_1,x_2,y)}{P(x_1,x_2)P(x_1,y)P(x_2,y)}$. If we know $\text{PInt}(x_1, x_2, y)$, we can recover $\text{PMI}(y; x_1, x_2)$ from the mutual information over the variable subsets, and then recover the joint distribution $P(x_1, x_2, y)$.

As the pointwise redundant information $\text{PRdn}(y; x_1, x_2)$ and the pointwise synergistic information $\text{PSyn}(y; x_1, x_2)$ are both higher-order interaction terms, their magnitudes are usually much smaller than the PMI terms. We assume they are approximately equal, and thus cancel each other when computing PInt. Given this, PInt is always 0. In the case of three words w_0, w_1, w_2, $\text{PInt}(w_0, w_1, w_2) = 0$ leads to $\text{PMI}(w_2; w_0, w_1) = \text{PMI}(w_2; w_0) + \text{PMI}(w_2; w_1)$.

References

Eneko Agirre, Enrique Alfonseca, Keith Hall, Jana Kravalova, Marius Paşca, and Aitor Soroa. 2009. A study on similarity and relatedness using distributional and wordnet-based approaches. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 19–27. Association for Computational Linguistics.

Sanjeev Arora, Yuanzhi Li, Yingyu Liang, Tengyu Ma, and Andrej Risteski. 2015. Random walks on discourse spaces: a new generative language model with applications to semantic word embeddings. *ArXiv e-prints*, arXiv:1502.03520 [cs.LG].

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, pages 1137–1155.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *The Journal of Machine Learning Research*, 3:993–1022.

Elia Bruni, Gemma Boleda, Marco Baroni, and Nam-Khanh Tran. 2012. Distributional semantics in technicolor. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 136–145. Association for Computational Linguistics.

Scott C. Deerwester, Susan T Dumais, and Richard A. Harshman. 1990. Indexing by latent semantic analysis. *J. Am. Soc. Inf. Sci.*

Paramveer Dhillon, Dean P Foster, and Lyle H Ungar. 2011. Multi-view learning of word embeddings via cca. In *Proceedings of Advances in Neural Information Processing Systems*, pages 199–207.

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

Manaal Faruqui, Yulia Tsvetkov, Dani Yogatama, Chris Dyer, and Noah A. Smith. 2015. Sparse overcomplete word vector representations. In *Proceedings of ACL 2015*.

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

Amir Globerson, Gal Chechik, Fernando Pereira, and Naftali Tishby. 2007. Euclidean embedding of co-occurrence data. *Journal of Machine Learning Research*, vol. 8 (2007):2265–2295, Oct.

Nicholas J. Higham. 1988. Computing a nearest symmetric positive semidefinite matrix. *Linear Algebra and its Applications*, 103(0):103 – 118.

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

Geoffrey Hinton. 2002. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800.

Paul W. Holland. 1973. Weighted Ridge Regression: Combining Ridge and Robust Regression Methods. NBER Working Papers 0011, National Bureau of Economic Research, Inc, September.

Daniel Hsu, Sham M Kakade, and Tong Zhang. 2012. A spectral algorithm for learning hidden markov models. *Journal of Computer and System Sciences*, 78(5):1460–1480.

Omer Levy and Yoav Goldberg. 2014a. Linguistic regularities in sparse and explicit word representations. In *Proceedings of CoNLL-2014*, page 171.

Omer Levy and Yoav Goldberg. 2014b. Neural word embeddings as implicit matrix factorization. In *Proceedings of NIPS 2014*.

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

Chenghua Lin and Yulan He. 2009. Joint sentiment/topic model for sentiment analysis. In *Proceedings of the 18th ACM conference on Information and Knowledge Management*, pages 375–384. ACM.

Minh-Thang Luong, Richard Socher, and Christopher D Manning. 2013. Better word representations with recursive neural networks for morphology. *CoNLL-2013*, 104.

Thomas Mach. 2012. *Eigenvalue Algorithms for Symmetric Hierarchical Matrices*. Dissertation, Chemnitz University of Technology.

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

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

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013c. Linguistic regularities in continuous space word representations. In *Proceedings of HLT-NAACL 2013*, pages 746–751.

Andriy Mnih and Geoffrey Hinton. 2007. Three new graphical models for statistical language modelling. In *Proceedings of the 24th International Conference on Machine learning*, pages 641–648. ACM.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. *Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014)*, 12.

Kira Radinsky, Eugene Agichtein, Evgeniy Gabrilovich, and Shaul Markovitch. 2011. A word at a time: Computing word relatedness using temporal semantic analysis. In *Proceedings of the 20th International Conference on World Wide Web*, WWW '11, pages 337–346, New York, NY, USA. ACM.

Nathan Srebro, Tommi Jaakkola, et al. 2003. Weighted low-rank approximations. In *Proceedings of ICML 2003*, volume 3, pages 720–727.

Karl Stratos, Do-kyum Kim, Michael Collins, and Daniel Hsu. 2014. A spectral algorithm for learning class-based n-gram models of natural language. In *Proceedings of the Association for Uncertainty in Artificial Intelligence*.

Karl Stratos, Michael Collins, and Daniel Hsu. 2015. Model-based word embeddings from decompositions of count matrices. In *Proceedings of ACL 2015*.

Mingkui Tan, Ivor W. Tsang, Li Wang, Bart Vandereycken, and Sinno Jialin Pan. 2014. Riemannian pursuit for big matrix recovery. In *Proceedings of ICML 2014*, pages 1539–1547.

Nicholas Timme, Wesley Alford, Benjamin Flecker, and John M Beggs. 2014. Synergy, redundancy, and multivariate information measures: an experimentalist's perspective. *Journal of Computational Neuroscience*, 36(2):119–140.

Hanna M Wallach. 2006. Topic modeling: beyond bag-of-words. In *Proceedings of the 23rd international conference on Machine learning*, pages 977–984. ACM.

Paul L Williams and Randall D Beer. 2010. Non-negative decomposition of multivariate information. *arXiv preprint arXiv:1004.2515*.

Yan Yan, Mingkui Tan, Ivor Tsang, Yi Yang, Chengqi Zhang, and Qinfeng Shi. 2015. Scalable maximum margin matrix factorization by active riemannian subspace search. In *Proceedings of IJCAI 2015*.

Dani Yogatama, Manaal Faruqui, Chris Dyer, and Noah A Smith. 2015. Learning word representations with hierarchical sparse coding. In *Proceedings of ICML 2015*.

Torsten Zesch, Christof Müller, and Iryna Gurevych. 2008. Using wiktionary for computing semantic relatedness. In *Proceedings of AAAI 2008*, volume 8, pages 861–866.

Chengxiang Zhai and John Lafferty. 2004. A study of smoothing methods for language models applied to information retrieval. *ACM Transactions on Information Systems (TOIS)*, 22(2):179–214.

Peilin Zhao, Steven CH Hoi, and Rong Jin. 2011a. Double updating online learning. *The Journal of Machine Learning Research*, 12:1587–1615.

Wayne Xin Zhao, Jing Jiang, Jianshu Weng, Jing He, Ee-Peng Lim, Hongfei Yan, and Xiaoming Li. 2011b. Comparing twitter and traditional media using topic models. In *Advances in Information Retrieval (Proceedings of the 33rd Annual European Conference on Information Retrieval Research)*, pages 338–349. Springer.

Reading Documents for Bayesian Online Change Point Detection

Taehoon Kim and **Jaesik Choi**
School of Electrical and Computer Engineering
Ulsan National Institute of Science and Technology
Ulsan, Korea
{carpedm20,jaesik}@unist.ac.kr

Abstract

Modeling non-stationary time-series data for making predictions is a challenging but important task. One of the key issues is to identify long-term changes accurately in time-varying data. *Bayesian Online Change Point Detection (BO-CPD)* algorithms efficiently detect long-term changes without assuming the Markov property which is vulnerable to local signal noise. We propose a *Document based BO-CPD (DBO-CPD)* model which automatically detects long-term temporal changes of continuous variables based on a novel dynamic Bayesian analysis which combines a non-parametric regression, the Gaussian Process (GP), with generative models of texts such as news articles and posts on social networks. Since texts often include important clues of signal changes, DBO-CPD enables the accurate prediction of long-term changes accurately. We show that our algorithm outperforms existing BO-CPDs in two real-world datasets: stock prices and movie revenues.

1 Introduction

Time series data depends on the latent dependence structure which changes over time. Thus, stationary parametric models are not appropriate to represent such dynamic non-stationary processes. Change point analysis (Smith, 1975; Stephens, 1994; Chib, 1998; Barry and Hartigan, 1993) focuses on formal frameworks to determine whether a change has taken place without assuming the Markov property which is vulnerable to local signal noise. When change points are identified, each part of the time series is approximated by specified parametric models under the stationary assumptions. Such change point detection models have successfully been applied to a variety of data, such as stock markets (Chen and Gupta, 1997; Hsu, 1977; Koop and Potter, 2007), analyzing bees' behavior (Xuan and Murphy, 2007), forecasting climates (Chu and Zhao, 2004; Zhao and Chu, 2010), and physics experiments (von Toussaint, 2011). However, offline-based change point analysis suffers from slow retrospective inference which prevents real-time analysis.

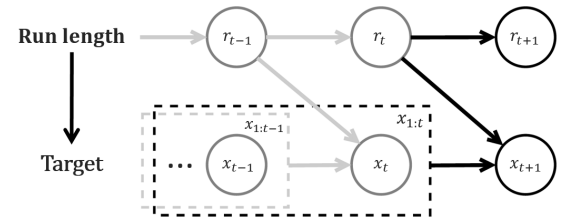
Bayesian Online Change Point Detection (BO-CPD) (Adams and MacKay, 2007; Steyvers and Brown, 2005; Osborne, 2010; Gu et al., 2013) overcomes this restriction by exploiting efficient online inference algorithms. BO-CPD algorithms efficiently detect long-term changes by analyzing continuous target values with the Gaussian Process (GP), a non-parametric regression method. The GP-based CPD model is simple and flexible. However, it is not straightforward to utilize rich external data such as texts in news articles and posts in social networks.

In this paper, we propose a novel BO-CPD model that improves the detection of change points in continuous signals by incorporating the rich external information implicitly written in texts on top of the long-term change analysis of the GP. In particular, our model finds causes of signal changes in news articles which are influential sources of markets of interests.

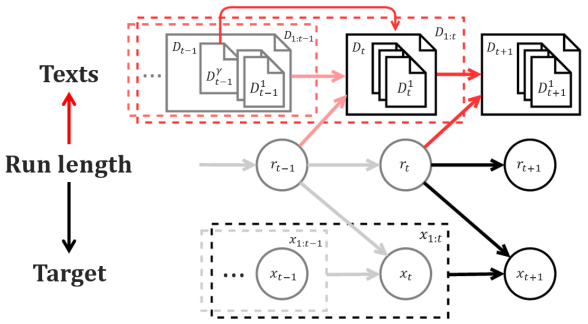
Given a set of news articles extracted from the Google News service and a sequence of target, continuous values, our new model, Document-based Bayesian Online Change Point Detection (DBO-CPD), learns a generative model which represents the probability of a news article given the run length (a length of consecutive observations without a change). By using the new prior, DBO-CPD models a dynamic hazard rate (h) which determines the rate at which change points occur.

In the literature, important information is extracted from news articles (Nothman et al., 2012;

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1610–1619,
Lisbon, Portugal, 17-21 September 2015.



(a) BO-CPD



(b) DBO-CPD (this work)

Figure 1: This figures illustrates a graphical representation of BO-CPD and our DBO-CPD model. x_t, r_t, and D_t represent a continuous variable of interest, the run length (hidden) variable, and documents, respectively. Our modeling contribution is to add texts $D_{1:t}$ for the accurate prediction of the run length r_{t+1}.

Schumaker and Chen, 2009; Gidófalvi and Elkan, 2001; Fung et al., 2003; Fung et al., 2002; Schumaker and Chen, 2006), tweets on Twitter (Si et al., 2013; Wang et al., 2012; Bollen et al., 2011; St Louis et al., 2012), online chats (Kim et al., 2010; Gruhl et al., 2005), and blog posts (Peng et al., 2015; Mishne and Glance, 2006).

In experiments, we show that DBO-CPD can effectively distinguish whether an abrupt change is a change point or not in real-world datasets (see Section 3.1). Compared to previous BO-CPD models which explain the changes by human manual mappings, our DBO-CPD automatically explains the reasons why a change point has occurred by connecting the numerical sequence of data and textual features of news articles.

2 Bayesian Online Change Point Detection

This section will review our research problem, the change point detection (CPD) (Barry and Hartigan, 1993), and the Bayesian Online Change Point Detection (BO-CPD) (Adams and MacKay, 2007) and our model, Document Based Online Change Point Detection (DBO-CPD).

Let $x_t \in \mathbb{R}$ be a data observation at time t. We assume that a sequence of data $(x_1, x_2, ..., x_t)$ is composed of several non-overlapping productive partitions (Barry and Hartigan, 1992). The boundaries that separate the partitions is called the change points. Let r be the random variable that denotes the *run length*, which is the number of time steps since the last change point was detected. r_t is the current run at time t. $x_t^{(r_t)}$ denotes the most recent data corresponding to the run r_t.

2.1 Online Recursive Detection

To make an optimal prediction of the next data x_{t+1}, one may need to consider all possible run lengths $r_t \in \mathbb{N}$ and a probability distribution over run length r_t. Given a sequence of data up to time t, $x_{1:t} = (x_1, x_2, ..., x_t)$, the run length prediction problem is formalized as computing the joint probability of random variables $P(x_{t+1}, x_{1:t})$. This distribution can be calculated in terms of the posterior distribution of run length at time t, $P(r_t|x_{1:t})$, as follows:

$$\begin{aligned} P(x_{t+1}, x_{1:t}) &= \sum_{r_t} P(x_{t+1}|r_t, x_t^{(r_t)}) P(r_t|x_{1:t}) \\ &= \sum_{r_t} P(x_{t+1}|x_t^{(r_t)}) P(r_t|x_{1:t}) \end{aligned} \quad (1)$$

The predictive distribution $P(x_{t+1}|r_t, x_t^{(r_t)})$ depends only on the most recent r_t observations $x_t^{(r_t)}$. The posterior distribution of run length $P(r_t|x_{1:t})$ can be computed recursively:

$$P(r_t|x_{1:t}) = \frac{P(r_t, x_{1:t})}{P(x_{1:t})} \quad (2)$$

where:

$$P(x_{1:t}) = \sum_{r_t} P(r_t, x_{1:t}). \quad (3)$$

The joint distribution over run length r_t and data $x_{1:t}$ can be derived by summing $P(r_t, r_{t-1}, x_{1:t})$ over r_{t-1}:

$$\begin{aligned} P(r_t, x_{1:t}) &= \sum_{r_{t-1}} P(r_t, r_{t-1}, x_{1:t}) \\ &= \sum_{r_{t-1}} P(r_t, x_t|r_{t-1}, x_{1:t-1}) P(r_{t-1}, x_{1:t-1}) \\ &= \sum_{r_{t-1}} P(r_t|r_{t-1}) P(x_t|r_{t-1}, x_t^{(r_t)}) P(r_{t-1}, x_{1:t-1}). \end{aligned}$$

This formulation updates the posterior distribution of the run length given the prior over r_t from r_{t-1} and the predictive distribution of new data.

However, the existing BO-CPD model (Adams and MacKay, 2007) specifies the conditional prior on the change point $P(r_t|r_{t-1})$ in advance. This approach may lead to model biased predictions because the update formula highly relies on the predefined, fixed hazard rate (h). Furthermore, BO-CPD is incapable of incorporating external information that implicitly influences the observation and explains the reasons for the current change of the long-term trend.

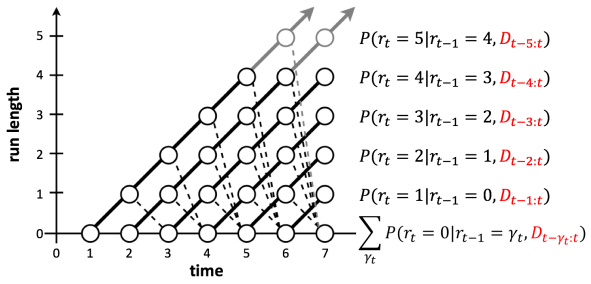


Figure 2: This figure illustrates the recursive updates of the posterior probability in the DBO-CPD model. Even the BO-CPD model only uses current and previous run length to calculate the posterior, DBO-CPD can utilize the series of text documents to compute the conditional probability accurately.

2.2 Document-based Bayesian Online Change Point Detection

This section explains our DBO-CPD model. To represent the text documents, we add a variable $\boldsymbol{D}$ which denotes a series of text documents related to the observed data as shown in Figure 1. Let D_t be a set of N_t text documents $D_t^1, D_t^2, ..., D_t^{N_t}$ that are indexed at time of publication t, where N_t is the number of documents observed at time t. Then, we can rewrite the joint probability over the run length as:

$$P(r_t, x_{1:t}) = \sum_{r_{t-1}} \sum_{D_t^{(r_{t-1})}} P\left(r_t|r_{t-1}, D_t^{(r_{t-1})}\right) \cdot P\left(x_t|r_{t-1}, x_t^{(r_{t-1})}\right) P(r_{t-1}, x_{1:t-1}) \quad (4)$$

where $D_t^{(r_t)}$ $(= D_{t-r_t+1:t})$ is the set of the r_t most recent documents. Figure 2 illustrates the recursive updates of posterior probability where solid lines indicate that the probability mass is passed upwards and dotted lines indicate the probability that the current run length r_t is set to zero.

Given documents $D_t^{(r_t)}$, the conditional probability is represented as follows:

$$\begin{aligned} & P\left(r_t = \gamma{+}1|r_{t-1} = \gamma, D_t^{(\gamma)}\right) \\ &= \frac{P\left(r_{t-1} = \gamma, D_t^{(\gamma)}|r_t = \gamma{+}1\right) P(r_t = \gamma{+}1)}{\sum_{\bar{\gamma}=1}^{\gamma+1} P\left(r_{t-1} = \gamma, D_t^{(\gamma)}|r_t = \bar{\gamma}\right) P(r_t = \bar{\gamma})} \\ &= \frac{P\left(r_{t-1} = \gamma, D_t^{(\gamma)}|r_t = \gamma + 1\right) P_{gap}(\gamma{+}1)}{\sum_{\bar{\gamma}=1}^{\gamma+1} P\left(r_{t-1} = \gamma, D_t^{(\gamma)}|r_t = \bar{\gamma}\right) P_{gap}(\bar{\gamma})} \end{aligned}$$

where P_{gap} is the distribution of intervals between consecutive change-points. As the BO-CPD model (Adams and MacKay, 2007), we assume the simplest case where the probability of a change-point at every step is constant if the length of a segment is modeled by a discrete exponential (geometric) distribution as:

$$P_{gap}(r_t|\lambda) = \lambda exp^{-\lambda r_t} \quad (5)$$

where $\lambda > 0$, a *rate parameter*, is the parameter of the distribution.

The update rule for the prior distribution on r_t makes the computation of the joint distribution tractable, $\sum_{\bar{\gamma}=1}^{\gamma+1} P(r_{t-1}{=}\gamma, D_t^{(\gamma)}|r_t{=}\bar{\gamma}) \cdot P_{gap}(\bar{\gamma})$. Because r_t can only be increased to $\gamma + 1$ or set to 0, the conditional probability is as follows:

$$\begin{aligned} & P(r_t = \gamma + 1|r_{t-1} = \gamma, D_t^{(\gamma)}) \\ &= \frac{T_D(t, \gamma|\gamma{+}1)}{T_D(t, \gamma|\gamma{+}1) + T_D(t, \gamma|0)} \end{aligned} \quad (6)$$

where the function $T_D(t, \alpha|\bar{\alpha})$ is an abbreviation of $P\left(r_{t-1}{=}\alpha, D_t^{(\alpha)}|r_t{=}\bar{\alpha}\right)$. In Equation (6), $T_D(t, \gamma|\gamma{+}1){=}P(r_{t-1}{=}\gamma, D_t^{(\gamma)}|r_t{=}\gamma{+}1)$ is the joint probability of the run length r_{t-1} and a set of documents $D_t^{(\gamma)}$ when no change has occurred at time t and the run length becomes $\gamma + 1$. Therefore, we can simplify the equation by removing $r_{t-1}{=}\gamma$ from the condition as follows:

$$T_D(t, \gamma|\gamma{+}1) = P(D_t^{(\gamma)}|r_t{=}\gamma + 1). \quad (7)$$

We represent the distribution of words by the *bag-of-words* model. Let D_t^i be the set of M words that is part of the ith document at time t, i.e. $D_t^i = \{d_t^{i,1}, d_t^{i,2}, ..., d_t^{i,M}\}$. In the model, we assume that the probability of word $d_t^{i,j}$ is independent and identically distributed (iid) given a run

length parameter r_t. In this setting, the conditional probability of the words takes the following form:

$$P\left(D_t^{(\gamma)}|r_t=\gamma{+}1\right)=\frac{1}{Z}\prod_{i,j}P\left(d_t^{i,j}|r_t=\gamma{+}1\right). \tag{8}$$

The conditional probability $P(d_t^{i,j}|r_t{=}\gamma{+}1)$ is represented by two generative models, $\phi_{\mathbf{wf}}$ and $\phi_{\mathbf{wi}}$ which illustrates *word frequency* and *word impact*, respectively. The key intuition of *word frequency* is that a word tends to close to a change point if a word has been frequently seen in articles, published when there was a rapid change. The key intuition of *word impact* is that how much does a word lose information in time which will be discussed in next section. In our paper, we use the unnormalized beta distribution of the weights of words to represent the exponential decays. The probability $P\left(D_t^{(\gamma)}|r_t{=}\gamma+1\right)$ can be represented recursively as:

$$\begin{aligned}
P\left(D_t^{(\gamma)}|r_t{=}\gamma{+}1\right) &= P\left(D_t^{(\gamma)}|\gamma{+}1\right)\\
&\propto \phi_{\mathbf{wi}}(D_t^{(\gamma)}|\gamma{+}1)\cdot\phi_{\mathbf{wf}}(D_t^{(\gamma)}|\gamma{+}1)\\
&= \phi_{\mathbf{wi}}(D_t|\gamma{+}1)\cdot\phi_{\mathbf{wf}}(D_t|\gamma{+}1)\\
&\quad\cdot\phi_{\mathbf{wi}}(D_{t-1}^{(\gamma-1)}|r_{t-1}{=}\gamma)\cdot\phi_{\mathbf{wf}}(D_{t-1}^{(\gamma-1)}|r_{t-1}{=}\gamma)\\
&= \prod_{i,j}\phi_{\mathbf{wi}}(d_t^{i,j}|\gamma{+}1)\cdot\phi_{\mathbf{wf}}(d_t^{i,j}|\gamma{+}1) \qquad (9)\\
&\quad\cdot\phi_{\mathbf{wi}}(D_{t-1}^{(\gamma-1)}|r_{t-1}{=}\gamma)\cdot\phi_{\mathbf{wf}}(D_{t-1}^{(\gamma-1)}|r_{t-1}{=}\gamma)
\end{aligned}$$

where:

$$\phi_{\mathbf{wf}}(d_t^{x,y}|\gamma)=\frac{count(d_t^{x,y},r_t=\gamma)}{\sum_{i,j}count(d_t^{i,j},r_t=\gamma)}.$$

Here, $\phi_{\mathbf{wi}}(d_t^{x,y}|\gamma)$ and $\phi_{\mathbf{wf}}(d_t^{x,y}|\gamma)$ are empirical potentials which contribute to represent $P(d_t^{i,j}|\gamma)$. $\phi_{\mathbf{wi}}(\cdot)$ is explained in Section 2.3. Here, $count(E)$ is the number of times event E appears in the dataset. In Equation (9), τ_t is the time gap (difference) between t and the time when a document is generated, and $d^{i,j}$ represents a document without considering the time domain.

$T_D(t,\gamma|0)$ is represented as follows:

$$\begin{aligned}
&P(r_{t-1}{=}\gamma, D_t^{(\gamma)}|r_t{=}0)\\
&= P(r_{t-1}{=}\gamma|r_t=0)P(D_t^{(\gamma)}|r_t{=}0)\\
&= H(\gamma{+}1)P(D_t^{(\gamma)}|r_t{=}0)
\end{aligned}$$

where $H(\tau)$ is the *hazard function* (Forbes et al., 2011),

$$H(\tau)=\frac{P_{gap}(\tau)}{\sum_{t=\tau}^{\infty}P_{gap}(t)}. \tag{10}$$

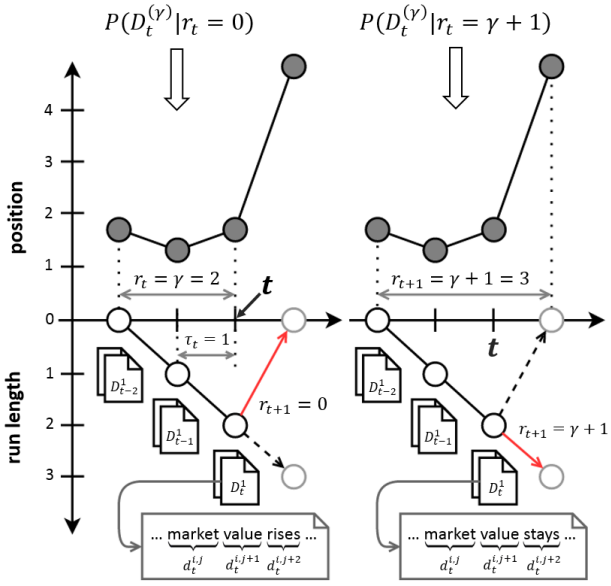


Figure 3: This figure illustrates how our Equation (9) is calculated and how it determines whether a change occurs or not. If the same data is given, BO-CPD gives us the same answer to a question whether an abrupt change at time t is a change point or not. However, DBO-CPD uses documents D_t^{γ} for its prediction to incorporate the external information which cannot be inferred only from the data.

When P_{gap} is the discrete exponential distribution, the hazard function is constant at $H(\tau)=1/\lambda$ (Adams and MacKay, 2007).

As an illustrative example, suppose that we found a rapid change in Google stock three days ago. Today at $t=3$, we want to know how the articles are written and whether it will affect the change tomorrow ($t=4$). As shown in Figure 3, we can calculate what degree a word, for example *rises* or *stays*, is likely to appear in articles published since today, which is $P(D_t^{(\gamma)}|r_t=\gamma{+}1)$, and this probability leads us to predict run lengths from the texts. Documents for each $\tau_t=0,1$ and 2 are generated from the generative models with a given predicted run length through recursive calculation of the Bayesian models which enables online prediction as shown in Equation (9). This is the main contribution of this paper that enables DBO-CPD to infer change points accurately with information included in text documents.

2.3 Generative Models Trained from Regression

Let $D \in \mathbb{R}^{T \times N \times M}$ be N documents of news articles which consist of M vocabulary over time domain T. $D_t^i \in \mathbb{R}^M$ is the ith document of a set of documents generated at time t, and define $r \in \mathbb{R}^N$ as the corresponding set of the run length, which is a time gap between the time when the document is generated and the next change point occurs. Then, given a text document D_t^i, we seek to predict the value of run length r by learning a parameterized function f:

$$\hat{r} = f(D_t^i; \mathbf{w}) \tag{11}$$

where $\mathbf{w} \in \mathbb{R}^d$ are the weights of text features for $d_t^{i,1}, d_t^{i,2}, ..., d_t^{i,M}$ which compose documents D_t^i. From a collection of N documents, we use linear regression which is trained by solving the following optimization problem:

$$\min_{\mathbf{w}, D_t^i} f(D_t^i; \mathbf{w}) \equiv \mathbf{C} \sum_{\mathbf{i=1}}^{\mathbf{N}} \xi(\mathbf{w}, \mathbf{D_t^i}, \mathbf{r_t}) + \mathbf{r}(\mathbf{w}) \tag{12}$$

where $r(\mathbf{w})$ is the regularization term and $\xi(\mathbf{w}, \mathbf{D_t^i}, \mathbf{r_t})$ is the loss function. Parameter $C > 0$ is a user-specified constant for balancing $r(\mathbf{w})$ and the sum of losses.

Let h be a function from a document into a vector-space representation $\in \mathbb{R}^d$. In linear regression, the function f takes the form:

$$f(D_t^i; \mathbf{w}) = h(D_t^i)^\top \mathbf{w} + \epsilon \tag{13}$$

where ϵ is Gaussian noise.

Figure 4 illustrates how we trained a linear regression model on a sample article. One issue is that the run length can not be trained directly. Suppose that we train $r_5 = 0$ into regression, the weight $\mathbf{w}$ of the model will become 0 even though the set of words contained in $D_5^j, \forall j \in \{1, ..., T\}$ is composed of salient words which can incur a possible future change point. To solve this interpretability problem, we trained the weight in the inverse exponential domain for the predicted variable, predicting e^{-r_t} instead of r_t. In this setting, the predicted run-length takes the form:

$$e^{-\hat{r_t}} = h(D_t)^\top \mathbf{w} + \epsilon. \tag{14}$$

By this method, the regression model can give a high weight to a word which often appears close to change points. We can interpret that highly weighted words d are more closely related to an outbreak of changes than lower weighted words.

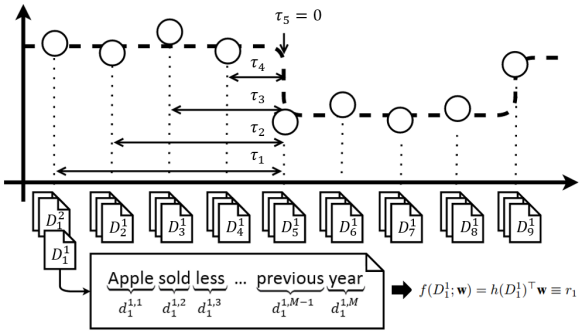


Figure 4: This figure illustrates a graphical representation of how we train a generative model from a regression problem. We use a regression model to predict time gap r_t between the release date of article and the nearest future change point. The weights of regression model are changed into the negative exponential scale to be considered as *word impact*.

With $\mathbf{w}$, we can rewrite the probability of d, τ_t given $\mathbf{w}$ as:

$$\begin{aligned} \phi_{\mathbf{wi}}(d, \tau_t) &\propto \mathbf{w}_d \cdot (\exp(-1/\mathbf{w}_d))^{\tau_t} \\ &= \mathbf{w}_d \cdot \exp(-\tau_t/\mathbf{w}_d). \end{aligned} \tag{15}$$

The potential, $\phi_{\mathbf{wi}}$, can also be represented recursively as follows:

$$\phi_{\mathbf{wi}}(d, \tau_{t+1}) = \phi_{\mathbf{wi}}(d, \tau_t) \cdot \exp(-1/\mathbf{w}_d), \tag{16}$$

since given a word d, $\tau_{t+1} = \tau_t + 1$ holds.

3 Experiments

Now we explain experiments of DBO-CPD in two real-world datasets, stock prices and movie revenues. The first case is the historical end-of-day stock prices of five information technology corporations. In the second dataset, we examine daily film revenues averaged by the number of theaters.

3.1 Datasets

In the stock price dataset, we gather data for five different companies: Apple (AAPL), Google (GOOG), IBM (IBM), Microsoft (MSFT), and Facebook (FB). These companies were selected because they were the top 5 ranked in market value in 2015.

We chose these technology companies because the announcement of new IT products and features and the interests of public media tend to be higher

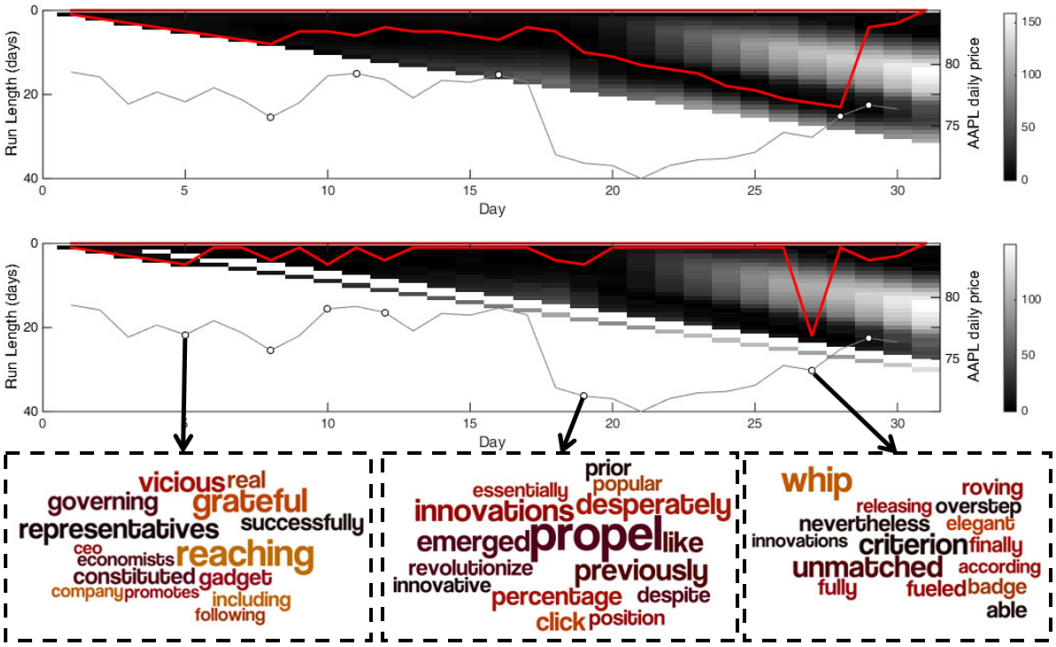


Figure 5: (a) Two plots show the results of BO-CPD (**top**) and DBO-CPD (**middle**) on Apple stock prices in January 2014. The stock price is plotted in light gray, with the predictive change points drawn as small circles. The red line represents the most likely predicted run-lengths for each day. The **bottom** figures are a set of visualizations of the top 15 strongly weighted words which are found at selected change points which BO-CPD is unable to predict. The size of each word represents the weight of its textual features learned during the training of the regression model.

and lead to many news articles. We use the historical stock price data from the Google Finance service.[1].

category	words	documents	words/doc
AAPL	15.0M	29,459	509
AAPL:N	11.0M	18,896	581
GOOG	15.0M	29,422	511
GOOG:N	8.2M	13,658	603
IBM	26.7M	45,741	583
IBM:N	3.4M	4,741	726
MSFT	20.5M	35,905	570
MSFT:N	3.5M	5,070	681
FB	18.9M	38,168	495
FB:N	4.3M	6,625	645
KNGHT	14.4M	16,874	852
INCPT	12.1M	17,155	705
AVGR	3.5M	6,476	537
FRZ	6.8M	15,021	454
INTRS	4.2M	7,846	538

Table 1: Dimensions of the datasets used in this paper, after tokenizing and filtering the news articles. '**:N**' means the articles are collected with additional 'NASDAQ:' search query.

The second dataset is a set of movie revenues averaged by the number of theaters for five months from the release date of film. We target 5 different movies: The Dark Knight (KNGHT), Inception (INCPT), The Avengers (AVGR), Frozen (FRZ) and Interstellar (INTRS), because these movies are on highest-grossing movie list and also are screened recently. The cumulative daily revenue per theater is collected from 'Box Office Mojo' (www.boxofficemojo.com).

News articles are collected from Google News and we use *Google search queries* to extract specific articles related to each dataset in a specific time period. During the online article crawling, we store not only the titles of articles, HTML documents, and publication dates, but also the number of related articles. The number of articles is used to differentiate the weight of news articles during the training of regression. In the case of stock price data, we use two different queries to decrease noise. First, we search with the company name such as 'Google'. Then, we use queries specific to stock 'NASDAQ:' to make the content of articles to be highly relevant to the stock market. In case of movie data, we search with the movie title with the additional word 'movie' to only collect articles related to the target movie.

With these collected articles, we used two ar-

[1]https://www.google.com/finance

ticle extractors, *newspaper* (Ou-Yang, 2013) and *python-goose* (Grangier, 2013), to automate the text extraction of 291,057 HTML documents. After preprocessing, we could successfully extract texts from 287,389 (98.74%) HTMLs.

3.2 Textual Feature Representation

After extracting texts from HTMLs, we tokenize the texts into words. We use three different tokenization methods which are downcasing the characters, punctuation removal, and removing English stop words. Table 1 shows the statistics on the corpora of collected news articles.

With these article corpora, we use a *bag-of-words* (BoW) representation to change each word into a vector representation where words from articles are indexed and then weighted. Using these vectors, we adopt three document representations, TF, TFIDF, and LOG1P, which extend BoW representation. TF and TFIDF (Sparck Jones, 1972) calculate the importance of a word to a set of documents based on term frequency. LOG1P (Kogan et al., 2009) calculates the logarithm of the word frequencies.

3.3 Training BO-CPD

As we noted earlier, we use BO-CPD to train the regression model to learn high weight for words which are more related to changes. When we choose the parameters for the Gaussian Process of BO-CPD, we try to find the value which makes the distance of intervals between predicted change points around 1-2 weeks. This is because we assume that the information included in the articles will have an immediate effect on the data right after it is published to the public, so the external information in texts will indicate the short-term causes for a future change.

For the reasonable comparison of BO-CPD and DBO-CPD, we use the same parameter for the Gaussian Process in both models. After several experiments we found that $a = 1$ and $b = 1$ for the Gaussian Process and $\lambda_{gap} = 250$ is appropriate to train BO-CPD in the stock and film datasets. We separate the training and testing examples for cross-validation at a ratio of $2 : 1$ for each year. Then we train each model differently by year.

3.4 Learning the strength parameter w from Regression

The weight **w** of the regression model gives us an intuition of how a word is important which affect to the length of the current run. With the predicted run length calculated in Section 3.3, we change the run length domain $r \in \mathbb{R}$ into $0 \leq r \leq 1$ by predicting e^{r_t} rather than r_t to solve the interpretability problem. Therefore, we can think of a high weight w_i as a powerful word which changes the current run length r to 0. To maintain the scalability of **w**, we normalize the weight by rescaling the range into $\mathbf{w} \in [-1, 1]$. With the word representation calculated in Section 3.2, we train the regression model by using the number of relevant articles as the importance weight of training.

	2010	2011	2012	2013	2014
AAPL BO-CPD	14.93	16.33	16.24	14.44	17.63
AAPL DBO-CPD I	**14.81**	16.22	16.20	**14.21**	17.12
AAPL DBO-CPD II	15.15	**16.20**	**16.14**	14.40	**17.11**
GOOG BO-CPD	**15.03**	15.65	15.49	19.43	19.04
GOOG DBO-CPD I	15.48	15.92	**15.21**	19.24	19.07
GOOG DBO-CPD II	15.31	**15.62**	15.36	**19.20**	**19.02**
IBM BO-CPD	17.10	17.83	17.42	16.25	16.30
IBM DBO-CPD I	17.66	**17.81**	17.40	16.20	**16.04**
IBM DBO-CPD II	**17.04**	17.82	**17.38**	**16.14**	16.39
MSFT BO-CPD	12.41	11.91	14.51	15.60	17.25
MSFT DBO-CPD I	12.33	12.60	14.48	**14.92**	**16.43**
MSFT DBO-CPD II	**12.21**	**11.79**	**14.46**	15.00	16.46
FB BO-CPD	N/A	N/A	**12.32**	13.07	16.68
FB DBO-CPD I	N/A	N/A	12.34	13.00	16.24
FB DBO-CPD II	N/A	N/A	12.43	**12.98**	**16.25**

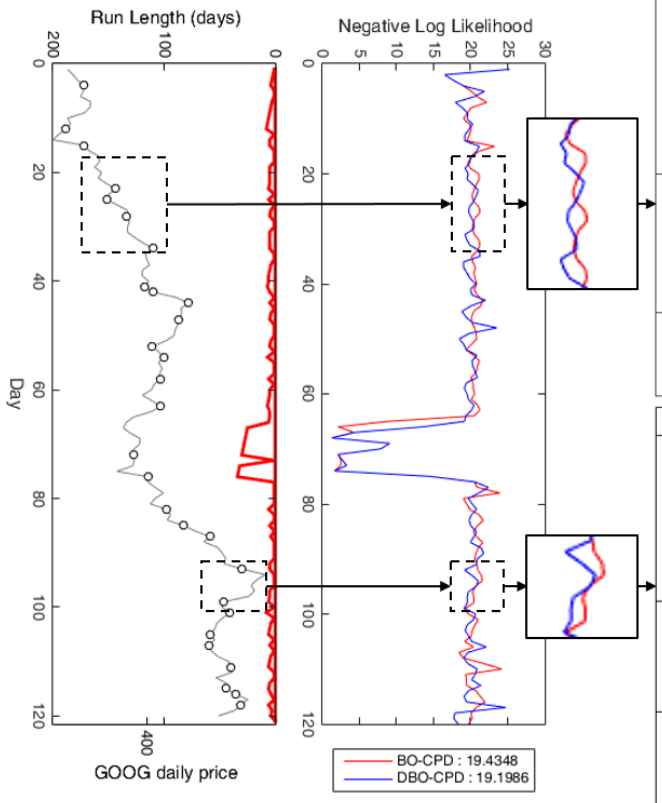
Table 2: Negative log likelihood of five stocks (Apple, Google, IBM, Microsoft, and Facebook) without and with our model per year from 2010 to 2014. DBO-CPD I represents the experiments without 'NASDAQ:' as a search query and DBO-CPD II is the result of articles searched with 'NASDAQ:'. Facebook data is not available before the year 2012.

3.5 Results

We evaluate the performance of BO-CPD and DBO-CPD by comparing the negative log likelihood (NLL) (Turner et al., 2009) of two models at time t as:

$$\log p(x_{1:T}|\mathbf{w}) = \sum_{t=1}^{T} \log p(x_t|x_{1:t-1}, \mathbf{w}).$$

We calculate the marginal NLL by year and the results are described in Table 2 and Table 3. (Facebook data is not available before the year 2012.) The difference between DBO-CPD I and DBO-CPD II is whether the search queries include 'NASDAQ'. In stock data sets of 5 years, our model outperforms BO-CPD in Apple, Google, IBM, Microsoft dataset. The improvements of



Rank	From Jan 25 to Feb 13 at 2013 from GOOG
1	The tech giant Google has been **steadily** building a portfolio of sustainability-related projects that put it squarely in the vanguard of corporate **leadership**, with a long and growing list that includes green data centers, solar power installations, open **source** alternative energy **maps** based on Google **Earth** and even amanure-to-energy project.
2	Google **announced** today that it has updated Google **Earth** for desktop with 100,000 **new tours** and a **million** photos from Panoramio, a geolocation-oriented photo share site that Google acquired in 2007.
3	Google Shopping was a **big** bet for Google because until September it was free for **retailers** to **list products** that were **searchable** in Google Shopping.

Rank	From May 14 to May 21 at 2013 from GOOG
1	Google's **director** of Google Maps, Daniel Graf, demoed the **new** version onstage and showed **new** features, **including integrated** lists from Zagat, more reviews in more places and a revamped directions and navigations experience (which now includes real-time incidence alerts and **dynamic** rerouting).
2	At Google's **developer**-focused I/O conference today, Google's Hugo Barra **announced** that Google Play has just passed 48 billion **app installs** — and had a **massive** 2.5 billion just last month.
3	Case in point: according to a **post** on the official Google Commerce **blog**, Google Wallet **support** has been baked into Gmail, so users will **soon** be able to send each other **money** by simply shooting each other emails.

Figure 6: (b) The **left** plot illustrates daily stock prices of Google in 2013 from early January to late May. The black line represents the stock price, black circles indicate the predicted change points, and the red line shows the predicted run length calculated by DBO-CPD. The **middle** plot shows the negative log likelihood (NLL) of BO-CPD and DBO-CPD on the same data. The overall marginal NLL of DBO-CPD (19.1964) is smaller than BO-CPD (19.3438). The two zoomed intervals are the two longest intervals where the negative log likelihood of DBO-CPD is smaller than BO-CPD. The **right** table shows the sentences whose run length predicted by the regression model (described in Section 2.3) are the highest at the two zoomed points, which means the sentences are likely to appear near feature change points. The boldface words are the top 5 most strongly-weighted terms in the regression model.

DBO-CPD compared to the BO-CPD is statistically significant with 90% confidence in the four stocks except for stock of Facebook. We also found that most of the DBO-CPD II shows better results than DBO-CPD I and BO-CPD in most datasets due to noise reduction of texts through the additional search query 'NASDAQ:'. Out of 23 datasets, APPL in 2010 and FB in 2012 are the only datasets where NLLs of BO-CPD is smaller (better) than NLLs of DBO-CPD.

One of the advantages of using a linear model is that we can investigate what the model discovers about different terms. As shown in Figure 5, we can find negative semantic words such as *vicious*, *whip*, and *desperately*, and words representing the status of a company like *propel*, *innovations*, and *grateful* are the most strongly-weighted terms in the regression model. We analyze and visualize some change points where NLL of DBO-CPD is lower than NLL of BO-CPD. The results are shown in Figure 6 and three sentences are the top 3 most weighted sentences in the regression model for two changes with the boldface words of top 5 strongly weighted terms like the terms *big*, *money*, and *steadily*. A particularly interesting case is the term *earth* which is found between Jan. 25 and Feb. 13 in 2013. After we investigated articles where the sentence is included, we found that Google announced a new tour guide feature in Google Earth on Jan. 31 and after this announce-

	NLL
KNGHT BO-CPD	39.76
KNGHT DBO-CPD I	**39.54**
INCPT BO-CPD	55.60
INCPT DBO-CPD I	**55.54**
AVGR BO-CPD	32.12
AVGR DBO-CPD I	**32.10**
FRZ BO-CPD	51.25
FRZ DBO-CPD I	**51.04**
INT BO-CPD	38.49
INT DBO-CPD I	**38.31**

Table 3: Negative log likelihood (NLL) of five movies (The Dark Knight, Inception, Avengers, Frozen, and Interstellar) without and with our model for 1 year from the release date of each movie.

ment the stock price increased. We can also find that the word *million* is also a positive term which can predict a new change in the near feature.

4 Conclusions

In this paper, we propose a novel generative model for online inference to find change points from non-stationary time-series data. Unlike previous approaches, our model can incorporate external information in texts which may includes the causes of signal changes. The main contribution of this paper is to combine the generative model for online change points detection and a regression model learned from the weights of words in documents. Thus, our model accurately infers the conditional prior of the change points and automatically explains the reasons of a change by connecting the numerical sequence of data and textual features of news articles.

5 Future work

Our DBO-CPD can be improved further by incorporating more external information beyond documents. In principle, our DBO-CPD can incorporate other features if they are vectorized into a matrix form. Our implementation currently only uses the simple bag of words models (TF, TFIDF and LOG1P) to improve the baseline GP-based CPD models by bringing documents into change point detection. One possible direction of future work would explore ways to fully represent the rich information in texts by extending the text features and language representations like continuous bag-of-words (CBOW) models (Mikolov et al., 2013) or Global vectors for word representation (GloVe) (Pennington et al., 2014).

Acknowledgments

This work was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) grant funded by the Ministry of Science, ICT & Future Planning (MSIP) (NRF- 2014R1A1A1002662), the NRF grant funded by the MSIP (NRF-2014M2A8A2074096).

References

Ryan Prescott Adams and David JC MacKay. 2007. Bayesian online changepoint detection. *arXiv preprint arXiv:0710.3742*.

Daniel Barry and John A Hartigan. 1992. Product partition models for change point problems. *The Annals of Statistics*, pages 260–279.

Daniel Barry and John A Hartigan. 1993. A bayesian analysis for change point problems. *Journal of the American Statistical Association*, 88(421):309–319.

Johan Bollen, Huina Mao, and Xiaojun Zeng. 2011. Twitter mood predicts the stock market. *Journal of Computational Science*, 2(1):1–8.

Jie Chen and AK Gupta. 1997. Testing and locating variance changepoints with application to stock prices. *Journal of the American Statistical Association*, 92(438):739–747.

Siddhartha Chib. 1998. Estimation and comparison of multiple change-point models. *Journal of econometrics*, 86(2):221–241.

Pao-Shin Chu and Xin Zhao. 2004. Bayesian change-point analysis of tropical cyclone activity: The central north pacific case. *Journal of Climate*, 17(24):4893–4901.

Catherine Forbes, Merran Evans, Nicholas Hastings, and Brian Peacock. 2011. *Statistical distributions*. John Wiley & Sons.

Gabriel Pui Cheong Fung, Jeffrey Xu Yu, and Wai Lam. 2002. News sensitive stock trend prediction. In *Advances in knowledge discovery and data mining*, pages 481–493. Springer.

Gabriel Pui Cheong Fung, Jeffrey Xu Yu, and Wai Lam. 2003. Stock prediction: Integrating text mining approach using real-time news. In *IEEE International Conference on Computational Intelligence for Financial Engineering*, pages 395–402.

Gyozo Gidófalvi and Charles Elkan. 2001. Using news articles to predict stock price movements. *Department of Computer Science and Engineering, University of California, San Diego*.

Xavier Grangier. 2013. Python-goose - article extractor.

Daniel Gruhl, Ramanathan Guha, Ravi Kumar, Jasmine Novak, and Andrew Tomkins. 2005. The predictive power of online chatter. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 78–87.

William Gu, Jaesik Choi, Ming Gu, Horst Simon, and Kesheng Wu. 2013. Fast change point detection for electricity market analysis. In *IEEE International Conference on Big Data*, pages 50–57.

Der-Ann Hsu. 1977. Tests for variance shift at an unknown time point. *Applied Statistics*, pages 279–284.

Su Nam Kim, Lawrence Cavedon, and Timothy Baldwin. 2010. Classifying dialogue acts in one-on-one live chats. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 862–871.

Shimon Kogan, Dimitry Levin, Bryan R Routledge, Jacob S Sagi, and Noah A Smith. 2009. Predicting risk from financial reports with regression. In *Proceedings of Human Language Technologies: Conference of the North American Chapter of the Association for Computational Linguistics*, pages 272–280.

Gary Koop and Simon M. Potter. 2007. Estimation and forecasting in models with multiple breaks. *The Review of Economic Studies*, 74(3):pp. 763–789.

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

Gilad Mishne and Natalie S Glance. 2006. Predicting movie sales from blogger sentiment. In *AAAI Spring Symposium: Computational Approaches to Analyzing Weblogs*, pages 155–158.

Joel Nothman, Matthew Honnibal, Ben Hachey, and James R Curran. 2012. Event linking: Grounding event reference in a news archive. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 228–232.

Michael Osborne. 2010. *Bayesian Gaussian processes for sequential prediction, optimisation and quadrature*. Ph.D. thesis, Oxford University New College.

Lucas Ou-Yang. 2013. newspaper - news, full-text, and article metadata extraction.

Baolin Peng, Jing Li, Junwen Chen, Xu Han, Ruifeng Xu, and Kam-Fai Wong. 2015. Trending sentiment-topic detection on twitter. In *Computational Linguistics and Intelligent Text Processing*, pages 66–77. Springer.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. *Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014)*, 12:1532–1543.

Robert Schumaker and Hsinchun Chen. 2006. Textual analysis of stock market prediction using financial news articles. *AMCIS 2006 Proceedings*, page 185.

Robert P Schumaker and Hsinchun Chen. 2009. Textual analysis of stock market prediction using breaking financial news: The azfin text system. *ACM Transactions on Information Systems (TOIS)*, 27(2):12.

Jianfeng Si, Arjun Mukherjee, Bing Liu, Qing Li, Huayi Li, and Xiaotie Deng. 2013. Exploiting topic based twitter sentiment for stock prediction.

AFM Smith. 1975. A bayesian approach to inference about a change-point in a sequence of random variables. *Biometrika*, 62(2):407–416.

Karen Sparck Jones. 1972. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 28(1):11–21.

Connie St Louis, Gozde Zorlu, et al. 2012. Can twitter predict disease outbreaks? *BMJ*, 344.

DA Stephens. 1994. Bayesian retrospective multiple-changepoint identification. *Applied Statistics*, pages 159–178.

Mark Steyvers and Scott Brown. 2005. Prediction and change detection. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1281–1288.

Ryan Turner, Yunus Saatci, and Carl Edward Rasmussen. 2009. Adaptive sequential bayesian change point detection.

Udo von Toussaint. 2011. Bayesian inference in physics. *Reviews of Modern Physics*, 83(3):943.

Hao Wang, Dogan Can, Abe Kazemzadeh, François Bar, and Shrikanth Narayanan. 2012. A system for real-time twitter sentiment analysis of 2012 us presidential election cycle. In *Proceedings of the ACL 2012 System Demonstrations*, pages 115–120. Association for Computational Linguistics.

Xiang Xuan and Kevin Murphy. 2007. Modeling changing dependency structure in multivariate time series. In *Proceedings of the 24th International Conference on Machine Learning (ICML)*, pages 1055–1062.

Xin Zhao and Pao-Shin Chu. 2010. Bayesian change-point analysis for extreme events (typhoons, heavy rainfall, and heat waves): An rjmcmc approach. *Journal of Climate*, 23(5):1034–1046.

Recognizing Textual Entailment Using Probabilistic Inference

Lei Sha, Sujian Li, Tingsong Jiang, Baobao Chang, Zhifang Sui
Key Laboratory of Computational Linguistics, Ministry of Education
School of Electronics Engineering and Computer Science, Peking University
Collaborative Innovation Center for Language Ability, Xuzhou 221009 China
{shalei, lisujian, tingsong, chbb, szf}@pku.edu.cn

Abstract

Recognizing Text Entailment (RTE) plays an important role in NLP applications including question answering, information retrieval, etc. In recent work, some research explore "deep" expressions such as discourse commitments or strict logic for representing the text. However, these expressions suffer from the limitation of inference inconvenience or translation loss. To overcome the limitations, in this paper, we propose to use the predicate-argument structures to represent the discourse commitments extracted from text. At the same time, with the help of the YAGO knowledge, we borrow the distant supervision technique to mine the implicit facts from the text. We also construct a probabilistic network for all the facts and conduct inference to judge the confidence of each fact for RTE. The experimental results show that our proposed method achieves a competitive result compared to the previous work.

1 Introduction

For the natural language, a common phenomenon is that there exist a lot of ways to express the same or similar meaning. To discover such different expressions, the Recognising Textual Entailment (RTE) task is proposed to judge whether the meaning of one text (denoted as H) can be inferred (entailed) from the other one (T)(Dagan et al., 2006). For many natural language processing applications like question answering, information retrieval which involve the diversity of natural language, recognising textual entailments is a critical step.

PASCAL Recognizing Textual Entailment (RTE) Challenges (Dagan et al., 2006) have witnessed a variety of excellent systems which intend to recognize the textual entailment instances. These systems mainly employ "shallow" techniques, including heuristics, term overlap, syntactic dependencies(Vanderwende et al., 2006; Jijkoun and de Rijke, 2005; Malakasiotis and Androutsopoulos, 2007; Haghighi et al., 2005). As Hickl (2008) stated, the shallow approaches do not work well for long sentences for the missing of underlying information which needs to be mined from the surface level expression.

Recently, some deep techniques are developed to mine the facts latent in the text. Hickl (2008) proposed the concept of discourse commitments which can be seen as the set of propositions inferred from the text, and used a series of syntax-level and semantic-level rules to extract the commitments from the T-H pairs. Then the RTE task is reduced to the identification of the commitments from T which are most likely to support the inference of the commitments from H. From the work of Hickl (2008), we can see that a deep understanding of text is critical to the RTE performance and discourse commitments can serve a good media to understanding text. However, the limitation of Hickl (2008)'s work is, the extracted discourse commitments are still from the original text and do not explore the implicit meaning latent behind the text.

Another kind of deep methods involves first transferring natural language to logic representation and then conducting strict logic inference based on the logic representations (de Salvo Braz et al., 2006; Tatu and Moldovan, 2006; Wotzlaw and Coote, 2013). Through logic inference, some implicit knowledge behind the text can be mined. However, it is not easy to translate the natural language text into formal logic expressions and the translation process inevitably suffer from great information loss.

Through analysis above, in our work, we pro-

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1620–1625,
Lisbon, Portugal, 17-21 September 2015.

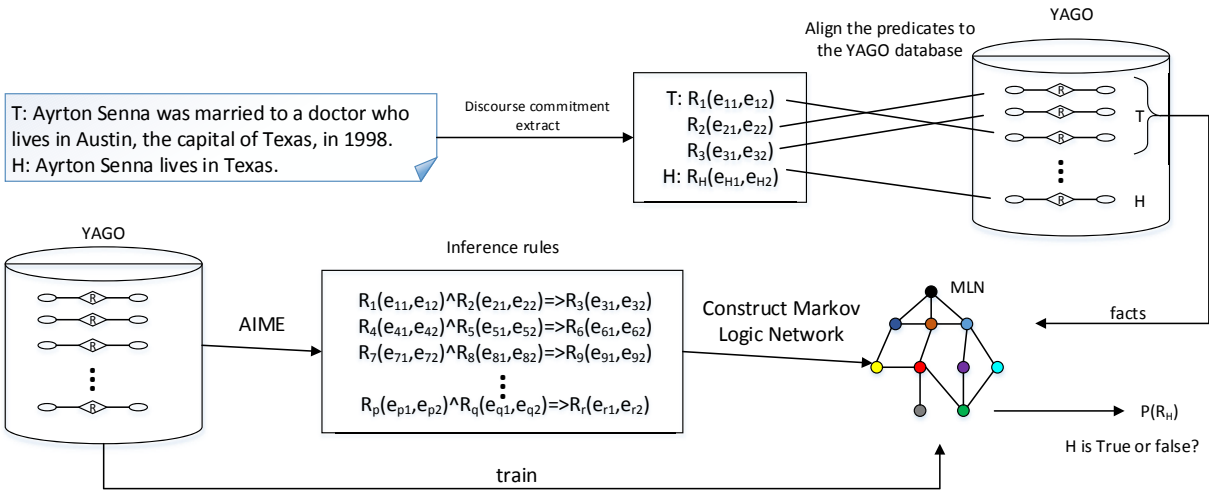


Figure 1: The Framework of our RTE system

pose to use the predicate-argument structure to represent the extracted discourse commitments. Inspired by the work of (Mintz et al., 2009), we make use of the external knowledge YAGO and borrow the distant supervision technique to mine implicit facts for the extracted predicates. For example,

Ayrton Senna was married to a doctor who lives in Austin, the capital of Texas, in 1998.

We translate this example into the predicate-argument structures such as bemarried(Senna, doctor), livein(doctor, Austin), captial(Austin, Texas). Then through distant supervision, we can get some new facts livein(Senna, Austin), livein(Senna, Texas).

To judge the confidence of the new facts, we construct a probabilistic network with all the facts and adopt the Markov Logic Network (MLN) to calculate the probability of each new fact, which can be further used to recognize text entailments.

2 Our RTE System

To make full use of the underlying information in sentences and lessen the effect brought by natural language's vagueness, we design a RTE system which is composed of three stages, as shown in Figure 1.

First, we decompose the sentences in T-H pair to a series of discourse commitments as Hickl (2008) did. Since the syntax of these commitments are very simple, we can directly transform them to predicates (or 3-arg tuples). Then we use YAGO, a large semantic database including several thousands relations, to provide distant supervision. The predicates in T are matched to YAGO facts due to some metrics. At last, we use the Markov Logic Network(MLN) (Richardson and Domingos, 2006) to infer the correctness of the predicates in H. The MLN is constructed using the inference rules ("soft" rules) generated by AMIE system (Galárraga et al., 2013) on YAGO. Each rule has a weight which should be trained by real world facts. Using this framework, we can apply the "soft" logic to the textual entailment recognition task.

2.1 Extracting Predicates from Sentences

Discourse commitment is our baseline system proposed by Hickl (2008) which can decompose one sentence to a series of shorter and simpler sentences which completely contain the origin sentence's information. One of the advantages of discourse commitments is that it can use a lot of syntax-level and semantic level rules to extract the underlying information of one sentence. For example, the following T-H pair can be decomposed as Figure 2. The discourse commitments of T contains the information: Ayrton Senna married in 1998, which is not easy to be captured by "shallow" methods.

T : Ayrton Senna was married to a doctor who lives in Austin, the capital of Texas, in 1998.

H : Ayrton Senna lives in Texas.

Since we need to infer new facts using the extracted commitments, we transfer all the commit-

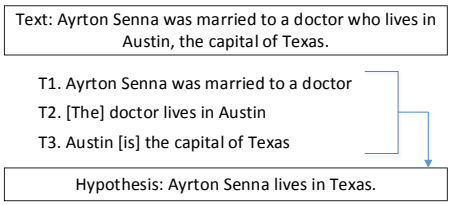


Figure 2: Text Commitments Example

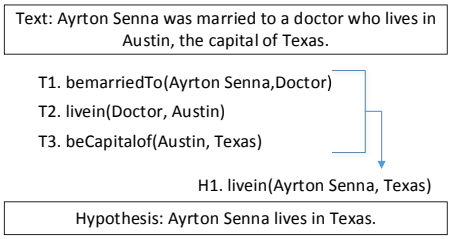


Figure 3: Text Predicates Example

ments to predicates. For example, the commitments in Figure 2 can be transformed to the predicates (or triples) shown in Figure 3. We use REVERB (Fader et al., 2011) to extract the triples (predicate + 2 Arguments). To make the inference process in the next section more convenient, we order that all of the arguments should be NPs. Therefore, we check if the arguments in the triples contain or have overlap with any of the NPs, replace it with that NP, and the predicates are successfully extracted.

2.2 Distant supervision with YAGO

The goal of distant supervision is to use the knowledge of YAGO (Mahdisoltani et al., 2014) to help the textual entailment recognition task. The facts in YAGO have various type of connections with each other. We think this connection is very useful for RTE. Therefore, the predicates in the T-H pair should be matched to the YAGO facts for making advantage of the connection information.

Since YAGO is very large, the common predicates can easily matched to YAGO facts in most cases. However, YAGO cannot contain every predicate in T. We run DIRT (Lin and Pantel, 2001) system on 1GB text random sampled from Gigaword corpus and for each predicate we choose the top-10 similar predicates as its synonymous predicate. If the origin predicate cannot be found in YAGO, we instead check for the top-10 similar predicates. If we still cannot find a match, that means this predicate has very little connection with other predicates and cannot be supervised by YAGO.

2.3 Probabilistic Inference

The goal of MLN (Richardson and Domingos, 2006) is to implement the probabilistic inference, or "soft" logic inference. MLN is constructed by an inference rule base. Each rule has a weight which needs to be well trained by real word facts. We use AMIE to mine inference rules from YAGO.

AMIE[1] (Galárraga et al., 2013) is a state-of-the-art inference rule mining system. The motivation of AMIE is that KBs themselves often already contain enough information to derive and add new facts. If, for example, a KB contains the fact that a child has a mother, then the mother's husband is most likely the father.

$$motherOf(m, c) \wedge marriedTo(m, f) \Rightarrow fatherOf(f, c)$$

AMIE can mine such inference rules from large KBs. The inference rules can be directly used for constructing Markov logic network in the next section. In addition, the process of mining inference rules is quite efficient so that it is very helpful for our RTE task.

We use AMIE only to extract the inference rules. After the inference rules are prepared, we can construct a MLN. We give the related facts in YAGO to the MLN, then the weights of each inference rule can tune to a best fit for these facts. After the weights of each inference rule are well trained, the MLN is well prepared to use. Given the predicates in T, we first select all the related rules to construct a simple MLN, and then give the MLN some facts. After that, the MLN will calculate the probabilities of the unknown new facts. The arguments of the new facts are the permutations of all the ground atoms (or entities). For example, if we give the facts "hasChild (Cliton, Chelsea)" and "IsMarriedto (Cliton, Hillary)", the MLN will output the probability of "hasChild (Cliton, Hillary)", "hasChild (Hillary, Chelsea)", "IsMarriedto (Cliton, Chelsea)", etc. Obviously, the probability of "hasChild (Hillary, Chelsea)" may be the highest, so that it is most likely to be true. The MLN constructing and inferring can be implemented by

[1] http://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga/amie/

Approach	Accuracy
Term overlap (Zanzotto and Moschitti, 2006)	67.50%
Graph Matching (MacCartney et al., 2006)	65.33%
Classification-Based (Hickl et al., 2006)	77.00%
Discourse Commitment (Hickl, 2008)	84.93%
Strict logic (Tatu and Moldovan, 2006)	71.59%
Our Framework	85.16%

Table 1: Performance of Inference based RTE

Alchemy[2], which provides a series of algorithms for statistical relational learning and probabilistic logic inference, based on the Markov logic representation.

After the new facts are inferred, we check if these facts can cover the predicates in H. If so, we decide T entails H.

3 Experiment

We evaluate the performance of our framework for RTE on the PASCAL RTE-2[3] and RTE-3[4] datasets, which has 1600 examples. We use the YAGO2 for aligning predicates and mining inference rules. YAGO2 contains more than 940K facts and about 470K entities. We run the AMIE system on YAGO2 for only one time to get all inference rules (about more than 1.8K in total). For each T-H pair, we only choose a portion of related inference rules to construct MLN. The chosen rules must contain at least one predicate which occurred in the predicates of T-H pair. We only use the MLN to infer when the discourse commitment paraphrasing cannot identify a T-H pair as "Entailment", which is a back-off method.

We compare our result with 5 baseline systems: (1) Zanzotto and Moschitti (2006)'s simple term-overlap measure, (2) MacCartney et al. (2006)'s semantic graph-mapping approach, (3) Hickl et al. (2006)'s classification-based term alignment approach. (4) Hickl (2008)'s discourse commitment based Alignment, (5) Tatu and Moldovan (2006)'s strict logic based method. The comparison of the 5 baselines and our framework is shown in Table 1. Since we only need to judge "Yes" or "No" for the 1600 examples, the precision is equal to the recall, so that we only report the precision.

According to the Table 1, the performance of our framework is higher than Hickl (2008)'s baseline, which is significant (Wilcoxon signed-rank test, $p < 0.05$). The reason is that we have added the inference portion to Hickl (2008)'s method. Therefore, some T-H pairs which had to be judged by semantic reasoning can be corrected by our framework. For instance, T is "Hughes loved his wife, Gracia, and was absolutely obsessed with his little daughter Elicia." and H is "Gracia's daughter is Elicia." It is not easy for the former baselines to recognize this entailment, but our framework can easily recognize it to be "true". In this way, our framework has achieved a higher result.

[2]http://alchemy.cs.washington.edu/

[3]http://pascallin.ecs.soton.ac.uk/Challenges/RTE2/

[4]http://pascallin.ecs.soton.ac.uk/Challenges/RTE3/

4 Related work

Textual Entailment Recognizing (RTE) task has been widely studied by many previous works. Firstly, the method based on similarity and overlap (Malakasiotis and Androutsopoulos, 2007; Jijkoun and de Rijke, 2005; Wan et al., 2006). This kind of methods can help solve the paraphrase recognition problem, which is a subset of RTE. Another important similarity-based method is tree kernel (Zanzotto and Moschitti, 2006), which rely on the cross-pair similarity between two pairs (T', H') and (T'', H'').

Secondly, some approaches extract the knowledge in T-H pair and check if the knowledge in T contains the knowledge in H. Hickl (2008) transformed the T-H pair into discourse commitments, reducing the RTE task to the identification of the commitments from a T which support the inference of the H. Other works map the text to logical meaning representations, and then strict logic entailment methods, possibly by invoking theorem provers.

Thirdly, some works make use of statistical classifiers which leverages a wide variety of features. The language expression of each T-H pair are represented by a feature vector $\langle f_1, f_2 \cdots f_m \rangle$. The feature vector contains the scores of different similarity measures applied to the pair, and possibly other features.

There are also other works based on predicate-argument representations with Markov Logic for RTE, such as Rios et al. (2014) and Beltagy et al. (2013). However, they did not use discourse commitments to extract predicate-argument triples, which may lead to severe information loss.

5 Conclusion

This paper introduced a new framework to solve the Textual Entailment Recognizing task. This

framework makes full use of Markov logic network for probabilistic inference. We hold that probabilistic inference is better than strict logic method since transforming from language form to strict logic form could lose a lot of information. Therefore it is extremely hard for the theorem provers to perform well.

In addition, we use YAGO database for distant supervision. The predicates extracted from T-H pair are first aligned with YAGO. If it succeeds, the inference procedure of MLN will become much more accurate. In addition, the inference rules for constructing MLN are also extracted from YAGO database using AIME system.

This framework can correctly recognize the entailment T-H pairs which must be judged using inference. This is our improvement over the previous work.

Acknowledgement

This research is supported by National Key Basic Research Program of China (No.2014CB340504) and National Natural Science Foundation of China (No.61375074,61273318). The contact authors of this paper are Sujian Li and Baobao Chang.

References

Islam Beltagy, Cuong Chau, Gemma Boleda, Dan Garrette, Katrin Erk, and Raymond Mooney. 2013. Deep semantics with probabilistic logical form. In *Proceedings of the Second Joint Conference on Lexical and Computational Semantics (* SEM-13)*.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The pascal recognising textual entailment challenge. In *Machine learning challenges. evaluating predictive uncertainty, visual object classification, and recognising tectual entailment*, pages 177–190. Springer.

Rodrigo de Salvo Braz, Roxana Girju, Vasin Punyakanok, Dan Roth, and Mark Sammons. 2006. An inference model for semantic entailment in natural language. In *Machine Learning Challenges. Evaluating Predictive Uncertainty, Visual Object Classification, and Recognising Tectual Entailment*, pages 261–286. Springer.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1535–1545. Association for Computational Linguistics.

Luis Antonio Galárraga, Christina Teflioudi, Katja Hose, and Fabian Suchanek. 2013. Amie: association rule mining under incomplete evidence in ontological knowledge bases. In *Proceedings of the 22nd international conference on World Wide Web*, pages 413–422. International World Wide Web Conferences Steering Committee.

Aria D Haghighi, Andrew Y Ng, and Christopher D Manning. 2005. Robust textual inference via graph matching. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 387–394. Association for Computational Linguistics.

Andrew Hickl, John Williams, Jeremy Bensley, Kirk Roberts, Bryan Rink, and Ying Shi. 2006. Recognizing textual entailment with lccs groundhog system. In *Proceedings of the Second PASCAL Challenges Workshop*.

Andrew Hickl. 2008. Using discourse commitments to recognize textual entailment. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, pages 337–344. Association for Computational Linguistics.

Valentin Jijkoun and Maarten de Rijke. 2005. Recognizing textual entailment using lexical similarity. In *Proceedings of the PASCAL Challenges Workshop on Recognising Textual Entailment*, pages 73–76.

Dekang Lin and Patrick Pantel. 2001. Dirt@ sbt@ discovery of inference rules from text. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 323–328. ACM.

Bill MacCartney, Trond Grenager, Marie-Catherine de Marneffe, Daniel Cer, and Christopher D Manning. 2006. Learning to recognize features of valid textual entailments. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 41–48. Association for Computational Linguistics.

Farzaneh Mahdisoltani, Joanna Biega, and Fabian Suchanek. 2014. Yago3: A knowledge base from multilingual wikipedias. In *7th Biennial Conference on Innovative Data Systems Research*. CIDR 2015.

Prodromos Malakasiotis and Ion Androutsopoulos. 2007. Learning textual entailment using svms and string similarity measures. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pages 42–47. Association for Computational Linguistics.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on*

Natural Language Processing of the AFNLP: Volume 2-Volume 2, pages 1003–1011. Association for Computational Linguistics.

Matthew Richardson and Pedro Domingos. 2006. Markov logic networks. *Machine learning*, 62(1-2):107–136.

Miguel Rios, Lucia Specia, Alexander Gelbukh, and Ruslan Mitkov. 2014. Statistical relational learning to recognise textual entailment. In *Computational Linguistics and Intelligent Text Processing*, pages 330–339. Springer.

Marta Tatu and Dan Moldovan. 2006. A logic-based semantic approach to recognizing textual entailment. In *Proceedings of the COLING/ACL on Main conference poster sessions*, pages 819–826. Association for Computational Linguistics.

Lucy Vanderwende, Arul Menezes, and Rion Snow. 2006. Microsoft research at rte-2: Syntactic contributions in the entailment task: an implementation. In *Proceedings of the Second PASCAL Challenges Workshop*.

Stephen Wan, Mark Dras, Robert Dale, and Cécile Paris. 2006. Using dependency-based features to take the para-farce out of paraphrase. In *Proceedings of the Australasian Language Technology Workshop*, volume 2006.

Andreas Wotzlaw and Ravi Coote. 2013. A logic-based approach for recognizing textual entailment supported by ontological background knowledge. *arXiv preprint arXiv:1310.4938*.

Fabio Massimo Zanzotto and Alessandro Moschitti. 2006. Automatic learning of textual entailments with cross-pair similarities. In *International conference on computational linguistics and the 44th Annual meeting of the Association for computational linguistics*, volume 1, pages 401–408. Association for Computational Linguistics.

Chinese Semantic Role Labeling with Bidirectional Recurrent Neural Networks

Zhen Wang, Tingsong Jiang, Baobao Chang, Zhifang Sui
Key Laboratory of Computational Linguistics, Ministry of Education
School of Electronics Engineering and Computer Science, Peking University
Collaborative Innovation Center for Language Ability, Xuzhou 221009 China
wzpkuer@gmail.com, {tingsong, chbb, szf}@pku.edu.cn

Abstract

Traditional approaches to Chinese Semantic Role Labeling (SRL) almost heavily rely on feature engineering. Even worse, the long-range dependencies in a sentence can hardly be modeled by these methods. In this paper, we introduce bidirectional recurrent neural network (RNN) with long-short-term memory (LSTM) to capture bidirectional and long-range dependencies in a sentence with minimal feature engineering. Experimental results on Chinese Proposition Bank (CPB) show a significant improvement over the state-of-the-art methods. Moreover, our model makes it convenient to introduce heterogeneous resource, which makes a further improvement on our experimental performance.

1 Introduction

Semantic Role Labeling (SRL) is defined as the task to recognize arguments for a given predicate and assign semantic role labels to them. Because of its ability to encode semantic information, there has been an increasing interest in SRL on many languages (Gildea and Jurafsky, 2002; Sun and Jurafsky, 2004). Figure 1 shows an example in Chinese Proposition Bank (CPB) (Xue and Palmer, 2003), which is a Chinese corpus annotated with semantic role labels.

Traditional approaches to Chinese SRL often extract a large number of handcrafted features from the sentence, even its parse tree, and feed these features to statistical classifiers such as CRF, MaxEnt and SVM (Sun and Jurafsky, 2004; Xue, 2008; Ding and Chang, 2008; Ding and Chang, 2009; Sun, 2010). However, these methods suffer from three major problems. Firstly, their performances are heavily dependent on feature engineering, which needs domain knowledge and laborious work of feature extraction and selection. Secondly, although sophisticated features are designed, the long-range dependencies in a sentence can hardly be modeled. Thirdly, a specific annotated dataset is often limited in its scalability, but the existence of heterogenous resource, which has very different semantic role labels and annotation schema but related latent semantic meaning, can alleviate this problem. However, traditional methods cannot relate distinct annotation schemas and introduce heterogeneous resource with ease.

WORD:	警察	正在	调查	事故	原因
	Police	now	investigate	accident	cause
ROLE:	[A0]	[AM-TMP]	REL	[A1]	
IOBES:	S-A0	S-AM-TMP	REL	B-A1	E-A1

Figure 1: A sentence with semantic roles labeled from CPB.

Concerning these problems, in this paper, we propose bidirectional recurrent neural network (RNN) with long-short-term memory (LSTM) to solve the problem of Chinese SRL. Our approach makes the following contributions:

- We formulate Chinese SRL with bidirectional LSTM RNN model. With bidirectional RNN, the dependencies in a sentence from both directions can be captured, and with LSTM architecture, long-range dependencies can be well modeled. The test results on the bechmark dataset CPB show a significant improvement over the state-of-the-art methods.
- Compared with previous work that relied on a huge number of handcrafted features, our model can achieve much better performance only with minimal feature engineering.
- The framework of our model makes the introduction of heterogeneous resource efficient

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1626–1631,
Lisbon, Portugal, 17-21 September 2015.

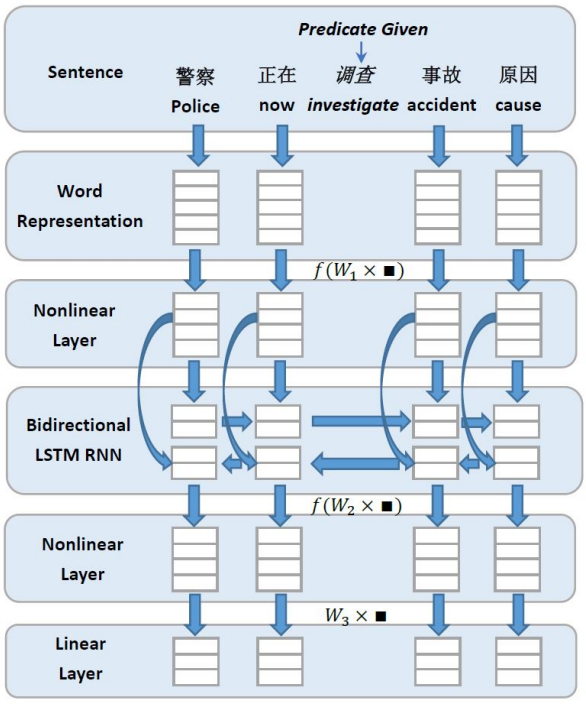


Figure 2: The model architecture.

and convenient, and this can further improve our experimental performance.

2 Chinese SRL with RNN

Following previous work, we regard Chinese SRL as a task of sequence labeling, which assigns a label for each word in the sequence. To identify the boundary information of semantic roles, we adopt the IOBES tagging schema for the labels as shown in Figure 1. For sequence labeling, it is important to capture dependencies in the sequence, especially for the problem of SRL, where the semantic role label for a word not only relies on its local information, but also is determined by long-range dependencies from other words. The advantage of RNN is the ability to better capture the contextual information, which is beneficial to capture dependencies in SRL. Moreover, we enrich the basic RNN model with bidirectional LSTM RNN, which can model bidirectional and long-range dependencies simultaneously.

2.1 Model Architecture

The architecture of our approach is illustrated in Figure 2. Given a sentence, we first get representation for each word to be labeled. Then after a nonlinear transformation, bidirectional LSTM RNN layer is designed to combine the local information of a word and its contextual information from both directions. With a nonlinear layer to form more complex features, a linear output layer follows. For each word to be labeled, there is an output vector, whose each dimension is a score corresponding to a kind of semantic role label.

2.2 Word Representation

Word representation captures the features locally embedded around the word. The features used in our work are: the current word, the current POS tag, the predicate, left and right words, left and right POS tags, distance to the predicate. Note that these features are all basic information about the word, hence we alleviate the heavy job of feature engineering. All these features are introduced by embeddings. After concatenation, we get the word representation feature vector.

To get more complex features, we adopt a nonlinear transformation:

$$z_t = f(W_1 x_t) \quad 1 \leq t \leq N$$

where x_t is the word representation of the t-th word, $W_1 \in \mathbb{R}^{n_1 \times n_0}$, n_0 is the length of word representation, f is an activation function and we use *tanh* in our experiments, N is the number of words to be labeled in the sequence.

2.3 Bidirectional LSTM RNN

Representation z_t only captures the local information. Here we adopt RNN to capture contextual information. Traditional RNN has the problem of vanishing or exploding gradients, which means the long-term dependencies can hardly be modeled. LSTM is designed to mitigate this problem.

At each word position t, the LSTM RNN computes six internal vectors: $\widetilde{C}$, g_i, g_f, g_o, C_t and h_t for the memory cell, which is a structure used in LSTM to store information. $\widetilde{C}$ computes the candidate value for the state of the memory cell:

$$\widetilde{C} = f(W_c z_t + U_c h_{t-1} + b_c)$$

The activations of the memory cell's input gate, forget gate and output gate are defined as:

$$g_j = \sigma(W_j z_i + U_j h_{t-1} + b_j)$$

where j stands for i, f or o. σ is taken $sigmoid$ in experiments. Then we can compute C_t, the memory cell's new state at position t:

$$C_t = g_i \odot \widetilde{C} + g_f \odot C_{t-1}$$

where $\odot$ indicates elementwise vector multiplication. With the new state of the memory cell, we can compute the value of output state h_t:

$$h_t = g_o \odot f(C_t)$$

h_t contains the information not only from local representation z_t, but also from previous output state h_{t-1}, hence can capture dependencies in a sentence. Because the dependencies forward and backward are both important to label semantic roles, we extend LSTM with bidirectional approach, resulting in:

$$a_t = [\overrightarrow{h_t}^T, \overleftarrow{h_t}^T]^T \quad 1 \leq t \leq N$$

Further, a nonlinear transformation follows:

$$v_t = f(W_2 a_t) \quad 1 \leq t \leq N$$

where $W_2 \in \mathbb{R}^{n_3 \times n_2}$, n_2 is the dimension of a_t.

2.4 Output Representation

For each word to be labeled, we adopt linear transformation to get the output vector o_t:

$$o_t = W_3 v_t \quad 1 \leq t \leq N$$

$W_3 \in \mathbb{R}^{n_4 \times n_3}$, where n_4 is the number of semantic role labels in IOBES tagging schema. Therefore, the resulting vector o_t for the t-th word is of length n_4, and each dimension corresponds to the score of a certain semantic role label.

2.5 Training Criteria

Because there are dependencies among word labels in a sentence, isolated training approach which independently considers each word will be inappropriate. Therefore, we adopt sentence tag approach, in which we encourage the correct path of tags, while discouraging all other valid paths.

Given all our training examples:

$$T = (x^{(i)}, y^{(i)})$$

where $x^{(i)}$ denotes the i-th training sentence, $y^{(i)}$ is the corresponding N_i (the number of words to be labeled) dimension vector, which indicates the correct path of tags, and $y_t^{(i)} = k$ means the t-th word has the k-th semantic role label. The score of $x^{(i)}$ along the path $y^{(i)}$ is defined as follows:

$$s(x^{(i)}, y^{(i)}, \theta) = \sum_{t=1}^{N_i} o_{t y_t^{(i)}}$$

where θ is an ensemble of all the parameters in the network.

The log likelihood with a single sample is then:

$$\begin{aligned} logp(y^{(i)}|x^{(i)}, \theta) &= log \frac{\exp\left(s(x^{(i)}, y^{(i)}, \theta)\right)}{\sum_{y\prime} \exp\left(s(x^{(i)}, y\prime, \theta)\right)} \\ &= s(x^{(i)}, y^{(i)}, \theta) - log \sum_{y\prime} \exp\Big(s(x^{(i)}, y\prime, \theta)\Big) \end{aligned}$$

where $y\prime$ ranges from all the valid paths of tags.

The full log likelihood of the whole training corpus is as follows:

$$J(\theta) = \sum_{i=1}^{T} logp(y^{(i)}|x^{(i)}, \theta)$$

To compute the network parameter θ, we maximize the log likelihood $J(\theta)$ using stochastic gradient ascent in the experiments.

2.6 Introducing Heterogeneous Resource

A single annotated corpus with semantic role labels is often limited in its scalability. Heterogeneous resource in our work is defined as another dataset annotated with semantic roles, which also provides predicate-argument structure annotation, but uses very different semantic role labels and annotation schema. However, in spite of these differences, the latent semantic meaning may be highly correlated. Therefore, the introduction of heterogeneous data can alleviate the problem of scalability with a single annotated corpus.

Traditional approaches hardly concern the existence of heterogeneous resource and are difficult to relate different annotation schemas, but in the framework of our model, heterogeneous data can be introduced in a relatively convenient way. Specifically, we learn a bidirectional LSTM RNN model based on heterogeneous data, then with the fine-tuned word embeddings we initialize the model on our experimental dataset. The principle behind is that the words almost convey the same semantic meaning albeit in distinct annotation schemas. The introduction of heterogenous resource in this way is efficient and can lead to performance improvement on our experiment.

3 Experiments

We conduct experiments to compare our model with previous landmark methods on the benchmark dataset CPB for Chinese SRL. The result

Remark	Choice
Word embedding dimension	$n_{word} = 50$
POS tag dimension	$n_{pos} = 20$
Distance dimension	$n_{dis} = 20$
Nonlinear layer	$n_1 = 200$
RNN layer	$n_h = 100$
Nonlinear layer	$n_3 = 100$
Learning rate	$\alpha = 10^{-3}$

Table 1: Hyper parameters of our model.

Method	F1(%)
Xue (2008)	71.90
Collobert and Weston (2008)	74.05
Sun et al. (2009)	74.12
Yang and Zong (2014)	75.31
Ours (Random Initialization)	**77.09**
+ Standard Pre-training	77.21
+ Heterogenous Resource	**77.59**

Table 2: Results comparison on CPB dataset.

reveals that even with our basic model, which does not resort to other resources, our approach can significantly outperform all of the competitors. Moreover, we enrich our work with introducing heterogenous resource to make a further improvement on performance. And the result also shows the influence of heterogeneous resource is more evident than the standard method of pre-training for word embeddings.

3.1 Experimental Setting

To facilitate comparison with previous work, we conduct experiments on the standard benchmark dataset CPB 1.0.[1] We follow the same data setting as previous work (Xue, 2008; Sun et al., 2009), which divided the dataset into three parts: 648 files (from chtb_081.fid to chtb_899.fid) are used as the training set. The development set includes 40 files, from chtb_041.fid to chtb_080.fid. The test set includes 72 files, which are chtb_001.fid to chtb_040.fid, and chtb_900.fid to chtb_931.fid. We use another annotated corpus[2] with distinct semantic role labels and annotation schema, which is designed by ourselves for other projects, as heterogeneous resource. This labeled dataset has 17,308 annotated sentences, and the semantic roles concerned are like "agent" and "patient", resulting in 21 kinds of types, which are all distinct from the semantic roles defined in CPB. We use the development set of CPB for model selection, and the hyper parameter setting of our model is reported in Table 1.

3.2 Chinese SRL Performance

Table 2 summarizes our SRL performance compared to previous landmark results. The work of Collobert and Weston (2008) was conducted on English SRL, we implement their approach on CPB for comparison. As indicated by this table, our approach significantly outperforms previous state-of-the-art methods even with all parameters randomly initialized, that is without introducing other resources. This result can prove the ability of our model to capture useful dependencies for Chinese SRL with minimal feature engineering.

Further, we conduct experiments with the introduction of heterogenous resource. Previous work found that the performance can be improved by pre-training the word embeddings on large unlabeled data and using the obtained embeddings to make initialization. With the result in Table 2, it is true that these pre-trained word embeddings have a good effect on our performance (we use *word2vec*[3] on Chinese Gigaword Corpus for word pre-training). However, as shown in Table 2, compared to standard pre-training, the influence of heterogenous data is more evident. We can explain this difference via the distinction between these two kinds of methods for performance improvement. The information provided by standard pre-training with unlabeled data is more general, while that of heterogenous resource is more relevant to our task, hence is more informative and evident.

4 Related Work

Semantic Role Labeling (SRL) was first defined by Gildea and Jurafsky (2002), who presented a a system based on statistical classifiers trained on hand-annotated corpus FrameNet. Sun and Jurafsky (2004) did the preliminary work on Chinese SRL without any large semantically annotated corpus and produced promising results. After CPB (Xue and Palmer, 2003) was built, Xue and Palmer (2005) and Xue (2008) produced more complete and systematic research on Chinese SRL. Ding and Chang (2009) established a

[1] https://catalog.ldc.upenn.edu/LDC2005T23

[2] This Chinese dataset is available on request.

[3] https://code.google.com/p/word2vec/

word based Chinese SRL system, which is quite different from the previous parsing based ones. Sun et al. (2009) extended the work of Chen et al. (2006), performed Chinese SRL with shallow parsing, which took partial parses as inputs. Yang and Zong (2014) proposed multi-predicate SRL, which showed improvements both on English and Chinese Proposition Bank. Different from most work relying on a large number of handcrafted features, Collobert and Weston (2008) proposed a convolutional neural network for SRL. Their approach achieved competitive performance on English SRL without requiring task specific feature engineering. However, by max-pooling operation, the convolution approach only preserved the most evident features in a sentence, thus can only weakly model the dependencies. With our bidirectional LSTM RNN model, this problem can be well alleviated.

Our model is based on recurrent neural network (RNN), which uses iterative function loops to store contextual information. To remedy the problem of vanishing and exploding gradients when training the standard RNN, Hochreiter and Schmidhuber (1997) proposed long-short-term memory (LSTM), which has been shown capable of storing and accessing information over very long time spans. Bidirectional RNN (Schuster and Paliwal, 1997) and bidirectional LSTM RNN (Graves et al., 2005) are the extensions of RNN and LSTM RNN with the capability of capturing contextual information from both directions in the sequence. In recent years, RNN has shown the state-of-the-art results in many NLP problems such as language modeling (Mikolov et al., 2010) and machine translation (Sutskever et al., 2014; Bahdanau et al., 2014). Sundermeyer et al. (2014) also used bidirectional LSTM RNN model to improve strong baselines when modeling translation. More recently, Zhou and Xu (2015) proposed LSTM RNN approach for English Semantic Role Labeling, which shared similar idea with our model. However, the features used and the network architecture were different from ours. Moreover, it is delightful that our work can achieve a rather good result with a relatively simpler model architecture.

5 Conclusion

In this paper, we formulate Chinese SRL problem with the framework of bidirectional LSTM RNN model. In our approach, the bidirectional and long-range dependencies in a sentence, which are important for Chinese SRL, can be well modeled. And with the framework of deep neural network, the heavy job of feature engineering is much alleviated. Moreover, our model makes the introduction of heterogenous data, which can alleviate the problem of scalability with a single annotated corpus, more convenient. Experiments show that our approach achieves much better results than previous work, and the introduction of heterogenous resource can make further improvement on performance.

Acknowledgments

This research is supported by National Key Basic Research Program of China (No.2014CB340504) and National Natural Science Foundation of China (No.61375074,61273318). The contact authors of this paper are Baobao Chang and Zhifang Sui.

References

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

Wenliang Chen, Yujie Zhang, and Hitoshi Isahara. 2006. An empirical study of chinese chunking. In *Proceedings of the COLING/ACL on Main conference poster sessions*, pages 97–104. Association for Computational Linguistics.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM.

Weiwei Ding and Baobao Chang. 2008. Improving chinese semantic role classification with hierarchical feature selection strategy. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 324–333. Association for Computational Linguistics.

Weiwei Ding and Baobao Chang. 2009. Word based chinese semantic role labeling with semantic chunking. *International Journal of Computer Processing Of Languages*, 22(02n03):133–154.

Daniel Gildea and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. *Computational linguistics*, 28(3):245–288.

Alex Graves, Santiago Fernández, and Jürgen Schmidhuber. 2005. Bidirectional lstm networks for improved phoneme classification and recognition. In *Artificial Neural Networks: Formal Models and*

Their Applications–ICANN 2005, pages 799–804. Springer.

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

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010*, pages 1045–1048.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *Signal Processing, IEEE Transactions on*, 45(11):2673–2681.

Honglin Sun and Daniel Jurafsky. 2004. Shallow semantic parsing of chinese. In *Proceedings of NAACL 2004*, pages 249–256.

Weiwei Sun, Zhifang Sui, Meng Wang, and Xin Wang. 2009. Chinese semantic role labeling with shallow parsing. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 3-Volume 3*, pages 1475–1483. Association for Computational Linguistics.

Weiwei Sun. 2010. Improving chinese semantic role labeling with rich syntactic features. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 168–172. Association for Computational Linguistics.

Martin Sundermeyer, Tamer Alkhouli, Joern Wuebker, and Hermann Ney. 2014. Translation modeling with bidirectional recurrent neural networks. In *Proceedings of the Conference on Empirical Methods on Natural Language Processing, October*.

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

Nianwen Xue and Martha Palmer. 2003. Annotating the propositions in the penn chinese treebank. In *Proceedings of the second SIGHAN workshop on Chinese language processing-Volume 17*, pages 47–54. Association for Computational Linguistics.

Nianwen Xue and Martha Palmer. 2005. Automatic semantic role labeling for chinese verbs. In *IJCAI*, volume 5, pages 1160–1165. Citeseer.

Nianwen Xue. 2008. Labeling chinese predicates with semantic roles. *Computational linguistics*, 34(2):225–255.

Haitong Yang and Chengqing Zong. 2014. Multi-predicate semantic role labeling. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 363–373.

Jie Zhou and Wei Xu. 2015. End-to-end learning of semantic role labeling using recurrent neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1127–1137, Beijing, China, July. Association for Computational Linguistics.

Unsupervised Negation Focus Identification with Word-Topic Graph Model

Bowei Zou **Qiaoming Zhu** **Guodong Zhou***

Natural Language Processing Lab, School of Computer Science and Technology
Soochow University, Suzhou, 215006, China
zoubowei@gmail.com, {qmzhu, gdzhou}@suda.edu.cn

Abstract

Due to the commonality in natural language, negation focus plays a critical role in deep understanding of context. However, existing studies for negation focus identification major on supervised learning which is time-consuming and expensive due to manual preparation of annotated corpus. To address this problem, we propose an unsupervised word-topic graph model to represent and measure the focus candidates from both lexical and topic perspectives. Moreover, we propose a document-sensitive biased PageRank algorithm to optimize the ranking scores of focus candidates. Evaluation on the *SEM 2012 shared task corpus shows that our proposed method outperforms the state of the art on negation focus identification.

1 Introduction

Negation is used to reverse the polarity of part of statements that are otherwise affirmative by default (Blanco and Moldovan, 2011), which is common in natural language. *Negation focus* is defined as the special part in sentence, which is most prominently or explicitly negated by a negative expression. For example, sentence (1) could be interpreted as *He stopped, but not until he got to Jackson Hole* with a positive part *he stopped* and a negative part *until he got to Jackson Hole*.

(1) *He did**n't** stop until he got to Jackson Hole.*

Our previous work (Zou et al., 2014) showed that contextual information plays a critical role on negation focus identification. For better illustration of this conclusion, they manually analyze the evidences for 100 negation focuses. It is surprising that 77 focuses can be identified with help of contextual information. This indicates that negation focus is often related with what authors repeatedly states in context. In this paper, we thus focus on graph-based ranking methods (Mihalcea and Tarau, 2004) which first build a word graph according to word co-occurrences within document, and then use random walk algorithms (e.g., PageRank) to measure word importance.

However, for negation focus identification, the graph-based methods may suffer from the following two problems: (a) the words in graph-based methods are strongly connected by co-occurrence rather than semantic content, which do not necessarily guarantee that they are relevant to the negation focus in context; and (b) identifying a negation focus may be affected by not only the relatedness of surrounding words but also its importance in current document which is not considered in standard random walk algorithms.

To address the above problems, we propose a word-topic graph model by adding a topical layer on the original word layer to capture the semantic clues from both lexical and topic perspectives. Besides, a document-sensitive PageRank algorithm is also proposed to optimize the graph model by considering the document's major topics. Experimental results indicate that our word-topic graph model outperforms other baseline methods. Moreover, our model is unsupervised and requires only un-annotated text for training.

The rest of this paper is organized as follows. Section 2 overviews the related work. Section 3 introduces our word-topic graph model with contextual discourse information. Section 4 reports the experimental results and analysis. Finally, we conclude our work in Section 5.

* Corresponding author

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1632–1636,
Lisbon, Portugal, 17-21 September 2015.

2 Related Work

So far there is little work on negation focus identification, which was pioneered by Blanco and Moldovan (2011) who investigated the negation phenomenon in semantic relations and proposed a supervised learning approach to identify the focus of a negation expression. However, although Morante and Blanco (2012) proposed negation focus identification as one of the *SEM'2012 shared tasks, only one team (Rosenberg and Bergler, 2012) participated in. They identified negation focus by three heuristic rules.

Our previous work (Zou et al., 2014) demonstrates the effectiveness of contextual information for negation focus identification. On this basis, we further optimize the graph model in both the topical layer and the PageRank algorithm in this paper.

In recent years, many algorithms are widely used to incorporate word graph models and topical information within random walk. Our work is originally inspired by Liu et al. (2010). Their method runs decomposed Topical PageRank (TPR) for each topic separately, and then calculates the word scores with respect to different topics. When setting the edge weights, only word co-occurrence is considered. Different from their work, our word-topic graph model runs on a two-layers (word layer and topical layer) graph model and sets the edge weights by measuring both word similarity and topic distribution.

3 Methods

The word-topic graph model consists of word layer and topical layer, as shown in Figure 1. While the word layer is constructed according to word co-occurrence within a sliding window, which expresses the cohesion relationship between words in the context, the topical layer is to refine the graph model over the discourse contextual information.

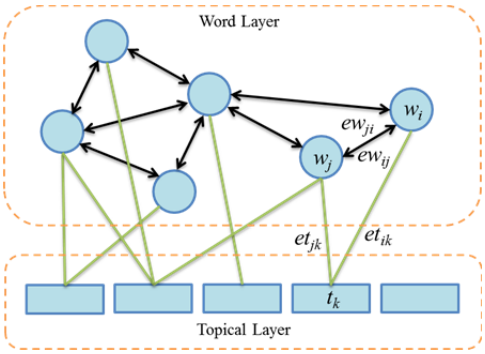


Figure 1. Word-topic graph model.

3.1 Constructing Word Layer

The word layer is constructed according to word co-occurrence within a sliding window, which expresses the cohesion relationship between words in the context. It can be denoted as $L_{word}(W, E_w)$, where vertex set $W=\{w_i\}$ represents the set of words in one document and link set $E_w=\{e_{ij}|w_i,\ w_j \in W\}$ represents the set of directed edges between these words. Note that only content words are considered. Namely, we consider nouns, verbs, adjectives, and adverbs.

The link directions are added from the first word pointing to other words within a sliding *s*-width sentence window. Directed edge ew_{ij} is weighted to represent the relatedness between word w_i and word w_j in a document with transition probability $P_w(j|i)$ from i to j, which is normalized as follows:

$$P_w(j \mid i) = \frac{sim(w_i, w_j)}{\sum_{k:w_i \to w_k} sim(w_i, w_k)}, \quad (1)$$

where the denominator represents the out-degree of vertex w_i , and $sim(w_i,w_j)$ denotes the similarity between word w_i and w_j. In this paper, both corpus-based and knowledge-based methods are evaluated to calculate the similarity between words.

- Word co-occurrence. If word w_i and word w_j occur in a *s*-width sliding sentence window, $sim(w_i,w_j)$ increases 1.
- WordNet similarity (Miller, 1995). In this paper, we adopt the *path similarity* function to measure relatedness of nouns and verbs, and adopt the *similar to* function to measure relatedness of adjectives and adverbs by using the NLTK toolkit[1] (Bird et al., 2009).

Note that $sim(w_i,w_i) = 0$ to avoid self-transition, and $sim(w_i,w_j)$ and $sim(w_j,w_i)$ may not be equal.

3.2 Preliminaries for Topical Layer

To infer the latent topic distributions of words, Latent Dirichlet Allocation (LDA) (Blei et al., 2003), a typical of topic model, is directly applied. By the set of topics which derive from a corpus, we can obtain:

- $P(t|w)$, the probability of topic t given word w, which indicates how much that word w focuses on topic t, and
- $P(t|d)$, the probability of topic t given document d, which indicates how much that document d focuses on topic t.

[1] http://www.nltk.org/

Then, the similarity between two words or between word w_i and document d can be measured by the similarity between their corresponding topic distributions. Formally, we denote a topic distribution as θ, and measure the similarity by using:

- Dot-product. We consider the topic distributions as vectors and apply the dot-product, a geometrically motivated function, to calculate the similarity:

$$Inner(\theta_{w_i}, \theta_{w_j}) = \theta_{w_i} \cdot \theta_{w_j} = \sum_{t_k \in T} P(t_k \mid w_i) \cdot P(t_k \mid w_j), \quad (2)$$

$$Inner(\theta_{w_i}, \theta_d) = \theta_{w_i} \cdot \theta_d = \sum_{t_k \in T} P(t_k \mid w_i) \cdot P(t_k \mid d). \quad (3)$$

- Kullback Leibler (KL) divergence (Lin, 1991). Considering the asymmetry (Eq.(4)), we obtain a symmetrized measure by Eq.(5).

$$D(\theta_i, \theta_j) = \sum_{t_k \in T} P_i(t_k) \log_2 \frac{P_i(t_k)}{P_j(t_k)}. \quad (4)$$

$$D_{KL}(\theta_i, \theta_j) = D(\theta_i, \theta_j) + D(\theta_j, \theta_i). \quad (5)$$

Note that $D_{KL}(\theta_i, \theta_j)$ is undefined if $P_i(t_k)=0$ or $P_j(t_k)=0$ for any $t_k \in T$. For this reason, only the topics which make both $P_i(t_k)\neq 0$ and $P_j(t_k)\neq 0$ are adopted to calculate the KL divergence between two topic distributions.

3.3 Word-Topic Graph Model

The word layer can well capture the relatedness between words, but just partially model the negation focus since it is more directly related with topic than content. Therefore, we add one more layer to refine the graph model over the topical information, as shown in Figure 1. Formally, the word-topic graph is defined as $G_{topic}(W, T, E_w, E_t)$, where vertex set $T=\{t_i\}$ represents the set of topics in all of documents in corpus and link set $E_t = \{et_{ij} | w_i \in W, t_j \in T\}$ represents the set of undirected edges between words and topics.

Considering that the topical layer can provide more contextual semantic information, we refine the relatedness between words by using a topical transition probability $P_t(j|i)$ which is calculated by two kinds of measurements:

$$P_t(j \mid i) = \frac{sim(\theta_{w_i}, \theta_{w_j})}{\sum_{k: w_i \to w_k} sim(\theta_{w_i}, \theta_{w_k})}. \quad (6)$$

Here, the similarity is measured by the dot-product or the KL divergence (using reciprocals). On this basis, the word transition probability $P_w(j|i)$ is updated as following:

$$P_w^{'}(j \mid i) = \mu \cdot P_w(j \mid i) + (1-\mu) \cdot P_t(j \mid i). \quad (7)$$

where $\mu \in [0,1]$ is the coefficient controlling the relative contributions from the lexical information and the topical information.

Moreover, the weights of word vertices are calculated by a PageRank algorithm. In standard PageRank (Page et al., 1998), words are set to be the same value, which indicates there is equal importance to all of words in a document. However, intuitively, we should allocate higher weights to those words with high relevance to the document. Therefore, we assign a document-sensitive value to word w_i:

$$R_d(w_i) = \frac{sim(\theta_{w_i}, \theta_d)}{\sum_{w_k \in W} sim(\theta_{w_k}, \theta_d)} \quad (8)$$

and calculate the weights of word vertices iteratively by using a biased PageRank algorithm:

$$Score^{(n+1)}(w_i) = \lambda \sum_{j \neq i} Score^{(n)}(w_j) P_w^{'}(j \mid i) + (1-\lambda) R_d(w_i). \quad (9)$$

All of the PageRank algorithms are terminated when the number of iterations reaches 100 or the difference of each vertex between consecutive iterations is less than 0.001.

Finally, according to the annotation guidelines (Blanco and Moldovan, 2011), the focus is always a full text of a semantic role. Thus, we select all of semantic roles in sentence as candidate focuses for ranking. The ranking score of a candidate focus f is computed by averaging the scores of all words inside the candidate:

$$score_{avg}(f) = \frac{\sum_{w_i \in f} score(w_i)}{Count(f, \bullet)}, \quad (10)$$

where $count(f,\bullet)$ denotes the number of content words within the candidate. Then the top ranked candidate is chosen as the negation focus.

4 Experimental Results

To evaluate the performance of our word-topic graph model for negation focus identification, we carry out experiments on the *SEM'2012 shared task corpus[2]. As a freely downloadable resource, the corpus is annotated on top of PropBank, which uses the WSJ section of the Penn TreeBank. In particular, negation focus annotation on this corpus is restricted to verbal negations (Blanco and Moldovan, 2011). In total, this corpus provides 3,544 instances of negation focus annotations. Although for each instance, the corpus only provides the current sentence, the previous and next sentences as its context, we sort to

[2] http://www.clips.ua.ac.be/sem2012-st-neg/

the Penn TreeBank[3] to obtain the corresponding document as its discourse context. For fair comparison, we adopt the same partition as *SEM'2012 shared task in our experiments.

We evaluate our results in terms of accuracy. To see whether an improvement is statistically significant, we conduct significance testing using the paired t-test.

For estimating the topical transition probability $P_t(j|i)$ and the document-sensitive value $R_d(w_i)$, we employ GibbsLDA++[4], an LDA model using Gibbs Sampling technique for parameter estimation and inference (Griffiths, 2002). We set the parameters $\alpha = 50/T$ and $\beta = 0.1$ as Griffiths and Steyvers (2004) suggested.

4.1 Influences of Parameters

There are two major parameters in our models that may influence the performance, including: (a) the damping factor μ of the word transition probability $P'_w(j|i)$ (Eq.(7)) and (b) the damping factor λ of the word-topic graph model (Eq.(9)).

Figure 2 shows the accuracy when varying μ from 0.1 to 0.9 with an interval of 0.1 and when varying λ from 0.05 to 1 with an interval of 0.05. We notice that the best performance is achieved when μ=0.6. It indicates that the direct lexical information contributes slightly more than the topical information. The results also show the complementarity between these two kinds of information on negation focus identification.

For λ, it has very little, if any, effect on performance, when λ is set from 0.5 to 0.85. It indicates that the contextual information (the first term in Eq.(9)) contributes more than the document information (the second term in Eq.(9)) on negation focus identification.

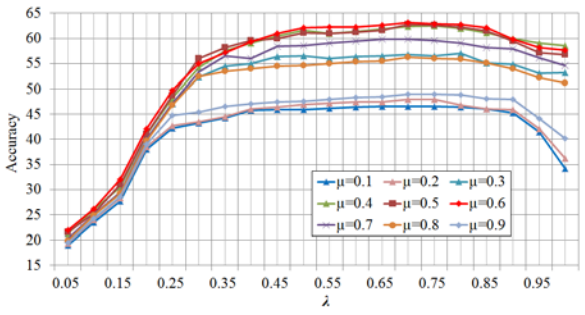


Figure 2. Influence of the damping factors μ and λ.

Moreover, the results also show that these two parameters have little impact in a certain range on performance (μ:0.4~0.6; λ:0.5~0.85), which suggests that the approach is robust to a certain extent. Therefore, we set μ=0.6 and λ=0.7 in the following experiments.

Besides, we also evaluate the other minor parameters in our model. Due to space limit, we do not report all of results here and set parameters to the following values: setting window size s=1 (the previous and next sentences) and the number of topic T=40, adopting the word co-occurrence similarity to calculate the similarity between words, and using dot-product to measure both $P_t(j|i)$ and $R_d(w_i)$.

4.2 Comparison with Other Methods

In the word-topic graph models, two primary improvements are proposed: (a) updating the word transition probability $P_w(j|i)$ by adding a topical layer ("TL"), and (b) assigning a document-sensitive value to word node ("DS").

model	Acc.
WLM	52.61
WTGM (TL)	65.74
WTGM(TL+DS)	69.39

Table 1. Performance of the word-topic graph model.

Table 1 shows that the word-topic graph model (WTGM) is significantly better (+16.78%, p<0.01) than the graph model with only word layer (WLM), which justifies the effectiveness of the topical layer. In addition, the results also indicate that the word-topic graph model not only takes the topical information into account ("TL"), but also considers the semantic relationship in current document ("DS").

We select two supervised baseline methods to compare with our word-topic graph model. One is a decision tree-based system described in Blanco and Moldovan (2011), and the other one is a SVM-based system which takes advantage of both syntactic features and contextual features (Zou et al., 2014).

system	Acc.
B&M(2011)	63.20
Zou et al.(2014)	67.14
Ours	69.39

Table 2. Comparison Results.

Table 2 shows that our word-topic graph model performs significantly better than the two others by 6.19% (p<0.01) and 2.25% (p<0.01), respectively. The results support our viewpoint that the topical information in context can help to find the negation focus, and the word-topic graph model we proposed is effective. Moreover, it is also worth noting that our method is

[3] http://www.cis.upenn.edu/~treebank/
[4] http://gibbslda.sourceforge.net/

unsupervised, which does not need the prior knowledge for training, while the other two supervised baselines employ the golden features, such as the POS tag, constituent tree, and dependency tree.

5 Conclusion

In this paper, we propose an unsupervised word-topic graph model, which represents and measures the word importance by using contextual information from both lexical and topical perspectives. And then, we propose a document-sensitive biased PageRank algorithm to calculate the ranking scores of negation focus candidates. Experimental results show that our method achieves better performance than other baselines without any annotated data.

The main shortcoming of our method is that not all of negation focus can be identified by the context. As our statistics, at least 17% of samples are hard to be determined by human beings when ignoring the information in current sentence. Therefore, in future work, we will focus on investigating an effective method to integrate the local lexical/syntactic information and the global contextual discourse information.

Acknowledgments

This research is supported by the National Natural Science Foundation of China, No.61272260, No.61331011, No.61273320, and the Major Project of College Natural Science Foundation of Jiangsu Province, No.11KJA520003. The authors would like to thank the anonymous reviewers for their insightful comments and suggestions.

Reference

Steven Bird, Edward Loper, and Ewan Klein. 2009. Natural Language Processing with Python. O'Reilly Media Inc.

Eduardo Blanco and Dan Moldovan. 2011. Semantic Representation of Negation Using Focus Detection. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, pages 581-589, Portland, Oregon, June 19-24, 2011.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993-1022, January.

Tom Griffiths. 2002. Gibbs sampling in the generative model of Latent Dirichlet Allocation. Tech. rep., Stanford University.

Thomas L. Griffiths and Mark Steyvers. 2004. Finding scientific topics. In *Proceedings of the National Academy of Sciences of the United States of America*, 101(Suppl. 1), 5228-5235.

Jianhua Lin. 1991. Divergence measures based on Shannon entropy. *IEEE Transactions on Information Theory*, 37(14), 145-151.

Zhiyuan Liu, Wenyi Huang, Yabin Zheng, and Maosong Sun. 2010. Automatic Keyphrase Extraction via Topic Decomposition. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 366-376, MIT, Massachusetts, USA, 9-11 October 2010.

Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into texts. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 404-411.

George A. Miller. 1995. Wordnet: a lexical database for english. Commun. ACM, 38(11):39-41.

Roser Morante and Eduardo Blanco. 2012. *SEM 2012 Shared Task: Resolving the Scope and Focus of Negation. *In Proceedings of the First Joint Conference on Lexical and Computational Semantics*, pages 265-274, Montreal, Canada, June 7-8, 2012.

Lawrence Page, Sergey Brin, Rajeev Motwani, et al. 1998. The pagerank citation ranking: Bringing order to the web. *Technical report*, Stanford University.

Sabine Rosenberg and Sabine Bergler. 2012. UConcordia: CLaC Negation Focus Detection at *Sem 2012. *In Proceedings of the First Joint Conference on Lexical and Computational Semantics*, pages 294-300, Montreal, Canada, June 7-8, 2012.

Bowei Zou, Qiaoming Zhu, and Guodong Zhou. 2014. Negation Focus Identification with Contextual Discourse Information. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 522-530, Baltimore, Maryland, USA, June 23-25 2014.

Reverse-engineering Language: A Study on the Semantic Compositionality of German Compounds

Corina Dima
Collaborative Research Center 833
University of Tübingen, Germany
corina.dima@uni-tuebingen.de

Abstract

In this paper we analyze the performance of different composition models on a large dataset of German compound nouns. Given a vector space model for the German language, we try to reconstruct the observed representation (the corpus-estimated vector) of a compound by composing the observed representations of its two immediate constituents. We explore the composition models proposed in the literature and also present a new, simple model that achieves the best performance on our dataset.

1 Introduction

Vector space models of language like the ones presented in (Collobert et al., 2011b; Mikolov et al., 2013; Pennington et al., 2014) create good representations for the *individual words* of a language. However, the words in a language can be combined into infinitely many distinct, well-formed *phrases* and *sentences*. Creating meaningful, reusable representations for such longer word sequences is still an open problem.

In this paper we focus on building representations for syntactic units *just above the word level*, by exploring compositional models for *compounds*. Bauer (2001) defines a compound as "a lexical unit made up of two or more elements, each of which can function as a lexeme independent of the other(s) in other contexts" (e.g. *apple tree*). The vast majority of compounds are compositional, i.e. we can understand the meaning of the compound if we know the meaning of its constituent words. We would like to equip the vector space model with a composition function able to construct a *composite* representation for *apple tree* from the representations of *apple* and *tree*. The composite representation should ideally be indistinguishable from its *observed* representation, i.e. the representation learned directly by the language model if the compound is part of the dictionary.

We situate our investigations in the context of the German language, a language where compounds represent an important fraction of the vocabulary. Baroni et al. (2002) analyzed the 28 million words German APA news corpus and discovered that compounds account for 47% of the word types but only 7% of the overall token count, with 83% of compounds having a corpus frequency of 5 or lower. The high productivity of the compounding process makes the compositional approach the most tractable way to create meaningful representations for all the compounds that have been or will be coined by the speakers of the German language.

German compounds have a strategic advantage for our study: they are generally written as a contiguous word, irrespective of how many constituents they have. Our example English compound, *apple tree*, translates into the German compound *Apfelbaum*, with the head *Baum* "tree" and the modifier *Apfel* "apple". Because the compound is written as a single word, we can directly learn the representations for the compound and for its constituents. Given a large dataset of German compounds together with their immediate constituents, and the corresponding distributed representations for each of the individual words, one can try to reverse-engineer the compounding process and learn the parameters of a function that combines the representation of the constituents into the representation of the compound. More formally, we are interested in learning a composition function f such that

$$c^{comp} = f(m^{obs}, h^{obs})$$

where $c^{comp} \in \mathbb{R}^n$ is the composite representation of the compound and $m^{obs}, h^{obs} \in \mathbb{R}^n$ are the observed representations of its modifier and its head. The function should minimize J, the mean squared error between the composite (c^{comp}) and

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1637–1642,
Lisbon, Portugal, 17-21 September 2015.

the observed (c^{obs}) representations of the $|C|$ compounds in the training set:

$$J = \sum_{i=1}^{|C|} \frac{1}{n} \sum_{j=1}^{n} (c_{ij}^{comp} - c_{ij}^{obs})^2$$

Several compositionality models have already been proposed in the literature (Mitchell and Lapata, 2010; Baroni and Zamparelli, 2010; Socher et al., 2012). In this paper we evaluate several of the proposed composition functions and also present a new composition model which outperforms all previous models on a dataset of German compounds.

2 Word Representations and Compounds Dataset

We trained 4 vector space language models for German (with 50, 100, 200 and 300 dimensions respectively) using the GloVe package (Pennington et al., 2014) and a 10 billion token raw-text corpus extracted from the DECOW14AX corpus (Schäfer, 2015). We use a vocabulary of 1,029,270 (1M) words, obtained by selecting all the words with a minimum frequency of 100 (the full vocabulary had 50M unique words). We used the default GloVe training parameters, the only modifications being the use of a symmetric context when constructing the co-occurence matrix (10 words to the left and to the right of the target word) and training each model for 15 iterations. All the vector spaces were normalized to the $L2$-norm, first across features then across samples using *scikit-learn* (Pedregosa et al., 2011).

The German compounds dataset used in the experiments is a subset of the 54759 compounds available in GermaNet 9.0[1]. The compounds in the list were automatically split and manually post-corrected (Henrich and Hinrichs, 2011). Each entry in the list is a triple of the form (compound, modifier, head). We filtered the entries in the list, keeping only those where all three words have a minimum frequency of 500 in the support corpus used to create the vector space representations. The reason for the filtering step is that a "well-learned" representation (based on a sufficiently large number of contexts) should allow for a more accurate reconstruction than a representation based only on a few contexts. The filtered dataset contains 34497 entries. This dataset was randomized and partitioned into `train`, `test` and `dev` splits according to the 70-20-10 rule. The dataset contains 8580 unique modifiers and heads, and a dictionary of 41732 unique words. 1345 compounds appear as the modifier or head of another compound.

[1] http://www.sfs.uni-tuebingen.de/lsd/compounds.shtml

3 12 ways to Represent A Compound

We adopt a notation similar to the one introduced in (Mitchell and Lapata, 2010), where the composite representation p is the result of applying a composition function f to the vectors u and v. In this study we tested the following composition functions:

1. $p = v$, the second constituent of the compound
2. $p = u$, the first constituent of the compound
3. $p = u \odot v$, component-wise vector multiplication
4. $p = (u \cdot u)v + (\lambda - 1)(u \cdot v)u$, dilation
5. $p = 0.5u + 0.5v$, vector addition
6. $p = \lambda u + \beta v$, weighted vector addition, where the λ and β are estimated using the training set. Models 1 through 6 were introduced in (Mitchell and Lapata, 2010).
7. $p = Uv$, where $v \in \mathbb{R}^n$ is the vectorial representation of the head word (given) and $U \in \mathbb{R}^{n \times n}$ is a matrix representation for the modifier, estimated with the help of the training data. The model estimates one matrix for each word that is used as a modifier. Referred to as *alm* in (Baroni and Zamparelli, 2010) and as *Lexfunc* in (Dinu et al., 2013b).
8. $p = M_1 u + M_2 v$, where $M_1, M_2 \in \mathbb{R}^{n \times n}$ are two matrices that modify the first and the second constituent vectors, respectively. In contrast to the previous model, this model estimates just one matrix for all the modifiers and one matrix for all the head words. Refered to as EAM in (Zanzotto et al., 2010) and as *Fulladd* in (Dinu et al., 2013b).
9. $p = g(W[u; v])$, where: $[u; v] \in \mathbb{R}^{2n \times 1}$ is the concatenation of the individual word vectors; $W \in \mathbb{R}^{n \times 2n}$ is a global matrix that: (i) combines the individual dimensions of the concatenated input vector $[u; v]$; (ii) brings the composite representation back into the $\mathbb{R}^{n \times 1}$ space; g is an element-wise function, in our experiments the hyperbolic tangent *tanh*. Introduced in (Socher et al., 2010).
10. $p = g(W[Vu; Uv])$. Introduced in (Socher

et al., 2012), it is a generalization of model 7. Each word is represented using an $\mathbb{R}^{n \times n}$ matrix and a $\mathbb{R}^n$ vector. The vectors are given, while the matrices are estimated using the training data. Referred to as *Fulllex* in (Dinu et al., 2013b).

11. $p = u \odot u' + v \odot v''$, the additive mask model (*Addmask*) and
12. $p = g(W[u \odot u'; v \odot v''])$, the global matrix mask model (*Wmask*), both presented in subsection 3.1.

Models 1 through 8 were tested using the implementations available in the DISSECT toolkit (Dinu et al., 2013a). As a side note, the *Lexfunc* implementation in DISSECT does not produce a composite representation for 11.5% of the our test data, where a word does not appear as a modifier during training. Therefore, we reimplemented the *Lexfunc* model and solved the missing training material problem by initializing the matrix for all the words in the dictionary with $I + \epsilon$, the identity matrix plus a small amount of Gaussian noise. This type of initialization was proposed by (Socher et al., 2012), and allows the model to back-off to the model $p = v$ when there is no data to estimate the parameters of the modifier matrix. We also reimplemented models 9 and 10, which were used in (Socher et al., 2010; Socher et al., 2012), as the existing implementations are part of a more complex recursive architecture aimed at constructing representations for full sentences.

3.1 The *mask* models

The newly introduced mask models build upon the idea that when a word w enters a composition process, there is some variation in its meaning depending on whether it is the first or the second element of the composition. Think, for instance, of the compounds *company* ***car*** and ***car*** *factory*. In the first case, *car* has its primary denotation, that of a road vehicle. In the second case, what matters more about the *car* is its *product* aspect, the fact that it is an "artifact produced in a factory". A good representation of the word *car* should encode both aspects. Likewise, a good composition model should be able to select from the individual word representations only those aspects that are relevant for the composition process.

We want to give the composition model the possibility to deal with these slight sense variations, so we train, for each word in the dictionary, two *masks*, one for the case when it is the first word in the composition process and one for when it is the second word. The *masks* of the word w represented by $u \in \mathbb{R}^n$ are two vectors $u', u'' \in \mathbb{R}^n$. The mask vectors are initialized with a vector of all ones, $\mathbf{1}$, and estimated with the help of the training data. Each time w is the first word in the composition process, it is represented as the element-wise multiplication of the vector u and the mask u', $u \odot u'$. When w is the second word in the composition, it is represented by the element-wise multiplication of u and the mask u'', $u \odot u''$.

It is important to note that the initial vector representations remain fixed during the learning process. The learning process only affects the mask vectors. The composite representation of a compound like *car factory* is obtained by combining the masked representations, $u_{car} \odot u'_{car}$ and $v_{factory} \odot v''_{factory}$. We tried two different combination methods: (i) $p = u \odot u' + v \odot v''$, called *Addmask* (model 11), where the masked representations are combined via component-wise addition, and (ii) $p = g(W[u \odot u'; v \odot v''])$, called *Wmask* (model 12), where the combination of the masked representations is made via a global matrix $W \in \mathbb{R}^{n \times 2n}$ and a nonlinearity g (*tanh*), similar to model 10.

3.2 Implementing composition models

Models 7, 9 and 10 and the mask models were implemented using neural network architectures in the Torch7 library (Collobert et al., 2011a). We use the mean squared error as a training criterion, and optimize all models using *Adagrad* (Duchi et al., 2011) and a mini-batch of 100 samples. The hyperparameters were chosen by testing different parameter values and evaluating their performance on the `dev` set. To avoid overfitting we used early stopping (Prechelt, 1998). All the implemented models keep the input vectors fixed during the composition process.

Training the mask models entails estimating modifier and head masks for every word in the dictionary $\mathcal{D}$. The two types of masks to be learned can be formalized as two matrices $W_M, W_H \in \mathbb{R}^{n \times |\mathcal{D}|}$, where n is the size of the initial word representations. The masks of the word $w_i \in \mathcal{D}$ are the i^{th} rows in W_M and W_H. In Torch7 such representations can be learned using *lookup table layers* (Collobert et al., 2011b), which map matrix indices to the corresponding row vector.

The masked representation of the modifier is obtained by first feeding the index of the word to LT_{W_M}, the modifier lookup table, to obtain the modifier mask, and then multiplying the modifier mask with the initial representation for the modifier. The masked representation of the head is obtained in a similar manner via a lookup operation in LT_{W_H}, the head lookup table. The *Addmask* and *Wmask* models differ only in the composition method used after the masking process: the masked representations are directly added together in the case of *Addmask* and are passed through a composition matrix $W \in \mathbb{R}^{n \times 2n}$ and a nonlinearity g in the case of *Wmask*. The two matrices W_M, W_H are initialized with all ones and are modified via backpropagation during the training process.

4 Evaluation and Results

The twelve composition models presented in Section 3 were evaluated using word representations of increasing size (described in Section 2). All the models are trained on the `train` split and tested on the `test` split. We used the rank evaluation method proposed by (Baroni and Zamparelli, 2010) for a similar task: first, we generate a composite representation for each of the 6901 compounds in the test set; then, we use the cosine similarity to rank each composite representation with respect to the observed representations of the 41732 unique words in the dataset dictionary. If the observed representation is the nearest neighbour, the composition is assigned the rank 1. Similar to (Baroni and Zamparelli, 2010), we assign the rank 1000 ($\geq$1K) when the observed representation is not one of the nearest 1000 neighbours of the composite representation. We then compute the first, second and third quartiles (Q1, Q2, Q3) across all the compounds in the test set. A Q1 value of 2 means that the first 25% of the data was only assigned ranks 1 and 2. Similarly, Q2 and Q3 refer to the ranks assigned to the first 50% and 75% of data, respectively. The results of our evaluation are displayed in Table 1.

The observed representation of the head (model 1) was used as a strong baseline for the compound composition task. Two of the tested models, multiplicative (model 3) and dilation (model 4) score worse than the head baseline, while the additive models (5 and 6) score only slightly above it. The fact that the worst performing model is the multiplicative model is surprising considering its good performance in previous studies (Mitchell and Lapata, 2010). This might be either a side-effect of the normalization procedure, or a genuine incompatibility of this compositionality model with the vectorial representations produced by GloVe.

The new *Addmask* and *Wmask* models (introduced in Section 3.1) perform very well, with *Wmask* producing the best results on the test dataset across all dimensions. It is interesting to note that the linguistically motivated *Lexfunc* and *Fulllex* models, which build dedicated representations for each individual constituent, are outperformed by a simple model like *Fulladd*, that only learns two modification matrices, one for each position. The explanation is, in our opinion, that the available training material is not enough for training all the parameters of the complex *Lexfunc* and *Fulllex* models, but good enough for the more simple *Fulladd*.

The mask models are computationally cheaper than models like *Lexfunc* and *Fulllex*, as they they only train $2n$ parameters for each word in the vocabulary, and not n^2 parameters like the aforementioned models. They manage to strike a balance and learn a dedicated representation for each constituent with a small number of parameters, thus performing better than the more complex models.

We used non-parametric statistical tests to detect significant differences between the results obtained by the models. We focused our analysis on the best performing 4 models: model 9, which we will label the *Matrix* model, *Fulladd* (model 8), *Addmask* (model 11) and *Wmask* (model 12). The comparison takes into account two separate factors: (i) differences between the models using representations of the same size; (ii) differences in the performance of the same model using representations of different sizes.

A Friedman test on the ranks obtained by the 4 selected models on representations of size 300 showed that there is a significant difference between the models ($p < 0.01$). Pairwise comparisons (using the Wilcoxon signed rank test and Bonferroni corrections) showed that there is a significant difference ($p < 0.01$) between all but one pair of models, namely the *Matrix* and the *Addmask* models ($p = 0.9$). The same test confirmed that there are significant differences in the performance of the best model *Wmask* when using representations of different sizes ($p < 0.01$). Pairwise

no	f	I	50d			100d			200d			300d		
			Q1	**Q2**	**Q3**	**Q1**	**Q2**	**Q3**	**Q1**	**Q2**	**Q3**	**Q1**	**Q2**	**Q3**
1	$p = v$	D	66	445	≥1K	36	202	≥1K	33	197	989	29	174	884
2	$p = u$	D	445	≥1K	≥1K	215	≥1K	≥1K	171	917	≥1K	144	808	≥1K
3	$p = u \odot v$	D	≥1K	≥1K	≥1K	≥1K	≥1K	≥1K	≥1K	≥1K	≥1K	≥1K	≥1K	≥1K
4	$p = (u \cdot u)v + (\lambda - 1)(u \cdot v)u$	D	75	492	≥1K	38	213	≥1K	35	209	≥1K	30	181	926
5	$p = 0.5u + 0.5v$	D	85	408	≥1K	29	137	600.5	28	140	637	24	120	553
6	$p = \lambda u + \beta v$	D	62	329	≥1K	23.5	118	556	23	121	568	20	105	503
7	$p = Uv$ (on 88.5% of test data)	D	38	415	≥1K	15	147	≥1K	9	61	636	8	47	443
7	$p = Uv$	R	10	88	829	8	64	595	7	48	479.5	7	51	526.5
10	$p = g(W[Vu; Uv])$	R	3	18	178	3	12	111	3	16	188	4	26	334
9	$p = g(W[u; v])$	R	4	19	137	3	11	64	**2**	7	33	**2**	7	29
11	$p = u \odot u' + v \odot v''$	N	3	12	85	3	8	45	3	7	30	3	7	27
8	$p = M_1u + M_2v$	D	4	19	135	3	10	61	**2**	7	33	**2**	**6**	27
12	$p = g(W[u \odot u'; v \odot v''])$	N	**2**	**9**	**62**	**2**	**7**	**35**	**2**	**6**	**25**	**2**	**6**	**24**

Table 1: Quartiles for the 6901 composite representations in the test set, ranked with respect to the observed representations. Best possible rank is 1. D marks the models tested with DISSECT, R marks reimplementations of existing models and N marks new models.

comparisons showed that *Wmask* model significantly improves its performance ($p < 0.01$) when using word representations of increasing size (50, 100, 200 and 300 dimensions).

The twelve composition models were also compared in terms of the mean squared error (MSE) objective function, by computing the MSE between the composite and the observed representation of the compounds in the test set. The best scoring models in the rank evaluation were also the best in the MSE evaluation. However, the difference in performance between the best and the worst models was considerably smaller: the MSE of the multiplicative model is only twice as large as the MSE of the best performing *Wmask* model. This is in contrast to the rank evaluation where the multiplicative model assigned the observed representations in the test set only ranks ≥ 1000, while *Wmask* assigned ranks ≤ 25 to 75% of the test data. Additional investigations are necessary to estimate the impact of different objective functions on the performance of compositional models.

5 Comparison to related work

The experiments reported in this paper are, to the best of our knowledge, the first large scale experiments on the composition of German compounds. Other studies (Kisselew et al., 2015; Lazaridou et al., 2013) focused on morphologically complex words in German and English respectively. In terms of the size of the training and test material, our experiments are closest to the adjective-noun experiments in (Baroni and Zamparelli, 2010) and (Dinu et al., 2013b) where the lexical function model performed the best, with lowest reported median ranks (Q2) above 100.

6 Conclusions

Twelve composition models were evaluated on the task of building compositional representations for German compounds. The best results (median rank 6) were obtained by the newly introduced *Wmask* model, $p = g(W[u \odot u'; v \odot v''])$. The results show that it is possible to learn a composition function specific to compounds, an idea which we would like to further explore using existing compound datasets for English (Ó Séaghdha, 2008; Tratz and Hovy, 2010). The implementation of the newly introduced composition methods can be downloaded from the author's website.

Acknowledgments

The author would like to thank Emanuel Dima, Erhard Hinrichs, Daniël de Kok, Dörte de Kok and Jianqiang Ma, as well as the anonymous reviewers for their insightful comments and suggestions. Financial support for the research reported in this paper was provided by the German Research Foundation (DFG) as part of the Collaborative Research Center "Emergence of Meaning" (SFB 833), project A3.

References

Marco Baroni and Roberto Zamparelli. 2010. Nouns are vectors, adjectives are matrices: Representing adjective-noun constructions in semantic space. In *Proceedings of the Conference on Empirical*

Methods in Natural Language Processing (EMNLP 2010), pages 1183–1193.

Marco Baroni, Johannes Matiasek, and Harald Trost. 2002. Predicting the components of German nominal compounds. In F. van Harmelen, editor, *Proceedings of the 15th European Conference on Artificial Intelligence (ECAI)*, pages 470–474.

Laurie Bauer. 2001. Compounding. In Martin Haspelmath, editor, *Language Typology and Language Universals*. Mouton de Gruyter, The Hague.

Ronan Collobert, Koray Kavukcuoglu, and Clément Farabet. 2011a. Torch7: A Matlab-like environment for machine learning. In *BigLearn, NIPS Workshop*, number EPFL-CONF-192376.

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

Georgiana Dinu, The Pham Nghia, and Marco Baroni. 2013a. DISSECT - DIStributional SEmantics Composition Toolkit. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL 2013)*, pages 31–36, Sofia, Bulgaria.

Georgiana Dinu, The Pham Nghia, and Marco Baroni. 2013b. General estimation and evaluation of compositional distributional semantic models. In *Workshop on Continuous Vector Space Models and their Compositionality*, Sofia, Bulgaria.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159.

Verena Henrich and Erhard W. Hinrichs. 2011. Determining Immediate Constituents of Compounds in GermaNet. In *Proceedings of Recent Advances in Natural Language Processing (RANLP 2011)*, pages 420–426, Hissar, Bulgaria.

Max Kisselew, Sebastian Padó, Alexis Palmer, and Jan Šnajder. 2015. Obtaining a Better Understanding of Distributional Models of German Derivational Morphology. In *Proceedings of the 11th International Conference on Computational Semantics (IWCS 2015)*, pages 58–63, London, UK.

Angeliki Lazaridou, Marco Marelli, Roberto Zamparelli, and Marco Baroni. 2013. Compositional-ly Derived Representations of Morphologically Complex Words in Distributional Semantics. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL 2013)*, pages 1517–1526, Sofia, Bulgaria.

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

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

Diarmuid Ó Séaghdha. 2008. *Learning compound noun semantics*. Ph.D. thesis, Computer Laboratory, University of Cambridge. Published as University of Cambridge Computer Laboratory Technical Report 735.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014)*, volume 12.

Lutz Prechelt. 1998. Early stopping - but when? In *Neural Networks: Tricks of the trade*, pages 55–69. Springer.

Roland Schäfer. 2015. Processing and querying large web corpora with the COW14 architecture. In *Challenges in the Management of Large Corpora (CMLC-3)*.

Richard Socher, Christopher D. Manning, and Andrew Y. Ng. 2010. Learning continuous phrase representations and syntactic parsing with recursive neural networks. In *Proceedings of the NIPS-2010 Deep Learning and Unsupervised Feature Learning Workshop*, pages 1–9.

Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1201–1211.

Stephen Tratz and Eduard Hovy. 2010. A taxonomy, dataset, and classifier for automatic noun compound interpretation. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL-10)*, Uppsala, Sweden.

Fabio Massimo Zanzotto, Ioannis Korkontzelos, Francesca Fallucchi, and Suresh Manandhar. 2010. Estimating Linear Models for Compositional Distributional Semantics. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 1263–1271.

Event Detection and Factuality Assessment with Non-Expert Supervision

Kenton Lee[†], **Yoav Artzi**[‡*], **Yejin Choi**[†], **and Luke Zettlemoyer**[†]

[†] Computer Science & Engineering, University of Washington, Seattle, WA 98195
{kentonl, yejin, lsz}@cs.washington.edu

[‡] Dept. of Computer Science and Cornell Tech, Cornell University, New York, NY 10011
yoav@cs.cornell.edu

Abstract

Events are communicated in natural language with varying degrees of certainty. For example, if you are "hoping for a raise," it may be somewhat less likely than if you are "expecting" one. To study these distinctions, we present scalable, high-quality annotation schemes for event detection and fine-grained factuality assessment. We find that non-experts, with very little training, can reliably provide judgments about what events are mentioned and the extent to which the author thinks they actually happened. We also show how such data enables the development of regression models for fine-grained scalar factuality predictions that outperform strong baselines.

1 Introduction

Interpretation of events—determining what the author claims did or did not happen—is important for many NLP applications, such as news article summarization or biomedical information extraction. However, detecting events and assessing their factuality is challenging. For example, while most non-copular verbs are events, words in general vary with use (e.g. "trade route" vs "*trade* with Iraq"). Events also have widely varying, context-dependent factuality cues, such as event interactions (e.g. "*prevent* easy *access*) and cue words (e.g. "*ordered* to" vs. "*expected* to"). As shown in Figure 1, these are common challenges that a model of event factuality must address.

In this paper, we present new data and models for these tasks, demonstrating that non-experts can provide high-quality annotations which enable fine-grained, scalar judgments of factuality. Unlike previous work, we do not use a detailed specification of exactly what events and factuality classes should be. Instead, we simply ask non-experts to find words describing things that the author claims could have happened, and rate each possibility on a scale of -3 (certainly did not happen) to 3 (certainly did). Figure 1 shows that non-expert workers—when their judgments are aggregated—consistently find a wide range of events and recognize the subtle differences in implied factuality. For example, the event *set* gets a score of 2.6, indicating that it likely but not certainly occurred, since it was *ordered*, whereas the *ordered* event, gets a score of 3.0.

We gather data for event detection and factuality, reusing sentences from the TempEval-3 corpus (Uzzaman et al., 2013). Our approach produces high-quality labels with modest costs. We also introduce simple but highly effective models for both tasks that outperform strong baselines. In

(1) U.S. embassies and military installations around the world were *ordered*$^{(3.0)}$ to *set*$^{(2.6)}$ up barriers and *tighten*$^{(2.6)}$ security to *prevent*$^{(1.8)}$ easy *access*$^{(-2.4)}$ by unauthorized people.

(2) Intel's most powerful computer chip has flaws that could *delay*$^{(0.8)}$ several computer makers' marketing *efforts*$^{(2.6)}$, but the "bugs" aren't *expected*$^{(-2.6)}$ to *hurt*$^{(-2.0)}$ Intel.

(3) President Bush on Tuesday *said*$^{(3.0)}$ the United States may *extend*$^{(1.6)}$ its naval *quarantine*$^{(2.6)}$ to Jordan's Red Sea port of Aqaba to *shut*$^{(1.4)}$ off Iraq's last unhindered trade route.

(4) He also *said*$^{(3.0)}$ of *trade*$^{(-0.8)}$ with Iraq: "There are no shipments at the moment."

Figure 1: Example annotations with italicized event mentions and crowdsourced scalar factuality values $u \in [-3.0, 3.0]$. Positive (or negative) values indicate the extent to which the author claims the events happened (or not).

[*] Work done at the University of Washington.

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1643–1648,
Lisbon, Portugal, 17-21 September 2015.

particular, our factuality regression model uses a learning objective that combines the advantages of LASSO and support vector regression, enabling it to effectively consider sparse lexical cues. By providing scalar factuality judgments for events, our models enable more fine-grained reasoning than previously considered. The corpus and learned models are available online.[1]

2 Related Work

While event definitions have been proposed in several prior studies, existing approaches vary in how they model various linguistic forms such as nominal events, stative events, generic events, and light verbs (Pustejovsky et al., 2003; Palmer et al., 2005; Meyers et al., 2004; Kim et al., 2009; Song et al., 2015). Even with a formal and precise account of events, training annotators to learn all such linguistic intricacies remains a practical challenge. Instead of *definition*-driven instructions, we propose *example*-driven instructions and show their effectiveness.

Previous studies have modeled event factuality assessment as a binary (Diab et al., 2009; Prabhakaran et al., 2010) or multi-class (Sauri and Pustejovsky, 2009) classification task, and they relied on expert annotators. A softer representation was proposed and crowdsourced by de Marneffe et al. (2012), who advocated for representing factuality from the reader's perspective as a distribution of categories, but their annotation process requires manual normalization of the text. In contrast, we model factuality from the author's perspective with scalar values, and we have an end-to-end crowdsourced annotation pipeline.

More recently, Soni et al. (2014) investigated a related problem for quoted statements on Twitter, and they also crowdsourced factuality annotations to learn regression models. While their approach is similar, we focus on predicting factuality for events that occur in every sentence. Without the restrictions of their task, we must reason about a larger variety of contextual cues.

Our method of evaluating annotator agreement (Section 3) is related to the crowdsourcing study by Snow et al. (2008), who showed that pooled non-experts can match or outperform single expert annotators. In contrast, we approximate expert judgments by independently sampling and aggregating sets of non-expert judgments.

[1] http://lil.cs.washington.edu/fact

Data	Documents	Sentences	Tokens
Train	192	2909	73220
Dev.	64	1060	26146
Test	20	274	7004

Figure 2: Corpus statistics.

3 Data Annotation

We use a two-stage annotation pipeline to create the labels shown in Figure 1. Event mentions are first detected, followed by factuality judgments. As motivated in Section 1, we use instructions that are easily understandable by workers with no linguistic training and improve overall quality by aggregating multiple judgments to get the final label.

Event Annotation Given a sentence, we highlight one token at a time and ask workers if it refers to an event. We use the following instructions:

> *We consider events to be things that may or may not occur either in the past, present or future (e.g., earthquake, meeting, jumping, talking, etc.). In some cases, it is not so clear whether a word is referring to an event or not. Consider these harder cases to be events.*

along with 25 example annotations that covered a large variety of cases such as nominals, statives, generic events, light verbs, and non-events. These examples include both toy sentences and sentences from the corpus to annotate. For efficiency, we did not annotate a short list of stop words, copular verbs, and auxiliaries.

Factuality Annotation For factuality, we present a sentence with one highlighted event token at a time with the following prompt:

> *On a scale from 3 to -3, rate how likely the highlighted event did or will happen according to the author of the sentence.*

along with 17 examples to calibrate the annotator's judgments, including negated, conditional, hedged, generic, and nested events. The responses -3, 0, and 3 were given explicit interpretations. 3 and -3 denote respectively that the target event certainly did or did not happen according to the author. 0 denotes that the author is neutral and expresses no bias towards the event's factuality.

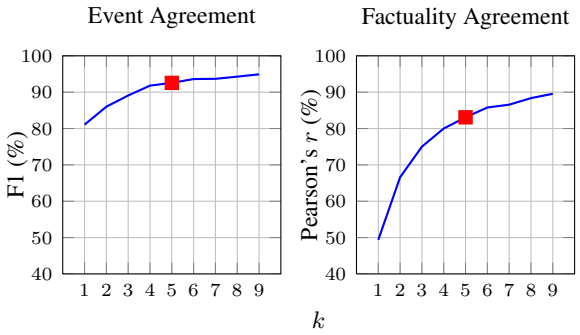


Figure 3: Agreement statistics as a function of k, the number of judgments aggregated. We choose $k = 5$ in both tasks for our experiments, as denoted by the red square.

Data collection We gathered data on CrowdFlower.[2] For quality control, annotators are randomly presented test questions with known answers. For each example, we collect and aggregate 5 judgments, as described below. For comparison, we annotated TempEval-3 (Uzzaman et al., 2013), keeping the existing test split and randomly holding out a quarter of the training examples to create a development set. Figure 2 shows data statistics. The annotation cost is 0.5¢ per judgment for detection and 2¢ per judgment for factuality.

Aggregated Agreement We introduce a simple scheme to measure agreement with aggregate data, for example when the majority class from a pool of judgments is used for the final label. Instead of comparing individuals, we want to know how often the aggregates will agree, if we were to have different groups of annotators doing the task.

Formally, we assume N samples $\{(x_i, y_i) \mid i = 1, \ldots, N\}$, where each x_i is a token within a sentence, and $y_i = \{y_i^j \mid j = 1, \ldots, M\}$ is the set of M judgments for x_i. Let $\mathcal{Y}$ be the set of possible labels, $\mathcal{Y} = \{-1, 1\}$ for detection and $\mathcal{Y} = [-3, 3]$ for factuality. Let $\text{AGG} : \mathcal{Y}^k \rightarrow \mathcal{Y}$ be an aggregation function, which maps k judgments to a single aggregate one. For event detection, we set $\text{AGG}(y^1, \ldots, y^k)$ to return the majority value from the set of judgments $\{y^1, \ldots, y^k\}$. For factuality, we set $\text{AGG}(y^1, \ldots, y^k) = \frac{1}{k} \sum_{j=1}^{k} y^j$, which computes the mean value.

To estimate the agreement between aggregates of k judgments, we collect pairs of disjoint subsets of size k from the M judgments. Given y_i, we define the set of aggregate judgment pairs: $\{(\text{AGG}(y'), \text{AGG}(y'')) \mid y', y'' \subset y_i \wedge |y'| = |y''| = k \wedge y' \cap y'' = \emptyset\}$.[3] To measure how well these aggregates agree, we treat $\text{AGG}(y')$ as a candidate hypothesis and $\text{AGG}(y'')$ as the gold label and compute the appropriate evaluation metric to measure aggregate agreement. We use the F1 score for detection and Pearson's correlation for factuality, as described in Section 5.

We experiment with $k = 1, \ldots, 9$ for 100 sentences, allowing aggregates of up to 9 judgments, as seen in Figure 3. Aggregate agreement for both tasks improve with larger k, but returns quickly diminish. Therefore, we chose $k = 5$ for the full data collection to reasonably trade off between quality and quantity. In absolute terms, the agreement at this level is strong (92.6% F1 for detection and 83.1% correlation for factuality), demonstrating that aggregate non-expert judgments can produce high-quality annotations.

		FactBank Labels								
		CT-	PR-	PS-	PS+	PR+	CT+	CTu	NA	Uu
Discretized ratings	-3	39	0	0	0	0	0	0	0	29
	-2	29	2	0	0	0	0	0	0	44
	-1	16	4	1	0	0	3	0	0	58
	0	15	0	5	2	0	7	0	1	95
	1	7	0	1	30	4	27	2	0	337
	2	4	1	0	20	42	260	0	0	564
	3	2	0	0	1	10	2760	0	0	771

Figure 4: Confusion matrix between FactBank labels and our discretized factuality ratings.

Comparison to FactBank We compare our factuality ratings, rounded to the nearest integer, to FactBank annotations (author source only) for overlapping events. The confusion matrix from Figure 4 shows there is strong correlation between our ratings and FactBank labels with specified certainties and polarities. These labels are CT-, PR-, PS-, PS+, PR+, and CT+, corresponding to events that are seen as (certainly/probably/possibly) (not happening/happening).

We differ most significantly in events labeled Uu (underspecified) by FactBank, which consist largely of nested events, such as "Sandors said he'd *double* his money" or "Sandors hoped he'd *double* his money." While FactBank annotators would label both *double* events as Uu, our annotations can indicate nuances based on the author's wording (i.e., *said* vs. *hoped*). The large variation

[2] http://crowdflower.com

[3] In practice, we sample judgment pairs rather then computing all possible combinations.

in the Uu column of the confusion matrix suggests that the factuality of an event is rarely perceived as completely neutral, even when the author does not commit to a belief in the event's occurrence.

4 Approach

Learning For the detection task, we learn a linear SVM classification model. For the factuality task, we assume a dataset with N examples of labeled events $\{(x_i, y_i) \mid i = 1, \ldots, N\}$, and we learn a regression model: $y_i = w^\top \phi(x_i)$. We introduce a learning objective for regression:

$$\min_w \|w\|_1 + C \sum_{i=1}^{N} \max(0, |y_i - w^\top \phi(x_i)| - \epsilon)$$

that combines the advantages of LASSO (Tibshirani, 1996) and support vector regression (Drucker et al., 1997). It induces sparse feature weights while being insensitive to errors less than ϵ.

Features For the detection model, we include features given the input word x: (1) lemma of x, (2) part of speech of x, (3) indicator for whether x is a hyponym of the *event* synset in WordNet and the part of speech of x, (4) Brown clusters of x and its part of speech, and (5) all dependency paths from x up to length 1. For the factuality model, given the input event mention x, we include: (1) lemma of x, (2) part of speech of x, and (3) all dependency paths from x up to length 2.

For dependency paths, we include all edge labels, the target word is omitted, and each node may or may not be lexicalized; we include all possible configurations. For example in "John did not expect to *return*", the dependency path: *not*←[**neg**]—*expect*—[**xcomp**]→*return*, would produce the following features:

not←[**neg**]—*expect*—[**xcomp**]→⟨∗⟩
⟨∗⟩←[**neg**]—*expect*—[**xcomp**]→⟨∗⟩
not←[**neg**]—⟨∗⟩—[**xcomp**]→⟨∗⟩
⟨∗⟩←[**neg**]—⟨∗⟩—[**xcomp**]→⟨∗⟩

These dependency features allow for context-dependent reasoning, including many of the cases in Figure 1 where the factuality of an event depends on the identity of a neighboring verb.

5 Experimental Setup

Baselines For detection, we include a baseline reimplementation of the NAVYTIME (Chambers, 2013) classification detector, one of the top performers in the TempEval-3 event detection task.

For factuality, we include three baselines: (1) A one-vs.-rest multi-class classifier (DISCRETE) using our features (Section 4) and labels that are discretized by rounding to the nearest integer, (2) a regression model (SVR) trained with the standard SVR objective using our features, and (3) a regression model (PRABHAKARAN) trained with the standard SVR objective using features from Prabhakaran et al. (2010). These features are highly informative, but their lexical features are restricted to a small set of manually defined words.

Implementation Details The SVM models (NAVYTIME, DISCRETE, SVR, PRABHAKARAN, and our detection model) were trained with SVM-Light (Joachims, 1999). We use CPLEX[4] to solve the linear program optimizing the regression objective in Section 4. All hyperparameters were tuned on the development set.

We use the Stanford dependency parser (de Marneffe et al., 2006) for extracting dependency path and part-of-speech features. We use WordNet (Miller, 1995) to generate lemma and hyponym features. Brown clusters with 100, 320, 1000, and 3200 clusters from Turian et al. (2010) are used in the detection features.

Evaluation Metrics We use the standard F1 score for the evaluation of detection. For event factuality, we report two metrics, the mean absolute error (MAE) relative to the gold standard labels and Pearson's correlation coefficient. While MAE is an intuitive metric that evaluates the absolute fit of the model, Pearson's r better captures how well a system is able to recover the variation of the annotations. Pearson's r is also conveniently normalized such that $r = 0$ for a system that blindly chooses the best *a priori* output and $r = 1$ for a system that makes no error.

6 Results

Detection Results Figure 5 shows development and test results for detection event mentions.[5] We see a small drop in precision and large gains in recall, but a significant increase in F1, primarily

[4] http://tiny.cc/cplex

[5] We performed two-sided bootstrap resampling statistical significance tests (Graham et al., 2014). In Figures 5 and 6, asterisks indicate that the difference from the best system is statistically significant ($p < 0.05$).

Model	Dev.			Test		
	P	R	F1	P	R	F1
Our system	**90.1**	**90.9**	**90.5**	85.5*	**87.8**	**86.6**
NAVYTIME	84.7*	79.6*	82.1*	**87.7**	78.3*	82.7*

Figure 5: Results for the detection task.

Model	Dev.		Test	
	MAE	r	MAE	r
Our system	**46.2**	**74.9**	**51.1**	**70.8**
SVR	50.3*	74.8	57.1*	69.4
DISCRETE	50.3*	68.6*	52.4	62.2*
PRABHAKARAN	58.7*	51.1*	62.0*	50.8*

Figure 6: Results for the factuality task.

due to the use of distributional features and more general dependency features.

Factuality Results Figure 6 shows development and test results for predicting the factuality of gold-labeled event mentions. Our system shows an overall improvement in performance over all baselines, demonstrating that the regression model works well for this data. It is able to make more graded judgments that correlate with the aggregate opinions of untrained annotators. As shown in Figure 8, which compares the mean average error for different buckets of factuality labels, we observe the largest gains over PRABHAKARAN in examples with low factuality, where lexical cues are especially critical.

Error Analysis We manually studied 50 development samples where our factuality model produced the largest absolute errors. Figure 7 summarizes the error types. The biggest challenge is the wide variety of sparse lexical cues. For example, the sentences "Wong Kwan will be lucky to *break* even" and "That *sale* could still fall through if financing problems develop" require modeling the influence of "lucky to" and "fall through." Even when these types of features do appear in the training data, they tend to be very rare.

We also find cases that require inference over longer distances than our model permits. Consider the sentence "Mesa had rejected a general proposal from StatesWest to *combine* the two carriers." To know that *combine* is not likely to happen, we must infer that it is conditioned on the proposal, which was rejected. Finally, we find that world knowledge and pragmatic inference is sometimes required. For example, in the sentence "There was no hint of *trouble* in the last

Error type	%
Missed lexical cue (unseen in training)	52
Missed lexical cue (seen in training)	12
Long distance inference	16
World knowledge & pragmatics	12
Annotation error	8

Figure 7: Error types for the 50 examples with the largest absolute development error.

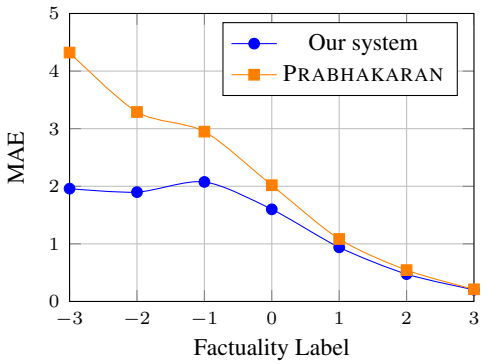


Figure 8: Mean absolute error in the development set for different labels rounded to the nearest integer. Our system's improvement is greater when predicting events with low factuality, which requires modeling sparse lexical cues.

conversation between controllers and TWA pilot Steven Snyder," the pragmatic implication that *trouble* likely happened requires common knowledge about flights.

7 Conclusion

We studied event detection and scalar factuality prediction, demonstrating that non-expert annotator can, in aggregate, provide high-quality data and introducing simple models that perform well on each task. There is significant room for future work to improve the results, including jointly modeling the factuality of multiple events and integrating factuality models into information extraction and question answering systems.

Acknowledgments

This research was supported in part by the NSF (IIS-1252835, IIS-1408287), DARPA under the DEFT program through the AFRL (FA8750-13-2-0019), an Allen Distinguished Investigator Award, and a gift from Google. The authors thank Mark Yatskar, Luheng He, and Mike Lewis for helpful discussions, and the anonymous reviewers for helpful comments.

References

Nathanael Chambers. 2013. Navytime: Event and time ordering from raw text. In *Second Joint Conference on Lexical and Computational Semantics*.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of the 5th International Conference on Language Resources and Evaluation*.

Marie-Catherine de Marneffe, Christopher D. Manning, and Christopher Potts. 2012. Did it happen? The pragmatic complexity of veridicality assessment. *Computational Linguistics*, 38(2):301–333.

Mona T Diab, Lori Levin, Teruko Mitamura, Owen Rambow, Vinodkumar Prabhakaran, and Weiwei Guo. 2009. Committed belief annotation and tagging. In *Proceedings of the Third Linguistic Annotation Workshop*, pages 68–73. Association for Computational Linguistics.

Harris Drucker, Christopher JC Burges, Linda Kaufman, Alex J Smola, and Vladimir Vapnik. 1997. Support vector regression machines. In *Advances in Neural Information Processing Systems*, pages 155–161.

Yvette Graham, Nitika Mathur, and Timothy Baldwin. 2014. Randomized significance tests in machine translation. In *Proceedings of the ACL 2014 Ninth Workshop on Statistical Machine Translation*.

Thorsten Joachims. 1999. Svmlight: Support vector machine. *SVM-Light Support Vector Machine http://svmlight.joachims.org/, University of Dortmund*, 19(4).

Jin-Dong Kim, Tomoko Ohta, Sampo Pyysalo, Yoshinobu Kano, and Jun'ichi Tsujii. 2009. Overview of bionlp'09 shared task on event extraction. In *Proceedings of the Workshop on Current Trends in Biomedical Natural Language Processing: Shared Task*, pages 1–9. Association for Computational Linguistics.

Adam Meyers, Ruth Reeves, Catherine Macleod, Rachel Szekely, Veronika Zielinska, Brian Young, and Ralph Grishman. 2004. The nombank project: An interim report. In *HLT-NAACL 2004 workshop: Frontiers in corpus annotation*, pages 24–31.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational linguistics*, 31(1):71–106.

Vinodkumar Prabhakaran, Owen Rambow, and Mona Diab. 2010. Automatic committed belief tagging. In *Coling 2010: Posters*, pages 1014–1022, Beijing, China, August. Coling 2010 Organizing Committee.

James Pustejovsky, Patrick Hanks, Roser Sauri, Andrew See, Robert Gaizauskas, Andrea Setzer, Dragomir Radev, Beth Sundheim, David Day, Lisa Ferro, et al. 2003. The timebank corpus. In *Corpus linguistics*.

Roser Sauri and James Pustejovsky. 2009. Factbank: a corpus annotated with event factuality. *Language Resources and Evaluation*, 43(3):227–268.

Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Y Ng. 2008. Cheap and fast—but is it good?: evaluating non-expert annotations for natural language tasks. In *Proceedings of the conference on empirical methods in natural language processing*, pages 254–263. Association for Computational Linguistics.

Zhiyi Song, Ann Bies, Stephanie Strassel, Tom Riese, Justin Mott, Joe Ellis, Jonathan Wright, Seth Kulick, Neville Ryant, and Xiaoyi Ma. 2015. From light to rich ere: Annotation of entities, relations, and events. *Proceedings of the 3rd Workshop on EVENTS at the NAACL-HLT*, pages 89–98.

Sandeep Soni, Tanushree Mitra, Eric Gilbert, and Jacob Eisenstein. 2014. Modeling factuality judgments in social media text. In *Proceedings of the Association for Computational Linguistics (ACL)*, Baltimore, MD.

Robert Tibshirani. 1996. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 384–394. Association for Computational Linguistics.

N. Uzzaman, H. Llorens, L. Derczynski, M. Verhagen, J. Allen, and J. Pustejovsky. 2013. Semeval-2013 task 1: Tempeval-3: Evaluating time expressions, events, and temporal relations. In *Proceedings of the International Workshop on Semantic Evaluation*.

Large-Scale Acquisition of Entailment Pattern Pairs by Exploiting Transitivity

Julien Kloetzer[*] **Kentaro Torisawa**[‡] **Chikara Hashimoto**[§] **Jong-Hoon Oh**[¶]
Information Analysis Laboratory,
National Institute of Information and Communications Technology (NICT), Kyoto, Japan
{[*]julien, [‡]torisawa, [§]ch, [¶]rovellia}@nict.go.jp

Abstract

We propose a novel method for acquiring entailment pairs of binary patterns on a large-scale. This method exploits the transitivity of entailment and a self-training scheme to improve the performance of an already strong supervised classifier for entailment, and unlike previous methods that exploit transitivity, it works on a large-scale. With it we acquired 138.1 million pattern pairs with 70% precision with such non-trivial lexical substitution as "*use Y to distribute X*"→"*X is available on Y*" whose extraction is considered difficult. This represents 50.4 million more pattern pairs (a 57.5% increase) than what our supervised baseline extracted at the same precision.

1 Introduction

Recognizing textual entailment (Geffet and Dagan, 2005; Androutsopoulos and Malakasiotis, 2009; Zanzotto et al., 2009; Berant et al., 2011) is an important task for many NLP applications, such as relation extraction (Romano et al., 2006) or question-answering (Harabagiu and Hickl, 2006). Text L ***entails*** text R if the information written in the latter can be deduced from the information written in the former. As building blocks to recognize entailment relations between texts, numerous works have focused on recognizing entailment relations between patterns, such as "*grew up in X*"→"*lived in X*" or "*X grew up in Y*"→"*X lived in Y*" (Lin and Pantel, 2001; Weeds and Weir, 2003a; Hashimoto et al., 2009; Berant et al., 2011; Kloetzer et al., 2013b).

We propose in this paper a method for acquiring on a very large-scale, entailment pairs of such class-dependent binary patterns as "*underwent* X_{exam} *on* Y_{date}"→"X_{exam} *carried out on* Y_{date}". Our starting point is a supervised baseline trained with 83,800 manually labeled pattern pairs detailed in Kloetzer et al. (2013b). Its top 205 million output pairs have an estimated 80% precision, but this baseline's performance is saturated. Table 1 shows the baseline's average precision when varying its amount of hand-labeled training data. Since the average precision only improves slightly with additional training data, the investment in hand-labeling additional training data is difficult to justify.

Quantity of training data	Average precision
Baseline plus 5,000 training data samples	**49.0%**
Baseline: 83,800 training data samples	48.8%
Baseline minus 10,000 training data samples	48.5%
Baseline minus 20,000 training data samples	47.8%

Table 1: Average precision for baseline method with various amounts of training data

To improve our baseline further, we exploit the transitivity property of entailment to automatically generate *new features* for it. The entailment is transitive; if we detect that L entails C and C entails R, we can infer an entailment relation between L and R even if no such relation was detected beforehand. Based on this idea, we propose a self-training scheme that works in the following way. For pattern pair $\langle P,Q\rangle$, we use the baselines output to find all the chains of patterns from P to Q that are linked by entailment relations, which we call *transitivity paths*, and encode the information related to them as new features to judge the validity of pair $\langle P,Q\rangle$. Our expectation is that even if our supervised baseline fails to judge $\langle P,Q\rangle$ as an entailment pair, the existence of paths

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1649–1655,
Lisbon, Portugal, 17-21 September 2015.

from P to Q that are comprised of pairs judged as entailments by our baseline might strongly suggest that P entails Q; hence, adding our new features to the baseline should help it make better decisions based on the information encoded in the features. This self-training approach is the first that encodes the information contained in transitivity paths as features for a classifier, and as such it differs from previous state-of-the-art methods that exploit transitivity to extract new pairs using Integer Linear Programming (Berant et al., 2011) or that auto-generate training data (Kloetzer et al., 2013a).

From a corpus of 600 million web pages, we show that our proposed method extracted 217.8 million entailment pairs in Japanese with 80% precision[1], which is a 6% increase over the 205.3 million pairs output by our baseline with identical precision. It also extracted 138.1 million entailment pairs with 70% precision with non-trivial lexical substitution (generally deemed difficult to extract), which is a 50.4 million pair increase (57.5% size improvement) over the 87.7 million pairs output by our baseline with the same precision. These include such pairs as "*use X to distribute Y*"→"*Y is available on X*", "*underwent X on Y*"→"*X carried out on Y*", "*start X at Y*"→"*Y's X*" or "*attach X to Y*"→"*put X on Y*". Even though we only present results for the Japanese language, we believe that our method should be applicable to other languages as well. This is because none of the few language dependent features of our classifier are strictly needed by the baseline or our proposed method, and its performance boost is unrelated to these features.

2 Related Works

The task of recognizing entailment between texts has been proposed by Dagan et al. (2006) and intensively researched (Malakasiotis and Androutsopoulos, 2007; Szpektor et al., 2004; Androutsopoulos and Malakasiotis, 2009; Dagan et al., 2009; Hashimoto et al., 2009; Berant et al., 2011) using a various range of techniques, including Integer Linear Programming (Berant et al., 2011), machine learning with SVMs (Malakasiotis and Androutsopoulos, 2007), and probabilistic models (Wang and Manning, 2010; Shnarch et al., 2011). Entailment recognizer or entailment data sets have been used in such fields as relation extraction (Romano et al., 2006) and question-answering (Harabagiu and Hickl, 2006; Tanaka et al., 2013). In this work, we are interested into recognizing entailment between syntactic patterns, which can then be used as building blocks in a complete entailment recognition system (Shnarch et al., 2011). Recognizing entailment between patterns has generally been studied using unsupervised techniques (Szpektor et al., 2004; Hashimoto et al., 2009; Weeds and Weir, 2003b), although we showed that supervised techniques naturally obtain stronger performance (Kloetzer et al., 2013b).

The two works that are most closely related to our work are Berant et al. (2011) and Kloetzer et al. (2013a), both of which exploit transitivity to improve the result of a baseline classifier. Berant et al. (2011) proposed an entailment recognition method for binary patterns that exploits Integer Linear Programming techniques (ILP) to expand the results of an SVM classifier. This method encodes into an ILP problem an entailment graph, which is a valued graph where nodes and edges respectively represent patterns and their entailment relations, and the values equal the SVM classifiers score output. The problems variables $E_{PQ} \in \{0, 1\}$ indicate whether pattern pairs (here, $\langle P,Q \rangle$) have an entailment relation, and the goal is to maximize the sum of the scores of the pairs selected as entailment relations $\{\langle P, Q\rangle | E_{PQ} = 1\}$. In (Kloetzer et al., 2013a), we proposed a contradiction acquisition method that uses a training data expansion scheme; it automatically generates new contradictions by exploiting transitivity and adds the highest scoring contradictions based on a novel score (CDP) to the training data of the original classifier. The score is based on the assumption that if pair $\langle P,Q \rangle$, when chained by transitivity to other pairs $\langle Q,R_i \rangle$, generally leads to correct entailment pairs $\langle P,R_i \rangle$, then *all* pairs $\langle P,R_i \rangle$ should be correct entailment pairs. Although this work was designed for contradiction recognition, it is easily adapted to entailment.

3 Target Data and Baseline Classifiers

Target Pattern Pairs We extracted our binary patterns from the TSUBAKI corpus (Shinzato et al., 2008) of 600 million Japanese web pages. Binary patterns are defined as sequences of words on the path of dependency relations connecting two nouns in a sentence and have two variables. "*use Y to distribute X*" and "*X is available on Y*" are such

[1]Examples are given in English for convenience

binary patterns. Like previous works (De Saeger et al., 2009; Berant et al., 2011; Kloetzer et al., 2013a), we pose restrictions on the noun-pairs that co-occur with each pattern using word classes to disambiguate their various potential meanings: "X_{book} *by* Y_{author}" and "$X_{building}$ *by* $Y_{location}$". We used the EM-based noun clustering algorithm presented inKazama and Torisawa (2008) to classify one million nouns into 500 semantic classes. Our target set, to which we apply all of our classifiers, is set Σ of around 11 billion class-dependent pattern pairs for which both patterns share at least three co-occurring noun-pairs.

Baseline Classifier Our baseline classifier (**BASE**) is an SVM classifier trained with about 83,800 binary pattern pairs that were hand-labeled as entailment (25,436 pairs, 30.4% of the total) or non-entailment (58,361 pairs). We trained the classifier using SVMlight software[2] with a polynomial kernel of degree 2.

Following previous work (Kloetzer et al., 2013b), we used three types of features in **BASE**: **surface features** indicate clues like the presence of n-grams or measure the string overlap between two patterns; **database features** exploit existing language resources; and **distributional similarity scores** measure the patterns' semantic similarity based on the nouns that co-occur with them. See Kloetzer et al. (2013b) for more details about **BASE** s features.

4 Proposed Method

Our method consists of the following three steps:

Step 1 Chain together the entailment pairs provided by our baseline classifier **BASE** to form transitivity paths; if $P \rightarrow Q$ and $Q \rightarrow R$, then create path $P \rightarrow Q \rightarrow R$.

Step 2 Train new classifiers with features that encode the information contained in the transitivity paths obtained in Step 1.

Step 3 Combine the output of these classifiers with that of baseline classifier **BASE**.

Figure 1 shows an overview of our method, and we describe its details in the following sections.

[2]http://svmlight.joachims.org/

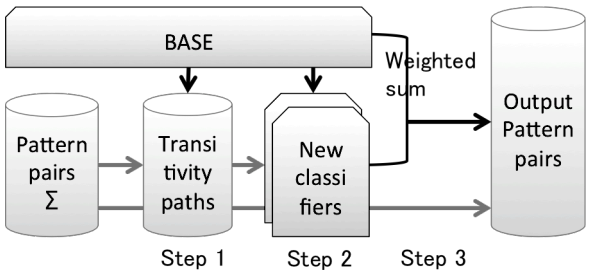


Figure 1: Overview of proposed method

4.1 Step 1: Transitivity Paths

We generate chains of entailment pairs (or transitivity paths) in the following way. First, we extract from the output of the baseline classifier **BASE** set $E(\theta)$ of the pattern pairs for which **BASE** returns a score over given threshold θ: $E(\theta) = \{\langle P, Q\rangle \in \Sigma | S_{BASE}(P, Q) \geq \theta\}$, where $S_{BASE}(P, Q)$ is the score returned by **BASE** for pattern pair $\langle P,Q\rangle$. The higher θ is, the greater the precision of the pairs in $E(\theta)$ should be. Then by chaining the entailment pairs from $E(\theta)$ together, we build sets of transitivity paths composed of two entailment pairs $Tr(\theta, 1)$ for $\theta \in \{0, -\infty\}$ and of three entailment pairs $Tr(\theta, 2)$ for $\theta = 0$. Since additional chaining is computationally expensive, we stopped at paths that consist of three pairs.

4.2 Step 2: Training New Classifiers

In this step, we train new classifiers by adding new features to **BASE**. The training data, the classifier software, and the settings are the same as for **BASE**. For given pattern pair $\langle P,R\rangle$, $Path(P, R, \theta, N)$ is the set of all the transitivity paths in $Tr(\theta, N)$ that lead to pair $\langle P,R\rangle$. We encode the information contained in these paths in three new feature sets.

Before explaining these three new feature sets, we define three scoring functions for the transitivity paths to assess their quality; the *MinScore* of a path is the minimum of scores returned by **BASE** for each pair in the path, and *ArScore* and *GeoScore* are the arithmetic and geometric averages of the scores returned by **BASE** for each pair in the path. Each of the three feature sets is computed for each of the three scoring functions, but we just mention *MinScore* in our explanations due to space limitations.

Feature set 1: scores of top-ranked paths Here we select the top ten paths of $Path(P, R, \theta, N)$ ranked by *MinScore* and use as features a new vec-

tor that consists of the following values: **(1)** the *MinScore* of each path and **(2)** the scores returned by **BASE** for each of the pairs in the ten paths. When there are fewer than ten paths, the missing features are set to 0.

Feature set 2: BASE features of the pairs in the highest ranking transitivity paths Here we select the transitivity path of $Path(P, R, \theta, N)$ with the highest *MinScore* and use the **BASE** feature values for each pair in the transitivity path as a new feature vector for pair $\langle P,R \rangle$.

Feature set 3: score distribution Given threshold α, we count the number of paths whose *MinScore* exceeds α. By varying α from lower bound low to upper bound up, we derive vector $[|\{p \in Path(P, R, \theta, N) | MinScore(p) \geq \alpha\}|]_{\alpha \in \{low, low+\beta, low+2*\beta, \ldots, up\}}$ and use it as a new feature vector for pair $\langle P,R \rangle$. We set $\beta = 0.1$ and low and up such that the score values returned by **BASE** are bounded by low and up.

4.3 Step 3: Optimization and Weighted Sum Classifier Combination

The final output of our method combines the outputs of **BASE** and two new classifiers: **(1)** a classifier with new features computed with 1-step transitivity paths ($N = 1$), and **(2)** another with new features computed with up to 2-step transitivity paths ($N \in \{1, 2\}$). We then use a weighted sum and compute score $S_{PROPOSED}(P, Q) = \sum_i n_i * S_i(P, Q)$ for each pair $\langle P,Q \rangle$. S_i represents the scores of the respective classifiers, and we set $n_0 + n_1 + n_2 = 100$ (n_i are all natural numbers).

For each potential combination of three weights n_i, we computed the average precision returned by our method on **DEV**, our development set, and selected for our final output the weight combination that gave the best average precision on **DEV**. The final classifiers weights obtained in our experiments were 62 for **BASE**, 30 for the classifier with 1-step transitivity features, and 8 for the one with the 1- and 2-step transitivity features.

Using the same method, we also performed ablation tests to remove the features that harmed the classifiers and ensured that every proposed new feature set and every scoring function were useful.

Finally, we confirmed that using a weighted sum for our proposed method returned higher average precision than Stacking (Wolpert, 1992), which is a more standard combination method, or than using the output of any of the new classifiers alone.

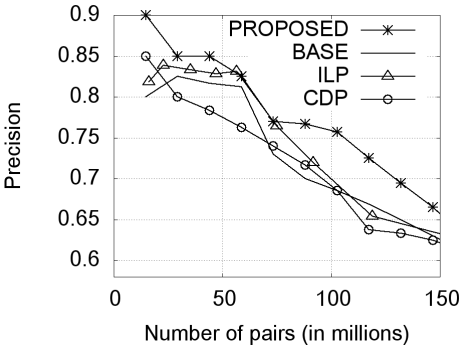


Figure 2: Precision curves for **PROPOSED** and baseline methods for *non-similar* pairs

5 Experiments

In this section, we evaluate our proposed method in a large-scale setting and compare it to **BASE** and to state-of-the-art methods based on ILP (Berant et al., 2011) and automatic training data expansion (Kloetzer et al., 2013a). We also indicate that our method shows the best performance gain for pattern pairs with non-trivial lexical substitutions, which are more difficult to acquire.

Evaluation Method For our evaluation, we prepared test set **TEST** of 15,000 pattern pairs randomly sampled from Σ (our target set of 11 billion pairs). The pairs were annotated by three humans (not the authors) who voted to settle labeling discrepancies. We also prepared development set **DEV** of $5,000$ pattern pairs from Σ for tuning our method. The Kappa score was 0.55 for the annotation of these two sets.

We measured the performance of each method by computing its *average precision* (Manning and Schütze, 1999) on the **TEST** set. We used the average precision instead of the traditional F-value because the latters value greatly varies depending on where the classification boundary is drawn, even for similar rankings. We also drew precision curves for each method using the same **TEST** set.

Proposed Methods Performance We first show the performance of **PROPOSED** (our proposed method) and **BASE** (the baseline classifier). As another baseline, we consider **BASE +DEV** where the 5,000 samples of the **DEV** set were added to the **BASE** training data. We show the average precision for each of these three classifiers and the

Classifier	Average precision	Million pairs at 80% prec.
PROPOSED	**50.64%**	**217.8**
BASE + DEV	48.96%	202.4
BASE	48.79%	205.3
ILP	N/A	205.2
CDP	48.42%	198.0

Table 2: Average precision and entailment pairs obtained (in millions) for proposed method, baseline classifiers, and state-of-the-art methods

Classifier	Av. precision (*similar*)	Av. precision (*non-similar*)
PROPOSED	**78.73%**	**39.53%**
BASE + DEV	77.72%	37.24%
BASE	77.85%	36.98%

Table 3: Average precision for *similar* and *non-similar* pairs

number of pairs obtained at 80% precision in Table 2. We also show the performance of these classifiers over *similar* pattern pairs (both patterns share a content word) and *non-similar* pairs (they do not share a content word) in Table 3.

As mentioned in the introduction, **BASE +DEV** shows that the addition of 5,000 hand-labeled samples to the training data of **BASE** (a 6% increase) only improves the average precision performance by 0.17%. Our proposed method, on the other hand, exploits the same 5,000 new annotated samples for tuning its parameters and obtains a 1.85% gain of average precision. Using **PROPOSED**, we acquired 217.8 million pattern pairs with 80% precision, an improvement of 6.0% over **BASE**.

As shown in Table 3, **BASE** s performance is much lower for *non-similar* pairs like "*use Y to distribute X*"→"*X is available on Y*", which have non-trivial lexical substitutions and are more difficult to acquire than *similar* pairs. This is also where **PROPOSED** obtains the biggest gain in performance: an average precision of 39.53 compared to 36.98 for **BASE**. We show the precision curves we obtained when ranking the *non-similar* pairs with **BASE** and **PROPOSED** in Fig. 2. **PROPOSED** acquired 138.1 million *non-similar* pairs at 70% precision, which is an increase of 50.4 million pairs (a 57.5% size improvement) compared to **BASE** with the same precision. We believe that the strong performance of **BASE** for *similar* entailment pairs helped it discover, through transitivity, the variations of *non-similar* entailment pairs it could already detect.

Comparison to State-of-the-art Methods We also compared **PROPOSED** with two state-of-the-art methods that exploit transitivity: the ILP-based method of Berant et al. (2011) (**ILP**) and the training data expansion method we proposed in Kloetzer et al. (2013a) (**CDP**). The latter, which was initially designed for acquiring contradiction pairs, was adapted to acquire entailment for comparison purposes. The results of this comparison are summarized in Table 2, and the precision curves for these two methods as well over *non-similar* pairs are shown in Fig. 2. Our proposed method is the only one that provides stable improvement in our large-scale setting; at best, the other two just slightly outperform **BASE**. We believe that our feature encoding provides more information to the classifier than the raw scores in the transitivity paths that are exploited by the other state-of-the-art methods, and as such strengthens the performance.

As for explaining the poor performance of the state-of-the-art methods, ILP is unfortunately not tractable for big problems; our ILP solver failed to solve 82% of the independent problems we fed it due to insufficient memory even on 64-Gb memory machines, making **ILP** just slightly better than **BASE**. Our pattern graph is also sparser than that in Berant et al. (2011), and as such ILP might not be completely efficient. But we assume that even if we had used the graphs whole closure, the ILP problem instances would have become even less tractable, resulting in performance that only slightly exceeds **BASE**. As for **CDP**, since our baseline classifier already has more than 80,000 hand-annotated samples as training data, the addition of automatically generated training samples is actually harmful.

6 Conclusion

In this work, we proposed a method that exploits the transitivity relation of entailment in a self-training scheme and combines classifiers with a weighted sum. In our large-scale setting, our method outperforms state-of-the-art methods that are also based on a transitivity approach, including an ILP-based method. Using our proposed method, we acquired 217.8 million Japanese entailment pairs with 80% precision and 138.1 million non-trivial pairs with 70% precision. We are considering an extrinsic evaluation for these data such as the RTE test in future research.

References

Ion Androutsopoulos and Prodromos Malakasiotis. 2009. A survey of paraphrasing and textual entailment methods. *arXiv preprint arXiv:0912.3747*.

Jonathan Berant, Ido Dagan, and Jacob Goldberger. 2011. Global learning of typed entailment rules. In *Proceedings of ACL 2011*, pages 610–619.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The pascal recognising textual entailment challenge. In *Machine learning challenges. evaluating predictive uncertainty, visual object classification, and recognising tectual entailment*, pages 177–190. Springer.

Ido Dagan, Bill Dolan, Bernardo Magnini, and Dan Roth. 2009. Recognizing textual entailment: Rational, evaluation and approaches. *Natural Language Engineering*, 15(4):1–17.

Stijn De Saeger, Kentaro Torisawa, Jun'ichi Kazama, Kow Kuroda, and Masaki Murata. 2009. Large scale relation acquisition using class dependent patterns. In *Proceedings of ICDM 2009*, pages 764–769.

Maayan Geffet and Ido Dagan. 2005. The distributional inclusion hypotheses and lexical entailment. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 107–114. Association for Computational Linguistics.

Sanda Harabagiu and Andrew Hickl. 2006. Methods for using textual entailment in open-domain question answering. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 905–912. Association for Computational Linguistics.

Chikara Hashimoto, Kentaro Torisawa, Kow Kuroda, Stijn De Saeger, Masaki Murata, and Jun'ichi Kazama. 2009. Large-scale verb entailment acquisition from the web. In *Proceedings of EMNLP 2009*, volume 3, pages 1172–1181.

Junichi Kazama and Kentaro Torisawa. 2008. Inducing gazetteers for named entity recognition by large-scale clustering of dependency relations. *Proceedings of ACL 2008*, pages 407–415.

Julien Kloetzer, Stijn De Saeger, Kentaro Torisawa, Chikara Hashimoto, Jong-Hoon Oh, Kiyonori Ohtake, and Motoki Sano. 2013a. Two-stage method for large-scale acquisition of contradiction pattern pairs using entailment. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 693–703.

Julien Kloetzer, Stijn De Saeger, Kentaro Torisawa, Motoki Sano, Chikara Hashimoto, and Jun Gotoh. 2013b. Large-scale acquisition of entailment pattern pairs. In *Information Processing Society of Japan (IPSJ) Kansai-Branch Convention*.

Dekang Lin and Patrick Pantel. 2001. Dirt - discovery of inference rules from text. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 323–328.

Prodromos Malakasiotis and Ion Androutsopoulos. 2007. Learning textual entailment using SVMs and string similarity measures. In *Proceedings of the ACL- PASCAL Workshop on Textual Entailment and Paraphrasing*, pages 42–47.

Christopher D Manning and Hinrich Schütze. 1999. *Foundations of statistical natural language processing*. MIT press.

Lorenza Romano, Milen Kouylekov, Idan Szpektor, and Ido Dagan. 2006. Investigating a generic paraphrase-based approach for relation extraction. In *Proceedings of EACL*, pages 409–416.

Keiji Shinzato, Tomohide Shibata, Daisuke Kawahara, Chikara Hashimoto, and Sadao Kurohashi. 2008. TSUBAKI: an open search engine infrastructure for developing new information access methodology. *Proceedings of IJCNLP2008*.

Eyal Shnarch, Jacob Goldberger, and Ido Dagan. 2011. A probabilistic modeling framework for lexical entailment. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers - Volume 2*, HLT '11, pages 558–563, Stroudsburg, PA, USA. Association for Computational Linguistics.

Idan Szpektor, Hristo Tanev, Ido Dagan, Bonaventura Coppola, et al. 2004. Scaling web-based acquisition of entailment relations. In *Proceedings of EMNLP*, volume 4, pages 41–48.

Masahiro Tanaka, Stijn De Saeger, Kiyonori Ohtake, Chikara Hashimoto, Makoto Hijiya, Hideaki Fujii, and Kentaro Torisawa. 2013. Wisdom2013: A large-scale web information analysis system. In *Sixth International Joint Conference on Natural Language Processing*, pages 58–61.

Mengqiu Wang and Christopher D Manning. 2010. Probabilistic tree-edit models with structured latent variables for textual entailment and question answering. In *Proceedings of the 23rd International Conference on Computational Linguistics*, COLING '10, pages 1164–1172, Beijing, China. Association for Computational Linguistics. ACM ID: 1873912.

Julie Weeds and David Weir. 2003a. A general framework for distributional similarity. In *Proceedings of EMNLP 2003*, pages 81–88. Association for Computational Linguistics.

Julie Weeds and David Weir. 2003b. A general framework for distributional similarity. In *Proceedings of the 2003 conference on Empirical methods in natural language processing*, pages 81–88. Association for Computational Linguistics.

David H Wolpert. 1992. Stacked generalization. *Neural networks*, 5(2):241–259.

Fabio Massimo Zanzotto, Marco Pennacchiotti, and Alessandro Moschitti. 2009. A machine learning approach to textual entailment recognition. *Natural Language Engineering*, 15(04):551–582.

Context-Dependent Knowledge Graph Embedding

Yuanfei Luo[1,2], **Quan Wang**[1*], **Bin Wang**[1], **Li Guo**[1]
[1]Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
{luoyuanfei,wangquan,wangbin,guoli}@iie.ac.cn
[2]University of Chinese Academy of Sciences, Beijing, China

Abstract

We consider the problem of embedding knowledge graphs (KGs) into continuous vector spaces. Existing methods can only deal with explicit relationships within each triple, i.e., local connectivity patterns, but cannot handle implicit relationships across different triples, i.e., contextual connectivity patterns. This paper proposes context-dependent KG embedding, a two-stage scheme that takes into account both types of connectivity patterns and obtains more accurate embeddings. We evaluate our approach on the tasks of link prediction and triple classification, and achieve significant and consistent improvements over state-of-the-art methods.

1 Introduction

Knowledge Graphs (KGs) like WordNet (Miller, 1995), Freebase (Bollacker et al., 2008), and DBpedia (Lehmann et al., 2014) have become extremely useful resources for many NLP-related applications. A KG is a directed graph whose nodes correspond to entities and edges to relations. Each edge is a triple of the form (h, r, t), indicating that entities h and t are connected by relation r. Although powerful in representing complex data, the symbolic nature makes KGs hard to manipulate.

Recently, knowledge graph embedding has attracted much attention (Bordes et al., 2011; Bordes et al., 2013; Socher et al., 2013; Wang et al., 2015). It attempts to embed entities and relations in a KG into a continuous vector space, so as to simplify the manipulation while preserving the inherent structure of the original graph.

Most of the existing KG embedding methods model triples individually, ignoring the fact that entities connected to a same node are usually implicitly related to each other, even if they are not directly connected. Figure 1 gives two examples. `Shaquille_O_Neal` and `NBA` in the former example and `Nevada` and `Utah` in the latter example are implicitly related to each other, through the intermediate nodes `Phoenix_Suns` and `USA` respectively. We refer to such implicit relationships as *contextual connectivity patterns* (CCPs). Relationships explicitly represented in triples are referred to as *local connectivity patterns* (LCPs). In most of the existing methods, only LCPs are explicitly modeled.

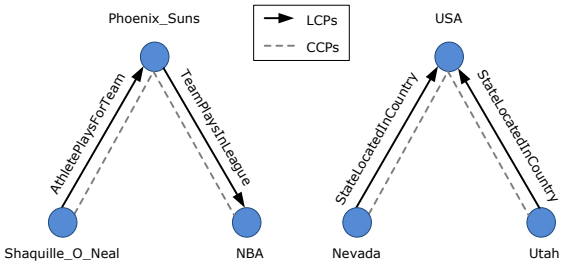


Figure 1: LCPs and CCPs.

This paper proposes a two-stage embedding scheme that explicitly takes into account both CCPs and LCPs, called *context-dependent KG embedding*. In the first stage, each CCP is formalized as a *knowledge path*, i.e., a sequence of entities and relations occurring in the pattern. A word embedding model is adopted to learn embeddings of entities and relations, by taking them as pseudo-words. The embeddings are enforced compatible within each knowledge path, and hence can capture CCPs. In the second stage, the learned embeddings are fine-tuned by an existing KG embedding technique. Since such a technique requires the embeddings to be compatible on each individual triple, LCPs are also encoded.

The advantages of our approach are three-fold. 1) It fully exploits both CCPs and LCPs, and can

*Corresponding author: Quan Wang.

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1656–1661,
Lisbon, Portugal, 17-21 September 2015.

obtain more accurate embeddings. 2) It is a general scheme, applicable to a wide variety of word embedding models in the first stage and KG embedding models in the second. 3) No auxiliary data is further required in the two-stage process, except for the original graph.

We evaluate our approach on two publicly available data sets, and achieve significant and consistent improvements over state-of-the-art methods in the link prediction and triple classification tasks. The learned embeddings are not only more accurate but also more stable.

2 Context-Dependent KG Embedding

We are given a KG with nodes corresponding to entities and edges to relations. Each edge is denoted by a triple (h, r, t), where h is the head entity, t the tail entity, and r the relation between them. Entities and relations are represented as vectors, matrices, or tensors in a continuous vector space. Context-dependent KG embedding aims to automatically learn entity and relation embeddings, by using observed triples $\mathcal{O}$ in a two-stage process.

2.1 Modeling CCPs

The first stage models CCPs conveyed in the KG. Each CCP is formalized as a knowledge path, i.e., a sequence of entities and relations occurring in the pattern. For the CCPs in Figure 1, the associated knowledge paths are:

"`Shaquille_O_Neal`, `AthletePlaysForTeam`, `Phoenix_Suns`, `TeamPlaysInLeague`, `NBA`"

"`Nevada`, `StateLocatedInCountry`, `USA`, `StateLocatedInCountry`, `Utah`".

We fix the length of knowledge paths to 5. During path extraction, we ignore the directionality of edges, and treat the KG as an undirected graph.[1]

Given the extracted knowledge paths, we employ word embedding models to pre-train the embeddings of entities and relations, by taking them as pseudo-words. We use two word embedding models: CBOW and Skip-gram (Mikolov et al., 2013a; Mikolov et al., 2013b). In CBOW, words in the context are projected to their embeddings and then summed. Based on the summed embedding, log-linear classifiers are employed to predict the current word. In Skip-gram, the current word is projected to its embedding, and log-linear classifiers are further adopted to predict its context. We restrain the context of a word (i.e. entity/relation) within each knowledge path. The entity and relation embeddings pre-trained in this way are required to be compatible within each knowledge path, and thus can encode CCPs.

Perozzi et al. (2014) and Goikoetxea et al. (2015) have proposed similar ideas, i.e., to generate random walks from online social networks or from the WordNet knowledge base, and then employ word embedding techniques on these random walks. But our approach has two differences. 1) It deals with heterogeneous graphs with different types of edges. Both nodes (entities) and edges (relations) are included during knowledge path extraction. However, the previous studies focus only on nodes. 2) We devise a two-stage scheme where the embeddings learned in the first stage will be fine-tuned in the second one, while the previous studies take such embeddings as final output.

2.2 Modeling LCPs

The second stage models LCPs conveyed in the KG. We employ three state-of-the-art KG embedding models, namely SME (Bordes et al., 2014), TransE (Bordes et al., 2013), and SE (Bordes et al., 2011) to fine-tune the pre-trained embeddings. These three models work in the following way. First, entities are represented as vectors, and relations as operators in an embedding space, characterized by vectors (SME and TransE) or matrices (SE). Then, for each triple (h, r, t), an energy function $f_r(h, t)$ is defined to measure its plausibility. Plausible triples are assumed to have low energies. Finally, to obtain entity and relation embeddings, a margin-based ranking loss, i.e.,

$$\mathcal{L} = \sum_{t^+ \in \mathcal{O}} \sum_{t^- \in \mathcal{N}_{t^+}} \left[\gamma + f_r(h, t) - f_r(h', t')\right]_+ ,$$

is minimized. Here, $t^+ = (h, r, t) \in \mathcal{O}$ is an observed (positive) triple; $\mathcal{N}_{t^+}$ is the set of negative triples constructed by replacing entities in t^+, and $t^- = (h', r, t') \in \mathcal{N}_{t^+}$; γ is a margin separating positive and negative triples; $[x]_+ = \max(0, x)$. Table 1 summarizes the entity/relation embeddings and the energy functions used in SME, TansE, and SE. For other KG embedding models, please refer to (Nickel et al., 2011; Riedel et al., 2013; Wang et al., 2014; Chang et al., 2014).

We adopt stochastic gradient descent to solve the minimization problem, by taking entity and relation embeddings pre-trained in the first stage as

[1] Two entities connected to a same node are always expected to have some implicit relationships, no matter how they are connected to the intermediate node.

Method	Entity/Relation embedding	Energy function
SME (linear) (Bordes et al., 2014)	$\mathbf{h}, \mathbf{t} \in \mathbb{R}^k$, $\mathbf{r} \in \mathbb{R}^k$	$f_r(\mathbf{h}, \mathbf{t}) = (\mathbf{W}_{u1}\mathbf{r} + \mathbf{W}_{u2}\mathbf{h} + \mathbf{b}_u)^T (\mathbf{W}_{v1}\mathbf{r} + \mathbf{W}_{v2}\mathbf{t} + \mathbf{b}_v)$
SME (bilinear) (Bordes et al., 2014)	$\mathbf{h}, \mathbf{t} \in \mathbb{R}^k$, $\mathbf{r} \in \mathbb{R}^k$	$f_r(\mathbf{h}, \mathbf{t}) = ((\underline{\mathbf{W}}_u \bar{\times}_3 \mathbf{r})\,\mathbf{h} + \mathbf{b}_u)^T ((\underline{\mathbf{W}}_v \bar{\times}_3 \mathbf{r})\,\mathbf{t} + \mathbf{b}_v)$
TransE (Bordes et al., 2013)	$\mathbf{h}, \mathbf{t} \in \mathbb{R}^k$, $\mathbf{r} \in \mathbb{R}^k$	$f_r(\mathbf{h}, \mathbf{t}) = \|\mathbf{h} + \mathbf{r} - \mathbf{t}\|_{\ell_1}$
SE (Bordes et al., 2011)	$\mathbf{h}, \mathbf{t} \in \mathbb{R}^k$, $\mathbf{R}_u, \mathbf{R}_v \in \mathbb{R}^{k \times k}$	$f_r(\mathbf{h}, \mathbf{t}) = \|\mathbf{R}_u\mathbf{h} - \mathbf{R}_v\mathbf{t}\|_{\ell_1}$

Table 1: Entity/Relation embeddings and energy functions used in KG embedding methods.

	# rel.	# ent.	# trip. (train/valid/test)			# path
WN18	18	40,943	141,442	5,000	5,000	5,674,308
NELL186	186	14,463	31,134	5,000	5,000	1,914,475

Table 2: Statistics of the data sets.

initial values.[2] The entity and relation embeddings fine-tuned in this way are required to be compatible within each triple, and thus can encode LCPs.

Socher et al. (2013) have proposed a similar idea, i.e., to use embeddings learned from an auxiliary corpus as initial values. However, linking entities recognized in an auxiliary corpus to those occurring in the KG is always a non-trivial task. Our approach requires no auxiliary data, and naturally avoids the entity linking task.

3 Experiments

We test our approach on the tasks of link prediction and triple classification. Two publicly available data sets are used. The first is WN18 released by Bordes et al. (2013)[3]. It is a subset of WordNet, consisting of 18 relations and the entities connected by them. The second is NELL186 released by Guo et al. (2015)[4], containing the most frequent 186 relations in NELL (Carlson et al., 2010) and the associated entities. Triples are split into training/validation/test sets, used for model training, parameter tuning, and evaluation respectively. Knowledge paths are extracted from training sets. Table 2 gives some statistics of the data sets.

To perform context-dependent KG embedding, we use CBOW and Skip-gram in the pre-training stage, and SME, TransE, and SE in the fine-tuning stage. We take randomly initialized SME, TransE, and SE as baselines, denoted as *-Random. We do not compare to the setting that employs only CBOW or Skip-gram, since it does not provide an energy function to calculate triple plausibility, which hinders the evaluation of both tasks.

[2]For SE, only entity vectors are initialized by pre-trained embeddings. Relation matrices are randomly initialized.

[3]https://everest.hds.utc.fr/doku.php?id=en:smemlj12

[4]http://www.aclweb.org/anthology/P/P15/

3.1 Link Prediction

Link prediction is to predict whether there is a specific relation between two entities.

Evaluation Protocol. For each test triple, the head is replaced by every entity in the KG, and the energy is calculated for each corrupted triple. Ranking the energies in ascending order, we get the rank of the correct answer. We can get another rank by corrupting the tail. We report two metrics on the test sets: Mean (averaged rank) and Hits@10 (proportion of ranks no larger than 10).

Implementation Details. To train CBOW and Skip-gram, we use the word2vec implementations[5]. 20 negative samples are drawn for each positive one. The context size is fixed to 5. To train SME, TransE, and SE, we use the implementations provided by the authors[6], with 100 mini-batches. We vary the learning rate in $\{0.01, 0.1, 1, 10\}$, the dimension k in $\{20, 50\}$, and the margin γ in $\{1, 2, 4\}$. The best model is selected by monitoring Hits@10 on the validation sets, with a total of at most 1000 iterations over the training sets.

Results. Table 3 reports the results on the test sets of WN18 and NELL186. The improvements of CBOW/Skip-gram over Random are also given. Statistically significant improvements are marked by ‡ (sign test, significance level 0.05). The results show that a pre-training stage consistently improves over the baselines for all the methods on both data sets. Almost all of the improvements are statistically significant.

3.2 Triple Classification

Triple classification aims to verify whether an unseen triple is correct or not.

Evaluation Protocol. Triples in the validation and test sets are labeled as positive instances. For each positive instance, we construct a negative instance by randomly corrupting the entities. During

[5]https://code.google.com/p/word2vec/

[6]https://github.com/glorotxa/SME

		Mean			Hits@10 (%)		
		Random	CBOW	Skip-gram	Random	CBOW	Skip-gram
WN18	SME (linear)	463.2	‡286.5 (↓38%)	226.9 (↓51%)	63.98	‡68.65 (↑7%)	‡70.01 (↑9%)
	SME (bilinear)	551.8	‡308.8 (↓44%)	‡279.2 (↓49%)	63.83	‡67.65 (↑6%)	‡67.53 (↑6%)
	TransE	723.1	‡293.0 (↓59%)	‡290.0 (↓60%)	78.50	‡79.67 (↑1%)	‡79.87 (↑2%)
	SE	960.0	‡426.2 (↓56%)	‡289.4 (↓70%)	71.53	‡76.05 (↑6%)	‡75.89 (↑6%)
NELL186	SME (linear)	595.5	‡371.9 (↓38%)	‡340.3 (↓43%)	29.82	‡34.22 (↑15%)	‡35.57 (↑19%)
	SME (bilinear)	375.2	‡305.0 (↓19%)	‡292.9 (↓22%)	37.45	‡39.31 (↑ 5%)	‡39.70 (↑ 6%)
	TransE	732.6	‡384.6 (↓48%)	‡384.6 (↓48%)	27.60	‡28.71 (↑ 4%)	‡30.52 (↑11%)
	SE	2307.0	‡1314.7 (↓43%)	‡412.2 (↓82%)	19.53	‡26.15 (↑34%)	‡31.12 (↑59%)

Table 3: Link prediction results on the test sets of WN18 and NELL186.

		Micro-ACC (%)			Macro-ACC (%)		
		Random	CBOW	Skip-gram	Random	CBOW	Skip-gram
WN18	SME (linear)	84.70	89.54 (↑6%)	89.16 (↑5%)	85.11	89.11 (↑5%)	90.57 (↑6%)
	SME (bilinear)	84.30	91.83 (↑9%)	90.68 (↑8%)	85.36	90.49 (↑6%)	89.89 (↑5%)
	TransE	94.60	96.98 (↑3%)	97.23 (↑3%)	86.74	93.46 (↑8%)	94.49 (↑9%)
	SE	94.71	96.46 (↑2%)	96.42 (↑2%)	87.99	92.05 (↑5%)	91.70 (↑4%)
NELL186	SME (linear)	88.59	89.95 (↑2%)	91.19 (↑3%)	84.42	85.70 (↑2%)	86.67 (↑3%)
	SME (bilinear)	88.74	93.22 (↑5%)	92.86 (↑5%)	83.41	89.70 (↑8%)	89.65 (↑7%)
	TransE	82.54	85.65 (↑4%)	85.33 (↑3%)	76.74	80.06 (↑4%)	80.06 (↑4%)
	SE	89.00	93.37 (↑5%)	93.07 (↑5%)	83.01	87.89 (↑6%)	87.98 (↑6%)

Table 4: Triple classification results on the test sets of WN18 and NELL186.

classification, a triple is predicted to be positive if the energy is below a relation-specific threshold δ_r; otherwise negative. We report two metrics on the test sets: micro-averaged accuracy (per-instance average) and macro-averaged accuracy (per-relation average).

Implementation Details. We use the same parameter settings as in the link prediction task. The relation-specific threshold δ_r is determined by maximizing Micro-ACC on the validation sets.

Results. Table 4 reports the results on the test sets of WN18 and NELL186. The results again demonstrate both the superiority and the generality of our approach.

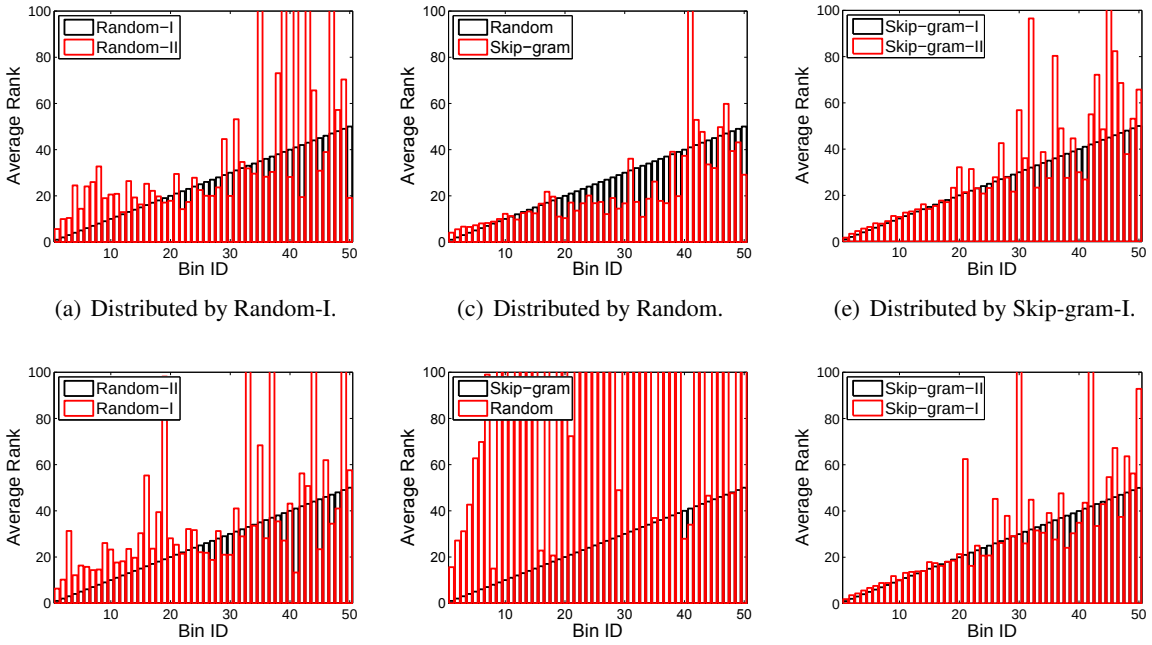
3.3 Discussions

This section is to explore why pre-training helps in KG embedding, specifically in link prediction.

We first test different random initializations in traditional KG embedding models. We run SME (linear) twice on WN18, with two different initialization settings. Both are randomly sampled from the same uniform distribution, but with different seeds, referred to as Random-I and Random-II. Each setting finally gets 10,000 ranks on the test set.[7] To better understand the difference between the two settings, we analyze the ranks individually, rather than reporting aggregated metrics (Mean and Hits@10). Specifically, we distribute the 10,000 instances into different bins according to the ranks given by one setting (e.g. Random-I). Instances assigned to the i-th bin have the same rank of i, that means, they are all ranked in the i-th position by this setting. Then, within each bin, we calculate the average rank of the instances given by the other setting (e.g. Random-II). If the average rank differs drastically from the bin ID, the instances in this bin are ranked significantly differently by the two settings. Figures 2(a) and 2(b) show the results, with the instances distributed according to Random-I and Random-II respectively. In both cases, we retain the bins with ID no larger than 50, covering about 85% of the instances. In most of the bins, the average rank (red bars in the figures) differs drastically from the bin ID (black bars in the figures), indicating that the ranks given by Random-I and Random-II are significantly different at the instance level. The results demonstrate the non-convexity of SME (linear): different initial values lead to different local minimum.

We further compare the settings of initial values 1) randomly sampled from a uniform distribution (Random) and 2) pre-trained by Skip-gram

[7] For each of the 5,000 test triples, both the head and the tail are corrupted and ranked.



(a) Distributed by Random-I. (c) Distributed by Random. (e) Distributed by Skip-gram-I.

(b) Distributed by Random-II. (d) Distributed by Skip-gram. (f) Distributed by Skip-gram-II.

Figure 2: Ranks obtained by different initialization strategies (best viewed in color).

(Skip-gram). The results are given in Figures 2(c) and 2(d). In most of the bins Skip-gram has an average rank lower than the bin ID (Figure 2(c)), while Random has an average rank much higher than the bin ID (Figure 2(d)), implying that Skip-gram performs better than Random-I at the instance level. The results indicate that pre-training might help in finding better initial values which lead to better local minimum.

Finally we test our two-stage KG embedding scheme where the skip-gram model itself is given two different initialization settings, say Skip-gram-I and Skip-gram-II. The results are given in Figures 2(e) and 2(f). In each of the first 20 bins, Skip-gram-I and Skip-gram-II get an average rank almost the same with the bin ID, implying that the two settings perform quite similarly, particularly at the highest ranking levels. The results indicate that a pre-training stage might help in obtaining more stable embeddings.

4 Conclusion

We have proposed a novel two-stage scheme for KG embedding, called context-dependent KG embedding. In the pre-training stage CCPs are encoded by a word embedding model, and in the fine-tuning stage LCPs are encoded by a traditional KG embedding model. Since both types of connectivity patterns are explicitly taken into account, our approach can obtain more accurate embeddings. Moreover, our approach is quite general, applicable to various word embedding and KG embedding models. Experimental results on link prediction and triple classification demonstrate the superiority, generality, and stability of our approach.

As future work, we plan to 1) Investigate the efficacy of longer CCPs (i.e. knowledge paths with lengths longer than 5). 2) Design a joint model that encodes LCPs and CCPs simultaneously. Moreover, our approach actually reveals the possibility of a broad idea, i.e., initializing an embedding model by another embedding model. We would also like to test the feasibility of other such strategies, e.g., initializing SME by TransE, so as to combine the benefits of both models.

Acknowledgments

We would like to thank the anonymous reviewers for their valuable comments and suggestions. This work is supported by the National Natural Science Foundation of China (grant No. 61402465), the Strategic Priority Research Program of the Chinese Academy of Sciences (grant No. XDA06030200), and the National Key Technology R&D Program (grant No. 2012BAH46B03).

References

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, pages 1247–1250.

Antoine Bordes, Jason Weston, Ronan Collobert, and Yoshua Bengio. 2011. Learning structured embeddings of knowledge bases. In *Proceedings of the 25th AAAI Conference on Artificial Intelligence*, pages 301–306.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems*, pages 2787–2795.

Antoine Bordes, Xavier Glorot, Jason Weston, and Yoshua Bengio. 2014. A semantic matching energy function for learning with multi-relational data. *Machine Learning*, 94(2):233–259.

Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R. Hruschka Jr., and Tom M. Mitchell. 2010. Toward an architecture for never-ending language learning. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence*, pages 1306–1313.

Kai-Wei Chang, Wen-tau Yih, Bishan Yang, and Christopher Meek. 2014. Typed tensor decomposition of knowledge bases for relation extraction. In *Proceedings of the 2014 Conference on Empirical Methods on Natural Language Processing*, pages 1568–1579.

Josu Goikoetxea, Aitor Soroa, and Eneko Agirre. 2015. Random walks and neural network language models on knowledge bases. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1434–1439.

Shu Guo, Quan Wang, Bin Wang, Lihong Wang, and Li Guo. 2015. Semantically smooth knowledge graph embedding. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 84–94.

Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N. Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, et al. 2014. Dbpedia: A large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web Journal*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. In *Proceedings of Workshop at International Conference on Learning Representations*.

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

George A Miller. 1995. Wordnet: A lexical database for english. *Communications of the ACM*, 38(11):39–41.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data. In *Proceedings of the 28th International Conference on Machine Learning*, pages 809–816.

Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 701–710.

Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M. Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In *Proceedings of the 2013 Conference on North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 74–84.

Richard Socher, Danqi Chen, Christopher D. Manning, and Andrew Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *Advances in Neural Information Processing Systems*, pages 926–934.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence*, pages 1112–1119.

Quan Wang, Bin Wang, and Li Guo. 2015. Knowledge base completion using embeddings and rules. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence*, pages 1859–1865.

Learning to Identify the Best Contexts for Knowledge-based WSD

Evgenia Wasserman-Pritsker
University of Haifa
Haifa, Israel, 31905
evgeniaw@is.haifa.ac.il

William W. Cohen
Carnegie Mellon University
Pittsburgh, PA 15213
wcohen@cs.cmu.edu

Einat Minkov
University of Haifa
Haifa, Israel, 31905
einatm@is.haifa.ac.il

Abstract

We outline a learning framework that aims at identifying useful contextual cues for knowledge-based word sense disambiguation. The usefulness of individual context words is evaluated based on diverse lexico-statistical and syntactic information, as well as simple word distance. Experiments using two different knowledge-based methods and benchmark datasets show significant improvements due to context modeling, beating the conventional window-based approach.

1 Introduction

Word sense disambiguation (WSD) is a key task of natural language processing. Unsupervised *knowledge-based* approaches to WSD (Navigli, 2009) make use of available lexical resources rather than rely on costly annotated data. Sense inference in this setting involves finding the word sense that agrees most with the specified context according to the information encoded in the knowledge base (KB). The popular Lesk (1986) method, for example, seeks to maximize word overlap between the dictionary glosses associated with the context words, and the glosses of candidate word senses. Similar methods are used in named entity disambiguation and linking to a KB (Hoffart et al., 2011).

Despite the sophistication of inference models developed, little attention has been given so far to context modeling for knowledge-based WSD. Context is represented by bag-of-words, where typically all context words are assigned equal importance (Navigli, 2009; Ling et al., 2014). However, every simple definition of context will include some unrelated or uninformative context words. Consider this usage of the word *church*: "An ancient stone church stands amid the fields , the sound of bells cascading from its tower". Known senses for 'church' according to Wordnet 3.0 (Fellbaum, 1998) correspond to a group of people, service, or a building. The latter sense is intended in this case, as one may conclude from the context words 'stone', 'stands' or 'tower'. We wish to focus on such meaningful cues and avoid the modeling of uninformative words ('ancient', 'sound').

In this work, a learning framework is proposed that is aimed at identifying contextual cues that are predictive of the target word's sense. The usefulness of a candidate context word for the disambiguation of the target word is evaluated based on syntactic and lexico-statistical information, as well as simple word distance. Indirect supervision is provided using noisy example labels induced automatically. Importantly, explicit lexical information is not encoded–the prediction model can thus be applied in settings where no sense-tagged examples are available of the target word type (see also (Szarvas et al., 2013)). Having assessed the usefulness of available context words given the learned model, we consider only the top scoring context words in performing WSD.

We believe this work to be the first to perform learning-based context selection for knowledge-based sense disambiguation. Empirical evaluation using two representative knowledge-based WSD methods and different benchmark datasets indicates on consistent improvements in performance due to context selection using the proposed approach.

2 Learned context selection models (LCS)

We first define the WSD task. Given a word mention w and available context Ctx, it is required to infer the intended sense $s^* \in S(w)$, where $S(w)$ is the set of known senses of w. Ctx may be a sentence, a paragraph, or a window over words that contain w.

Knowledge-based methods seek to maximize some measure of agreement, or *similarity*, between candidate word senses and a given context. We denote this as $Sim()$, where the sense inference procedure is defined as follows:

$$\hat{s}(w) = \arg\max_{s \in S(w)} Sim(s, Ctx) \quad (1)$$

Typically, Ctx is represented as a bag-of-words, and the similarity score $Sim(s, Ctx)$ is additive, i.e., it may be computed as a summation over the similarity scores between sense s and the individual context words $c_j \in Ctx$, using the general formula:

$$Sim(s, Ctx) = \sum_{c_j \in Ctx} weight(c_j) Sim(s, c_j) \quad (2)$$

According to this view, each context word serves as a sense disambiguation 'expert'. Context words are usually assigned uniform weight, i.e., $weight(c_j) = 1$. Alternatively, varying weights may reflect the reliability, or relevancy, of context word c_j in disambiguating target word w; ideally, unuseful context information would be down-weighted or discarded.

2.1 Learning

Our goal is to learn models that assess whether a candidate context word c_j serves as a reliable 'expert' in predicting the sense of target word w. We propose a distantly supervised learning scheme. Given sense-tagged instances of the form $\langle Ctx(w_i) \rangle$, we derive a dataset of context-target word pairs $\langle w_i, c_{ij} \rangle$, $c_{ij} \in Ctx(w_i)$. Defining whether context work c_{ij} is useful, or reliable, with respect to the disambiguation of w_i is not trivial, however. In particular, words that are perceived as relevant according to human judgment may not necessarily yield the correct prediction using the inference algorithm. We consider a context word to be reliable if it yields a correct sense prediction of the target word, as follows:

$$y(w_i, c_{ij}) = \begin{cases} 1, & \text{if } \arg\max_s Sim(s, c_{ij}) = s^*(w_i) \\ 0, & \text{otherwise} \end{cases} \quad (3)$$

As the similarity measures $Sim()$, as well as the reference knowledge base, are imperfect, the labels assigned in this fashion are expected to be noisy.

A context and target word pair is represented as a feature vector, as described below. Importantly, we avoid the representation of explicit lexical information, so that the learned models are applicable to word pairs of arbitrary word types.

Given a new instance at test time, the learned model is used to score the individual context words. One can then assign respective non-uniform weights to the context words (Eq. 2). Here, we take a *context selection* approach–a ranking is induced over the context words based on the predicted scores, and only the top ranked context words are modelled in the disambiguation process; that is, the selected context words are assigned weight 1.0, and the weight of the remaining context words is set to zero. As discussed in Sec.3, this design choice was found to give preferable results in preliminary experiments.

2.2 Feature Types

Various aspects may be modeled as features in this framework, describing properties of the context word c_j, as well as the relationship between the target-context word pair $\langle w, c_j \rangle$. In addition to simple *word distance*, we encode the following syntactic and lexico-statistical information.

Syntactic features. Word distance is further assessed in syntactic terms, denoting the length of the shortest dependency path linking the word pair, as well as the length of the shortest connecting path in a constituency parse tree (Swanson and Gordon, 2006; Huang and Lu, 2011). It may be useful to further encode information about the edge types that comprise the connecting path, as some dependency relations indicate more salient semantic relatedness than others (Padó and Lapata, 2007; Minkov and Cohen, 2013). In this work, if the target and context words are directly connected in the dependency graph, we include a feature indicating the label of the edge. The part-of-speech tag of the context word may provide another contextual cue (Yarowsky, 1993); dedicated features indicate whether c_j is tagged as *noun*, *verb*, *adjective* or *adverb*. We used the Stanford parser (de Marneffe et al., 2006) in our experiments.

Lexico-statistical information. We use the pointwise mutual information (PMI) measure (Turney, 2001) to assess the semantic relatedness between the context–target word pair. In general, we expect context words that are topically related to the target word to be useful for its disambiguation. To compute PMI, we obtained word frequencies from the large ukWaC corpus (Ferraresi et al., 2008), considering word co-occurrences over a window of five words. It has been indicated that highly frequent words are generally less topical, where this aspect is not fully captured by PMI (Han et al., 2013). We therefore model as complimentary information the inverse document frequency (Salton and McGill, 1983) of c_j, also computed using ukWaC. Finally, a context word is often ambiguous by itself, where low polysemy is correlated with topic-specificity (Han et al., 2013). We represent the number of known senses of the context word c_j based on WordNet.

3 Experiments

We consider two WSD methods representative of prevalent knowledge-based approaches, comparing against previously published results. The popular Lesk approach (1986) mentioned before computes $Sim(s, c_j)$ in terms of word overlap between the glosses of the senses of c_j and the gloss of s. There exist multiple variants of the Lesk algorithm (Kilgarriff and Rosenzweig, 2000; Banerjee and Pedersen, 2003; Ponzetto and Navigli, 2010). We experiment with *Gloss vectors* (GV) (Patwardhan and Pedersen, 2006). This method enriches WordNet glosses with glosses of hypernyms and other related senses, as well as with co-occurring words derived from raw text. GV scores were obtained using the WordNet::Similarity package (Pedersen et al., 2004).

Graph-based methods are also commonly used for sense disambiguation (Mihalcea, 2005; Hughes and Ramage, 2007). If the KB is represented as a graph, various metrics can be applied that reflect structural similarity between word senses represented as graph nodes. We consider the Personalized PageRank (PPR) algorithm, which has been shown to yield state-of-the-art WSD performance (Agirre and Soroa, 2009). According to the *linearity theorem* (Jeh and Widom, 2003), PPR scores can be computed for each of the context words separately, and then aggregated (Eq. 2). In this case, $Sim(s, c_j)$ equals the PPR score attributed to the node denoting sense s, having the graph walk initiated at a uniform distribution over the various senses of c_j. PPR scores were generated using the UKB software (Agirre and Soroa, 2009).[1]

	Word types	Target words	Context words	Pairwise acc. (PPR/ GV)
Koeling *et al.*	41	9.6K	121K	0.32/ 0.31
Semeval'07	35	16K	390K	0.35/ 0.33

Table 1: The experimental datasets: statistics

3.1 Datasets

We experiment with two benchmark datasets. The lexical sample due to Koeling et al. (2005) includes annotated instances of 41 selected nouns. About 300 example sentences are available per noun, retrieved evenly from three sources: the domain-specific *sports* and *finance* sections of Reuters corpus, and the general British National Corpus (BNC). The second dataset consists of all noun examples from the SemEval-2007 English lexical sample task (Pradhan et al., 2007), created from another corpus–the WSJ Treebank.

The two datasets were transformed into target-context word pairs. For every word pair $\langle w, c_j \rangle$, the scores $Sim(s, c_j)$, $s \in S(w)$, were generated using GV and PPR and WordNet 3.0 as the reference knowledge base. A context-target word pair was labeled as a positive example if it yielded a correct sense prediction, or as negative otherwise (Eq. 3). Table 1 details statistics of the original and respective word pair datasets, including the ratio of context words labeled as positive examples– as shown, this 'pairwise accuracy' is low, reaching up to 0.35.

3.2 Experimental setup

We experimented with several classification paradigms using the Weka learning suite (Hall et al., 2009). Learning had to be robust to label noise. We report results using Naive Bayes, due to both its good performance and efficiency. Following preliminary experiments, we adopted a context selection approach–the learned model is used to rank the available context words, where the top ranked words, obtained by applying ratio r, are selected as context. We tune r using training examples. The reported performance uses rough values of $r = 0.5$ for the Koeling *et al.* examples, which

[1] http://ixa2.si.ehu.es/ukb/; we used the bin file wn30+gloss, and the PPR_w2w graph walk variant.

	Koeling *el al.*		SemEval'07	
	GV	PPR	GV	PPR
Uniform	.389	.494	.370	.432
LCS:CW	$.410^{+5\%}$	$\mathbf{.511}^{+3\%}$	$.469^{+27\%}$	$.494^{+14\%}$
LCS:CD	$.411^{+6\%}$	$.510^{+3\%}$	$.480^{+30\%}$	$\mathbf{.507}^{+17\%}$

Table 2: Main results: recall performance

include individual sentences, and $r = 0.2$ for SemEval'07, where parahraphs of a few sentences are provided as context.

3.3 Results

Table 2 shows the results of applying context selection for each of the dataset and methods. As in previous works, performance is reported in terms of *recall*, defined as the ratio of correct sense predictions out of total number of target word mentions.[2] To avoid over fitting, we performed *cross word* evaluation, predicting contexts for all instances of each word type with a model trained using the other word types (LCS:CW). Concretely, the Koeling *et al.* dataset was split into 41 bins, according to the target word type. For each word type, we generated a model using the examples of the remaining (in this case, 40) word types. This cross word evaluation procedure was applied to both datasets. We further report the results of *cross dataset* experiments (LCS:CD), in which one dataset is used for training and the other for evaluation. As baseline, we use all of the available context words, weighting them uniformly ("uniform" in the table).

As shown, LCS yields substantial improvements over the "uniform" baseline. The improvement rate for each experiment is displayed in superscript. Recall increased at high rates on the SemEval dataset. This dataset is skewed, and much of these gains are attributed to large increase in recall for two word types, covering 27% of the examples. Improvements on the balanced dataset due to Koeling *et al.* were more modest, yet significant. Interestingly, improvement rates are higher using GV than PPR; we conjecture that PPR predictions are biased towards highly-connected graph nodes, being less sensitive to the local context defined. Remarkably, the results using cross-dataset training are comparable to or exceed the within-dataset CW results, showing generality and robustness of the proposed approach.

[2]Since predictions are generated for all examples, recall equals in this case to *precision*, and *accuracy*.

	GV	PPR
Uniform	.389	.494
Lexico-stat. features:		
PMI only	$.397^{+2.1\%}$	$.502^{+1.6\%}$
+IDF	$.403^{+3.6\%}$	$.503^{+1.8\%}$
+No. of senses	$.406^{+4.4\%}$	$.509^{+3.0\%}$
+Syntactic features:	$.411^{+5.7\%}$	$.510^{+3.2\%}$
+Word distance	$.410^{+5.4\%}$	$.511^{+3.4\%}$

Table 3: Feature ablation results using LCS:CW and the Koeling *et al.* dataset

		BNC	Sports	Finance
Uniform	PPR	.491	.437	.554
LCS:CW		$\mathbf{.502}^{+2\%}$	$.464^{+6\%}$	$.565^{+2\%}$
LCS:CD		$.501^{+2\%}$	$.459^{+5\%}$	$\mathbf{.570}^{+3\%}$
Uniform	GV	.382	.361	.423
LCS:CW		$.401^{+5\%}$	$.377^{+4\%}$	$.450^{+6\%}$
LCS:CD		$.400^{+5\%}$	$.386^{+7\%}$	$.448^{+6\%}$
AL&S'09		.438	.356	.469
H&L'11		.397	-	-
P&N'10		-	.420	.478
R&M'12		-	**.465**	.493

Table 4: Detailed results on the Koeling *et al.* dataset

Table 3 further shows the results of an ablation study, assessing the contribution of the various feature types by adding them incrementally. We found the contribution of the lexico-statistical features to be the largest. In particular, modeling PMI yielded the best performance when used as a standalone feature. This means that context words that are topically related to the target word are especially useful for knowledge-based WSD. Modeling IDF information led to further gains in performance. As discussed before, the two measures are complimentary, as common words are generally less topical. Representing the number of senses of the context words yielded further improvements. Overall, this combination of word features accounted for the majority of the total gains achieved. The syntactic features had a lesser impact, yet improved results further, mainly using the GV method. Finally, simple word distance was found to have little impact; similar behavior was observed elsewhere (Hoffart et al., 2011).

In another set of experiments, we evaluated and found LCS to be robust with respect to the ratio r– while performance using LCS varied, it improved over the baseline across the range $0 < r < 1$. In contrast, selecting equal-sized sets of context words using the window approach was found to

hurt performance.

Finally, we compare our results against previous works. Our approach outperforms the results obtained by unsupervised systems on the noun portion of the SemEval'07 dataset (Patwardhan et al., 2007; Mohammad et al., 2007), achieving recall of .507 vs. .497 (a higher result obtained by Mohammad *et al*). Table 4 presents LCS results separately for each of the source domains of the Koeling dataset for comparison purposes. Previous results using PPR and uniform context weighting reported by Agirre *et al.* (2009) (AL&S'09) are substantially lower than our baseline; we mainly attribute this to the different version of WordNet used.[3] Huang and Lu (2011) proposed a manually-tuned syntax-based context selection and weighting formula. They applied it in combination with the GV method, reporting improvement on BNC sentences only. Our baseline result using GV was lower (.382 vs. .390), however LCS yielded better final performance (.401 vs. .397). Compared with their work, we use learning and model richer types of evidence; with PPR and LCS, we report best results on the BNC sentences. Table 4 details also recent results obtained for the BNC and Sports portions of the dataset. Ponzetto and Navigli (2010) (P&N'10) enriched the WordNet graph with additional relations projected onto the graph from Wikipedia; the table reports their best results using a graph centrality measure (Navigli and Lapata, 2010). Raviv and Markovitch (2012) (R&M'12) reported state-of-the-art performance in the specialized domains using Wikipedia as the reference knowledge base. Each individual context word is represented in their work as a weighted vector of Wikipedia concepts, where sense inference is performed by maximizing cosine similarity between the centroid of the context vectors and a vector representation of each word sense. Our results using PPR and LCS exceed or roughly match their results without using the Wikipedia resource.

4 Conclusion

We presented a learning framework that identifies useful contextual cues for knowledge-based sense disambiguation. The generated models are non-lexicalized, and are therefore applicable to new word types. Existing approaches pay little attention to context selection, or perform simplistic context modeling, whereas the proposed approach effectively consolidates diverse types of evidence. In the future, we are interested in representing additional word relatedness measures in this framework, such as embedding-based word similarity (Wang et al., 2015). We are further interested in creating specialized models that fit different word classes, e.g., of particular part-of-speech. In general, the proposed approach may prove beneficial for additional tasks that model word meaning in context, such as lexical substitution and sense induction.

[3] They used WordNet 1.7, while we use version 3.0. Large performance gaps due to different versions of WordNet were reported elsewhere (Agirre and Soroa, 2009).

Acknowledgments

We wish to thank Ido Dagan, Shuly Wintner and the anonymous reviewers for their useful comments. This work was supported by BSF under grant 2010090.

References

Eneko Agirre and Aitor Soroa. 2009. Personalizing pagerank for word sense disambiguation. In *Proceedings of EACL*.

Eneko Agirre, Oier Lopez de Lacalle, and Aitor Soroa. 2009. Knowledge-based WSD on specific domains: performing better than generic supervised WSD. In *Proceedings of IJCAI*.

Satanjeev Banerjee and Ted Pedersen. 2003. Extended gloss overlaps as a measure of semantic relatedness. In *Proceedings of IJCAI*.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC*.

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

Adriano Ferraresi, Eros Zanchetta, Marco Baroni, and Silvia Bernardini. 2008. Introducing and evaluating ukWaC, a very large web-derived corpus of English. In *Proceedings of the WAC4 Workshop at LREC*.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. 2009. The WEKA data mining software: an update. *SIGKDD Exploratory Newsletter*, 11(1):10–18.

Lushan Han, Tim Finin, Paul McNamee, Anupam Joshi, and Yelena Yesha. 2013. Improving word similarity by augmenting PMI with estimates of word polysemy. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 25(6).

Johannes Hoffart, Mohamed A. Yosef, Ilaria Bordino, Hagen Furstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. 2011. Robust disambiguation of named entities in text. In *Proceedings of EMNLP*.

Heyan Huang and Wenpeng Lu. 2011. Knowledge-based word sense disambiguation with feature words based on dependency relation and syntax tree. *International Journal of Advancements in Computing Technology*, 3(8).

Thad Hughes and Daniel Ramage. 2007. Lexical semantic relatedness with random graph walks. In *Proceedings of EMNLP*.

Glen Jeh and Jennifer Widom. 2003. Scaling personalized web search. In *Proceedings of the 12th international conference on World Wide Web (WWW)*.

Adam Kilgarriff and Joseph Rosenzweig. 2000. English senseval:report and results. In *2nd International Conference on Language Resourcesand Evaluation (LREC)*.

Rob Koeling, Diana McCarthy, and John Carrol. 2005. Domain-specific sense distributions and predominant sense acquisition. In *Proceedings of HLT-EMNLP*.

Michael Lesk. 1986. Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone. In *Proceedings of the international conference on Systems documentation*.

Xiao Ling, Sameer Singh, and Daniel S. Weld. 2014. Context representation for named entity linking. In *Pacific Northwest Regional NLP Workshop*.

Rada Mihalcea. 2005. Unsupervised large-vocabulary word sense disambiguation with graph-based algorithms for sequence data labeling. In *Proceedings of HLT-EMNLP*.

Einat Minkov and William W. Cohen. 2013. Adaptive graph walk-based similarity measures for parsed text. *Natural Language Engineering*.

Saif Mohammad, Graeme Hirst, and Philip Resnik. 2007. Tor, TorMd: Distributional profiles of concepts for unsupervised word sense disambiguation. In *Proceedings of SemEval-2007*.

Roberto Navigli and Mirella Lapata. 2010. An experimental study on graph connectivity for unsupervised word sense disambiguation. *IEEE Transactions on Pattern Anaylsis and Machine Intelligence*, 32(4).

Roberto Navigli. 2009. Word sense disambiguation: A survey. *ACM Computing Surveys*, 41(2).

Sebastian Padó and Mirella Lapata. 2007. Dependency-based construction of semantic space models. *Computational Linguistics*, 33(2).

Siddharth Patwardhan and Ted Pedersen. 2006. Using wordnet-based context vectors to estimate the semantic relatedness of concepts. In *Proceedings of the EACL 2006 Workshop on Making Sense of Sense*.

Siddharth Patwardhan, Satanjeev Banerjee, and Ted Pedersen. 2007. UMND1: unsupervised word sense disambiguation using contextual semantic relatedness. In *Proceedings of SemEval-2007*.

Ted Pedersen, Siddharth Patwardhan, and Jason Michelizzi. 2004. WordNet::Similarity - measuring the relatedness of concepts. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*.

Simone Paolo Ponzetto and Roberto Navigli. 2010. Knowledge-rich word sense disambiguation rivaling supervised systems. In *Proceedings of ACL*.

Sameer S. Pradhan, Edward Loper, Dmitriy Dligach, and Martha Palmer. 2007. Semeval-2007 task 17: English lexical sample, SRL and all words. In *Proceedings of SemEval-2007*.

Ariel Raviv and Shaul Markovitch. 2012. Concept-based approach to word-sense disambiguation. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*.

Gerard Salton and Michael J. McGill. 1983. *Introduction to Modern Information Retrieval*. McGraw-Hill.

Reid Swanson and Andrew S. Gordon. 2006. A comparison of alternative parse tree paths for labeling semantic roles. In *the joint conference of the International Committee on Computational Linguistics and the Association for Computational Linguistics (COLING-ACL)*.

György Szarvas, Chris Biemann, and Iryna Gurevych. 2013. Supervised all-wwords lexical substitution using delexicalized features. In *Proceedings of HLT-NAACL*.

Peter Turney. 2001. Mining the web for synonyms: PMI-IR versus LSA on TOEFL. In *Proceedings of ECML*.

Jing Wang, Mohit Bansal, Kevin Gimpel, Brian D. Ziebart, and Clement T. Yu. 2015. A sense-topic model for word sense induction with unsupervised data enrichment. *Transactions of Computational Linguistics*, 3.

David Yarowsky. 1993. One sense per collocation. In *ARPA Human Language Technology Workshop*.

Measuring Prerequisite Relations Among Concepts

Chen Liang [†] **Zhaohui Wu**[‡] **Wenyi Huang**[†] **C. Lee Giles**[†]
[†]Information Sciences and Technology
[‡]Computer Science and Engineering
The Pennsylvania State University
University Park, PA
cul226@ist.psu.edu zzw109@psu.edu {wzh112,giles}@ist.psu.edu

Abstract

A *prerequisite* relation describes a basic relation among concepts in cognition, education and other areas. However, as a semantic relation, it has not been well studied in computational linguistics. We investigate the problem of measuring prerequisite relations among concepts and propose a simple link-based metric, namely *reference distance* (**RefD**), that effectively models the relation by measuring how differently two concepts refer to each other. Evaluations on two datasets that include seven domains show that our single metric based method outperforms existing supervised learning based methods.

1 Introduction

What should one know/learn before starting to learn a new area such as "deep learning"? A key for answering this question is to understand what a *prerequisite* is. A prerequisite is usually a concept or requirement before one can proceed to a following one. And the prerequisite relation exists as a natural dependency among concepts in cognitive processes when people learn, organize, apply, and generate knowledge (Laurence and Margolis, 1999). While there has been serious effort in understanding prerequisite relations in learning and education (Bergan and Jeska, 1980; Ohland et al., 2004; Vuong et al., 2011), it has not been well studied as a semantic relation in computational linguistics, where researchers focus more on lexical relations among lexical items (Miller, 1995) and fine-grained entity relations in knowledge bases (Mintz et al., 2009).

Instead of treating it as a relation extraction or link prediction problem using traditional machine learning approaches (Talukdar and Cohen, 2012; Yang et al., 2015), we seek to better understand prerequisite relations from a perspective of cognitive semantics (Croft and Cruse, 2004). Partially motivated by the theory of *frame semantics* (Fillmore, 2006), or, to understand a concept, one needs to understand all the related concepts in its "frame", we propose a metric that measures prerequisite relations based on a simple observation of human learning. When learning concept A, if one needs to refer to concept B for a lot of A's related concepts but not vice versa, B would more likely be a prerequisite of A than A of B. Specifically, we model a concept in a vector space using its related concepts and measure the prerequisite relation between two concepts by computing how differently the two's related concepts refer to each other, or *reference distance* (**RefD**).

Our simple metric **RefD** successfully reflects some properties of the prerequisite relation such as asymmetry and irreflexivity; and can be properly implemented for various applications using different concept models. We present an implementation of the metric using Wikipedia by leveraging the links as reference relations among concepts; and present a scalable prerequisite dataset construction method by crawling public available university course prerequisite websites and mapping them to Wikipedia concepts. Experimental results on two datasets that include seven domains demonstrate its effectiveness and robustness on measuring prerequisites. Surprisingly, our single metric based approach significantly outperforms baselines which use more sophisticated supervised learning. All the datasets are publicly available upon request.

Our main contributions include:

- A novel metric to measure the prerequisite relation among concepts that outperforms existing supervised learning baselines.
- A new dataset containing 1336 concept pairs in Computer Science and Math.

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1668–1674,
Lisbon, Portugal, 17-21 September 2015.

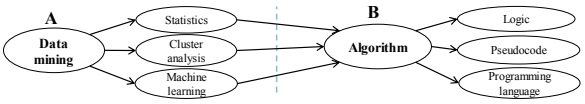


Figure 1: An example of the reference structure for two concepts ("Data mining" and "Algorithm") with a prerequisite relation.

2 Measuring Prerequisite Relations

Our goal is to design a function $f : \mathcal{C}^2 \rightarrow \mathcal{R}$ that maps a concept pair (A, B) to a real value that measures the extent to which A requires B as a prerequisite, where $\mathcal{C}$ is the concept space. How should a concept be represented in $\mathcal{C}$? According to the theory of frame semantics, one cannot understand a concept without access to all essential knowledge related to it. Such knowledge can be actually viewed as a set of related concepts. Thus, a concept could be represented by its related concepts in $\mathcal{C}$. For example, the concept "deep learning" may be represented by concepts such as "machine learning", "artificial neural network", etc.

Compared to prerequisites, a more common and observable relation among concepts is a *reference*, which widely exists in various forms such as hyperlinks, citations, notes, etc. Although a single evidence of reference does not indicate a prerequisite relation, a large number of such evidences might make a difference. For example, if most related concepts of A refer to B but few related concepts of B refer to A, then B is more likely to be a prerequisite of A, as shown in Figure 1. In order to measure prerequisite relations, we propose a *reference distance* (**RefD**), which is defined as

$$RefD(A,B) = \frac{\sum_{i=1}^{k} r(c_i,B)\cdot w(c_i,A)}{\sum_{i=1}^{k} w(c_i,A)} - \frac{\sum_{i=1}^{k} r(c_i,A)\cdot w(c_i,B)}{\sum_{i=1}^{k} w(c_i,B)} \quad (1)$$

where $\mathcal{C} = \{c_1, ..., c_k\}$ is the concept space; $w(c_i, A)$ weights the importance of c_i to A; and $r(c_i, A)$ is an indicator showing whether c_i refers to A, which could be links in Wikipedia, mentions in books, citations in papers, etc.

$RefD$ enables several useful properties for the prerequisite relation: 1) normalized: $RefD(A, B) \in [-1, 1]$; 2) asymmetric: $RefD(A, B)+RefD(B, A)=0$, which means if A is a prerequisite of B then B is not a prerequisite of A; and 3) irreflexive: $RefD(A, A)=0$, which means A is not a prerequisite of itself. To capture all three possible prerequisite relations between a concept pair, *RefD* is expected to satisfy the following constraints:

$$RefD(A,B) \in \begin{cases} (\theta, 1], & \text{if B is a prerequisite of A} \\ [-\theta, \theta], & \text{if no prerequisite relation} \\ [-1, -\theta), & \text{if A is a prerequisite of B} \end{cases}$$

where θ is a positive threshold.

Equation 1 provides a general framework to calculate *RefD*. In practice, we need to specify the concept space $\mathcal{C}$, the weight w, and the reference indicator function r.

3 Wikipedia-based RefD Implementation

We now implement *RefD* using Wikipedia. As a widely used open-access encyclopedia, Wikipedia provides relatively up-to-date and high quality knowledge and has been successfully utilized as explicit concepts (Gabrilovich and Markovitch, 2007). Moreover, the rich hyperlinks created by Wiki editors provide a natural way to calculate the reference indicator function r.

Specifically, the concept space $\mathcal{C}$ consists of all Wikipedia articles. $r(c, A)$ represents whether there is a link from Wiki article c to A. For $w(c, A)$, we experiment with two methods:

- *EQUAL*: A is represented by the concepts linked from it ($L(A)$) with equal weights.

$$w(c, A) = \begin{cases} 1 & \text{if } c \in L(A) \\ 0 & \text{if } c \notin L(A) \end{cases}$$

- *TFIDF*: A is represented by the concepts linked from it with TFIDF weights.

$$w(c, A) = \begin{cases} tf(c, A) * \log \frac{N}{df(c)} & \text{if } c \in L(A) \\ 0 & \text{if } c \notin L(A) \end{cases}$$

where $tf(c, A)$ is the number of times c being linked from A; N is the total number of Wikipedia articles; and $df(c)$ is the number of Wikipedia articles where c appears.

4 Experiments

In order to evaluate the proposed metric, we apply it to predicting prerequisite relations in Wikipedia, i.e., whether one article in Wikipedia is a prerequisite of another article. Given a pair of concepts (A,B), we predict whether B is a prerequisite of A or not. Both pairs where A is a prerequisite of

Dataset	Domain	# Pairs	# Prerequisites
CrowdComp	Meiosis	400	67
	Public-key Cryp.	200	27
	Parallel Postulate	200	25
	Newton's Laws	400	44
	Global Warming	400	43
Course	CS	678	108
	MATH	658	75

Table 1: Statistics of CrowdComp and Course Datasets

Domain	MaxEnt†	MaxEnt	EQUAL	TFIDF
Meiosis	51	60.2	53	55.7
Public-key Cryp.	67.1	60.3	55.1	57.7
Parallel Postulate	64.7	73.6	70.5	67.9
Newton's Laws	53.9	57.7	63.7	64.6
Global Warming	56.8	50.0	57.4	60.1
Average	58.7	60.4	60.0*	**61.2***

Table 2: Comparison of out-of-domain training accuracies of a MaxEnt classifier and *RefD* using EQUAL and TFIDF weighting. MaxEnt† is the number reported by Talukdar et al. (2012). MaxEnt shows the performance of our implementation. * indicates the difference between *RefD* and MaxEnt is statistically significant ($p < 0.01$).

B and pairs where no prerequisite relation exists will be viewed as negative examples.

RefD is tested on two datasets: CrowdComp dataset (Talukdar and Cohen, 2012) and a Course prerequisite dataset collected by us. We compare *RefD* with a Maximum Entropy (MaxEnt) classifier which exploits graph-based features such as PageRank scores and content-based features such as the category information, whether a title of concept is mentioned in the first sentence of the other concept, the number of times a concept is linked from the other, etc. (Talukdar and Cohen, 2012). All experiments use a Wikipedia dump of Dec 8, 2014.

4.1 Results on the CrowdComp Dataset

The CrowdComp dataset was collected using Amazon Mechanical Turk by Talukdar et al. (2012). It contains binary-labeled concept pairs from five different domains, including meiosis, public-key cryptography, the parallel postulate, Newton's laws of motion, and global warming. The label of the prerequisite relation for each pair is assigned using majority vote. Details of the dataset are shown in Table 1.

Following Talukdar et al. (2012), we evaluate

	CS				MATH			
	A	P	R	F	A	P	R	F
MaxEnt	72.8	87.6	53.2	66.1	69.0	78.1	53	63.1
EQUAL	76.4*	80.4	69.9	74.7*	**73.9***	78.4	67.3	**71.9***
TFIDF	**77.1***	82.3	69.1	**75.1***	70.3*	76.3	60.1	66.7*

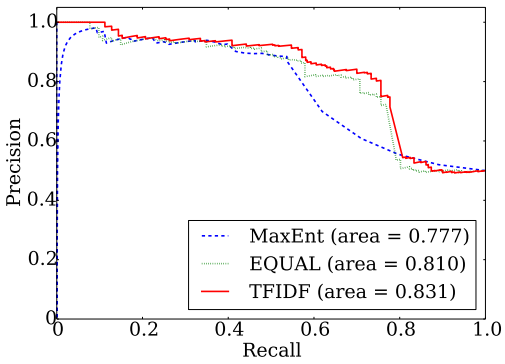
Table 3: Comparison of in-domain training accuracies, precision, recall, and F1 measure of MaxEnt and *RefD* using EQUAL and TFIDF weighting. * indicates the improvement over MaxEnt is statistically significant ($p < 0.01$).

different methods in a "leave one domain out" manner, where data from one domain is used for testing and data from other four for training. Classes in the training and testing set are balanced by oversampling the minority class. Table 2 lists the accuracies of different methods. In terms of average performance, *RefD* achieves comparable average accuracy as MaxEnt. When TFIDF is used to calculate w, *RefD* performs better than MaxEnt. Also we notice that our implementation of MaxEnt classifier achieves higher accuracy than reported in the original paper, which may be due to the difference between Wiki dumps used. In addition, we can see that there are large differences in performance across different domains, which is mainly due to two reasons. First, the coverage of Wikipedia for different domains may vary a lot. Some domains are more popular and thus edited more frequently, leading to a better quality of articles and a more complete link structure. Second, since the ground-truth labels are collected by crowdsourcing and there is no guarantee for workers' knowledge about a certain domain, the quality of labels for different domains varies.

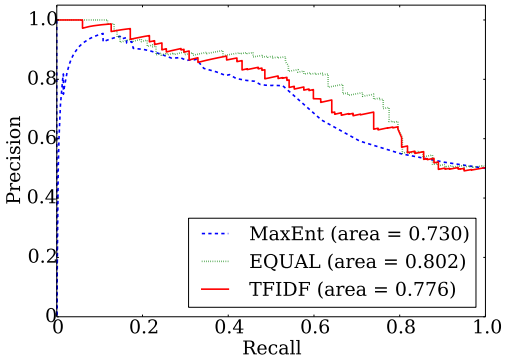
4.2 Results on the Course Dataset

We also built a Course dataset with the help of information available on a university's course website containing prerequisite relations between courses. For example, "CS 331 Data Structures and Algorithms" is a prerequisite for "CS 422 Data mining". We get the prerequisite pairs by crawling the website and linking the course to Wikipedia using simple rules such as title matching and content similarity. In order to get negative samples, we randomly sample 600 pairs using concepts appearing in the prerequisite pairs. All pairs are then checked by two domain experts by removing pairs with incorrect labels. Table 1 lists the information of the dataset.

Evaluation uses in-domain 5-fold cross-



(a) CS

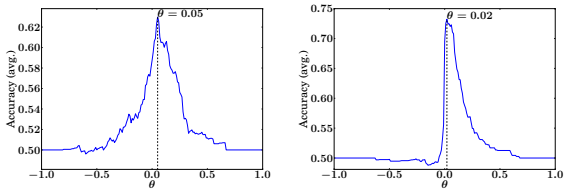


(b) MATH

Figure 2: Comparison of Precision-Recall curves of MaxEnt and *RefD* (using EQUAL and TFIDF weighting) on the Course dataset.

validation and classes are balanced by oversampling the minority class. Table 3 lists the performance comparison of different methods on accuracy, precision, recall and F1 score. We can see that *RefD* outperforms MaxEnt in terms of accuracy, recall, and F1 score on both CS and MATH domain. Because MaxEnt relies on many features but there are only limited distinct positive samples in the dataset, it is more likely to overfit the training data, which leads to high precision but low recall on test set. In order to better compare precision and recall, we plot the Precision-Recall curves of different methods, as shown in Figure 2. *RefD* shows a clear improvement in the area under the Precision-Recall curve.

Comparing two weighting methods, we find that TFIDF performs slightly better than EQUAL on CS while EQUAL has higher scores than TFIDF on MATH. Since how to compute w in *RefD* is a crucial problem, our ongoing work is to explore more sophisticated semantic representations to measure prerequisite relations. A natural extension to the two simple methods here is to represent



(a) CrowdComp (b) Course

Figure 3: Average accuracy on two datasets with a given threshold of *RefD* using TFIDF weighting.

a concept using WordNet (Miller, 1995), Explicit Semantic Analysis (Gabrilovich and Markovitch, 2007), or Word2vec embeddings (Mikolov et al., 2013). Incorporating these representations may improve the performance of *RefD*.

4.3 Parameter Analysis and Case Study

Since using *RefD* to predict prerequisites requires setting a threshold θ, we also investigate the relation between the threshold and the performance of prediction, as shown in Figure 3. We can see that a threshold of 0.05 for *RefD* using TFIDF achieves the highest average accuracy on the CrowdComp dataset while a threshold of 0.02 works the best for Course dataset. Empirically we find that a threshold between 0.02 and 0.1 yields a good performance for prerequisite prediction task.

We further explore the performance of *RefD* through a case study for the concept "deep learning" (denoted as c'). Specifically, for any concept c linked from c' we calculate *RefD*(c', c). Table 4 lists the *RefD* scores for different concepts using *EQUAL* weighting. The concepts on the left have negative *RefD* scores with high absolute values, which means that "deep learning" is a prerequisite of them. Meanwhile concepts on the right have high positive *RefD* scores, which means that "deep learning" requires knowing them first. For example, people may first need to have some knowledge of "machine learning", "artificial intelligence" and "algorithm" in order to learn "deep learning". Also we notice that concepts in the middle have *RefD* scores which are very close to 0, showing that there is no prerequisite relations between these concepts and "deep learning". However, since our *RefD* implementation is based on Wikipedia, it might not give an accurate measure for concepts if they have no Wikipedia articles or their articles are too short to provide an encyclopedic coverage, such as "discriminative model" and "feature engineering".

Concept	*RefD*	Concept	*RefD*	Concept	*RefD*
Deep belief network	-0.38	List of Nobel laureates	0.009	Machine learning	0.32
Neocognitron	-0.28	Neural development	0.009	Artificial neural network	0.31
Word embedding	-0.24	Watson (computer)	0.003	Artificial intelligence	0.15
Vanishing gradient problem	-0.22	Self-organization	8e-5	Algorithm	0.14
Feature learning	-0.17	Language model	-0.004	Statistical classification	0.13

Table 4: *RefD* scores between "deep learning" and the concepts linked from it. All scores are calculated by *RefD*('deep learning', concept).

Please note that our Wikipedia-based implementation is computationally efficient especially after precomputing weights and references and can be easily incorporated as a feature into existing supervised learning based methods.

5 Related Work

In the area of education, researchers have tried to find prerequisites based on the assessment data of students' performance (Scheines et al., 2014; Vuong et al., 2011). However, prerequisite relations have not been well studied in computer science, with only a few exceptions. Liu et al. (2011) studied learning-dependency between knowledge units using classification where a knowledge unit is a special text fragment containing concepts. We focus on more general prerequisite relations among concepts. Talukdar and Cohen (2012) applied a Maximum Entropy classifier to predict prerequisite structures in Wikipedia using various features such as a random walk with restart score and PageRank score. Instead of doing feature engineering, we propose to measure prerequisite relations using a single metric. Yang et al. (2015) proposed Concept Graph Learning to induce relations among concepts from prerequisite relations among courses, where the learned concept prerequisite relations are implicit and thus can not be evaluated. Our method is more interpretable for measuring prerequisite relations.

Our work is closely related to the study of semantic relations. One direction is automatic lexical relation extraction. Different methods have been proposed to discover hypernym-hyponym relations based on lexical patterns (Hearst, 1992; McNamee et al., 2008; Ritter et al., 2009), distributional similarity (Kotlerman et al., 2010), semantic word embeddings (Fu et al., 2014), etc. Another line is entity relation extraction, which can be performed by distant supervision (Mintz et al., 2009; Riedel et al., 2010), Open IE (Fader et al., 2011), and neural networks (Bordes et al., 2011; Lin et al., 2015).

In addition, semantic relatedness measures have been widely studied, where the key is to model the semantic representation based on a latent space, such as LSA (Deerwester et al., 1990), PLSA (Hofmann, 1999), LDA (Blei et al., 2003) and distributed word embeddings (Huang et al., 2012; Mikolov et al., 2013), or an explicit concept space, such as ESA (Gabrilovich and Markovitch, 2007), SSA (Hassan and Mihalcea, 2011), and SaSA (Wu and Giles, 2015). Our work can also be served as a basis for building concept hierarchy (Wang et al., 2015) and teaching/learning assistant tools (Liang et al., 2015).

6 Conclusions and Future Work

We studied the problem of measuring prerequisite relations among concepts and proposed *RefD*, a general, light-weight, and effective metric, to capture the relation. We presented Wikipedia-based implementations of *RefD* with two different weighting strategies. Experiments on two datasets including seven domains showed that our proposed metric outperformed existing baselines using supervised learning.

Promising future directions would be applying the framework of *RefD* to other contexts such as measuring the prerequisite relations or reading orders between papers and textbooks. In addition, *RefD* can be incorporated into existing supervised models for a more accurate measure. Also it would be meaningful to explore ranking different prerequisites of a concept. Besides the rich link structure we could take advantage of more content information from Wikipedia and other resources such as textbooks and scientific papers.

Acknowledgments

We gratefully acknowledge partial support from the National Science Foundation, technical support from Jian Wu, and helpful comments from the anonymous reviewers.

References

John R Bergan and Patrick Jeska. 1980. An examination of prerequisite relations, positive transfer among learning tasks, and variations in instruction for a seriation hierarchy. *Contemporary Educational Psychology*, 5(3):203–215.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022.

Antoine Bordes, Jason Weston, Ronan Collobert, and Yoshua Bengio. 2011. Learning structured embeddings of knowledge bases. In *Conference on Artificial Intelligence*, number EPFL-CONF-192344.

William Croft and D Alan Cruse. 2004. *Cognitive linguistics*. Cambridge University Press.

Scott C. Deerwester, Susan T Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. 1990. Indexing by latent semantic analysis. *JAsIs*, 41(6):391–407.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1535–1545. Association for Computational Linguistics.

Charles J Fillmore. 2006. Frame semantics. *Cognitive linguistics: Basic readings*, 34:373–400.

Ruiji Fu, Jiang Guo, Bing Qin, Wanxiang Che, Haifeng Wang, and Ting Liu. 2014. Learning semantic hierarchies via word embeddings. In *Proceedings of the 52th Annual Meeting of the Association for Computational Linguistics: Long Papers*, volume 1.

Evgeniy Gabrilovich and Shaul Markovitch. 2007. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *IJCAI*, volume 7, pages 1606–1611.

Samer Hassan and Rada Mihalcea. 2011. Semantic relatedness using salient semantic analysis. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2011, San Francisco, California, USA, August 7-11, 2011*.

Marti A Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th conference on Computational linguistics-Volume 2*, pages 539–545. Association for Computational Linguistics.

Thomas Hofmann. 1999. Probabilistic latent semantic indexing. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 50–57. ACM.

Eric H Huang, Richard Socher, Christopher D Manning, and Andrew Y Ng. 2012. Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 873–882. Association for Computational Linguistics.

Lili Kotlerman, Ido Dagan, Idan Szpektor, and Maayan Zhitomirsky-Geffet. 2010. Directional distributional similarity for lexical inference. *Natural Language Engineering*, 16(04):359–389.

Stephen Laurence and Eric Margolis. 1999. Concepts and cognitive science. *Concepts: core readings*, pages 3–81.

Chen Liang, Shuting Wang, Zhaohui Wu, Kyle Williams, Bart Pursel, Benjamin Brautigam, Sherwyn Saul, Hannah Williams, Kyle Bowen, and C. Lee Giles. 2015. Bbookx: An automatic book creation framework. In *Proceedings of the 2015 ACM Symposium on Document Engineering*.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of AAAI*.

Jun Liu, Lu Jiang, Zhaohui Wu, Qinghua Zheng, and Yanan Qian. 2011. Mining learning-dependency between knowledge units from text. *The VLDB Journal*, 20(3):335–345.

Paul McNamee, Rion Snow, Patrick Schone, and James Mayfield. 2008. Learning named entity hyponyms for question answering. In *IJCNLP*, pages 799–804.

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

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 1003–1011. Association for Computational Linguistics.

Matthew W Ohland, Amy G Yuhasz, and Benjamin L Sill. 2004. Identifying and removing a calculus prerequisite as a bottleneck in clemson's general engineering curriculum. *Journal of Engineering Education*, 93(3):253–257.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *Machine Learning and*

Knowledge Discovery in Databases, pages 148–163. Springer.

Alan Ritter, Stephen Soderland, and Oren Etzioni. 2009. What is this, anyway: Automatic hypernym discovery. In *AAAI Spring Symposium: Learning by Reading and Learning to Read*, pages 88–93.

Richard Scheines, Elizabeth Silver, and Ilya Goldin. 2014. Discovering prerequisite relationships among knowledge components. In *Proceedings of Educational Data Mining*, pages 355–356.

Partha Pratim Talukdar and William W Cohen. 2012. Crowdsourced comprehension: predicting prerequisite structure in wikipedia. In *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP*, pages 307–315. Association for Computational Linguistics.

Annalies Vuong, Tristan Nixon, and Brendon Towle. 2011. A method for finding prerequisites within a curriculum. In *Proceedings of Educational Data Mining*, pages 211–216.

Shuting Wang, Chen Liang, Zhaohui Wu, Kyle Williams, Bart Pursel, Benjamin Brautigam, Sherwyn Saul, Hannah Williams, Kyle Bowen, and C. Lee Giles. 2015. Concept hierarchy extraction from textbooks. In *Proceedings of the 2015 ACM Symposium on Document Engineering*.

Zhaohui Wu and C. Lee Giles. 2015. Sense-aware semantic analysis: A multi-prototypeword representation model usingwikipedia. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence*, pages 2188–2194.

Yiming Yang, Hanxiao Liu, Jaime G. Carbonell, and Wanli Ma. 2015. Concept graph learning from educational data. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining, WSDM 2015, Shanghai, China, February 2-6, 2015*, pages 159–168.

Adapting Phrase-based Machine Translation to Normalise Medical Terms in Social Media Messages

Nut Limsopatham and Nigel Collier
Department of Theoretical and Applied Linguistics
University of Cambridge
Cambridge, UK
{nl347,nhc30}@cam.ac.uk

Abstract

Previous studies have shown that health reports in social media, such as DailyStrength and Twitter, have potential for monitoring health conditions (e.g. adverse drug reactions, infectious diseases) in particular communities. However, in order for a machine to understand and make inferences on these health conditions, the ability to recognise when laymen's terms refer to a particular medical concept (i.e. text normalisation) is required. To achieve this, we propose to adapt an existing phrase-based machine translation (MT) technique and a vector representation of words to map between a social media phrase and a medical concept. We evaluate our proposed approach using a collection of phrases from tweets related to adverse drug reactions. Our experimental results show that the combination of a phrase-based MT technique and the similarity between word vector representations outperforms the baselines that apply only either of them by up to 55%.

1 Introduction

Social media, such as DailyStrength[1] and Twitter[2], is a fast growing and potentially rich source of *voice of the patient* data about experience in terms of benefits and side-effects of drugs and treatments (O'Connor et al., 2014). However, natural language understanding from social media messages is a difficult task because of the lexical and grammatical variability of the language (Baldwin et al., 2013; O'Connor et al., 2014). Indeed, language understanding by machines requires the ability to recognise when a phrase refers to a particular concept. Given a variable length phrase, an effective system should return a concept with the most similar meaning. Table 1 shows examples of mappings between Twitter phrases and medical concepts. For example, a Twitter phrase 'No way I'm gettin any sleep 2nite' might be mapped to the medical concept 'Insomnia' (SNOMED:193462001), when using the SNOMED-CT ontology (Spackman et al., 1997). The success of the mapping between social media phrases and formal medical concepts would enable an automatic integration between patient experiences and biomedical databases (Limsopatham and Collier, 2015). We refer to this mapping from social media phrases to medical concepts as *medical term normalisation*, which aims to determine the unique identifier of a medical concept that is mentioned in different forms in a free-text (Morgan et al., 2008).

Existing works, e.g. (Elkin et al., 2012; Gobbel et al., 2014; Wang et al., 2009), mostly focused on identifying medical concepts in medical documents. For example, Gobbel et al. (2014) proposed a naïve Bayesian-based technique to map phrases from clinical notes to medical concepts in the SNOMED-CT ontology. Wang et al. (2009) identified medical concepts regarding adverse drug events in electronic medical records. On the other hand, O'Connor et al. (2014) investigated the normalisation of medical terms in Twitter messages. In particular, they proposed to use the Lucene retrieval engine[3] to retrieve medical concepts that could be potentially mapped to a given Twitter phrase, when mapping between Twitter phrases and medical concepts.

In contrast, we argue that the medical text normalisation task can be achieved by using well-established phrase-based MT techniques, where we translate a text written in *a social media language* (e.g. 'No way I'm gettin any sleep 2nite') to a text written in *a formal medical language* (e.g.

[1] http://www.dailystrength.org/
[2] http://twitter.com
[3] http://lucene.apache.org/

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1675–1680,
Lisbon, Portugal, 17-21 September 2015.

Table 1: Examples of the mappings between social media messages and medical concepts.

Social media message	Description of corresponding medical concept
No way I'm gettin any sleep 2nite	Insomnia (SNOMED ID: 193462001)
kept me up for days	Insomnia (SNOMED ID: 193462001)
can't even focus forreal	Unable to concentrate (SNOMED ID: 60032008)
I should be studying for but literally can't	Unable to concentrate (SNOMED ID: 60032008)
DRUG makes u skinny	Weight loss (SNOMED ID: 89362005)
still tired as shit	Fatigue (SNOMED ID: 84229001)
wiggin out a little bit	Fidgeting (SNOMED ID: 247910009)
I'm happiest with _DRUG_	Cheerful mood (SNOMED ID: 112080002)
DRUG made me the most chipper person	Cheerful mood (SNOMED ID: 112080002)

'Insomnia') and then calculate the similarity between the translated phrase and the description of a medical concept. Indeed, in this work we investigate an effective adaptation of phrase-based MT to map a Twitter phrase to a medical concept. Moreover, we propose to combine the adapted phrase-based MT technique and the similarity between word vector representations to effectively map a Twitter phrase to a medical concept.

The main contributions of this paper are threefold:

1. We investigate the adaptation of phrase-based MT to map a Twitter phrase to a SNOMED-CT concept.
2. We propose to combine our adaptation of phrase-based MT and the similarity between word vector representations to map Twitter phrases to formal medical concepts.
3. We thoroughly evaluate the proposed approach using phrases from our collection of tweets related to the topic of adverse drug reactions (ADRs).

2 Related Work

Phrase-based MT models, e.g. (Koehn et al., 2003; Och and Ney, 2004), have been shown to be effective in translation between languages, as they learn local term dependencies, such as collocations, reorderings, insertions and deletions. Koehn et al. (2003) showed that a phrase-based MT technique markedly outperformed traditional word-based MT techniques on several benchmarks. In this work, we adapt the phrase-based MT technique of Koehn et al. (2003) for the medical text normalisation task. In particular, we use the phrase-based MT technique to translate phrases from *Twitter language* to *formal medical language*, before mapping the translated phrases to medical concepts based on the ranked similarity of their word vector representations.

Traditional approaches for creating word vector representations treated words as atomic units (Mikolov et al., 2013b; Turian et al., 2010). For instance, the one-hot representation used a vector with a length of the size of the vocabulary, where one dimension is on, to represent a particular word (Turian et al., 2010). Recently, techniques for learning high-quality word vector representations (i.e. distributed word representations) that could capture the semantic similarity between words, such as continuous bags of words (CBOW) (Mikolov et al., 2013b) and global vectors (GloVe) (Pennington et al., 2014), have been proposed. Indeed, these distributed word representations have been effectively applied in different systems that achieve state-of-the-art performances for several NLP tasks, such as MT (Mikolov et al., 2013a) and named entity recognition (Passos et al., 2014). In this work, beside using word vector representations to measure the similarity between translated Twitter phrases and the description of medical concepts, we use the similarity between word vector representations of the original Twitter phrase and the description of a medical concept to augment the adapted phrase-based MT technique.

3 Medical Term Normalisation

We discuss our adaptation of phrase-based MT for medical text normalisation in Section 3.1. Section 3.2 introduces our proposed approach for combining similarity score of word vector representations with the adapted phrase-based MT technique.

3.1 Adapting Phrase-based MT

We aim to learn a translation between a Twitter phrase (i.e. a phrase from a Twitter message) and a formal medical phrase (i.e. the description of a medical concept). For a given Twitter phrase phr_t, we find a suitable medical phrase phr_m using *a translation score*, based on a phrase-based model, as follows:

$$score_{translation}(phr_m|phr_t) = p(phr_m|phr_t) \quad (1)$$

where $p(phr_m|phr_t)$ can be calculated using any phrase-based MT technique, e.g. (Koehn et al., 2003; Och and Ney, 2004). We then rank translated phrases phr_m based on this translation score. The top-k translated phrases are used for identifying the corresponding medical concept.

However, the translated phrase phr_m may not be exactly matched with the description of any target medical concepts. We propose two techniques to deal with this problem. For the first technique, we rank the target concepts based on the cosine similarity between the vector representation of phr_m and the vector representation of the description of each concept $desc_c$:

$$sim_{cos}(phr_m, desc_c) = \frac{V_{phr_m} \cdot V_{desc_c}}{||V_{phr_m}|| \times ||V_{desc_c}||} \quad (2)$$

where V_{phr_m} and V_{desc_c} are the vector representations of phr_m and $desc_c$, respectively. Any technique for creating word vector representations (e.g. one-hot, CBOW and GloVe) can be used. Note that if a phrase (e.g. phr_m) contains several terms, we create a vector representation by summing the value of the same dimension of the vector representation of each word (i.e. element-wise addition).

On the other hand, the second technique also incorporates the ranked position r of the translated phrase phr_m when translated from the original phrase phr_t using Equation (1). Indeed, the second technique calculates the similarity score as follows:

$$sim_{rcos}(phr_m, desc_c) = \frac{1}{r} \cdot \frac{V_{phr_m} \cdot V_{desc_c}}{||V_{phr_m}|| \times ||V_{desc_c}||} \quad (3)$$

3.2 Combining Similarity Score with Phrase-based MT

As discussed in Section 2, word vector representations (e.g. created by CBOW or GloVe) can capture semantic similarity between words by itself. Hence, we propose to map a Twitter phrase phr_t to a medical concept c, which is represented with a description $desc_c$, by linearly combining the cosine similarity, between vector representations of the Twitter phrase phr_t and the description $desc_c$, with the similarity score computed using one of the adapted phrased-based MT techniques (introduced in Section 3.1), as follows:

$$sim_{combine}(phr_t, desc_c) = \frac{V_{phr_t} \cdot V_{desc_c}}{||V_{phr_t}|| \times ||V_{desc_c}||} + MT_a(phr_t, desc_c) \quad (4)$$

where $MT_a(phr_t, desc_c)$ is calculated using one of the adapted phrase-based MT techniques described in Section 3.1.

4 Experimental Setup

4.1 Test Collection[4]

To evaluate our approach, we use a collection of 25 million tweets related to adverse drug reactions (ADRs), from cognitive enhancers (Hanson et al., 2013) and anti-depressants (Schneeweiss et al., 2010). These tweets were collected using the Twitter Streaming API[5] by filtering on the name of a particular set of drugs that can have adverse reactions to the patients. Note that terms regarding adverse drug reaction (e.g. insomnia) were not used for capturing tweets. From this collection, we use 201 ADR phrases and their corresponding SNOMED-CT concepts annotated by a PhD-level computational linguist. These phrases were anonymised by replacing numbers, user IDs, URIs, locations, email addresses, dates and drug names with appropriate tokens e.g. *_NUMBER_*.

4.2 Evaluation Approach

We conduct experiments using 10-fold cross validation, where the Twitter phrases are randomly divided into 10 separated folds. We address this task as a ranking task, where we aim to rank the medical concept with the highest similarity score, e.g. calculated using Equation (2), at the top rank. Hence, we evaluate our approach using Mean Reciprocal Rank (MRR) measure (Craswell, 2009), which is an information retrieval measure based on the user model where the user wants to see

[4]The gold-standard mapping between the Twitter phrases and the SNOMED-CT concepts are available on Zenodo.org (DOI: http://dx.doi.org/10.5281/zenodo.27354)

[5]https://dev.twitter.com/streaming/public

only one relevant concept. In particular, MRR is based on the the reciprocal of the rank at which the first relevant concept is viewed in the ranking (e.g. MRR = 0.5 if the first mapped concept is wrong but the second is correct). We limit our evaluation at top 5 of the ranking (i.e. MRR-5). In addition, we compare the significant difference between the performance achieved by our proposed approach and the baselines using the paired t-test ($p < 0.05$).

4.3 Word Vector Representation

We use three different techniques, including one-hot, CBOW and GloVe, to create word vector representations used in our approach (see Section 3). In particular, the vocabulary for creating the one-hot representation includes all terms in the Twitter phrases and the descriptions of the target SNOMED-CT concepts. Meanwhile, we create word vector representations based on CBOW and GloVe by using the word2vec[6] and GloVe[7] implementations. We learn the vector representations from the collections of tweets and medical articles, respectively, using window size of 10 words. The tweet collection (denoted *Twitter*) contains 419,702,147 English tweets, which are related to 11 drug names and 6 cities, while the medical article collection (denoted *BMC*) includes all medical articles from the BioMed Central[8]. For both CBOW and GloVe, we create vector representations with vector sizes 50 and 200, respectively.

4.4 Learning Phrase-based Model

We use the phrase-based MT technique of Koehn et al. (2003), as implemented in the Moses toolkit (Koehn et al., 2007)[9] with default settings, to learn to translate from the Twitter language to the medical language. In particular, when training the translator, we show the learner pairs of the Twitter phrases and descriptions of the corresponding SNOMED-CT concepts.

5 Experimental Results

We evaluate 6 different instantiations of the proposed approach discussed in Section 3, including:

1. *bestMT*: set $k = 1$, when finding the translated phrase phr_m for a Twitter phrase phr_t (Equation (1)), before ranking target medical concepts for the translated phrase phr_m using Equation (2).
2. *top5MT*: similar to *bestMT*, but set $k = 5$.
3. *top5MTr*: similar to *top5MT*, but also consider the rank position of the translate phrases when ranking the target medical concepts by using Equation (3).
4. *bestMT+vSim*: incorporate with the ranking generated from *bestMT*, the cosine similarity between the vector representations of the Twitter phrase phr_t and the description $desc_c$ of target medical concepts by using Equation (4).
5. *top5MT+vSim*: similar to *bestMT+vSim*, but use the ranking from *top5MT*.
6. *top5MTr+vSim*: similar to *bestMT+vSim*, but use the ranking from *top5MTr*.

Another baseline is *vSim*, where we consider only the cosine similarity between the vector representations of the Twitter phrase phr_t and the description $desc_c$ of target medical concepts.

Table 2 compares the performance of these 6 instantiations and the *vSim* baseline in terms of MRR-5. We firstly observe that for the *vSim* baseline, excepting for word vector representation with vector size 50 learned using GloVe from the Twitter collection, word vector representations learned using either CBOW or GloVe are more effective than the one-hot representation. However, the difference between the MRR-5 performance is not statistically significant ($p > 0.05$, paired t-test). In addition, word vector representations learned either using CBOW or GloVe with vector size 200 is more effective than those with vector size 50.

Next, we find that our adaptation of phrase-based MT (i.e. *bestMT*, *top5MT* and *top5MTr*) significantly ($p < 0.05$) outperforms the *vSim* baseline. For example, with the one-hot representation, *top5MT* (MRR-5 0.2491) and *top5MTr* (MRR-5 0.2458) perform significantly ($p < 0.05$) better than *vSim* (MRR-5 0.1675) by up to 49%. Meanwhile, when using word vector representations with the vector size 200 learned using GloVe from the BMC collection, *top5MT* (MRR-5 0.2638) significantly ($p < 0.05$) outperforms *vSim* with either the GloVe vector representation (MRR-5 0.1869) or the one-hot representation (MRR-5 0.1675). We observe the similar trends in performance

[6] https://code.google.com/p/word2vec/

[7] http://nlp.stanford.edu/projects/glove/

[8] http://www.biomedcentral.com/about/datamining

[9] http://www.statmt.org/moses/

Table 2: MRR-5 performance of the proposed approach and the baselines. Significant differences ($p < 0.05$) compared to the cosine similarity (*vSim*) baselines with the one-hot representation, and with the corresponding distributed word representation (e.g. CBOW or GloVe) are denoted $^{\triangle}$ and $^{\blacktriangle}$, respectively.

Approach	One-hot	BMC				Twitter			
		CBOW		GloVe		CBOW		GloVe	
		50	200	50	200	50	200	50	200
vSim	0.1675	0.1771	0.1896	0.1840	0.1869	0.1812	0.1813	0.0936	0.1807
bestMT	0.2232	0.1926	0.2070	0.1803	0.2500$^{\triangle}$	0.2014	0.2047	0.1258	0.2138
top5MT	0.2491$^{\triangle}$	**0.1994**	0.2104	0.1879	**0.2638**$^{\triangle\blacktriangle}$	**0.2037**	0.2095	**0.1322**	0.2362
top5MTr	0.2458$^{\triangle}$	0.1982	0.2109	**0.1894**	0.2617$^{\triangle}$	**0.2037**	**0.2096**	**0.1322**	0.2310
bestMT+vSim	0.2420$^{\triangle}$	0.1910	0.1953	0.1860	0.2532$^{\triangle}$	0.1891	0.1954	0.1078	0.2374
top5MT+vSim	0.2556$^{\triangle}$	0.1916	**0.2144**	0.1726	0.2600$^{\triangle}$	0.1978	0.2068	0.1079	0.2405$^{\triangle}$
top5MTr+vSim	**0.2594**$^{\triangle}$	0.1861	0.2070	0.1802	0.2590$^{\triangle}$	0.1959	0.2027	0.1129	**0.2406**$^{\triangle}$

when using vector representations learned from the Twitter collection. These results show that our adapted phrase-based MT techniques are effective for the medical term normalisation task.

In addition, we observe the effectiveness of our combined approach (i.e. *bestMT+vSim*, *top5MT+vSim* and *top5MTr+vSim*), as it further improves the performance of the adapted phrase-based MT (i.e. *bestMT*, *top5MT* and *top5MTr*, respectively), when using the one-hot representation. For example, *top5MTr+vSim* achieves the MRR-5 of 0.2594, while the MRR-5 of *top5MTr* is 0.2458. However, the performance difference is not statistically significant. Meanwhile, when using the CBOW and GloVe vectors, the achieved performance is varied based on the collection (i.e. BMC or Twitter) used for learning the vectors and the size of the vectors.

6 Conclusions

We have introduced our approach that adapts a phrase-based MT technique to normalise medical terms in Twitter messages. We evaluate our proposed approach using a collection of phrases from tweets related to ADRs. Our experimental results show that the proposed approach significantly outperforms an effective baseline by up to 55%. For future work, we aim to investigate the modelling of learned vector representation, such as CBOW and GloVe, within a phrase-based MT model when normalising medical terms.

Acknowledgements

The authors gratefully acknowledge Nestor Alvaro (Sokendai, Japan) for providing access to the Twitter/SNOMED-CT annotations which were used to derive the test collection used in these experiments. The derived dictionary and a representative sample of the word vector representations (CBOW and GloVe at 200d) are made available on Zenodo.org (DOI: http://dx.doi.org/10.5281/zenodo.27354). We wish to thank funding support from the EPSRC (grant number EP/M005089/1).

References

Timothy Baldwin, Paul Cook, Marco Lui, Andrew MacKinlay, and Li Wang. 2013. How noisy social media text, how diffrnt social media sources. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 356–364.

Nick Craswell. 2009. Mean reciprocal rank. In *Encyclopedia of Database Systems*, pages 1703–1703. Springer.

Peter L Elkin, David A Froehling, Dietlind L Wahner-Roedler, Steven H Brown, and Kent R Bailey. 2012. Comparison of natural language processing biosurveillance methods for identifying influenza from encounter notes. *Annals of Internal Medicine*, 156(1_Part_1):11–18.

Glenn T Gobbel, Ruth Reeves, Shrimalini Jayaramaraja, Dario Giuse, Theodore Speroff, Steven H Brown, Peter L Elkin, and Michael E Matheny. 2014. Development and evaluation of raptat: a machine learning system for concept mapping of phrases from medical narratives. *Journal of biomedical informatics*, 48:54–65.

Carl L Hanson, Scott H Burton, Christophe Giraud-Carrier, Josh H West, Michael D Barnes, and Bret Hansen. 2013. Tweaking and tweeting: exploring twitter for nonmedical use of a psychostimulant drug (adderall) among college students. *Journal of medical Internet research*, 15(4).

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North*

American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1, pages 48–54. Association for Computational Linguistics.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th annual meeting of the ACL on interactive poster and demonstration sessions*, pages 177–180. Association for Computational Linguistics.

Nut Limsopatham and Nigel Collier. 2015. Towards the semantic interpretation of personal health messages from social media. *Proceedings of the 1st Workshop on Understanding the City with Urban Informatics (UCUI 2015) in conjunction with CIKM 2015* (in press). `https://www.repository.cam.ac.uk/handle/1810/249275`

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

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

Alexander A Morgan, Zhiyong Lu, Xinglong Wang, Aaron M Cohen, Juliane Fluck, Patrick Ruch, Anna Divoli, Katrin Fundel, Robert Leaman, Jörg Hakenberg, et al. 2008. Overview of biocreative ii gene normalization. *Genome biology*, 9(Suppl 2):S3.

Franz Josef Och and Hermann Ney. 2004. The alignment template approach to statistical machine translation. *Computational linguistics*, 30(4):417–449.

Karen O'Connor, Pranoti Pimpalkhute, Azadeh Nikfarjam, Rachel Ginn, Karen L Smith, and Graciela Gonzalez. 2014. Pharmacovigilance on twitter? mining tweets for adverse drug reactions. In *AMIA Annual Symposium Proceedings*, volume 2014, page 924. American Medical Informatics Association.

Alexandre Passos, Vineet Kumar, and Andrew McCallum. 2014. Lexicon infused phrase embeddings for named entity resolution. *arXiv preprint arXiv:1404.5367*.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. *Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014)*, 12:1532–1543.

Sebastian Schneeweiss, Amanda R Patrick, Daniel H Solomon, Colin R Dormuth, Matt Miller, Jyotsna Mehta, Jennifer C Lee, and Philip S Wang. 2010. Comparative safety of antidepressant agents for children and adolescents regarding suicidal acts. *Pediatrics*, pages peds–2009.

Kent A Spackman, Keith E Campbell, and Roger A Côté. 1997. Snomed rt: a reference terminology for health care. In *Proceedings of the AMIA annual fall symposium*, page 640. American Medical Informatics Association.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 384–394. Association for Computational Linguistics.

Xiaoyan Wang, George Hripcsak, Marianthi Markatou, and Carol Friedman. 2009. Active computerized pharmacovigilance using natural language processing, statistics, and electronic health records: a feasibility study. *Journal of the American Medical Informatics Association*, 16(3):328–337.

Script Induction as Language Modeling

Rachel Rudinger[1], **Pushpendre Rastogi**[1], **Francis Ferraro**[1], **and Benjamin Van Durme**[1,2]
[1]Center for Language and Speech Processing
[2]Human Language Technology Center of Excellence
Johns Hopkins University

Abstract

The *narrative cloze* is an evaluation metric commonly used for work on automatic script induction. While prior work in this area has focused on count-based methods from distributional semantics, such as pointwise mutual information, we argue that the narrative cloze can be productively reframed as a language modeling task. By training a discriminative language model for this task, we attain improvements of up to 27 percent over prior methods on standard narrative cloze metrics.

1 Introduction

Although the concept of *scripts* in artificial intelligence dates back to the 1970s (Schank and Abelson, 1977), interest in this topic has renewed with recent efforts to automatically induce scripts from text on a large scale. One particularly influential work in this area, Chambers and Jurafsky (2008), treats the problem of script induction as one of learning *narrative chains*, which they accomplish using simple textual co-occurrence statistics. For the novel task of learning narrative chains, they introduce a new evaluation metric, the *narrative cloze* test, which involves predicting a missing event from a chain of events drawn from text. Several follow-up works (Chambers and Jurafsky, 2009; Jans et al., 2012; Pichotta and Mooney, 2014; Rudinger et al., 2015) employ and extend Chambers and Jurafsky (2008)'s methods for learning narrative chains, each using the narrative cloze to evaluate their work. [1]

In this paper, we take the position that the narrative cloze test, which has been treated predominantly as a method for evaluating script knowledge, is more productively thought of simply as a language modeling task.[2] To support this claim, we demonstrate a marked improvement over previous methods on this task using a powerful discriminative language model – the Log-Bilinear model (LBL). Based on this finding, we believe one of the following conclusions must follow: either discriminative language models are a more effective technique for script induction than previous methods, or the narrative cloze test is not a suitable evaluation for this task.[3]

2 Task Definition

Following the definitions of Chambers and Jurafsky (2008), a **narrative chain** is "a partially ordered set of narrative events that share a common actor," where a **narrative event** is "a tuple of an event (most simply a verb) and its participants, represented as *typed dependencies*." (De Marneffe et al., 2006) Formally, $e := (v, d)$, where e is a narrative event, v is a verb lemma, and d is the syntactic dependency (*nsubj* or *dobj*) between v and the protagonist. As an example, consider the following narrative:

> John studied for the exam and aced it.
> His teacher congratulated him.

With John as protagonist, we have a sequence of three narrative events: $(study, nsubj)$, $(ace, nsubj)$, and $(congratulate, dobj)$.

In the **narrative cloze** test, a sequence of narrative events (like the example provided here) is extracted automatically from a document, and one

[1]A number of related works on script induction use alternative task formulations and evaluations. (Chambers, 2013; Cheung et al., 2013; Cheung and Penn, 2013; Frermann et al., 2014; Manshadi et al., 2008; Modi and Titov, 2014; Regneri et al., 2010)

[2]Manshadi et al. (2008) also take a language modeling approach to event prediction, although their experiments are not directly comparable.

[3]We note that, whether the narrative cloze was originally intended as a rigorous evaluation of script induction techniques or merely a preliminary metric, we are motivated by the observation that this evaluation has nonetheless become a standard metric for this task.

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1681–1686,
Lisbon, Portugal, 17-21 September 2015.

narrative event is removed; the task is to predict the missing event.

Data Each of the models discussed in the following section are trained and tested on chains of narrative events extracted from stories in the New York Times portion of the Gigaword corpus (Graff et al., 2003) with Concrete annotations (Ferraro et al., 2014). Training is on the entirety of the 1994–2006 portion (16,688,422 chains with 58,515,838 narrative events); development is a subset of the 2007–2008 portion (10,000 chains with 35,109 events); and test is a subset of the 2009–2010 portion (5,000 chains with 17,836 events). All extracted chains are of length two or greater.

Chain Extraction To extract chains of narrative events for training and testing, we rely on the (automatically-generated) coreference chains present in Concretely Annotated Gigaword. Each narrative event in an extracted chain is derived from a single mention in the corresponding coreference chain, i.e., it consists of the verb and syntactic dependency ($nsubj$ or $dobj$) that governs the head of the mention, if such a dependency exists. Overlapping mentions within a coreference chain are collapsed to a single mention to avoid redundant extractions.

3 Models

In this section we present each of the models we train for the narrative cloze evaluation. In a single narrative cloze test, a sequence of narrative events, $(e_1, \cdots, e_L)$, with an insertion point, k, for the missing event is provided. Given a fixed vocabulary of narrative events, $\mathcal{V}$, a candidate sequence is generated for each vocabulary item by inserting that item into the sequence at index k. Each model generates a score for the candidate sequences, yielding a ranking over the vocabulary items. The rank assigned to the actual missing vocabulary item is the score the model receives on that cloze test. In this case, we set $\mathcal{V}$ to include all narrative events, e, that occur at least ten times in training, yielding a vocabulary size of 12,452. All out-of-vocabulary events are converted to (and scored as) the symbol UNK.

3.1 Count-based Methods

Unigram Baseline (UNI) A simple but strong baseline introduced by Pichotta and Mooney (2014) for this task is the unigram model: candidates are ranked by their observed frequency in training, without regard to context.

Unordered PMI (UOP) The original model for this task, proposed by Chambers and Jurafsky (2008), is based on the pointwise mutual information (PMI) between events.

$$pmi(e_1, e_2) \propto \log \frac{C(e_1, e_2)}{C(e_1, *)C(*, e_2)} \quad (1)$$

Here, $C(e_1, e_2)$ is the number of times e_1 and e_2 occur in the same narrative event sequence, i.e., the number of times they "had a coreferring entity filling the values of [their] dependencies," and the ordering of e_1 and e_2 is not considered. In our implementation, individual counts are defined as follows:

$$C(e, *) := \sum_{e' \in \mathcal{V}} C(e, e') \quad (2)$$

This model selects the best candidate event in a given cloze test according to the following score:

$$\hat{e} = \arg\max_{e \in \mathcal{V}} \sum_{i=1}^{L} pmi(e, e_i) \quad (3)$$

We tune this model with an option to apply a modified version of discounting for PMI from Pantel and Ravichandran (2004).

Ordered PMI (OP) This model is a slight variation on Unordered PMI introduced by Jans et al. (2012). The only distinction is that $C(e_1, e_2)$ is treated as an asymmetric count, sensitive to the order in which e_1 and e_2 occur within a chain.

Bigram Probability (BG) Another variant introduced by Jans et al. (2012), the "bigram probability" model uses conditional probabilities rather than PMI to compute scores. In a cloze test, this model selects the following event:

$$\hat{e} = \arg\max_{e \in \mathcal{V}} \prod_{i=1}^{k} p(e|e_i) \prod_{i=k+1}^{L} p(e_i|e) \quad (4)$$

where $p(e_2|e_1) = \frac{C(e_1, e_2)}{C(e_1, *)}$ and $C(e_1, e_2)$ is asymmetric. We tune this model with an option to perform absolute discounting. Note that this model is not a bigram model in the typical language modeling sense.

Len	UNI	UOP	OP	BG	LBL2	LBL4	Tests
2	490	1887	2363	1613	**369**	371	5668
3	452	1271	1752	1009	**330**	334	2793
4	323	806	1027	502	**229**	232	1616
5	364	735	937	442	254	**243**	1330
6	347	666	891	483	257	**249**	942
7	330	629	838	468	241	**237**	630
8	259	466	510	278	208	**201**	512
9	299	610	639	348	198	**195**	396
10+	331	472	397	277	240	**229**	3949
ALL	400	1115	1382	868	294	**292**	17836

(a) Average Rank

Len	UNI	UOP	OP	BG	LBL2	LBL4	Tests
2	.148	.053	.077	.149	**.205**	.204	5668
3	.179	.043	.065	.164	**.217**	.215	2793
4	.226	.042	.064	.195	**.253**	.253	1616
5	.225	.049	.076	.213	.261	**.266**	1330
6	.213	.054	.079	.214	.254	**.263**	942
7	.213	.061	.092	.215	.243	**.247**	630
8	.235	.063	.091	.244	.268	**.278**	512
9	.259	.058	.107	.252	**.280**	.278	396
10+	.191	.082	.113	.193	.198	**.205**	3949
ALL	.186	.057	.083	.181	.221	**.223**	17836

(b) Mean Reciprocal Rank (MRR)

Len	UNI	UOP	OP	BG	LBL2	LBL4	Tests
2	23.9	09.4	11.9	23.8	34.0	**34.1**	5668
3	28.8	08.2	11.1	28.0	**36.3**	35.6	2793
4	33.9	07.7	14.4	32.2	**38.7**	**38.7**	1616
5	33.4	10.1	18.7	34.0	39.6	**40.3**	1330
6	34.8	10.9	22.2	36.8	40.5	**41.9**	942
7	32.5	12.2	24.0	34.9	**39.4**	39.2	630
8	36.7	13.7	21.7	38.7	41.6	**43.2**	512
9	37.9	15.2	28.5	39.1	41.7	**43.2**	396
10+	31.4	18.5	24.0	32.7	**35.7**	**35.7**	3949
ALL	29.5	11.6	16.8	29.8	36.5	**36.6**	17836

(c) Percent Recall at 10

Len	UNI	UOP	OP	BG	LBL2	LBL4	Tests
2	41.7	16.9	25.5	38.6	**51.2**	51.0	5668
3	46.8	20.2	30.2	45.0	**54.8**	54.0	2793
4	53.8	25.3	37.8	54.0	59.0	**60.0**	1616
5	52.5	29.9	40.5	54.3	59.1	**61.1**	1330
6	53.9	33.2	40.7	55.2	60.6	**61.7**	942
7	51.8	34.3	42.7	56.5	61.6	**63.8**	630
8	58.2	42.2	47.7	61.3	**67.2**	67.0	512
9	58.1	42.2	47.7	60.1	66.2	**67.0**	396
10+	49.9	47.4	50.1	54.2	58.4	**59.8**	3949
ALL	48.0	28.6	36.4	48.3	56.3	**56.8**	17836

(d) Percent Recall at 50

Table 1: Narrative cloze results bucketed by chain length for each model and scoring metric with best results in bold. The models are Unigram Model (UNI), Unordered PMI (UOP), Ordered PMI (OP), Bigram Probability Model (BG), Log-Bilinear Model N=2 (LBL2), Log-Bilinear Model N=4 (LBL4)

Skip N-gram We tune the previous three models (UOP, OP, and BG) with the skip n-gram counting methods introduced by Jans et al. (2012) for this task, varying the ways in which the counts, $C(e_1, e_2)$, are collected. Using skip-n counting, $C(e_1, e_2)$ is incremented every time e_1 and e_2 co-occur within a window of size n. We experiment with **skip-0** (consecutive events only), **skip-3** (window size 3), and **skip-all** (entire chain length) settings.

For each of the four narrative cloze scoring metrics we report on (average rank, mean reciprocal rank, recall at 10, and recall at 50), we tune the Unordered PMI, Ordered PMI, and Bigram Probability models over the following parameter space: $\{$**skip-0**, **skip-3**, **skip-all**$\}$ $\times$ $\{$**discount**, **no-discount**$\}$ $\times$ $\{$**T=4**, **T=10**, **T=20**$\}$, where **T** is a pairwise count threshold.

3.2 A Discriminative Method

Log-Bilinear Language Model (LBL) The Log-Bilinear language model is a language model that was introduced by Mnih and Hinton (2007). Like other language models, the LBL produces a probability distribution over the next possible word given a sequence of N previously observed words. N is a hyper-parameter that determines the size of the context used for computing the probabilities. While many variants of the LBL have been proposed since its introduction, we use the simple variant described below.

Formally, we associate one context vector $\mathbf{c}_e \in \mathbb{R}^d$, one bias parameter $b_e \in \mathbb{R}$, and one target vector $\mathbf{t}_e \in \mathbb{R}^d$ to each narrative event $e \in \mathcal{V} \cup \{$ UNK, BOS, EOS $\}$. $\mathcal{V}$ is the vocabulary of events and BOS, EOS, and UNK are the beginning-of-sequence, end-of-sequence, and out-of-vocabulary symbols, respectively. The probability of an event e that appears after a sequence $s = [s_1, s_2, \ldots, s_N]$ of context words is defined as:

$$p(e|s) = \frac{\exp(\mathbf{t}_e^\intercal \hat{\mathbf{t}}_s + b_e)}{\sum_{e' \in \mathcal{V} \cup \{ \text{ UNK, EOS } \}} \exp(\mathbf{t}_{e'}^\intercal \hat{\mathbf{t}}_s + b_{e'})}$$

$$\text{where } \hat{\mathbf{t}}_s = \sum_{j=1}^{N} \mathbf{m}_j \odot \mathbf{c}_{s_j}$$

The $\odot$ operator performs element-wise multiplication of two vectors. The parameters that are optimized during training are $\mathbf{m}_j \; \forall j \in [1, \ldots, N]$ and $\mathbf{c}_e, \mathbf{t}_e \; \forall e \in \mathcal{V} \cup \{$ UNK, BOS, EOS $\}$. To calculate the log-probability of a sequence of narrative events $E = (e_1, \ldots, e_L)$ we compute:

$$l(S) = \left(\sum_{i=1}^{n} \log(p(e_i | f_E(e_i))) \right) + \log(p(\text{EOS} | f_E(\text{EOS}))) \quad (5)$$

Here f_E is a function that returns the sequence of N words that precede the event e_i in the se-

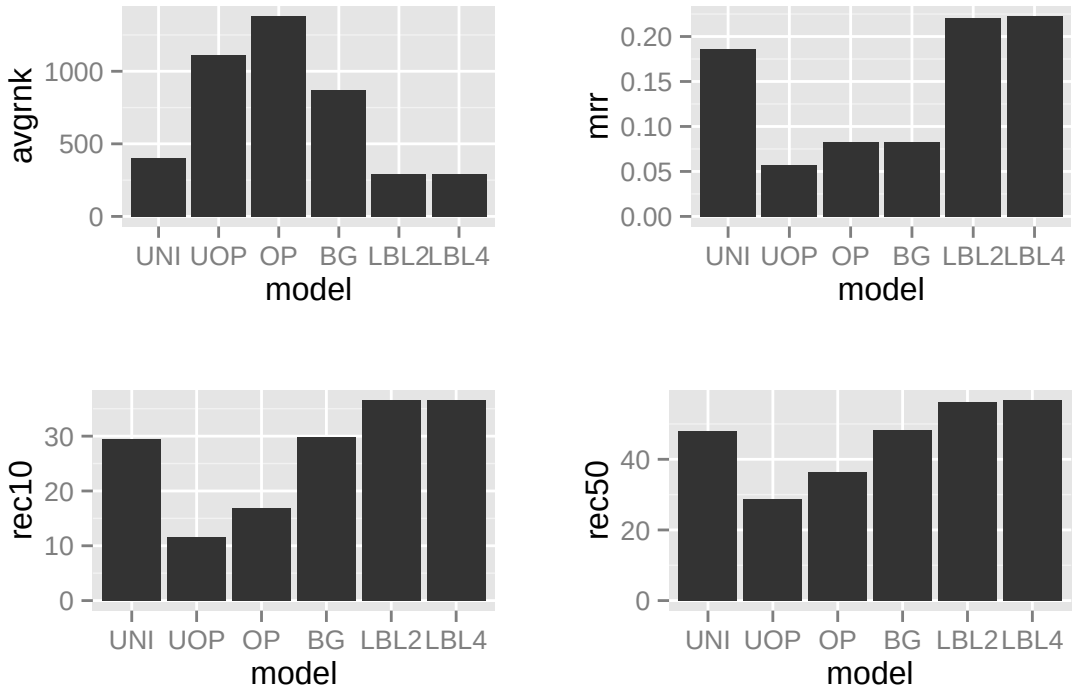


Figure 1: Narrative cloze results over all chain lengths. Unigram Model (UNI), Unordered PMI Model (UOP), Ordered PMI Model (OP), Bigram Probability Model (BG), Log-Bilinear Model with context size 2 or 4 (LBL2, LBL4). Average Rank (avgrnk), Mean Reciprocal Rank (mrr), % Recall at 10 (rec10), % Recall at 50 (rec50).

quence E' made by prepending N BOS tokens and appending a single EOS token to E.

The LBL models are trained by minimizing the objective described in Equation 5 for all the sequences in the training corpus. We used the OxLM toolkit (Paul et al., 2014) which internally uses Noise-Contrastive Estimation (NCE) (Gutmann and Hyvärinen, 2010) and processor parallelization for speeding up the training. For this task, we train LBL models with $N = 2$ (LBL2) and $N = 4$ (LBL4). In our experiments, increasing context size to $N = 6$ did not significantly improve (or degrade) performance.

4 Experimental Results

Table 1 shows the results of 17,836 narrative cloze tests (derived from 5,000 held-out test chains), with results bucketed by chain length. Performance is reported on four metrics: average rank, mean reciprocal rank, recall at 10, and recall at 50.

For each of the four metrics, the best overall performance is achieved by one of the two LBL models (context size 2 or 4); the LBL models also achieve the best performance on every chain length. Not only are the gains achieved by the discriminative LBL consistent across metrics and chain length, they are large. For average rank, the LBL achieves a 27.0% relative improvement over the best non-discriminative model; for mean reciprocal rank, a 19.9% improvement; for recall at 10, a 22.8% improvement; and for recall at 50, a 17.6% improvement. (See Figure 1.) Furthermore, note that both PMI models and the Bigram model have been individually tuned for each metric, while the LBL models have not. (The two LBL models are tuned only for overall perplexity on the development set.)

All models trend toward improved performance on longer chains. Because the unigram model also improves with chain length, it appears that longer chains contain more frequent events and are thus easier to predict. However, LBL performance is also likely improving on longer chains because of additional contextual information, as is evident from LBL4's slight relative gains over LBL2 on longer chains.

5 Conclusion

Pointwise mutual information and other related count-based techniques have been used widely to identify semantically similar words (Church and Hanks, 1990; Lin and Pantel, 2001; Tur-

ney and Pantel, 2010), so it is natural that these techniques have also been applied to the task of script induction. Qualitatively, PMI often identifies intuitively compelling matches; among the top 15 events to share a high PMI with $(eat, nsubj)$ under the Unordered PMI model, for example, we find events such as $(overeat, nsubj)$, $(taste, nsubj)$, $(smell, nsubj)$, $(cook, nsubj)$, and $(serve, dobj)$. When evaluated by the narrative cloze test, however, these count-based methods are overshadowed by the performance of a general-purpose discriminative language model.

Our decision to attempt this task with the Log-Bilinear model was motivated by the simple observation that the narrative cloze test is, in reality, a language modeling task. Does the LBL's success on this task mean that work in script induction should abandon traditional count-based methods for discriminative language modeling techniques? Or does it mean that an alternative evaluation metric is required to measure script knowledge? While we believe our results are sufficient to conclude that one of these alternatives is the case, we leave the task of determining which to future research.

Acknowledgments

This work was supported by the Paul Allen Institute for Artificial Intelligence (*Acquisition and Use of Paraphrases in a Knowledge-Rich Setting*), a National Science Foundation Graduate Research Fellowship (Grant No. DGE-1232825), the Johns Hopkins HLTCOE, and DARPA DEFT (FA8750-13-2-001, *Large Scale Paraphrasing for Natural Language Understanding*). We would also like to thank three anonymous reviewers for their feedback. Any opinions expressed in this work are those of the authors.

References

Nathanael Chambers and Dan Jurafsky. 2008. Unsupervised learning of narrative event chains. In *Proceedings of ACL-08: HLT*, pages 789–797, Columbus, Ohio. Association for Computational Linguistics.

Nathanael Chambers and Dan Jurafsky. 2009. Unsupervised learning of narrative schemas and their participants. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 602–610, Suntec, Singapore. Association for Computational Linguistics.

Nathanael Chambers. 2013. Event schema induction with a probabilistic entity-driven model. In *EMNLP*, volume 13, pages 1797–1807.

Jackie Chi Kit Cheung and Gerald Penn. 2013. Probabilistic domain modelling with contextualized distributional semantic vectors. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 392–401. Association for Computational Linguistics.

Jackie Chi Kit Cheung, Hoifung Poon, and Lucy Vanderwende. 2013. Probabilistic frame induction. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 837–846, Atlanta, Georgia, June. Association for Computational Linguistics.

Kenneth Ward Church and Patrick Hanks. 1990. Word association norms, mutual information, and lexicography. *Computational linguistics*, 16(1):22–29.

Marie-Catherine De Marneffe, Bill MacCartney, Christopher D Manning, et al. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC*, volume 6, pages 449–454.

Francis Ferraro, Max Thomas, Matthew R. Gormley, Travis Wolfe, Craig Harman, and Benjamin Van Durme. 2014. Concretely Annotated Corpora. In *4th Workshop on Automated Knowledge Base Construction (AKBC)*.

Lea Frermann, Ivan Titov, and Manfred Pinkal. 2014. A hierarchical bayesian model for unsupervised induction of script knowledge. *EACL 2014*, page 49.

David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2003. English gigaword. *Linguistic Data Consortium, Philadelphia*.

Michael Gutmann and Aapo Hyvärinen. 2010. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *International Conference on Artificial Intelligence and Statistics*, pages 297–304.

Bram Jans, Steven Bethard, Ivan Vulić, and Marie-Francine Moens. 2012. Skip n-grams and ranking functions for predicting script events. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 336–344, Avignon, France. Association for Computational Linguistics.

Dekang Lin and Patrick Pantel. 2001. Dirt - discovery of inference rules from text. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 323–328. ACM.

Mehdi Manshadi, Reid Swanson, and Andrew S Gordon. 2008. Learning a probabilistic model of event sequences from internet weblog stories. In *FLAIRS Conference*, pages 159–164.

Andriy Mnih and Geoffrey Hinton. 2007. Three new graphical models for statistical language modelling. In *Proceedings of the 24th international conference on Machine learning*, pages 641–648. ACM.

Ashutosh Modi and Ivan Titov. 2014. Inducing neural models of script knowledge. *CoNLL-2014*, page 49.

Patrick Pantel and Deepak Ravichandran. 2004. Automatically labeling semantic classes. In Daniel Marcu Susan Dumais and Salim Roukos, editors, *HLT-NAACL 2004: Main Proceedings*, pages 321–328, Boston, Massachusetts, USA. Association for Computational Linguistics.

Baltescu Paul, Blunsom Phil, and Hoang Hieu. 2014. Oxlm: A neural language modelling framework for machine translation. *The Prague Bulletin of Mathematical Linguistics*, 102(1):81–92.

Karl Pichotta and Raymond Mooney. 2014. Statistical script learning with multi-argument events. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 220–229, Gothenburg, Sweden. Association for Computational Linguistics.

Michaela Regneri, Alexander Koller, and Manfred Pinkal. 2010. Learning script knowledge with web experiments. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 979–988. Association for Computational Linguistics.

Rachel Rudinger, Vera Demberg, Ashutosh Modi, Benjamin Van Durme, and Manfred Pinkal. 2015. Learning to predict script events from domain-specific text. *Lexical and Computational Semantics (* SEM 2015)*, page 205.

Roger Schank and Robert Abelson. 1977. *Scripts, plans, goals and understanding: An inquiry into human knowledge structures*. Lawrence Erlbaum Associates, Hillsdale, NJ.

Peter D. Turney and Patrick Pantel. 2010. From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research*, 37(1):141–188, January.

Online Learning of Interpretable Word Embeddings

Hongyin Luo[1], **Zhiyuan Liu**[1,2] *, **Huanbo Luan**[1], **Maosong Sun**[1,2]

[1] Department of Computer Science and Technology, State Key Lab on Intelligent Technology and Systems, National Lab for Information Science and Technology, Tsinghua University, Beijing, China

[2] Jiangsu Collaborative Innovation Center for Language Competence, Jiangsu, China

Abstract

Word embeddings encode semantic meanings of words into low-dimension word vectors. In most word embeddings, one cannot interpret the meanings of specific dimensions of those word vectors. Non-negative matrix factorization (NMF) has been proposed to learn interpretable word embeddings via non-negative constraints. However, NMF methods suffer from scale and memory issue because they have to maintain a global matrix for learning. To alleviate this challenge, we propose online learning of interpretable word embeddings from streaming text data. Experiments show that our model consistently outperforms the state-of-the-art word embedding methods in both representation ability and interpretability. The source code of this paper can be obtained from `http://github.com/skTim/OIWE`.

1 Introduction

Word embeddings (Turian et al., 2010) aim to encode semantic meanings of words into low-dimensional dense vectors. As compared with traditional one-hot representation and distributional representation, word embeddings can better address the sparsity issue and have achieved success in many NLP applications recent years.

There are two typical approaches for word embeddings. The neural-network (NN) approach (Bengio et al., 2006) employs neural-based techniques to learn word embeddings. The matrix factorization (MF) approach (Pennington et al., 2014) builds word embeddings by factorizing word-context co-occurrence matrices. The MF approach requires a global statistical matrix, while the NN approach can flexibly perform learning from streaming text data, which is efficient in both computation and memory. For example, two recent NN methods, Skip-Gram and Continuous Bag-of-Word Model (CBOW) (Mikolov et al., 2013a; Mikolov et al., 2013b), have achieved impressive impact due to their simplicity and efficiency.

For most word embedding methods, a critical issue is that, we are unaware of what each dimension represent in word embeddings. Hence, the latent dimension for which a word has its largest value is difficult to interpret. This makes word embeddings like a black-box, and prevents them from being human-readable and further manipulation.

People have proposed non-negative matrix factorization (NMF) for word representation, denoted as non-negative sparse embedding (NNSE) (Murphy et al., 2012). NNSE realizes interpretable word embeddings by applying non-negative constraints for word embeddings. Although NNSE learns word embeddings with good interpretabilities, like other MF methods, it also requires a global matrix for learning, thus suffers from heavy memory usage and cannot well deal with streaming text data.

Inspired by the characteristics of NMF methods (Lee and Seung, 1999), we note that, non-negative constraints only allow additive combinations instead of subtractive combinations, and lead to a parts-based representation. Hence, the non-negative constraints derive interpretabilities of word embeddings. In this paper, we aim to design an online NN method to efficiently learn interpretable word embeddings. In order to achieve the goal of interpretable embeddings, we design projected gradient descent (Lin, 2007) for optimization so as to apply non-negative constraints on NN methods such as Skip-Gram. We also employ adaptive gradient descent (Sun et al., 2012) to speedup learning convergence. We name the proposed models as online interpretable word embeddings (OIWE).

*Corresponding author: Z. Liu (liuzy@tsinghua.edu.cn)

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1687–1692, Lisbon, Portugal, 17-21 September 2015.

For experiments, we implement OIWE based on Skip-Gram. We evaluate the representation performance of word embedding methods on the word similarity computation task. Experiment results show that, our OIWE models are significantly superior to other baselines including Skip-Gram, RNN and NNSE. We also evaluate the interpretability performance on the word intrusion detection task. The results demonstrate the effectiveness of OIWE as compared to NNSE.

2 Our Model

In this section, we first introduce Skip-Gram and then introduce the proposed online interpretable word embeddings based on Skip-Gram.

2.1 Skip-Gram

Skip-Gram (Mikolov et al., 2013b) is simple and effective to learn word embeddings. The objective of Skip-Gram is to make word vectors good at predicting its context words. More specifically, given a word sequence $\{w_1, w_2, \ldots, w_T\}$, Skip-Gram aims to maximize the average log probability

$$\frac{1}{T}\sum_{1}^{T}\Big(\sum_{-k\le j\le k, j\neq 0}\log \Pr(w_{t+j}|w_t)\Big), \quad (1)$$

where k is the context window size, and $\Pr(w_{t+j}|w_t)$ indicates the probability of seeing w_{t+j} in the context of w_t, which are measured with softmax function

$$\Pr(w_{t+j}|w_t) = \frac{\exp\big(\mathbf{w}_{t+j}\cdot\mathbf{w}_t\big)}{\sum_{w\in W}\exp\big(\mathbf{w}\cdot\mathbf{w}_t\big)}, \quad (2)$$

where $\mathbf{w}_{t+j}$ and $\mathbf{w}_t$ are word embeddings of w_{t+j} and w_t, and W is the vocabulary size. Since the computation of full softmax is time consuming, the techniques of hierarchical softmax and negative sampling (Mikolov et al., 2013b) are proposed for approximation.

Take negative sampling for example. The log probability $\Pr(w_{t+j}|w_t)$ can be approximate by

$$\log\sigma\big(\mathbf{w}_{t+j}\cdot\mathbf{w}_t\big) + \sum_{w\in N_t}\log\sigma\big(\mathbf{w}\cdot\mathbf{w}_t\big), \quad (3)$$

where $\sigma(x) = 1/(1+\exp(-x))$, and N_t is the set of negative samples as compared to the corresponding context word $\mathbf{w}_{t+j}$. The task can be regarded as to distinguish the context word $\mathbf{w}_{t+j}$ from negative samples.

For Skip-Gram with negative sampling, we can perform stochastic gradient descent for learning. The update rule for the positive/negative context words $u \in \{w_{t+j}\}\cup N_t$ is

$$\mathbf{u}^{i+1} = \mathbf{u}^i + \gamma\big[I_{w_t}(u) - \sigma(\mathbf{u}\cdot\mathbf{w}_t)\big]\mathbf{w}_t^i, \quad (4)$$

where $I_{w_t}(u) = 1$ when w is the positive context word of w_t and $I_{w_t}(u) = 0$ when w is negative, i is the iteration number, and γ is the learning rate. Correspondingly, the update rule for the input word w_t is

$$\mathbf{w}_t^{i+1} = \mathbf{w}_t^i + \gamma\sum_{u\in\{w_t\}\cup N_t}\big[I_{w_t}(u) - \sigma(\mathbf{u}\cdot\mathbf{w}_t)\big]\mathbf{u}_t^i. \quad (5)$$

We note that, the learning rate γ in Skip-Gram is shared by all word embeddings.

2.2 OIWE

In order to learn interpretable word embeddings, we have to make the word embeddings learned in Skip-Gram keep non-negative. In order to achieve this goal, we have to constrain the update rules in Equation (4) and (5) as follows:

$$\mathbf{x}_k^{i+1} = P\big[\mathbf{x}_k^i + \gamma\nabla f(\mathbf{x}_k)\big], \quad (6)$$

where x may be u or w_t, k is the corresponding dimension in word embedding $\mathbf{x}$, $\nabla f(\mathbf{x}_k)$ indicates the gradient corresponding to $\mathbf{x}_k$, and $P[\cdot]$ is defined as

$$P[x] = \begin{cases} x & \text{if } x > 0, \\ 0 & \text{if } x \le 0. \end{cases} \quad (7)$$

Motivated by the projected gradient descent methods for NMF (Lin, 2007), in this paper we propose two methods for Skip-Gram to realize the constraint in Equation (6).

Naive Projected Gradient (NPG). In NPG, we consider the most straightforward update strategy by simply setting

$$\mathbf{x}_k^{i+1} = \max\big(0, \mathbf{x}_k^i + \gamma\nabla f(\mathbf{x}_k)\big). \quad (8)$$

The method has been used for NMF (Lin, 2007) although the details are not discussed.

The NPG method only constrains the violated dimensions without taking the update consistency among dimensions of a word embedding into account. For example, if many dimensions encounter $\mathbf{x}_k^i + \gamma\nabla f(\mathbf{x}_k) < 0$ at the same time, which are set to 0 with Equation (8) with other

dimensions unchanged, the updated word embedding may heavily deviate from its semantic meaning. Hence, NPG may suffer from instable updating results. To address this issue, we propose to employ the following improved projected gradient method.

Improved Projected Gradient (IPG). In order to make the non-negative update more consistent among dimensions, we design an improved projected gradient by iteratively finding the most appropriate learning rate γ. The basic idea is that, we will find a good learning rate γ to make less dimensions violate the non-negative constraint.

More specifically, in Equation (6), for a learning rate γ, we define the *violation ratio* as

$$R(\gamma) = \frac{\left|\{k|\mathbf{x}_k^i > 0, \mathbf{x}_k^i + \gamma \nabla f(\mathbf{x}_k) < 0\}\right|}{K}, \quad (9)$$

where K is the dimension size of word embeddings. The violation ratio indicates how many dimensions violate the non-negative constraint and require to be set to 0. When the learning rate γ decreases, the violation ratio will also decrease, and the zero-setting in Equation (8) will bring less deviation to word embeddings.

We set a threshold δ for the violation ratio $R(\gamma)$ and a lower bound γ_L for the learning rate γ. Starting from an initial learning rate γ^0, we will repeatedly decrease the learning rate by

$$\gamma^{m+1} = \gamma^m \cdot \beta \quad (10)$$

with $0 < \beta < 1$ until

$$R(\gamma^{m+1}) < \delta \quad \text{or} \quad \gamma^{m+1} \leq \gamma_L, \quad (11)$$

and then update with Equation (8) using γ^{m+1}. In nature, the updating constraint of learning rate in Equation (11) play a similar role to Equation (13) in (Lin, 2007), which aims to prevent the projection operation from heavily deviating the word embeddings.

2.3 More Optimization Details

In experiments, we explore many optimization methods and find the following two strategies are important: (1) *Adaptive Gradient Descent*. Following the idea from (Sun et al., 2012), we maintain different learning rates γ_w for each word w, and the learning rates for those high-frequency words may decrease faster than those low-frequency words. This will speedup the convergence of word embedding learning. (2) *Unified Word Embedding Space*. Different from original Skip-Gram (Mikolov et al., 2013b) which learn embeddings of w_t and its context words w_{t+j} in two separate spaces, in this paper both w_t and its context words w_{t+j} share the same embedding space. Hence, a word embedding may get more opportunities for learning.

3 Experiments

In this section, we investigate the representation performance and interpretability of our OIWE models with other baselines including typical N-N and MF methods.

The representation performance is evaluated with the word similarity computation task, and the interpretability is evaluated with the word intrusion detection task. For the both tasks, we train our OIWE models using the `text8` corpus obtained from *word2vec* website[1], and the OIWE models achieve the best performance by setting the dimension number $K = 300$, $\beta = 0.6$, $\delta = 1/60$, and $\gamma_L = 2.5 \times 10^{-6}$.

3.1 Word Similarity Computation

Following the settings in (Murphy et al., 2012), we also select the following three sets for word similarity computation: (1) **WS-203**, the strict-similarity subset of 203 pairs (Agirre et al., 2009) selected from the wordsim-353 (Finkelstein et al., 2001), (2) **RG-65**, 65 concrete word pairs built by (Rubenstein and Goodenough, 1965) and (3) **MEN**, $3,000$ word pairs built by (Bruni et al., 2014). The performance is evaluated with the Spearman coefficient between human judgements and similarities calculated using word embeddings.

We select three baselines including Skip-Gram (Mikolov et al., 2013b), recurrent neural networks (RNN) (Mikolov et al., 2011) and NNSE (Murphy et al., 2012). For Skip-Gram, we report the result we learned using *word2vec* on `text8` corpus. The result of RNN is from (Faruqui and Dyer, 2014) and the one of NNSE is from (Murphy et al., 2012).

The evaluation results of word similarity computation are shown in Table 1. We can observe that: (1) The OIWE models consistently outperform other baselines. (2) IPG generally achieves better representation performance than

[1] `https://code.google.com/p/word2vec/`

Model	WS-203	RG-65	MEN
Skip-Gram	67.35	50.49	52.56
RNN	49.28	50.19	43.44
NNSE	51.06	56.48	-
OIWE-NPG	63.71	56.85	**57.60**
OIWE-IPG	**71.74**	**57.16**	56.68

Table 1: Spearman coefficient results (%) on word similarity computation.

NPG. This indicates consistent updates are important for learning of word embeddings. One can refer to `http://github.com/skTim/OIWE` for the evaluation results on more evaluation datasets.

3.2 Word Intrusion Detection

We evaluate interpretability of word embeddings with the task of word intrusion detection proposed by (Murphy et al., 2012). In this task, for each dimension we create a word set containing top-5 words in this dimension, and intruce a noisy word from the bottom half of this dimension which ranks high in other dimensions. Human editors are asked to check each word set and try to pick out the intrusion words, and the detection precision indicates the interpretability of word embedding models. Note that, for this task we do not perform normalization for word vectors.

Model	Precision
Skip-Gram	32.62
NNSE	92.00
OIWE-NPG	61.40
OIWE-IPG	**94.80**

Table 2: Experiment results (%) on word intrusion detection.

The evaluation results are shown in Table 2. We can observe that: (1) Skip-Gram performs poor in word intrusion detection without doubt since it is uninterpretable in nature. (2) The OIWE-NPG model achieves better interpretability as compared to Skip-Gram, but performs much worse than the OIWE-IPG model. The OIWE-IPG model achieves competitive interpretability with NNSE. This indicates that reducing violation rations in word embedding learning is crucial for preserving interpretability.

In Table 3, we show top-5 words for some dimensions, which clearly demonstrate semantic meanings of these dimensions. One can also refer to `http://github.com/skTim/OIWE` to find top-5 words for all dimensions.

No.	Top Words
1	type, form, way, kind, manner
2	translates, describes, combines, included, includes
3	gospel, baptism, jesus, faith, judaism
4	Franz, Johann, Wilhelm, Friedrich, von
25	prominent, famous, important, influential, popular

Table 3: Top words of some dimensions in word embeddings.

3.3 Influence of Dimension Numbers

The dimension number is an important configuration in word embeddings. In Fig. 1 we show the performance of OIWE and Skip-Gram on word similarity computation with varying dimension numbers.

From the figure, we can observe that: (1) The both models achieve their best performance under the same dimension number. This indicates that OIWE, to some extent, inherits the representation power of Skip-Gram. (2) The performance of OIWE seems to be more sensitive to dimension numbers. When the dimension number changes from 300 to 200 or 400, the performance drops much quickly than Skip-Gram. The reason may be as follows. OIWE has to concern about both representation ability of word embeddings and interpretability of each dimension. An appropriate dimension number is critical to make each dimension interpretable, just like the cluster number is important for clustering. On the contrary, Skip-Gram is much free to learn word embeddings only concerning about representation ability. (3) The performance of OIWE with various dimensions also varies on different evaluation datasets. For example, OIWE-IPG with $K = 400$ gets 68.74 on MEN, which is much better than that with $K = 300$. In future work, we will extensively investigate the characteristics of OIWE with respect to dimension numbers and other hyperparameters.

4 Conclusion and Future Work

In this paper, we present online interpretable word embeddings. The OIWE models perform project-

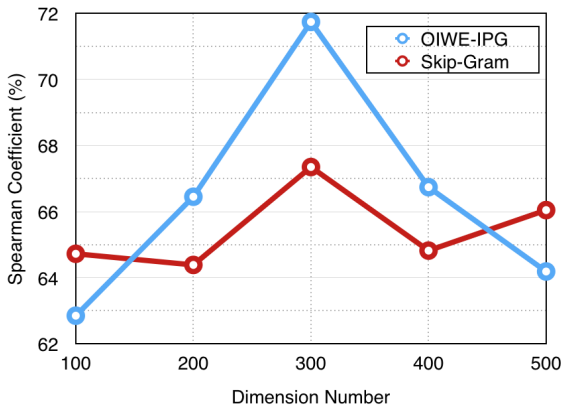


Figure 1: Influence of Dimension Number on Words Similarity

ed gradient descent to apply non-negative constraints on NN methods such as Skip-Gram. Experiment results on word similarity computation and word intrusion detection demonstrate the effectiveness and efficiency of our models in both representation ability and interpretability. We also note that, our models can be easily extended to other NN methods.

In future, we will explore the following research issues: (1) We will extensively investigate the characteristics of OIWE with respect to various hyperparameters including dimension numbers. (2) We will evaluate the performance of our OIWE models in various NLP applications. (3) We will also investigate possible extensions of our OIWE models, including multiple-prototype models for word sense embeddings (Huang et al., 2012; Chen et al., 2014), semantic compositions for phrase embeddings (Zhao et al., 2015) and knowledge representation (Bordes et al., 2013; Lin et al., 2015).

Acknowledgments

Zhiyuan Liu and Maosong Sun are supported by National Key Basic Research Program of China (973 Program 2014CB340500) and National Natural Science Foundation of China (NSFC No. 62102140). Huanbo Luan is supported by the National Natural Science Foundation of China (NSFC No. 61303075). This research is also supported by the Singapore National Research Foundation under its International Research Centre@Singapore Funding Initiative and administered by the IDM Programme.

References

Eneko Agirre, Enrique Alfonseca, Keith Hall, Jana Kravalova, Marius Pasca, and Aitor Soroa. 2009. A study on similarity and relatedness using distributional and wordnet-based approaches. In *Proceedings of HLT-NAACL*, pages 19–27.

Yoshua Bengio, Holger Schwenk, Jean-Sebastien Senecal, Frederic Morin, and Jean-Luc Gauvain. 2006. Neural probabilistic language models. In *Innovations in Machine Learning*, pages 137–186.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Proceedings of NIPS*, pages 2787–2795.

Elia Bruni, Nam-Khanh Tran, and Marco Baroni. 2014. Multimodal distributional semantics. *JAIR*, 49:1–47.

Xinxiong Chen, Zhiyuan Liu, and Maosong Sun. 2014. A unified model for word sense representation and disambiguation. In *Proceedings of EMNLP*, pages 1025–1035.

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

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2001. Placing search in context: The concept revisited. In *Proceedings of WWW*, pages 406–414. ACM.

Eric H Huang, Richard Socher, Christopher D Manning, and Andrew Y Ng. 2012. Improving word representations via global context and multiple word prototypes. In *Proceedings of ACL*, pages 873–882.

Daniel D Lee and H Sebastian Seung. 1999. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of AAAI*.

Chuan-bi Lin. 2007. Projected gradient methods for nonnegative matrix factorization. *Neural computation*, 19(10):2756–2779.

Tomas Mikolov, Stefan Kombrink, Lukas Burget, Jan Honza Cernocky, and Sanjeev Khudanpur. 2011. Extensions of recurrent neural network language model. In *Proceedings of ICASSP*, pages 5528–5531. IEEE.

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

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

Brian Murphy, Partha Pratim Talukdar, and Tom M Mitchell. 2012. Learning effective and interpretable semantic models using non-negative sparse embedding. In *Proceedings of COLING*, pages 1933–1950.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. *Proceedings of EMNLP*, 12:1532–1543.

Herbert Rubenstein and John B Goodenough. 1965. Contextual correlates of synonymy. *Communications of the ACM*, 8(10):627–633.

Xu Sun, Houfeng Wang, and Wenjie Li. 2012. Fast online training with frequency-adaptive learning rates for chinese word segmentation and new word detection. In *Proceedings of ACL*, pages 253–262.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of ACL*, pages 384–394.

Yu Zhao, Zhiyuan Liu, and Maosong Sun. 2015. Phrase type sensitive tensor indexing model for semantic composition. In *Proceedings of AAAI*.

A Strong Lexical Matching Method for the Machine Comprehension Test

Ellery Smith, Nicola Greco, Matko Bošnjak, Andreas Vlachos
Department of Computer Science
University College London
{e.smith,n.greco,m.bosnjak,a.vlachos}@cs.ucl.ac.uk

Abstract

Machine comprehension of text is the overarching goal of a great deal of research in natural language processing. The Machine Comprehension Test (Richardson et al., 2013) was recently proposed to assess methods on an open-domain, extensible, and easy-to-evaluate task consisting of two datasets. In this paper we develop a lexical matching method that takes into account multiple context windows, question types and coreference resolution. We show that the proposed method outperforms the baseline of Richardson et al. (2013), and despite its relative simplicity, is comparable to recent work using machine learning. We hope that our approach will inform future work on this task. Furthermore, we argue that MC500 is harder than MC160 due to the way question answer pairs were created.

1 Introduction

Machine comprehension of text is the central goal in NLP. The academic community has proposed a variety of tasks, such as information extraction (Sarawagi, 2008), semantic parsing (Mooney, 2007) and textual entailment (Androutsopoulos and Malakasiotis, 2010). However, these tasks assess performance on each task individually, rather than on overall progress towards machine comprehension of text.

To this end, Richardson et al. (2013) proposed the Machine Comprehension Test (MCTest), a new challenge that aims at evaluating machine comprehension. It does so through an open-domain multiple-choice question answering task on fictional stories requiring the common sense reasoning typical of a 7-year-old child. It is easy to evaluate as it consists of multiple choice questions. Richardson et al. (2013) also showed how the creation of stories and questions can be crowdsourced efficiently, constructing two datasets for the task, namely MC160 and MC500. In addition, the authors presented a lexical matching baseline which is combined with the textual entailment recognition system BIUTEE (Stern and Dagan, 2011).

In this paper we develop an approach based on lexical matching which we extend by taking into account the type of the question and coreference resolution. These components improve the performance on questions that are difficult to handle with pure lexical matching. When combined with BIUTEE, we achieved 74.27% accuracy on MC160 and 65.96% on MC500, which are significantly better than those reported by Richardson et al. (2013). Despite the simplicity of our approach, these results are comparable with the recent machine learning-based approaches proposed by Narasimhan and Barzilay (2015), Wang et al. (2015) and Sachan et al. (2015).

Furthermore, we examine the types of questions and answers in the two datasets. We argue that some types are relatively simple to answer, partly due to the limited vocabulary used, which explains why simple lexical matching methods can perform well. On the other hand, some questions require understanding of higher level concepts such as those of the story and its characters, and/or require inference. This is still beyond the scope of current NLP systems. However, we believe our analysis will be useful in developing new methods and datasets for the task. To that extent, we will make our code and analysis publicly available.[1]

2 Task description

MCTest is an open-domain multiple-choice question answering task on fictional stories consisting of two datasets, MC160 and MC500. The

[1] http://github.com/elleryjsmith/UCLMCTest

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1693–1698,
Lisbon, Portugal, 17-21 September 2015.

It was a terrible day to live in the zoo again for Pauly. It wasn't a terrible day for Zip, the monkey next to him, or Garth, the giraffe down the sidewalk, or Pat, the alligator in the pond, or for Bam the prairie dog, but it was a terrible day in the monkey cage for Pauly. Pauly didn't feel he belonged in the monkey cage because he wasn't a monkey. He was a sailor who had visited the zoo on vacation and fallen asleep on a bench right before closing time. The zoo worker saw how hairy he was and thought he was a monkey that had escaped from his cage, so they put him in a cage.

1. Where was Pauly when the zoo worker saw him?
A) Looking at the monkeys
B) Sailing on a boat
C) Asleep on a bench
D) Walking down a path

2. Why did Pauly feel he didn't belong in the monkey cage?
A) Because he wasn't a monkey
B) Because he was a zoo worker
C) Because he didn't fall asleep
D) Because he was hairy

Figure 1: An excerpt from story *mc500.train.44*

two datasets contain 160 and 500 stories respectively, with 4 questions per story, and 4 candidate answers per question (Figure 1). All stories and questions were crowd-sourced using Amazon Mechanical Turk.[2] MC160 was manually curated by Richardson et al., while MC500 was curated by crowdworkers. Both datasets are divided into training, development, and test sets. All development was conducted on the training and development sets; the test sets were used only to report the final results.

3 Scoring function

Richardson et al. (2013) proposed a sliding window algorithm that ranks the answers by forming the bag-of-words vector of each answer paired with the question text and then scoring them according to their overlap with the story text. We propose a modified version of this algorithm, which combines the scores across a range of window sizes.

More concretely, the algorithm of Richardson et al. (2013) passes a sliding window over the story, size of which is equal to the number of words in the question-answer pair. The highest overlap score between a story text window and the question-answer pair is taken as the score for the answer. Therefore, their algorithm makes a single pass over the story text per answer. In comparison, our system scores each answer by making multiple passes and summing the obtained scores. Concretely, on the first pass, we set the sliding window size to 2 tokens, and increment this size on each subsequent pass, up to a length of 30 tokens. We then combine this score with the overall number of matches of the question-answer pair across the story as a whole. This enables our algorithm to catch long-distance relations in the story. Similar to Richardson et al. (2013), we use a linear combination of this score with their distance-based scoring function, and we weigh tokens with their inverse document frequencies in each individual story.

By itself, this simple enhancement gives substantial improvements over the MSR baseline as shown in Table 1 (*Enhanced SW+D*), as it measures the overlap of the question-answer pair with multiple portions of the story text.

4 Incorporating linguistic analyses

We build upon our enhanced scoring function using stemming, rules taking into account the type of the question, and coreference. The improvements due to each of these components are presented in Table 1, and we discuss the application of coreference and the rules used in more detail in the following subsections.

	MC160	**MC500**
MSR SW+D	69.43%	63.01%
Enhanced SW+D	72.65%	63.57%
+Coreference Rules	+2.38%	+1.36%
+Negation	+0.75%	+0.36%
+Stemming	+2.04%	+0.50%
Final system	**75.77%**	**65.43%**

Table 1: Performance improvements on combined train and dev sets.

4.1 Coreference resolution

The entities mentioned in MCTest stories are frequently referred to by anaphoric expressions throughout the story and the questions, which is ignored by the described scoring function. Therefore, we substituted each mention in a coreference chain with its representative mention, applied the scoring function on the processed text

[2] http://www.mturk.com

and added the score to the original one. The chains and their represenatative mentions were obtained using the Stanford CoreNLP toolkit (Manning et al., 2014). We found that coreference improved performance on some question types, but decreased performance on others. Thus we developed a set of question rules in order to apply it selectively, which we discuss in the next section.

4.2 Rules

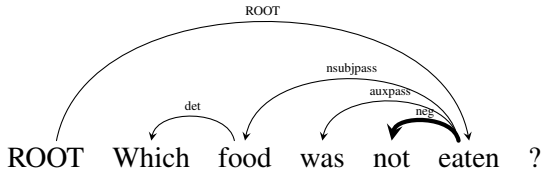


Figure 2: Dependency tree to detect negation

To account for the variety of questions in MCTest, we developed a set of rules to handle certain question types differently. To this purpose, we created rules which detect numerical, temporal, narrative and negation-based questions, and additionally partition questions by their *wh*-word.

By partitioning questions in different types, we found that *who* questions, which primarily deal with identifying a character in the story, benefit from the use of coreference chains described in the previous section. In addition, performance in questions aimed at selecting an appropriate noun, such as *which*, *where* or numerical questions, also improved with coreference. However, other question types, such as *why* questions, or questions concerning the story narrative, did not register any consistent improvement, and we opted not to use co-reference for them. This selective application of co-reference resulted in improvements on both datasets (Table 1).

We also identified negation questions as requiring special treatment. Some negation questions are trivially solvable by selecting an answer which does not appear in the text. However, our proposed function that scores answers according to the lexical overlap with the story text is unlikely to score answers not appearing in text highly. Motivated by this observation we invert the score of each word when a question with a negated root verb was detected, e.g. *"What did James not eat for dinner?"*, using Stanford Typed Dependencies (De Marneffe and Manning, 2008), as depicted in Figure 2. Due to this inversion a higher lexical overlap results in a lower score, improving accuracy on both MC160 and MC500 (+*negation* in Table 1)

In a similar fashion we detected numerical questions based on the presence of a POS tag for a cardinal number in either the question or any of the answers choices. Questions concerning the story's narrative (e.g. *"Which is the first character mentioned in the story"*) were detected using keywords (e.g. *character*, *book*, etc.). Additionally, we detected temporal questions such as *"What did Jane do before she went home?"* by the presence of a temporal modifier or temporal prepositions (e.g. *before*, *while*, etc.). Then we attempted to account for them by searching the text for the sentence indicating that she had gone home and reducing the weight for all subsequent sentences. However, since the improvements due to these rules were negligible, we did not include them in our final system. Nevertheless, these rules were helpful in analyzing problem areas in the datasets, as discussed in Section 6.

5 Results

We evaluated our system on MC160 and MC500 test sets and the results are shown in Table 2. Our proposed baseline outperforms the baseline of Richardson et al. (2013) by 4 and 3 points in accuracy on MC160 and MC500 respectively.[3] Our system is comparable to the MSR baseline with the RTE system BIUTEE (Stern and Dagan, 2011). If we linearly combine the RTE scores used in the MSR baseline with our method, we achieve 5 and 2.5 accuracy points higher than the best results achieved by Richardson et al. (2013).

Concurrently with ours, three other approaches to solving MCTest were developed and subsequently published a few months before our method. Narasimhan and Barzilay (2015) presented a discourse-level approach, which chooses an answer by utilising relations between sentences chosen as important. Despite is simplicity, our method is comparable in performance, suggesting that better lexical matching could help improve their model. Sachan et al. (2015) treated MCTest as a structured prediction problem, searching for a latent structure connecting the question, answer and the text, dubbed the answer-entailing structure. Their model performs better on MC500 (was

[3]We consider the updated MSR algorithms and results, together with partial credit accuracies, provided at `http://research.microsoft.com/en-us/um/redmond/projects/mctest/results.html`

	MC160	MC500
MSR SW+D	68.02%	59.93%
Final system	72.19%	62.67%*
MSR SW+D + RTE	69.27%	63.33%
Final system + RTE	74.27%*	65.96%*
Narasimhan & Barzilay	73.23%	63.75%
Sachan et al.	-	67.83%
Wang et al.	75.27%	69.94%

Table 2: Performance on the MC160 and MC500 test sets, including the results of all previous work. * denotes statistically significant ($p < 0.05$) improvement using McNemar's test, with respect to the MSR baseline (SW+D)

not tested on MC160), however the strength of our model is obtaining comparable results with a much simpler model. The work of Wang et al. (2015) is the most similar to ours, in the sense that they combine a baseline feature set with more advanced linguistic analyses, namely syntax, frame semantics, coreference, and word embeddings. Instead of a rule-based approach, they combine them through a latent-variable classifier achieving the current state-of-the-art performance on MCTest.

6 Discussion

Using the question-filtering rules mentioned in Section 4.2, we obtained individual accuracy scores per question type for the final system combined with RTE (Table 3). Note that these types are in three groups: i) wh-word questions (disjoint, questions without an wh-word are in *Other*), ii) classes of questions requiring non-trivial linguistic analysis and reasoning (not disjoint, not all questions considered), and iii) questions originally classified by crowdworkers, classifying whether the question can be answered by a single or multiple sentences in the story (disjoint).

Compared to the wh-word question type results of Narasimhan and Barzilay (2015), our approach performs better primarily on *why* questions (72.97% and 80.65% vs. 59.45% and 69.35% on MC160 and MC500 respectively) and slightly better on *how*, *when* and *what* questions on MC500. Additionally, our system is more successful in questions requiring multiple sentences to be answered correctly (70.31% and 63.3%vs. 65.23% and 59.9% on MC160 and MC500 respectively).

If we remove the RTE component from our system, the performance on relatively simple question types such as *what*, *who* and *where* remains practically the same, thus confirming that our approach can handle simple questions well. On the other hand, the performance on *why* questions drops without RTE, thus stressing the need for deeper text understanding.

Category	MC160	MC500
What	76.98% (126)	68.77% (317)
Who	71.43% (28)	58.44% (77)
When	80.00% (5)	100.00% (7)
How	72.62% (21)	50.58% (43)
Which	66.67% (6)	40.00% (25)
Where	91.67% (12)	68.97% (58)
Why	72.97% (37)	80.65% (62)
Whose	-	66.67% (3)
Other	0.00% (5)	25.00% (8)
Negation	53.33% (15)	34.48% (29)
Temporal	58.82% (17)	56.41% (39)
Numerical	69.32% (22)	48.26% (43)
Narrative	81.82% (11)	58.41% (26)
Quantifiers	70.00% (20)	53.38% (37)
Single	78.79% (112)	69.12% (272)
Multi	70.31% (128)	63.34% (328)

Table 3: Performance of our final system + RTE per question type on the test sets. The number of relevant questions is in parentheses.

There are several clear deficiencies in certain question types, particularly in handling negation. These errors provide a broad overview of the cases in which simple lexical techniques are not sufficient to determine the correct answer.

Many numerical questions, particularly in MC500, require the use of simple algebra over story elements, including counting characters and objects, and understanding temporal order. One question even requires calculating the probability of an event occurring, while another one calls for complex volumetric calculation. Answering questions such as these is beyond the capabilities of a lexical algorithm, and accuracies in this category are worse than on all questions. Additionally, lexical algorithms such as ours, which ignore predicate-argument structure, perform worse in the presence of quantifiers.

In MC500, the performance of our system on more abstract questions, concerning the overall narrative of the story, also demonstrates a sig-

nificant inadequacy of lexical-based algorithms. Questions such as "*What was the first character mentioned in the story?*", which relate to the overall narrative flow of the passage, or questions concerning the state of the story environment, such as "*Where is the story set?*", are difficult to solve without a system which understands the concept of a story. Typical question-answering methods would also struggle here.

Another type of challenging question are those which require an implicit temporal understanding of the text, i.e. questions concerning time without using a temporal modifier. For example, given a story which states that "*John is at the beach*", then later "*John went home*", a question such as "*What did John do at home?*" would prove itself difficult for traditional methods to answer. These questions are difficult to identify automatically by the form of the question alone, thus we cannot provide accuracies for them.

Our results confirm that it is easier to achieve better performance on MC160 with simple lexical techniques, while the MC500 has proved more resilient to the same improvements. We also observed that the MC500 registers smaller improvements in accuracy when adding components such as co-reference. This is a consequence of the design and curation process of the MC500 dataset, which stipulated that answers must not be contained directly within the story text, or if they are, that two or more misleading choices included.

Richardson et al. (2013) demonstrate that the MC160 and MC500 have similar ratings for clarity and grammar, and that humans perform equally well on both. However, in many cases MC500 appears to be designed in such a way to confuse lexical algorithms and encourage the use of more sophisticated techniques necessary to deal with phenomena such as elimination questions, negation, and common knowledge not explicitly written in the story.

7 Related work

The use of shallow methods for machine comprehension has been explored in previous work, for example Hirschman et al. (1999) used a bag-of-words to match question-answer pairs to sentences in the text, and choose the best pair with the best matching sentence. As discussed in our analysis, such systems cannot handle well questions involving negation and quantification. Numerical questions, which we found to be particularly challenging, have been the focus of recent work on algebra word problems (Kushman et al., 2014) for which dedicated systems have been developed.

MacCartney et al. (2006) demonstrated that a large set of rules can be used to recognize valid textual entailments. These consider phenomena such as polarity and quantification, similar to those we used in our analysis of the MCTest datasets. More complex methods, which attempt deeper modeling of text include Natural Logic (Angeli and Manning, 2014) and Combinatorial Categorial Grammars (Lewis and Steedman, 2013) combined with distributional models. While promising, these approaches have been developed primarily on sentence-level tasks, thus the stories in MCTest are likely to present additional challenges.

The recently proposed class of methods called Memory Network (Weston et al., 2014), uses neural networks and external memory to answer a simpler comprehension task. Though quite successful on toy tasks, those methods cannot yet be applied to MCTest as they require much larger training datasets than the ones available for this task.

A recent approach by Hermann et al. (2015) uses attention-based recurrent neural networks to attack the problem of machine comprehension. In this work, the authors show how to generate large amounts of data for machine comprehension exploiting news websites, and how to use novel architectures in deep learning to solve the task. However, due to the need for a large dataset for training, and the focus only on questions that take entities as answers, this approach has not been applied to MCTest.

8 Conclusion

In this paper we developed an approach to MCTest that combines lexical matching with simple linguistic analysis. We evaluated it on the two MCTest datasets, MC160 and MC500, and we showed that it improves upon the original baseline by 4 and 3 percentage points respectively, while being comparable to more complex approaches. In addition, our analysis highlighted the challenges involved and in particular in the MC500 dataset.

Acknowledgments

We would like to thank Tom Brown for his contributions in the early stages of this work.

References

Ion Androutsopoulos and Prodromos Malakasiotis. 2010. A survey of paraphrasing and textual entailment methods. *Journal of Artificial Intelligence Research*, pages 135–187.

Gabor Angeli and Christopher D Manning. 2014. Naturalli: Natural logic inference for common sense reasoning. In *Proceedings of EMNLP*.

Marie-Catherine De Marneffe and Christopher D Manning. 2008. Stanford typed dependencies manual. *URL http://nlp. stanford. edu/software/dependencies manual. pdf*.

Karl M. Hermann, Tomáš Kočiský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching Machines to Read and Comprehend. *ArXiv e-prints*.

Lynette Hirschman, Marc Light, Eric Breck, and John D Burger. 1999. Deep read: A reading comprehension system. In *Proceedings of ACL*, pages 325–332.

Nate Kushman, Yoav Artzi, Luke Zettlemoyer, and Regina Barzilay. 2014. Learning to automatically solve algebra word problems. In *Proceedings of ACL*, pages 271–281.

Mike Lewis and Mark Steedman. 2013. Combined distributional and logical semantics. *TACL*, 1:179–192.

Bill MacCartney, Trond Grenager, Marie-Catherine de Marneffe, Daniel Cer, and Christopher D Manning. 2006. Learning to recognize features of valid textual entailments. In *Proceedings of the HLT-NAACL*, pages 41–48.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of ACL: System Demonstrations*, pages 55–60.

Raymond J Mooney. 2007. Learning for semantic parsing. In *Computational Linguistics and Intelligent Text Processing*, pages 311–324.

Karthik Narasimhan and Regina Barzilay. 2015. Machine comprehension with discourse relations. In *Proceedings of ACL*.

Matthew Richardson, Christopher JC Burges, and Erin Renshaw. 2013. MCTest: A Challenge Dataset for the Open-Domain Machine Comprehension of Text. In *Proceedings of EMNLP*, pages 193–203.

Mrinmaya Sachan, Avinava Dubey, Eric P Xing, and Matthew Richardson. 2015. Learning answer-entailing structures for machine comprehension. In *Proceedings of ACL*.

Sunita Sarawagi. 2008. Information extraction. *Foundations and trends in databases*, 1(3):261–377.

Asher Stern and Ido Dagan. 2011. A Confidence Model for Syntactically-Motivated Entailment Proofs. In *Proceedings of Recent Advances in Natural Language Processing, (RANLP)*, pages 455–462.

Hai Wang, Mohit Bansal, Kevin Gimpel, and David McAllester. 2015. Machine comprehension with syntax, frames, and semantics. In *Proceedings of ACL: Short papers*.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2014. Memory networks. *arXiv preprint arXiv:1410.3916*.

Broad-coverage CCG Semantic Parsing with AMR

Yoav Artzi*
Dept. of Computer Science and Cornell Tech
Cornell University
New York, NY 10011
yoav@cs.cornell.edu

Kenton Lee **Luke Zettlemoyer**
Computer Science & Engineering
University of Washington
Seattle, WA 98195
{kentonl, lsz}@cs.washington.edu

Abstract

We propose a grammar induction technique for AMR semantic parsing. While previous grammar induction techniques were designed to re-learn a new parser for each target application, the recently annotated AMR Bank provides a unique opportunity to induce a single model for understanding broad-coverage newswire text and support a wide range of applications. We present a new model that combines CCG parsing to recover compositional aspects of meaning and a factor graph to model non-compositional phenomena, such as anaphoric dependencies. Our approach achieves 66.2 Smatch F1 score on the AMR bank, significantly outperforming the previous state of the art.

1 Introduction

Semantic parsers map sentences to formal representations of their meaning (Zelle and Mooney, 1996; Zettlemoyer and Collins, 2005; Liang et al., 2011). Existing learning algorithms have primarily focused on building actionable meaning representations which can, for example, directly query a database (Liang et al., 2011; Kwiatkowski et al., 2013) or instruct a robotic agent (Chen, 2012; Artzi and Zettlemoyer, 2013b). However, due to their end-to-end nature, such models must be relearned for each new target application and have only been used to parse restricted styles of text, such as questions and imperatives.

Recently, AMR (Banarescu et al., 2013) was proposed as a general-purpose meaning representation language for broad-coverage text, and work is ongoing to study its use for variety of applications such as machine translation (Jones et al., 2012) and summarization (Liu et al., 2015). The AMR meaning bank provides a large new corpus that, for the first time, enables us to study the problem of grammar induction for broad-coverage semantic parsing. However, it also presents significant challenges for existing algorithms, including much longer sentences, more complex syntactic phenomena and increased use of non-compositional semantics, such as within-sentence coreference. In this paper, we introduce a new, scalable Combinatory Categorial Grammar (CCG; Steedman, 1996, 2000) induction approach that solves these challenges with a learned joint model of both compositional and non-compositional semantics, and achieves state-of-the-art performance on AMR Bank parsing.

We map sentences to AMR structures in a two-stage process (Section 5). First, we use CCG to construct lambda-calculus representations of the compositional aspects of AMR. CCG is designed to capture a wide range of linguistic phenomena, such as coordination and long-distance dependencies, and has been used extensively for semantic parsing. To use CCG for AMR parsing we define a simple encoding for AMRs in lambda calculus, for example, as seen with the logical form z and AMR a in Figure 1 for the sentence *Pyongyang officials denied their involvement*. However, using CCG to construct such logical forms requires a new mechanism for non-compositional reasoning, for example to model the long-range anaphoric dependency introduced by *their* in Figure 1.

To represent such dependencies while maintaining a relatively compact grammar, we follow Steedman's (2011) use of generalized Skolem terms, a mechanism to allow global references in lambda calculus. We then allow the CCG derivation to mark when non-compositional reasoning is required with underspecified placeholders. For example, Figure 1 shows an underspecified logical form u that would be constructed by the grammar with the bolded placeholder ID indicating an un-

*Work done at the University of Washington.

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1699–1710,
Lisbon, Portugal, 17-21 September 2015.

resolved anaphoric reference. These placeholders are resolved by a factor graph model that is defined over the output logical form and models which part of it they refer to, for example to find the referent for a pronoun. Although primarily motivated by non-compositional reasoning, we also use this mechanism to underspecify certain relations during parsing, allowing for more effective search.

Following most work in semantic parsing, we consider two learning challenges: grammar induction, which assigns meaning representations to words and phrases, and parameter estimation, where we learn a model for combining these pieces to analyze full sentences. We introduce a new CCG grammar induction algorithm which incorporates ideas from previous algorithms (Zettlemoyer and Collins, 2005; Kwiatkowski et al., 2010) in a way that scales to the longer sentences and more varied syntactic constructions observed in newswire text. During lexical generation (Section 6.1), the algorithm first attempts to use a set of templates to hypothesize new lexical entries. It then attempts to combine bottom-up parsing with top-down recursive splitting to select the best entries and learn new templates for complex syntactic and semantic phenomena, which are re-used in later sentences to hypothesize new entries.

Finally, while previous algorithms (e.g., Zettlemoyer and Collins, 2005) have assumed the existence of a grammar that can parse nearly every sentence to update its parameters, this does not hold for AMR Bank. Due to sentence complexity and search errors, our model cannot produce fully correct logical forms for a significant portion of the training data. To learn from as much of the data as possible and accelerate learning, we adopt an early update strategy to generate effective updates from partially correct analyses (Section 6.2).

We evaluate performance on the publicly available AMR Bank (Banarescu et al., 2013) and demonstrate that our modeling and learning contributions are crucial for grammar induction at this scale and achieve new state-of-the-art results for AMR parsing (Section 8). In addition, we also present, for the first time, results without surface-form alignment heuristics, which demonstrates the need for future work, especially to generalize to other languages. The source code and learned models are available online.[1]

[1] http://yoavartzi.com/amr

x: Pyongyang officials denied their involvement.

a:
$(d/\text{deny-01}$
$\quad :\text{ARG0}\ (p/\text{person}$
$\quad\quad :\text{ARG0-of}\ (h/\text{have-org-role-91}$
$\quad\quad\quad :\text{ARG1}\ (c/\text{city}$
$\quad\quad\quad\quad :\text{name}\ (n/\text{name}\ :\text{op1}\text{“Pyongyang”}))$
$\quad\quad\quad :\text{ARG2}(o/\text{official})))$
$\quad :\text{ARG1}\ (i/\text{involve-01}\ :\text{ARG1}\ p))$

u:
$\mathcal{A}_1(\lambda d.\text{deny-01}(d) \wedge$
$\quad \text{ARG0}(d, \mathcal{A}_2(\lambda p.\text{person}(p) \wedge$
$\quad\quad \textbf{REL-of}(p, \mathcal{A}_3(\lambda h.\text{have-org-role-91}(h) \wedge$
$\quad\quad\quad \text{ARG1}(h, \mathcal{A}_4(\lambda c.\text{city}(c) \wedge$
$\quad\quad\quad\quad \text{name}(c, \mathcal{A}_5(\lambda n.\text{name}(n) \wedge$
$\quad\quad\quad\quad\quad \text{op1}(n, \text{PYONGYANG})))))) \wedge$
$\quad\quad\quad \textbf{REL}(h, \mathcal{A}_6(\lambda o.\text{official}(o)))))) \wedge$
$\quad \text{ARG1}(d, \mathcal{A}_7(\lambda i.\text{involve-01}(i) \wedge$
$\quad\quad \text{ARG1}(i, \mathcal{R}(\textbf{ID})))))$

z:
$\mathcal{A}_1(\lambda d.\text{deny-01}(d) \wedge$
$\quad \text{ARG0}(d, \mathcal{A}_2(\lambda p.\text{person}(p) \wedge$
$\quad\quad \text{ARG0-of}(p, \mathcal{A}_3(\lambda h.\text{have-org-role-91}(h) \wedge$
$\quad\quad\quad \text{ARG1}(h, \mathcal{A}_4(\lambda c.\text{city}(c) \wedge$
$\quad\quad\quad\quad \text{name}(c, \mathcal{A}_5(\lambda n.\text{name}(n) \wedge$
$\quad\quad\quad\quad\quad \text{op1}(n, \text{PYONGYANG})))))) \wedge$
$\quad\quad\quad \text{ARG2}(h, \mathcal{A}_6(\lambda o.\text{official}(o)))))) \wedge$
$\quad \text{ARG1}(d, \mathcal{A}_7(\lambda i.\text{involve-01}(i) \wedge$
$\quad\quad \text{ARG1}(i, \mathcal{R}(2)))))$

Figure 1: A sentence (x) paired with its AMR (a), underspecified logical form (u), which contains underspecified constants in bold that are mapped to AMR relations to generate the fully specified logical form (z).

2 Technical Overview

Task Let $\mathcal{X}$ be the set of all possible sentences and A the set of all AMR structures. Given a sentence $x \in \mathcal{X}$, we aim to generate an AMR $a \in A$. We define a simple, deterministic and invertible conversion process between AMRs and lambda-calculus logical forms; roughly speaking, each AMR variable gets its own lambda term, which is scoped as low as possible, and each AMR role becomes a binary predicate applied to these variables. Figure 1 shows an example, and the full details are provided in the supplementary materials. Therefore, henceforth we discuss the task of mapping a sentence $x \in \mathcal{X}$ to a logical form $z \in \mathcal{Z}$, where $\mathcal{Z}$ is the set of all logical forms. For example, in Figure 1, we would map the sentence x to the logical form z. We evaluate system performance using SMATCH (Cai and Knight, 2013).

Model Given a sentence x and lexicon Λ, we generate the set of possible derivations $\text{GEN}(x, \Lambda)$ using a two-stage process (Section 5). First, we use a weighted CCG to map x to an *underspecified logical form* u (Section 5.1), a logical form with placeholder constants for unresolved elements. For example, in the underspecified logical form u in Figure 1, the constants REL-of, REL and ID are placeholders. We then resolve

these placeholders by defining a factor graph to find their optimal mapping and generate the final logical form z. In the figure, REL-of is mapped to ARG0-of, REL to ARG2 and ID to 2.

Learning We assume access to a training set of N examples $\{(x_i, z_i) : i = 1 \dots N\}$, each containing a sentence x_i and a logical form z_i. Our goal is to learn a CCG, which constitutes learning the lexicon and estimating the parameters of both the grammar and the factor graph. We define a learning procedure (Section 6) that alternates between expanding the lexicon and updating the parameters. Learning new lexical entries relies on a two-pass process that combines learning the meaning of words and new syntactic structures, and supports learning with and without alignment heuristics (e.g., from Flanigan et al., 2014).

3 Related Work

The problem of learning semantic parsers has received significant attention. Algorithms have been developed for learning from different forms of supervision, including logical forms (Wong and Mooney, 2007; Muresan, 2011), question-answer pairs (Clarke et al., 2010; Liang et al., 2011; Cai and Yates, 2013; Kwiatkowski et al., 2013), sentences paired with demonstrations (Goldwasser and Roth, 2011; Chen and Mooney, 2011), conversational logs (Artzi and Zettlemoyer, 2011), distant supervision (Krishnamurthy and Mitchell, 2012, 2015; Reddy et al., 2014) and without explicit semantic supervision (Poon, 2013).

Although we are first to consider using CCG to build AMR representations, our work is closely related to existing methods for CCG semantic parsing. Previous CCG induction techniques have either used hand-engineered lexical templates (e.g., Zettlemoyer and Collins, 2005) or learned templates from the data directly (e.g., Kwiatkowski et al., 2010, 2012). Our two-pass reasoning for lexical generation combines ideas from both methods in a way that greatly improves scalability to long, newswire-style sentences. CCG has also been used for broad-coverage recovery of first-order logic representations (Bos, 2008; Lewis and Steedman, 2013). However, this work lacked corpora to evaluate the logical forms recovered.

AMR (Banarescu et al., 2013) is a general-purpose meaning representation and has been used in a number of applications (Pan et al., 2015; Liu et al., 2015). There is also work on recovering AMRs, including graph parsing (Flanigan et al., 2014), methods to build AMRs from dependency trees (Wang et al., 2015) and algorithms for aligning words to AMRs (Pourdamghani et al., 2014).

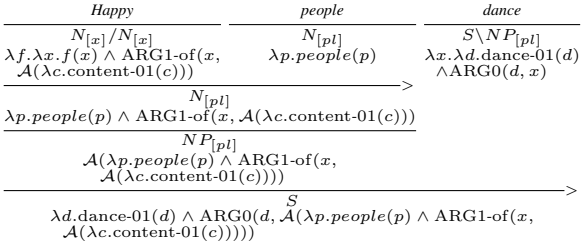


Figure 2: Example CCG tree with three lexical entries, two forward applications (>) and type-shifting of a plural noun to a noun phrase.

4 Background

Combinatory Categorial Grammar CCG is a categorial formalism that provides a transparent interface between syntax and semantics (Steedman, 1996, 2000). Section 7 details our instantiation of CCG. In CCG trees, each node is a category. Figure 2 shows a simple CCG tree. For example, $S\backslash NP_{[pl]} : \lambda x.\lambda d.\text{dance-01}(d) \wedge \text{ARG0}(d, x)$ is a category for an intransitive verb phrase. The syntactic type $S\backslash NP_{[pl]}$ indicates that an argument of type $NP_{[pl]}$ is expected and the returned syntactic type will be S. The backward slash $\backslash$ indicates the argument is expected on the left, while a forward slash $/$ indicates it is expected on the right. The syntactic attribute pl specifies that the argument must be plural. Attribute variables enforce agreement between syntactic attributes. For example, as in Figure 2, adjectives are assigned the syntax $N_{[x]}/N_{[x]}$, where x is used to indicate that the attribute of the argument will determine the attribute of the returned category. The simply-typed lambda calculus logical form in the category represents its semantic meaning. The typing system includes basic types (e.g., entity e, truth value t) and functional types (e.g., $\langle e, t \rangle$ is the type of a function from e to t). In the example category, $\lambda x.\lambda d.\text{dance-01}(d) \wedge \text{ARG0}(d, x)$ is a $\langle e, \langle e, t \rangle \rangle$-typed function expecting an ARG0 argument, and the conjunction specifies the roles of the dance-01 frame.

A CCG is defined by a lexicon and a set of combinators. The lexicon pairs words and phrases with their categorial meaning. For example, $dance \vdash \lambda x.\lambda d.\text{dance-01}(d) \wedge \text{ARG0}(d, x)$ pairs $dance$ with the category above. We adopt a factored representation of the lexicon (Kwiatkowski et al., 2011), where entries are dynamically generated by

combining *lexemes* and *templates*. For example, the above lexical entry can be generated by pairing the lexeme $\langle dance, \{\text{dance-01}\}\rangle$ with the template $\lambda v_1.[S\backslash NP : \lambda x.\lambda a.v_1(a) \wedge \text{ARG0}(a, x)]$.

Skolem Terms and IDs Generalized Skolem terms (henceforth, *Skolem terms*) for CCG were introduced by Steedman (2011) to capture complex dependencies with relatively local quantification. We define here a simplified version of the theory to represent entities and allow distant references. Let $\mathcal{A}$ be a $\langle\langle e, t\rangle, e\rangle$-typed predicate. Given a $\langle e, t\rangle$-typed logical expression C, the logical form $\mathcal{A}_n(C)$ is a Skolem term with the Skolem ID n. For example, $\mathcal{A}_2(\lambda y.\text{boy}(y))$ is a Skolem term that could represent the noun phrase *the boy*, which introduces a new entity. Skolem IDs are globally scoped, i.e., they can be referred from anywhere in the logical form without scoping constraints. To refer to Skolem terms, we define the $\langle id, e\rangle$-typed predicate $\mathcal{R}$. For example, the sentence *the boy loves himself* may be represented with $\mathcal{A}_1(\lambda x.\text{love-01}(x) \wedge \text{ARG0}(x, \mathcal{A}_2(\lambda y.\text{boy}(y))) \wedge \text{ARG1}(x, \mathcal{R}(2)))$, where $\mathcal{R}(2)$ references $\mathcal{A}_2(\lambda y.\text{boy}(y))$.

5 Mapping Sentences to Logical Form

Given a sentence x and lexicon Λ, the function $\textsc{Gen}(x, \Lambda)$ defines the set of possible derivations. Each derivation d is a tuple $\langle y, \mathcal{M}\rangle$, where y is a CCG parse tree and $\mathcal{M}$ is a mapping of constants from u, the *underspecified logical form* at the root of y, to their fully specified form.

5.1 Underspecified Logical Forms

An underspecified logical form represents multiple logical forms via a mapping function that maps its constants to sets of constants and Skolem IDs. For example, consider the underspecified logical form u at the top of Figure 3b. If, for example, REL can be mapped to manner or ARG2, then the sub-expression $\text{REL}(h, \mathcal{A}_6(\lambda o.\text{official}(o)))$ represents $\text{manner}(h, \mathcal{A}_6(\lambda o.\text{official}(o)))$ or $\text{ARG2}(h, \mathcal{A}_6(\lambda o.\text{official}(o)))$. During learning, we assume access to fully specified logical forms, which we convert to underspecified form as needed. In practice, all binary relations, except ARG0 and ARG1, and all Skolem ID references are underspecified.

Formally, let $\mathcal{C}$ be the set of all constants and $\mathcal{I}(u)$ the set of all Skolem IDs in the logical form u. Let $\mathcal{S}_u : \mathcal{C} \rightarrow 2^{\mathcal{C} \cup \mathcal{I}(u)}$ be a specification function, such that its inverse is deterministic. We call a constant c a *placeholder* if $|\mathcal{S}_u(c)| > 1$. Given an underspecified logical form u, applying $\mathcal{S}_u$ to all constants u contains, generates a set of fully specified logical forms.

We define $\mathcal{S}_u$ to be (a) $\mathcal{S}_u(\text{ID}) = \mathcal{I}(u)$, the set of Skolem IDs in u, (b) $\mathcal{S}_u(\text{REL}) = \{\text{part}, \text{ARG2}, \dots\}$, all 67 active AMR relations except ARG0 and ARG1, (c) $\mathcal{S}_u(\text{REL-of}) = \{\text{part-of}, \text{ARG0-of}, \dots\}$, all 33 passive relations, and otherwise (d) $\mathcal{S}_u(c) = c$. For example, in u in Figure 3b, the set of assignments to the ID placeholder is $\mathcal{I}(u) = \{1, 2, 3, 4, 5, 6, 7\}$.

5.2 Derivations

The first part of a derivation $d = \langle y, \mathcal{M}\rangle$ is a CCG parse tree y with an underspecified logical form u at its root. For example, Figure 3a shows such a CCG parse tree, where the logical form contains the placeholders REL, REL-of and ID.

The second part of the derivation is a function $\mathcal{M} : \textsc{Consts}(u) \rightarrow \mathcal{C} \cup \mathcal{I}(u)$, where $\textsc{Consts}(u)$ is the set of all occurrences of constants in u. For example, in Figure 3b, $\textsc{Consts}(u)$ contains, among others, three different occurrences of ARG1 and one of ID, and $\mathcal{M}$ maps REL to ARG2, REL-of to ARG0-of and ID to the Skolem ID 2. The set of potential assignments for each occurrence of constant c is $\mathcal{S}_u(c)$, and $\mathcal{M}$, which returns a single element for each constant, is a disambiguation of $\mathcal{S}_u$. Applying $\mathcal{M}$ to all constants in u results in the final logical form z.

Decomposing the derivation provides two advantages. First, we are able to defer decisions from the CCG parse to the factor graph, thereby considering fewer hypotheses during parsing and simplifying the computation. Second, we can represent distant references while avoiding the complex parse trees that would have been required to represent these dependencies with scoped variables instead of Skolem IDs.[2]

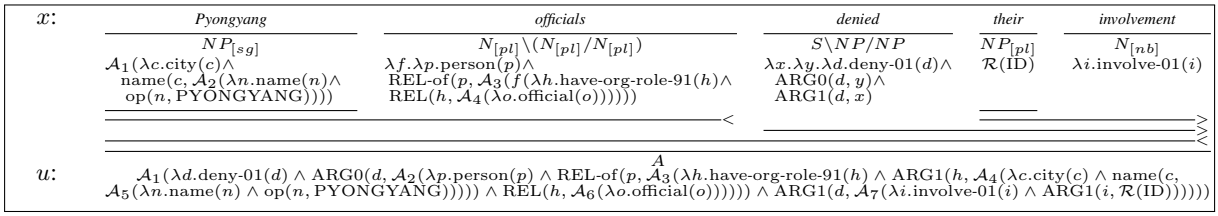
5.3 Model

Given a sentence x, we use a weighted log-linear CCG (Lafferty et al., 2001; Clark and Curran, 2007) to rank the space of possible parses under the grammar Λ. At the root of each CCG derivation is the underspecified logical form u.

To represent a probability distribution over $\mathcal{M}$, we build for each u a factor graph $G_u = \langle V, F, E\rangle$,

[2] Similar to mention clustering methods for co-reference resolution (Ng, 2010), IDs can be viewed as creating clusters.

(a) **CCG parse y:** Maps the sentence x to an underspecified logical form u (Section 5.1) with placeholders for unresolved decisions: ID for reference identifiers and the predicates REL and REL-of for unresolved relations.



(b) **Constant mapping $\mathcal{M}$:** Each constant in u, the logical form at the root of y, is mapped to a Skolem ID or a logical constant to create the fully specified logical form z, which can be converted to an AMR. Only mappings that modify constants are illustrated.

u: $\mathcal{A}_1(\lambda d.\text{deny-01}(d) \wedge \text{ARG0}(d, \mathcal{A}_2(\lambda p.\text{person}(p) \wedge \text{REL-of}(p, \mathcal{A}_3(\lambda h.\text{have-org-role-91}(h) \wedge \text{ARG1}(h, \mathcal{A}_4(\lambda c.\text{city}(c) \wedge \text{name}(c, \mathcal{A}_5(\lambda n.\text{name}(n) \wedge \text{op}(n, \text{PYONGYANG}))))) \wedge \text{REL}(h, \mathcal{A}_6(\lambda o.\text{official}(o)))))) \wedge \text{ARG1}(d, \mathcal{A}_7(\lambda i.\text{involve-01}(i) \wedge \text{ARG1}(i, \mathcal{R}(\text{ID})))))))$

z: $\mathcal{A}_1(\lambda d.\text{deny-01}(d) \wedge \text{ARG0}(d, \mathcal{A}_2(\lambda p.\text{person}(p) \wedge \text{ARG0-of}(p, \mathcal{A}_3(\lambda h.\text{have-org-role-91}(h) \wedge \text{ARG1}(h, \mathcal{A}_4(\lambda c.\text{city}(c) \wedge \text{name}(c, \mathcal{A}_5(\lambda n.\text{name}(n) \wedge \text{op}(n, \text{PYONGYANG}))))) \wedge \text{ARG2}(h, \mathcal{A}_6(\lambda o.\text{official}(o)))))) \wedge \text{ARG1}(d, \mathcal{A}_7(\lambda i.\text{involve-01}(i) \wedge \text{ARG1}(i, \mathcal{R}(2))))))$

Figure 3: A complete derivation for the sentence *Pyongyang officials denied their involvement*.

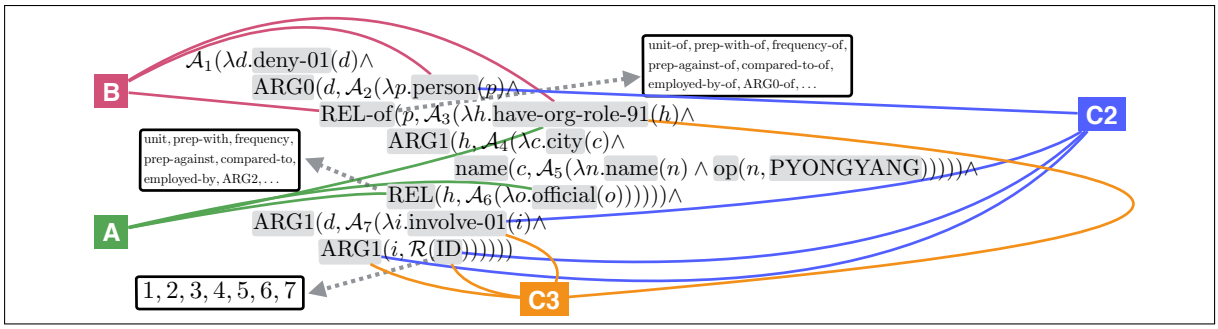


Figure 4: A visualization of the factor graph constructed for the derivation in Figure 3. Variables are marked with gray background. The set of possible assignments, marked with a dashed arrow, is only specified for placeholders (REL-of, REL and ID). Only a subset of the factors are included (A, B, C2 and C3). Solid lines represent edges. Factor A captures selectional preference between the types have-org-role-91 and official to determine the relation REL. Factor B does the same for person and have-org-role-91 to determine REF-of. Factors C2 and C3 account for selectional preferences when resolving ID. In C2, we consider the assignment 2, which will create a relation of type ARG1 between the types involve-01 and person. C3 similarly considers the assignment 3.

where $V = \textsc{Consts}(u)$ is the set of variables, F is the set of factors and E is the set of edges. Each edge is of the form (v, f) where $v \in V$ and $f \in F$. Figure 4 shows the factor graph used in generating the derivation in Figure 3, including all the variables and a subset of the factors. For each variable $v_c \in V$ such that $c \in \textsc{Consts}(u)$ the set of possible assignments is determined by $\mathcal{S}_u(c)$.

To generate the factors F and edges E we use the function $\Phi(V')$ that maps a set of variables $V' \subseteq V$ to a factor f and a set of edges, each one of the form (v, f), where $v \in V'$. Factors express various features (Section 7), such as selectional preferences and control structures. In the figure, Factor A captures the selectional preference for the assignment of the relation REL between have-org-role-91 and official. Factor B captures a similar preference, this time to resolve REL-of. Factor $C2$ captures a selectional preference triplet *involve*-01/ARG1/*person* that will be created if ID is resolved to the Skolem ID 2. Finally, $C3$ captures a similar preference for resolving ID to 3. Since the assignment of many of the variables is fixed, i.e., they are fully specified constants, in practice our factor graph representation simply conditions on them.

Derivations are scored using a log-linear model that includes both CCG parse features and those defined by the factor graph. Let $\mathcal{D}(z)$ be the subset of derivations with the final logical form z and $\theta \in \mathbb{R}^l$ be a l-dimensional parameter vector. We define the probability of the logical form z as

$$p(z|x; \theta, \Lambda) = \sum_{d \in \mathcal{D}(z)} p(d|x; \theta, \Lambda) \ ,$$

and the probability of a derivation d is defined as

$$p(d|x; \theta, \Lambda) = \frac{e^{\theta \cdot \phi(x,d)}}{\sum_{d' \in \textsc{Gen}(x,\Lambda)} e^{\theta \cdot \phi(x,d')}} \ , \qquad (1)$$

where $\phi(x, d) \in \mathbb{R}^l$ is a feature vector (Section 7).

5.4 Inference

To compute the set of derivations $\text{GEN}(x, \Lambda)$ we define a two-stage process. We first run the CCG parser to generate underspecified logical forms. Following previous work (Zettlemoyer and Collins, 2005), we use CKY parsing to enumerate the top-K underspecified logical forms.[3] During the CKY chart construction, we ignore Skolem IDs when comparing categories. This allows us to properly combine partial derivations and to fully benefit from the dynamic programming. We dynamically generate lexical entries for numbers and dates using regular expression patterns and for named-entities using a recognizer. For every underspecified logical form u, we construct a factor graph and use beam search to find the top-L configurations of the graph.[4]

During learning, we use the function $\text{GENMAX}(x, z, \theta, \Lambda)$ to get all derivations that map the sentence x to the logical form z, given parameters θ and lexicon Λ. To compute GENMAX, we follow Zettlemoyer and Collins (2005) and collect constant co-occurrence counts from z to prune from the CKY chart any category that cannot participate in a derivation leading to z. Since only constant names are changed during the second stage, setting the factor graph to get z is trivial: if the underspecified logical form is identical to z except the placeholders, we replace the placeholders with the correct final assignment, otherwise the derivation cannot result in z.

6 Learning

Learning the two-stage model requires inducing the entries of the CCG lexicon Λ and estimating the parameters θ, which score both stages of the derivation. We assume access to a training set of N examples $D = \{(x_i, z_i) : i = 1 \dots N\}$, each containing a sentence x_i and a logical form z_i. This data does not include information about the lexical entries and CCG parsing operations required to construct the correct derivations. We consider all these decisions as latent.

The main learning algorithm (Algorithm 1) starts by initializing the lexicon (line 1) and then processes the data T times (line 2), each time alternating between batch expansion of the lexicon and a sequence of mini-batch parameter updates. An iteration starts with a batch pass to expand the lexicon. The subroutine GENENTRIES, described in Section 6.1 and Algorithm 2, is called to generate a set of new entries for each sample (line 5).

Next, we update the parameters θ with mini-batch updates. Given a mini-batch size of M, we use the procedure $\text{SUB}(D, i, M)$ to get the i-th segment of the data D of size M. We process this segment (line 10) to accumulate the

Algorithm 1 The main learning algorithm.

Input: Training set $D = \{(x_i, z_i) : i = 1 \dots N\}$, number of iterations T, mini-batch size M, seed lexicon Λ_0 and learning rate μ.
Definitions: $\text{SUB}(D, i, j)$ is the set of the next j samples from D starting at i. $\text{GENMAX}(x, z, \theta, \Lambda)$ is the set of viterbi derivations from x with the final result z given parameters θ and lexicon Λ. $\text{LEX}(d)$ is the set of lexical entries used in the derivation d. $\text{COMPUTEGRAD}(x, z, \theta, \Lambda)$ computes the gradient for sentence x and logical form z, given the parameters θ and lexicon Λ, and it described in Section 6.2. $\text{ADAGRAD}(\Delta)$ applies a per-feature learning rate to the gradient Δ (Duchi et al., 2011).
Output: Lexicon Λ and model parameters θ.

1: $\Lambda \leftarrow \Lambda_0$
2: **for** $t = 1$ to T **do**
3: » Generate entries and update the lexicon.
4: **for** $i = 1$ to N **do**
5: $\lambda_{\text{new}} \leftarrow \lambda_{\text{new}} \cup \text{GENENTRIES}(x_i, z_i, \theta, \Lambda)$
6: $\Lambda \leftarrow \Lambda \cup \lambda_{\text{new}}$
7: » Compute and apply mini-batch gradient updates.
8: **for** $i = 1$ to $\lceil \frac{N}{M} \rceil$ **do**
9: $\Delta \leftarrow \vec{0}$
10: **for** (x, z) in $\text{SUB}(D, i, M)$ **do**
11: » Compute and aggregate the gradient.
12: $\Delta \leftarrow \Delta + \text{COMPUTEGRAD}(x, z, \theta, \Lambda)$
13: $\theta \leftarrow \theta + \mu \text{ADAGRAD}(\Delta)$
14: » Get all correct viterbi derivations.
15: $V \leftarrow \bigcup_{(x,z) \in D} \text{GENMAX}(x, z, \theta, \Lambda)$
16: » Retain only entries from derivations in V.
17: $\Lambda \leftarrow \bigcup_{d \in V} \text{LEX}(d)$
18: **return** Λ and θ

Algorithm 2 GENENTRIES: Procedure to generate lexical entries from one training sample. See Section 6.1 for details.

Input: Sample (x, z), model parameters θ and lexicon Λ.
Definitions: $\text{GENLEX}(x, z, \Lambda)$ and $\text{RECSPLIT}(z, z, \theta, \Lambda)$ are defined in Section 6.1.
Output: Set of lexical entries λ.

1: » Augment lexicon with sample-specific entries.
2: $\Lambda_+ \leftarrow \Lambda \cup \text{GENLEX}(x, z, \Lambda)$
3: » Get max-scoring correct derivations.
4: $\mathcal{D}_+ \leftarrow \text{GENMAX}(x, z, \Lambda_+, \theta)$
5: **if** $|\mathcal{D}_+| > 0$ **then**
6: » Return entries from max-scoring derivations.
7: **return** $\bigcup_{d \in \mathcal{D}_+} \text{LEX}(d)$
8: **else**
9: » Top-down splitting to generate new entries.
10: **return** $\text{RECSPLIT}(x, z, \theta, \Lambda_+)$

[3] See Artzi et al. (2014) for a description of this process and how to approximate the partition function in Equation 1.

[4] Experiments with loopy belief propagation showed it to be slower and less effective for our task.

mini-batch gradient Δ by calling the procedure COMPUTEGRAD(x, z, θ, Λ) (line 12), which computes the gradient for x and z given θ and Λ, as described in Section 6.2. We use AdaGrad (Duchi et al., 2011) parameter updates (line 13).

Each iteration concludes with removing all lexical entries not used in max-scoring correct derivations, to correct for overgeneration (lines 14-17).

6.1 Lexicon Expansion: GENENTRIES

Given a sentence x, a logical form z, parameters θ and a lexicon Λ, GENENTRIES(x, z, θ, Λ) (Algorithm 2) computes a set of lexical entries, such that there exists at least one derivation d using these entries from x to z. We first use GENLEX(x, z, Λ) to generate a large set of potential lexical entries from u, the underspecified form of z, by generating lexemes (Section 4) and pairing them with all templates in Λ. We then use a two-pass process to select the entries to return. The set of generated lexemes is a union of: (a) the set G_{gen} that includes all pairings of subsets of constants from z with spans in x up to length k_{gen} and (b) the set that is constructed by matching named-entity constants[5] in z with their corresponding mentions in the text to create new lexemes with potentially any other constant (for lexemes with multiple constants). Λ is augmented with the generated set of lexical entries to create Λ_+ (line 2).

First Pass Given the augmented lexicon Λ_+, we compute the set $\mathcal{D}_+ = \text{GENMAX}(x, z, \theta, \Lambda_+)$ (line 4). Following Artzi and Zettlemoyer (2013b), we constrain the set of derivations to include only those that use at most one lexeme from G_{gen}. If generating new lexemes is sufficient to derive z from x, $\mathcal{D}_+$ will contain these derivations and we return their lexical entries to be added to the lexicon Λ (lines 5-7). Otherwise, we proceed to do a second pass, where we try to generate new templates to parse the sentence.

Second Pass: RECSPLIT In this pass we try to generate max-scoring derivations in a top-down process. Starting from u, the underspecified form of z, we search for CCG parsing steps that will connect to existing partial derivations in the CKY chart to create a complete parse tree. Since the space of possible operations is extremely large, we use CCGBank (Hockenmaier and Steedman, 2007) categories to prune, as described below.

The second pass is executed by calling RECSPLIT$(x, z, \theta, \Lambda_+)$, which returns a set of lexical entries to add to the model (line 10). We recursively apply the splitting operation introduced by Kwiatkowski et al. (2010). Given a CCG category, splitting outputs all possible category pairs that could have originally generated it. For example, given the category $S\backslash NP \vdash \lambda y.\lambda d.\text{deny-01}(d) \wedge \text{ARG0}(d, y) \wedge \text{ARG1}(d, \mathcal{A}_1(\lambda i.\text{involve-01}(i) \wedge \text{ARG1}(i, \mathcal{R}(\text{ID}))))$, one of the possible splits will include the categories $S\backslash NP/NP \vdash \lambda x.\lambda y.\lambda d.\text{deny-01}(d) \wedge \text{ARG0}(d, y) \wedge \text{ARG1}(d, x)$ and $NP \vdash \mathcal{A}_1(\lambda i.\text{involve-01}(i) \wedge \text{ARG1}(i, \mathcal{R}(\text{ID})))$ which would combine with forward application ($>$). Kwiatkowski et al. (2010) present the full details.[6] The process starts from u, the underspecified form of z, and recursively applies the splitting operation while ensuring that: (1) there is at most one entry from G_{gen} or one entry where both the template and lexemes are new in the derivation, (2) each parsing step must have at least one child that may be constructed from an existing partial derivation, and (3) for each new parsing step, the syntax of a newly generated child must match the syntax of a CCGBank category for the same span. To search the space of derivations we populate a CKY chart and do a top-down beam search, where in each step we split categories for smaller spans.

6.2 Gradient Computation: COMPUTEGRAD

Given a sentence x, its labeled logical form z, parameters θ and lexicon Λ, the procedure COMPUTEGRAD(x, z, θ, Λ) computes the gradient for the sample (x, z). Let $\mathcal{D}^*(z) = \text{GENMAX}(x, z, \theta, \Lambda)$, the set of max-scoring correct derivations. The *hard* gradient update is:

$$\frac{1}{|\mathcal{D}^*(z)|} \sum_{d \in \mathcal{D}^*(z)} \phi(x_i, d) - E_{p(d, |x_i; \theta, \Lambda)}[\phi(x_i, d)] \quad , \quad (2)$$

where $\phi(x, d) \in \mathbb{R}^l$ is a l-dimensional feature vector (Section 5.3) and the positive portion of the gradient, rather than using expected features, averages over all max-scoring correct derivations.

Early updates To generate an effective update when no correct derivation is observed, we follow Collins and Roark (2004) and do an early update if $\mathcal{D}^*(z)$ is empty or if GEN(x, Λ), the set

[5] Named-entity constants are created from name instances when converting from AMR to lambda calculus. See the supplementary material for the exact procedure.

[6] Unlike Kwiatkowski et al. (2010), we also introduce syntactic attributes (e.g., pl, sg) when splitting.

of derivations for x, does not contain a derivation with the correct final logical form z. Given the partial derivations, our gradient computation is identical to Equation 2. However, in contrast to Collins and Roark (2004) our data does not include gold derivations. Therefore, we attempt to identify max-scoring partial derivations that may lead to the correct derivation. We extract sub-expressions from u,[7] the underspecified form of z, and search the CKY chart for the top-scoring non-overlapping spans that contain categories with these logical forms. We use the partial derivations leading to these cells to compute the gradient.

The benefit of early updates is two-fold. First, as expected, it leads to higher quality updates that are focused on the errors the model makes. Second, given the complexity of the data, it allows us to have updates for many examples that would be otherwise ignored. In our experiments, we observe this behavior with nearly 40% of the training set.

7 Experimental Setup

Data, Tools and Metric For evaluation, we use AMR Bank release 1.0 (LDC2014T12). We use the proxy report portion, which includes newswire articles from the English Gigaword corpus, and follow the official split for training, development and evaluation (6603/826/823 sentences). We use EasyCCG (Lewis and Steedman, 2014) trained with the re-banked CCGBank (Hockenmaier and Steedman, 2007; Honnibal et al., 2010) to generate CCGBank categories, the Illinois Named Entity Tagger (Ratinov and Roth, 2009) for NER, Stanford CoreNLP (Manning et al., 2014) for tokenization and part-of-speech tagging and UW SPF (Artzi and Zettlemoyer, 2013a) to develop our system. We use SMATCH (Cai and Knight, 2013) to evaluate logical forms converted back to AMRs.

CCG We use three syntactic attributes: singular sg, mass nouns nb and plural pl. When factoring lexical entries, we avoid extracting binary relations and references, and leave them in the template. We use backward and forward binary combinators for application, composition and crossing composition. We allow non-crossing composition up to the third order. We also add rules to handle punctuation and unary rules for type-shifting non-adjectives in adjectival positions and verb phrases in adverbial positions. We allow shifting of bare plurals, mass nouns and named entities to noun phrases. To avoid spurious ambiguity during parsing, we use normal-form constraints (Hockenmaier and Bisk, 2010). We use five basic lambda calculus types: entity e, truth value t, identifier id, quoted text txt and integer i.

Features During CCG parsing, we use indicator features for unary type shifting, crossing composition, lexemes, templates and dynamically generated lexical entries. We also use indicators for co-occurrence of part-of-speech tags and syntactic attributes, repetitions in logical conjunctions and attachments in the logical form. In the factor graph, we use indicator features for control structures, parent-relation-child selectional preferences and for mapping a relation to its final form. See the supplementary material for a detailed description.

Initialization and Parameters We created the seed lexicon from the training data by sampling and annotating 50 sentences with lexical entries, adding entries for pronouns and adding lexemes for all alignments generated by JAMR (Flanigan et al., 2014). We initialize features weights as follows: 10 for all lexeme feature for seed entries and entries generated by named-entity matching (Section 6.1), IBM Model 1 scores for all other lexemes (Kwiatkowski et al., 2011), -3 for unary type shifting and crossing composition features, 3 for features that pair singular and plural part-of-speech tags with singular and plural attributes and 0 for all other features. We set the number of iterations $T = 10$ and select the best model based on development results. We set the max number of tokens for lexical generation $k_{\text{gen}} = 2$, learning rate $\mu = 0.1$, CCG parsing beam $K = 50$, factor graph beam $L = 100$, mini batch size $M = 40$ and use a beam of 100 for GENMAX.

Two-pass Inference During testing, we perform two passes of inference for every sentence. First, we run our inference procedure (Section 5.4). If no derivations are generated, we run inference again, allowing the parser to skip words at a fixed cost and use the entries for *related* words if a word is unknown. We find related words in the lexicon using case, plurality and inflection string transformations. Finally, if necessary, we heuristically transform the logical forms at the root of the CCG parse trees to valid AMR logical forms. We set the cost of logical form transformation and word skipping to 10 and the cost of using related entries to 5.

[7] We extract all sub-expressions of type e, $\langle e, t\rangle$, $\langle e, t\rangle, \langle e, t\rangle\rangle$ or $\langle e, \langle e, t\rangle\rangle$ from u.

8 Results

Table 1 shows SMATCH test results. We compare our approach to the latest, *fixed* version of JAMR (Flanigan et al., 2014) available online,[8] the only system to report test results on the official LDC release. Our approach outperforms JAMR by 3 SMATCH F1 points, with a significant gain in recall. Given consensus inter-annotator agreement of 83 SMATCH F1 (Flanigan et al., 2014), this improvement reduces the gap between automated methods and human performance by 15%. Although not strictly comparable, Table 1 also includes results on the pre-release AMR Bank corpus, including the published JAMR results, their fixed results and the results of Wang et al. (2015).

Table 2 shows SMATCH scores for the developments set, with ablations. The supplementary material includes example output derivations and qualitative comparison to JAMR outputs. We first remove underspecifying constants, which leaves the factor graph to resolve only references. While the expressivity of the model remains the same, more decisions are considered during parsing, modestly impacting performance.

We also study the different methods for lexical generation. Skipping the second recursive splitting pass in GENENTRIES creates an interesting tradeoff. As we are unable to learn templates without splitting, we induce a significantly smaller lexicon (500K vs. 1.6M entries). Although we are unable to recover many syntactic constructions, our search problem is in general much simpler. We therefore see a relatively mild drop in overall performance (1.1 F1). Removing G_{gen} during lexical generation (Section 6.1) creates a more significant drop in performance (3.4 F1), demonstrating how considering all possible lexemes allows the system to recover entries that are not covered by heuristic alignments. We are also able for the first time to report AMR parsing results without any surface-form similarity heuristics, by removing both JAMR alignments and named-entity matching lexical generation (Section 6.1). The significant drop in performance (20 points F1) demonstrates the need for better alignment algorithm.

Finally, Figure 5 plots development SMATCH F1 with and without early updates. As expected, early updates increase the learning rate significantly and have a large impact on overall performance. Without early updates we are unable to learn from almost half of the data, and performance drops by nearly 15 points.

[8] JAMR is available at `http://tiny.cc/jamr`.

	P	R	F1
JAMR (fixed)	67.8	59.2	63.2
Our approach	66.8	65.7	66.3
Pre-release corpus results			
JAMR (Flanigan et al., 2014)	52.0	66.0	58.0
JAMR (fixed)	66.8	58.3	62.3
Wang et al. (2015)	64.0	62.0	63.0

Table 1: Test SMATCH results.

	P	R	F1
Full system	67.2	65.1	66.1
w/o underspecified constants	66.9	64.2	65.5
Lexical learning ablations			
w/o splitting	65.0	65.0	65.0
w/o G_{gen}	62.6	62.7	62.6
w/o surface-form similarity	55.9	38.5	45.6

Table 2: Development SMATCH results.

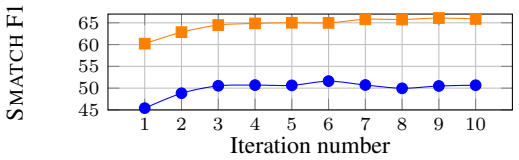


Figure 5: Development SMATCH F1 without early updates (●) and with early updates (■).

9 Conclusion

We described an approach for broad-coverage CCG induction for semantic parsing, including a joint representation of compositional and non-compositional semantics, a new grammar induction technique and an early update procedure. We used AMR as the target representation and present new state-of-the-art AMR parsing results.

While we focused on recovering non-compositional dependencies, other non-compositional phenomena remain to be studied. Although our technique is able to learn certain idioms as multi-word phrases, learning to recognize discontinuous idioms remains open. Similarly, resolving cross-sentence references, which are not annotated in AMR Bank, is important future work. Finally, we would like to reduce the dependency on surface-form heuristics, for example to better generalize to other languages.

Acknowledgements

This research was supported in part by a Microsoft Research PhD Fellowship, the NSF (IIS-1252835), DARPA under the DEFT program through the AFRL (FA8750-13-2-0019), an Allen Distinguished Investigator Award and a gift from Google. The authors thank Mark Yatskar, Tom Kwiatkowski, Chloé Kiddon, Eunsol Choi, Mike Lewis and the reviewers for their helpful advice.

References

Artzi, Y., Das, D., and Petrov, S. (2014). Learning compact lexicons for CCG semantic parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Artzi, Y. and Zettlemoyer, L. S. (2011). Bootstrapping semantic parsers from conversations. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Artzi, Y. and Zettlemoyer, L. S. (2013a). UW SPF: The University of Washington Semantic Parsing Framework.

Artzi, Y. and Zettlemoyer, L. S. (2013b). Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association for Computational Linguistics*, 1(1):49–62.

Banarescu, L., Bonial, C., Cai, S., Georgescu, M., Griffitt, K., Hermjakob, U., Knight, K., Koehn, P., Palmer, M., and Schneider, N. (2013). Abstract meaning representation for sembanking. In *Proceedings of the Linguistic Annotation Workshop*.

Bos, J. (2008). Wide-coverage semantic analysis with Boxer. In *Proceedings of the Conference on Semantics in Text Processing*.

Cai, Q. and Yates, A. (2013). Semantic parsing Freebase: Towards open-domain semantic parsing. In *Proceedings of the Joint Conference on Lexical and Computational Semantics*.

Cai, S. and Knight, K. (2013). Smatch: an evaluation metric for semantic feature structures. In *Proceedings of the Conference of the Association of Computational Linguistics*.

Chen, D. (2012). Fast online lexicon learning for grounded language acquisition. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Chen, D. L. and Mooney, R. J. (2011). Learning to interpret natural language navigation instructions from observations. In *Proceedings of the National Conference on Artificial Intelligence*.

Clark, S. and Curran, J. R. (2007). Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics*, 33(4):493–552.

Clarke, J., Goldwasser, D., Chang, M., and Roth, D. (2010). Driving semantic parsing from the world's response. In *Proceedings of the Conference on Computational Natural Language Learning*.

Collins, M. and Roark, B. (2004). Incremental parsing with the perceptron algorithm. In *Proceedings of the Annual Meeting on Association for Computational Linguistics*.

Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, pages 2121–2159.

Flanigan, J., Thomson, S., Carbonell, J., Dyer, C., and Smith, N. A. (2014). A discriminative graph-based parser for the Abstract Meaning Representation. In *Proceedings of the Conference of the Association of Computational Linguistics*.

Goldwasser, D. and Roth, D. (2011). Learning from natural instructions. In *Proceedings of the International Joint Conference on Artificial Intelligence*.

Hockenmaier, J. and Bisk, Y. (2010). Normal-form parsing for combinatory categorial grammars with generalized composition and type-raising. In *Proceedings of the International Conference on Computational Linguistics*.

Hockenmaier, J. and Steedman, M. (2007). CCGBank: A corpus of CCG derivations and dependency structures extracted from the Penn Treebank. *Computational Linguistics*, pages 355–396.

Honnibal, M., Curran, J. R., and Bos, J. (2010). Rebanking CCGBank for Improved NP Interpretation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Jones, B., Andreas, J., Bauer, D., Hermann, K. M., and Knight, K. (2012). Semantics-based machine translation with hyperedge replacement grammars. In *Proceedings of the International Conference on Computational Linguistics*.

Krishnamurthy, J. and Mitchell, T. (2012). Weakly supervised training of semantic parsers. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*.

Krishnamurthy, J. and Mitchell, T. M. (2015). Learning a compositional semantics for Freebase with an open predicate vocabulary. *Transactions of the Association for Computational Linguistics*, 3.

Kwiatkowski, T., Choi, E., Artzi, Y., and Zettlemoyer, L. S. (2013). Scaling semantic parsers with on-the-fly ontology matching. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Kwiatkowski, T., Goldwater, S., Zettlemoyer, L. S., and Steedman, M. (2012). A probabilistic model of syntactic and semantic acquisition from child-directed utterances and their meanings. *Proceedings of the Conference of the European Chapter of the Association of Computational Linguistics*.

Kwiatkowski, T., Zettlemoyer, L. S., Goldwater, S., and Steedman, M. (2010). Inducing probabilistic CCG grammars from logical form with higher-order unification. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Kwiatkowski, T., Zettlemoyer, L. S., Goldwater, S., and Steedman, M. (2011). Lexical generalization in CCG grammar induction for semantic parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Lafferty, J., McCallum, A., and Pereira, F. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the International Conference on Machine Learning*.

Lewis, M. and Steedman, M. (2013). Combined distributional and logical semantics. *Transactions of the Association for Computational Linguistics*, 1.

Lewis, M. and Steedman, M. (2014). A* CCG parsing with a supertag-factored model. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Liang, P., Jordan, M., and Klein, D. (2011). Learning dependency-based compositional semantics. In *Proceedings of the Conference of the Association for Computational Linguistics*.

Liu, F., Flanigan, J., Thomson, S., Sadeh, N., and Smith, N. A. (2015). Toward abstractive summarization using semantic representations. In *Proceedings of the North American Association for Computational Linguistics*.

Manning, C. D., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S. J., and McClosky, D. (2014). The Stanford CoreNLP natural language processing toolkit. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Muresan, S. (2011). Learning for deep language understanding. In *Proceedings of the International Joint Conference on Artificial Intelligence*.

Ng, V. (2010). Supervised noun phrase coreference research: The first fifteen years. In *Proceedings of the annual meeting of the association for computational linguistics*.

Pan, X., Cassidy, T., Hermjakob, U., Ji, H., and Knight, K. (2015). Unsupervised entity linking with Abstract Meaning Representation. In *Proceedings of the North American Association for Computational Linguistics*.

Poon, H. (2013). Grounded unsupervised semantic parsing. In *Association for Computational Linguistics (ACL)*.

Pourdamghani, N., Gao, Y., Hermjakob, U., and Knight, K. (2014). Aligning English strings with Abstract Meaning Representation graphs. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Ratinov, L. and Roth, D. (2009). Design challenges and misconceptions in named entity recognition. In *Proceedings of the Conference on Computational Natural Language Learning*.

Reddy, S., Lapata, M., and Steedman, M. (2014). Large-scale semantic parsing without question-answer pairs. *Transactions of the Association for Computational Linguistics*, 2.

Steedman, M. (1996). *Surface Structure and Interpretation*. The MIT Press.

Steedman, M. (2000). *The Syntactic Process*. The MIT Press.

Steedman, M. (2011). *Taking Scope*. The MIT Press.

Wang, C., Xue, N., Pradhan, S., and Pradhan, S. (2015). A transition-based algorithm for AMR parsing. In *Proceedings of the North American Association for Computational Linguistics*.

Wong, Y. and Mooney, R. J. (2007). Learning synchronous grammars for semantic parsing with

lambda calculus. In *Proceedings of the Conference of the Association for Computational Linguistics*.

Zelle, J. and Mooney, R. J. (1996). Learning to parse database queries using inductive logic programming. In *Proceedings of the National Conference on Artificial Intelligence*.

Zettlemoyer, L. S. and Collins, M. (2005). Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*.

Semantically Conditioned LSTM-based Natural Language Generation for Spoken Dialogue Systems

Tsung-Hsien Wen, Milica Gašić, Nikola Mrkšić,
Pei-Hao Su, David Vandyke and **Steve Young**
Cambridge University Engineering Department,
Trumpington Street, Cambridge, CB2 1PZ, UK
{thw28,mg436,nm480,phs26,djv27,sjy}@cam.ac.uk

Abstract

Natural language generation (NLG) is a critical component of spoken dialogue and it has a significant impact both on usability and perceived quality. Most NLG systems in common use employ rules and heuristics and tend to generate rigid and stylised responses without the natural variation of human language. They are also not easily scaled to systems covering multiple domains and languages. This paper presents a statistical language generator based on a semantically controlled Long Short-term Memory (LSTM) structure. The LSTM generator can learn from unaligned data by jointly optimising sentence planning and surface realisation using a simple cross entropy training criterion, and language variation can be easily achieved by sampling from output candidates. With fewer heuristics, an objective evaluation in two differing test domains showed the proposed method improved performance compared to previous methods. Human judges scored the LSTM system higher on informativeness and naturalness and overall preferred it to the other systems.

1 Introduction

The natural language generation (NLG) component provides much of the persona of a spoken dialogue system (SDS), and it has a significant impact on a user's impression of the system. As noted in Stent et al. (2005), a good generator usually depends on several factors: adequacy, fluency, readability, and variation. Previous approaches attacked the NLG problem in different ways. The most common and widely adopted today is the *rule-based* (or *template-based*) approach (Cheyer and Guzzoni, 2007; Mirkovic and Cavedon, 2011). Despite its robustness and adequacy, the frequent repetition of identical, rather stilted, output forms make talking to a *rule-based* generator rather tedious. Furthermore, the approach does not easily scale to large open domain systems(Young et al., 2013; Gašić et al., 2014; Henderson et al., 2014). Hence approaches to NLG are required that can be readily scaled whilst meeting the above requirements.

The *trainable generator* approach exemplified by the HALOGEN (Langkilde and Knight, 1998) and SPaRKy system (Stent et al., 2004) provides a possible way forward. These systems include specific trainable modules within the generation framework to allow the model to adapt to different domains (Walker et al., 2007), or reproduce certain style (Mairesse and Walker, 2011). However, these approaches still require a handcrafted generator to define the decision space within which statistics can be used for optimisation. The resulting utterances are therefore constrained by the predefined syntax and any domain-specific colloquial responses must be added manually.

More recently, *corpus-based* methods (Oh and Rudnicky, 2000; Mairesse and Young, 2014; Wen et al., 2015) have received attention as access to data becomes increasingly available. By defining a flexible learning structure, *corpus-based* methods aim to learn generation directly from data by adopting an over-generation and reranking paradigm (Oh and Rudnicky, 2000), in which final responses are obtained by reranking a set of candidates generated from a stochastic generator. Learning from data directly enables the system to mimic human responses more naturally, removes the dependency on predefined rules, and makes the system easier to build and extend to other domains. As detailed in Sections 2 and 3, however, these existing approaches have weaknesses in the areas of training data efficiency, accuracy and naturalness.

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1711–1721,
Lisbon, Portugal, 17-21 September 2015.

This paper presents a statistical NLG based on a semantically controlled Long Short-term Memory (LSTM) recurrent network. It can learn from unaligned data by jointly optimising its sentence planning and surface realisation components using a simple cross entropy training criterion without any heuristics, and good quality language variation is obtained simply by randomly sampling the network outputs. We start in Section 3 by defining the framework of the proposed neural language generator. We introduce the semantically controlled LSTM (SC-LSTM) cell in Section 3.1, then we discuss how to extend it to a deep structure in Section 3.2. As suggested in Wen et al. (2015), a backward reranker is introduced in Section 3.3 to improve fluency. Training and decoding details are described in Section 3.4 and 3.5.

Section 4 presents an evaluation of the proposed approach in the context of an application providing information about venues in the San Francisco area. In Section 4.2, we first show that our generator outperforms several baselines using objective metrics. We experimented on two different ontologies to show not only that good performance can be achieved across domains, but how easy and quick the development lifecycle is. In order to assess the subjective performance of our system, a quality test and a pairwise preference test are presented in Section 4.3. The results show that our approach can produce high quality utterances that are considered to be more natural and are preferred to previous approaches. We conclude with a brief summary and future work in Section 5.

2 Related Work

Conventional approaches to NLG typically divide the task into sentence planning and surface realisation. Sentence planning maps input semantic symbols into an intermediary form representing the utterance, e.g. a tree-like or template structure, then surface realisation converts the intermediate structure into the final text (Walker et al., 2002; Stent et al., 2004). Although statistical sentence planning has been explored previously, for example, generating the most likely context-free derivations given a corpus (Belz, 2008) or maximising the expected reward using reinforcement learning (Rieser and Lemon, 2010), these methods still rely on a pre-existing, handcrafted generator. To minimise handcrafting, Stent and Molina (2009) proposed learning sentence planning rules directly from a corpus of utterances labelled with Rhetorical Structure Theory (RST) discourse relations (Mann and Thompson, 1988). However, the required corpus labelling is expensive and additional handcrafting is still needed to map the sentence plan to a valid syntactic form.

As noted above, *corpus-based* NLG aims at learning generation decisions from data with minimal dependence on rules and heuristics. A pioneer in this direction is the class-based n-gram language model (LM) approach proposed by Oh and Rudnicky (2000). Ratnaparkhi (2002) later addressed some of the limitations of class-based LMs in the over-generation phase by using a modified generator based on a syntactic dependency tree. Mairesse and Young (2014) proposed a phrase-based NLG system based on factored LMs that can learn from a semantically aligned corpus. Although active learning (Mairesse et al., 2010) was also proposed to allow learning online directly from users, the requirement for human annotated alignments limits the scalability of the system. Another similar approach casts NLG as a template extraction and matching problem, e.g., Angeli et al. (2010) train a set of log-linear models to make a series of generation decisions to choose the most suitable template for realisation. Kondadadi et al. (2013) later show that the outputs can be further improved by an SVM reranker making them comparable to human-authored texts. However, template matching approaches do not generalise well to unseen combinations of semantic elements.

The use of neural network-based (NN) approaches to NLG is relatively unexplored. The stock reporter system ANA by Kukich (1987) is perhaps the first NN-based generator, although generation was only done at the phrase level. Recent advances in recurrent neural network-based language models (RNNLM) (Mikolov et al., 2010; Mikolov et al., 2011a) have demonstrated the value of distributed representations and the ability to model arbitrarily long dependencies. Sutskever et al. (2011) describes a simple variant of the RNN that can generate meaningful sentences by learning from a character-level corpus. More recently, Karpathy and Fei-Fei (2014) have demonstrated that an RNNLM is capable of generating image descriptions by conditioning the network model on a pre-trained convolutional image feature representation. Zhang and Lapata (2014) also describes interesting work using RNNs to generate

Chinese poetry. A forerunner of the system presented here is described in Wen et al. (2015), in which a forward RNN generator, a CNN reranker, and a backward RNN reranker are trained jointly to generate utterances. Although the system was easy to train and extend to other domains, a heuristic gate control was needed to ensure that all of the attribute-value information in the system's response was accurately captured by the generated utterance. Furthermore, the handling of unusual slot-value pairs by the CNN reranker was rather arbitrary. In contrast, the LSTM-based system described in this paper can deal with these problems automatically by learning the control of gates and surface realisation jointly.

Training an RNN with long range dependencies is difficult because of the vanishing gradient problem (Bengio et al., 1994). Hochreiter and Schmidhuber (1997) mitigated this problem by replacing the sigmoid activation in the RNN recurrent connection with a self-recurrent memory block and a set of multiplication gates to mimic the read, write, and reset operations in digital computers. The resulting architecture is dubbed the Long Short-term Memory (LSTM) network. It has been shown to be effective in a variety of tasks, such as speech recognition (Graves et al., 2013b), handwriting recognition (Graves et al., 2009), spoken language understanding (Yao et al., 2014), and machine translation (Sutskever et al., 2014). Recent work by Graves et al. (2014) has demonstrated that an NN structure augmented with a carefully designed memory block and differentiable read/write operations can learn to mimic computer programs. Moreover, the ability to train deep networks provides a more sophisticated way of exploiting relations between labels and features, therefore making the prediction more accurate (Hinton et al., 2012). By extending an LSTM network to be both deep in space and time, Graves (2013) shows the resulting network can used to synthesise handwriting indistinguishable from that of a human.

3 The Neural Language Generator

The generation model proposed in this paper is based on a recurrent NN architecture (Mikolov et al., 2010) in which a 1-hot encoding $\mathbf{w}_t$ of a token[1] w_t is input at each time step t conditioned on a recurrent hidden layer $\mathbf{h}_t$ and outputs the probability distribution of the next token w_{t+1}. Therefore, by sampling input tokens one by one from the output distribution of the RNN until a stop sign is generated (Karpathy and Fei-Fei, 2014) or some constraint is satisfied (Zhang and Lapata, 2014), the network can produce a sequence of tokens which can be lexicalised [2] to form the required utterance.

3.1 Semantic Controlled LSTM cell

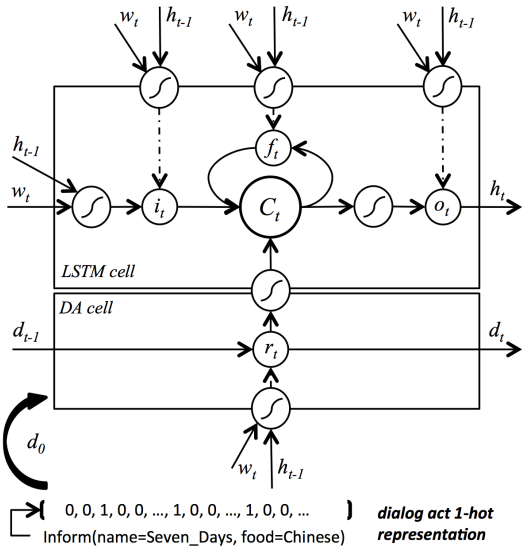


Figure 1: Semantic Controlled LSTM cell proposed in this paper. The upper part is a traditional LSTM cell in charge of surface realisation, while the lower part is a sentence planning cell based on a sigmoid control gate and a dialogue act (DA).

Long Short-term Memory (Hochreiter and Schmidhuber, 1997) is a recurrent NN architecture which uses a vector of memory cells $\mathbf{c}_t \in \mathbb{R}^n$ and a set of elementwise multiplication gates to control how information is stored, forgotten, and exploited inside the network. Of the various different connectivity designs for an LSTM cell (Graves, 2013; Zaremba et al., 2014), the architecture used in this paper is illustrated in Figure 3.1 and defined by the following equations,,

$$\mathbf{i}_t = \sigma(\mathbf{W}_{wi}\mathbf{w}_t + \mathbf{W}_{hi}\mathbf{h}_{t-1}) \quad (1)$$

$$\mathbf{f}_t = \sigma(\mathbf{W}_{wf}\mathbf{w}_t + \mathbf{W}_{hf}\mathbf{h}_{t-1}) \quad (2)$$

$$\mathbf{o}_t = \sigma(\mathbf{W}_{wo}\mathbf{w}_t + \mathbf{W}_{ho}\mathbf{h}_{t-1}) \quad (3)$$

$$\hat{\mathbf{c}}_t = tanh(\mathbf{W}_{wc}\mathbf{w}_t + \mathbf{W}_{hc}\mathbf{h}_{t-1}) \quad (4)$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \hat{\mathbf{c}}_t \quad (5)$$

$$\mathbf{h}_t = \mathbf{o}_t \odot tanh(\mathbf{c}_t) \quad (6)$$

[1] We use *token* instead of *word* because our model operates on text for which slot values are replaced by its corresponding slot tokens. We call this procedure delexicalisation.

[2] The process of replacing slot token by its value.

where σ is the sigmoid function, $\mathbf{i}_t, \mathbf{f}_t, \mathbf{o}_t \in [0,1]^n$ are input, forget, and output gates respectively, and $\hat{\mathbf{c}}_t$ and $\mathbf{c}_t$ are proposed cell value and true cell value at time t. Note that each of these vectors has a dimension equal to the hidden layer $\mathbf{h}$.

In order to ensure that the generated utterance represents the intended meaning, the generator is further conditioned on a control vector $\mathbf{d}$, a 1-hot representation of the dialogue act (DA) type and its slot-value pairs. Although a related work (Karpathy and Fei-Fei, 2014) has suggested that reapplying this auxiliary information to the RNN at every time step can increase performance by mitigating the *vanishing gradient problem* (Mikolov and Zweig, 2012; Bengio et al., 1994), we have found that such a model also omits and duplicates slot information in the surface realisation. In Wen et al. (2015) simple heuristics are used to turn off slot feature values in the control vector $\mathbf{d}$ once the corresponding slot token has been generated. However, these heuristics can only handle cases where slot-value pairs can be identified by exact matching between the delexicalised surface text and the slot value pair encoded in $\mathbf{d}$. Cases such as binary slots and slots that take don't care values cannot be explicitly delexicalised in this way and these cases frequently result in generation errors.

To address this problem, an additional control cell is introduced into the LSTM to gate the DA as shown in Figure 1. This cell plays the role of *sentence planning* since it manipulates the DA features during the generation process in order to produce a surface realisation which accurately encodes the input information. We call the resulting architecture Semantically Controlled LSTM (SC-LSTM). Starting from the original DA 1-hot vector $\mathbf{d}_0$, at each time step the DA cell decides what information should be retained for future time steps and discards the others,

$$\mathbf{r}_t = \sigma(\mathbf{W}_{wr}\mathbf{w}_t + \alpha\mathbf{W}_{hr}\mathbf{h}_{t-1}) \quad (7)$$

$$\mathbf{d}_t = \mathbf{r}_t \odot \mathbf{d}_{t-1} \quad (8)$$

where $\mathbf{r}_t \in [0,1]^d$ is called the reading gate, and α is a constant. Here $\mathbf{W}_{wr}$ and $\mathbf{W}_{hr}$ act like keyword and key phrase detectors that learn to associate certain patterns of generated tokens with certain slots. Figure 3 gives an example of how these detectors work in affecting DA features inside the network. Equation 5 is then modified so that the cell value $\mathbf{c}_t$ also depends on the DA,

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \hat{\mathbf{c}}_t + tanh(\mathbf{W}_{dc}\mathbf{d}_t) \quad (9)$$

After updating Equation 6 by Equation 9, the output distribution is formed by applying a softmax function g, and the distribution is sampled to obtain the next token,

$$P(w_{t+1}|w_t, w_{t-1}, ...w_0, \mathbf{d}_t) = g(\mathbf{W}_{ho}\mathbf{h}_t) \quad (10)$$

$$w_{t+1} \sim P(w_{t+1}|w_t, w_{t-1}, ...w_0, \mathbf{d}_t). \quad (11)$$

3.2 The Deep Structure

Deep Neural Networks (DNN) enable increased discrimination by learning multiple layers of features, and represent the state-of-the-art for many applications such as speech recognition (Graves et al., 2013b) and natural language processing (Collobert and Weston, 2008). The neural language generator proposed in this paper can be easily extended to be deep in both space and time by stacking multiple LSTM cells on top of the original structure. As shown in Figure 2, *skip connections* are applied to the inputs of all hidden layers as well as between all hidden layers and the outputs (Graves, 2013). This reduces the number of processing steps between the bottom of the network and the top, and therefore mitigates the vanishing gradient problem (Bengio et al., 1994) in the vertical direction. To allow all hidden layer information to influence the reading gate, Equation 7 is changed to

$$\mathbf{r}_t = \sigma(\mathbf{W}_{wr}\mathbf{w}_t + \sum_l \alpha_l \mathbf{W}^l_{hr}\mathbf{h}^l_{t-1}) \quad (12)$$

where l is the hidden layer index and α_l is a layer-wise constant. Since the network tends to overfit when the structure becomes more complex, the dropout technique (Srivastava et al., 2014) is used to regularise the network. As suggested in (Zaremba et al., 2014), dropout was only applied to the non-recurrent connections, as shown in the Figure 2. It was not applied to word embeddings since pre-trained word vectors were used.

3.3 Backward LSTM reranking

One remaining problem in the structure described so far is that the LSTM generator selects words based only on the preceding history, whereas some sentence forms depend on the backward context. Previously, bidirectional networks (Schuster and

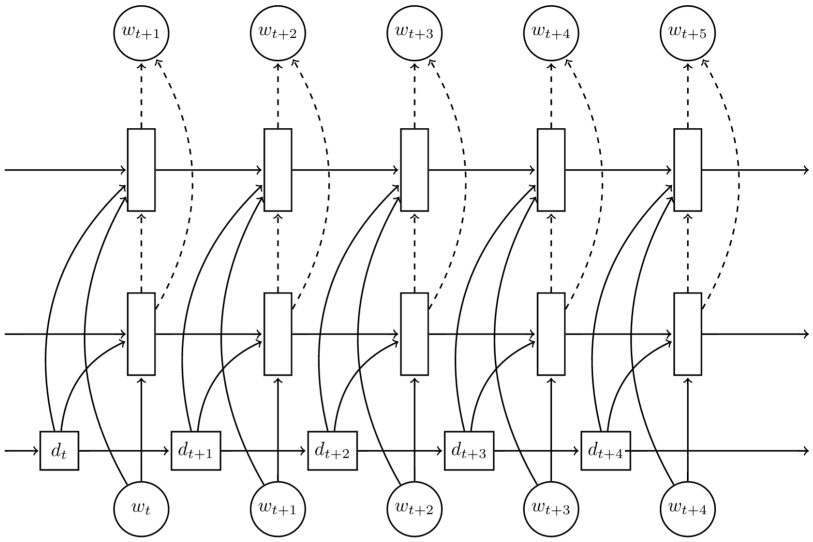


Figure 2: The Deep LSTM generator structure by stacking multiple LSTM layers on top of the DA cell. The skip connection was adopted to mitigate the vanishing gradient, while the dropout was applied on dashed connections to prevent co-adaptation and overfitting.

Paliwal, 1997) have been shown to be effective for sequential problems (Graves et al., 2013a; Sundermeyer et al., 2014). However, applying a bidirectional network directly in the SC-LSTM generator is not straightforward since the generation process is sequential in time. Hence instead of integrating the bidirectional information into one network, we trained another SC-LSTM from backward context to choose best candidates from the forward generator outputs. In our experiments, we also found that by tying the keyword detector weights $\mathbf{W}_{wr}$ (see Equations 7 and 12) of both the forward and backward networks together makes the generator less sensitive to random initialisation.

3.4 Training

The forward generator and the backward reranker were both trained by treating each sentence as a mini-batch. The objective function was the cross entropy error between the predicted word distribution $\mathbf{p}_t$ and the actual word distribution $\mathbf{y}_t$ in the training corpus. An l_2 regularisation term was added to the objective function for every 10 training examples as suggested in Mikolov et al. (2011b). However, further regularisation was required for the reading gate dynamics. This resulted in the following modified cost function for each mini-match (ignoring standard l_2),

$$F(\theta) = \sum_t \mathbf{p}_t^{\mathrm{T}} log(\mathbf{y}_t) + \|\mathbf{d}_T\| + \sum_{t=0}^{T-1} \eta \xi^{\|\mathbf{d}_{t+1} - \mathbf{d}_t\|} \quad (13)$$

where $\mathbf{d}_T$ is the DA vector at the last word index T, and η and ξ are constants set to 10^{-4} and 100, respectively. The second term is used to penalise generated utterances that failed to render all the required slots, while the third term discourages the network from turning more than one gate off in a single time step. The forward and backward networks were structured to share the same set of word embeddings, initialised with pre-trained word vectors (Pennington et al., 2014). The hidden layer size was set to be 80 for all cases, and deep networks were trained with two hidden layers and a 50% dropout rate. All costs and gradients were computed and stochastic gradient descent was used to optimise the parameters. Both networks were trained with back propagation through time (Werbos, 1990). In order to prevent overfitting, early stopping was implemented using a held-out validation set.

3.5 Decoding

The decoding procedure is split into two phases: (a) over-generation, and (b) reranking. In the over-generation phase, the forward generator conditioned on the given DA, is used to sequentially generate utterances by random sampling of the predicted next word distributions. In the reranking phase, the cost of the backward reranker $F_b(\theta)$ is computed. Together with the cost $F_f(\theta)$ from the forward generator, the reranking score R is com-

puted as:

$$R = -(F_f(\theta) + F_b(\theta) + \lambda \text{ERR}) \quad (14)$$

where λ is a tradeoff constant, and the slot error rate ERR is computed by exact matching the slot tokens in the candidate utterances,

$$\text{ERR} = \frac{p+q}{N} \quad (15)$$

where N is the total number of slots in the DA, and p, q is the number of missing and redundant slots in the given realisation. Note that the ERR reranking criteria cannot handle arbitrary slot-value pairs such as binary slots or slots that take the don't care value because they cannot be delexicalised and exactly matched. λ is set to a large value in order to severely penalise nonsensical outputs.

4 Experiments

4.1 Experimental Setup

The target application for our generation system is a spoken dialogue system providing information about certain venues in San Francisco. In order to demonstrate the scalability of the proposed method and its performance in different domains, we tested on two domains that talk about restaurants and hotels respectively. There are 8 system dialogue act types such as *inform* to present information about restaurants, *confirm* to check that a slot value has been recognised correctly, and *reject* to advise that the user's constraints cannot be met. Each domain contains 12 attributes (slots), some are common to both domains and the others are domain specific. The detailed ontologies for the two domains are provided in Table 1. To form a training corpus for each domain, dialogues collected from a previous user trial (Gašić et al., 2015) of a statistical dialogue manager were randomly sampled and shown to workers recruited via the Amazon Mechanical Turk (AMT) service. Workers were shown each dialogue turn by turn and asked to enter an appropriate system response in natural English corresponding to each system DA. For each domain around 5K system utterances were collected from about 1K randomly sampled dialogues. Each categorical value was replaced by a token representing its slot, and slots that appeared multiple times in a DA were merged into one. After processing and grouping each utterance according to its delexicalised DA, we obtained 248 distinct DAs in the restaurant domain and 164 in the hotel domain. The average number of slots per DA for each domain is 2.25 and 1.95, respectively.

	SF Restaurant	SF Hotel
act type	inform, inform_only, reject, confirm, select, request, reqmore, goodbye	
shared	name, type, *pricerange, price, phone, address, postcode, *area, *near	
specific	*food *goodformeal ***kids-allowed**	***hasinternet** ***acceptscards** ***dogs-allowed**

bold=binary slots, *=slots can take "don't care" value

Table 1: Ontologies used in the experiments.

The system was implemented using the Theano library (Bergstra et al., 2010; Bastien et al., 2012), and trained by partitioning each of the collected corpus into a training, validation, and testing set in the ratio 3:1:1. The frequency of each action type and slot-value pair differs quite markedly across the corpus, hence up-sampling was used to make the corpus more uniform. Since our generator works stochastically and the trained networks can differ depending on the initialisation, all the results shown below[3] were averaged over 5 randomly initialised networks. For each DA, we overgenerated 20 utterances and selected the top 5 realisations after reranking. The BLEU-4 metric was used for the objective evaluation (Papineni et al., 2002). Multiple references for each test DA were obtained by mapping them back to the distinct set of DAs, grouping those delexicalised surface forms that have the same DA specification, and then lexicalising those surface forms back to utterances. In addition, the slot error rate (ERR) as described in Section 3.5 was computed as an auxiliary metric alongside the BLEU score. However, for the experiments it is computed at the corpus level, by averaging slot errors over each of the top 5 realisations in the entire corpus. The trade-off weights α between keyword and key phrase detectors as mentioned in Section 3.1 and 3.2 were set to 0.5.

4.2 Objective Evaluation

We compared the single layer semantically controlled LSTM (*sc-lstm*) and a deep version with

[3]Except human evaluation, in which only one set of networks was used.

Method	SF Restaurant		SF Hotel	
	BLEU	ERR(%)	BLEU	ERR(%)
hdc	0.451	0.0	0.560	0.0
kNN	0.602	0.87	0.676	1.87
classlm	0.627	8.70	0.734	5.35
rnn w/o	0.706	4.15	0.813	3.14
lstm w/o	0.714	1.79	0.817	1.93
rnn w/	0.710	1.52	0.815	1.74
lstm w/	0.717	0.63	0.818	1.53
sc-lstm	0.711	0.62	0.802	0.78
+deep	**0.731**	**0.46**	**0.832**	**0.41**

Table 2: Objective evaluation of the top 5 realisations. Except for handcrafted (*hdc*) and k-nearest neighbour (*kNN*) baselines, all the other approaches ranked their realisations from 20 over-generated candidates.

two hidden layers (+*deep*) against several baselines: the handcrafted generator (*hdc*), k-nearest neighbour (*kNN*), class-based LMs (*classlm*) as proposed in Oh and Rudnicky (2000), the heuristic gated RNN as described in Wen et al. (2015) and a similar LSTM variant (*rnn w/* & *lstm w/*), and the same RNN/LSTM but without gates (*rnn w/o* & *lstm w/o*). The handcrafted generator was developed over a long period of time and is the standard generator used for trialling end-to-end dialogue systems (for example (Gašić et al., 2014)). The kNN was implemented by computing the similarity of the test DA 1-hot vector against all of the training DA 1-hot vectors, selecting the nearest and then lexicalising to generate the final surface form. The objective results are shown in Table 2. As can be seen, none of the baseline systems shown in the first block (*hdc*, *kNN*, & *classlm*) are comparable to the systems described in this paper (*sc-lstm* & +*deep*) if both metrics are considered. Setting aside the difficulty of scaling to large domains, the handcrafted generator's (*hdc*) use of predefined rules yields a fixed set of sentence plans, which can differ markedly from the real colloquial human responses collected from AMT, while the class LM approach suffers from inaccurate rendering of information. Although the *kNN* method provides reasonable adequacy i.e. low ERR, the BLEU is low, probably because of the errors in the collected corpus which *kNN* cannot handle but statistical approaches such as LMs can by suppressing unlikely outputs.

The last three blocks in Table 2 compares the proposed method with previous RNN approaches.

Method	**Informativeness**	**Naturalness**
+deep	2.58	**2.51**
sc-lstm	**2.59**	2.50
rnn w/	2.53	2.42^{*}
classlm	2.46^{**}	2.45

$^{*}p < 0.05$ $^{**}p < 0.005$

Table 3: Real user trial for utterance quality assessment on two metrics (rating out of 3), averaging over top 5 realisations. Statistical significance was computed using a two-tailed Student's t-test, between deep and all others.

Pref.%	**classlm**	**rnn w/**	**sc-lstm**	**+deep**
classlm	-	46.0	40.9^{**}	37.7^{**}
rnn w/	54.0	-	43.0	35.7^{*}
sc-lstm	59.1^{*}	57	-	47.6
+deep	62.3^{**}	64.3^{**}	52.4	-

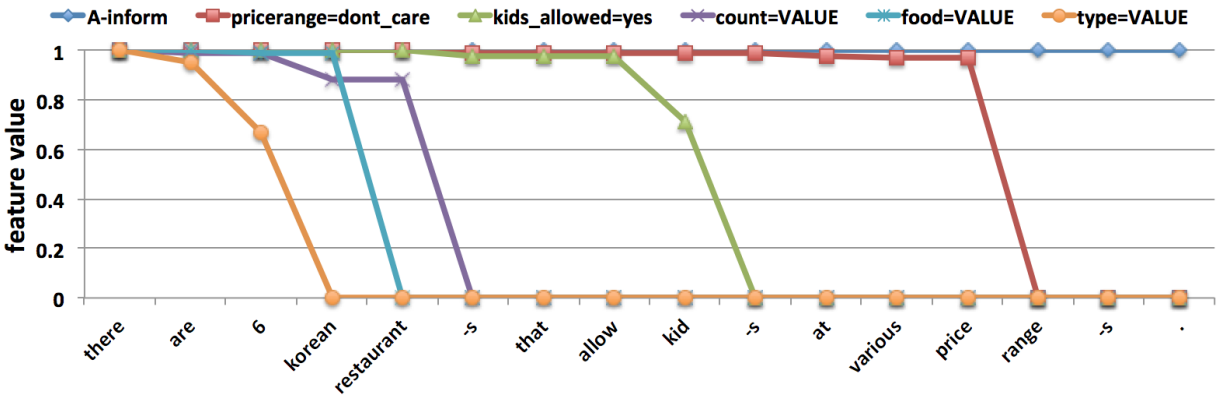
$^{*}p < 0.05$ $^{**}p < 0.005$

Table 4: Pairwise preference test among four systems. Statistical significance was computed using two-tailed binomial test.

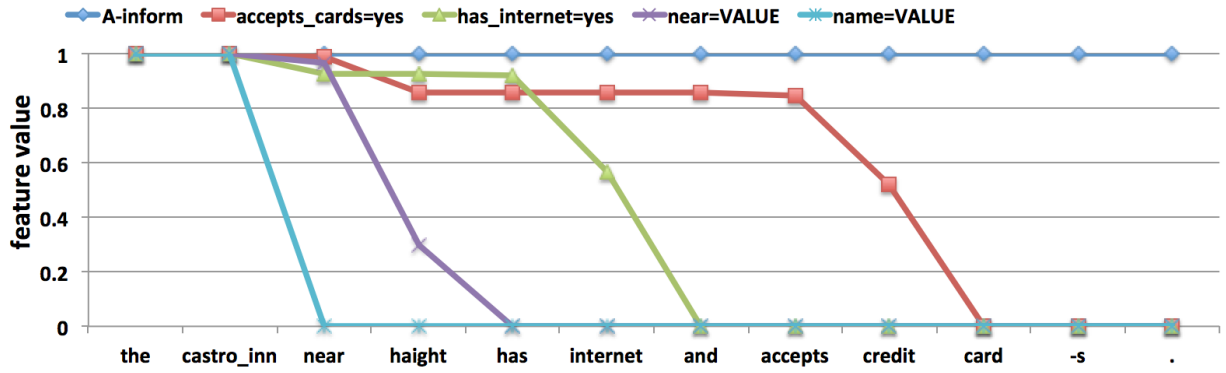
LSTM generally works better than vanilla RNN due to its ability to model long range dependencies more efficiently. We also found that by using gates, whether learned or heuristic, gave much lower slot error rates. As an aside, the ability of the SC-LSTM to learn gates is also exemplified in Figure 3. Finally, by combining the learned gate approach with the deep architecture (+*deep*), we obtained the best overall performance.

4.3 Human Evaluation

Since automatic metrics may not consistently agree with human perception (Stent et al., 2005), human testing is needed to assess subjective quality. To do this, a set of judges were recruited using AMT. For each task, two systems among the four (*classlm*, *rnn w/*, *sc-lstm*, and +*deep*) were randomly selected to generate utterances from a set of newly sampled dialogues in the restaurant domain. In order to evaluate system performance in the presence of language variation, each system generated 5 different surface realisations for each input DA and the human judges were asked to score each of them in terms of informativeness and naturalness (rating out of 3), and also asked to state a preference between the two. Here *informativeness*



(a) An example realisation from SF restaurant domain



(b) An example realisation from SF hotel domain

Figure 3: Examples showing how the SC-LSTM controls the DA features flowing into the network via its learned semantic gates. Despite errors due to sparse training data for some slots, each gate generally learned to detect words and phrases describing a particular slot-value pair.

is defined as whether the utterance contains all the information specified in the DA, and *naturalness* is defined as whether the utterance could plausibly have been produced by a human. In order to decrease the amount of information presented to the judges, utterances that appeared identically in both systems were filtered out. We tested 1000 DAs in total, and after filtering there were approximately 1300 generated utterances per system.

Table 3 shows the quality assessments which exhibit the same general trend as the objective results. The SC-LSTM systems (*sc-lstm* & *+deep*) outperform the class-based LMs (*classlm*) and the RNN with heuristic gates (*rnn w/*) in both metrics. The deep SC-LSTM system (*+deep*) is significantly better than the class LMs (*classlm*) in terms of *informativeness*, and better than the RNN with heuristic gates (*rnn w/*) in terms of *naturalness*. The preference test results are shown in Table 4. Again, the SC-LSTM systems (*sc-lstm & +deep*) were significantly preferred by the judges. Moreover, the judges recorded a strong preference for the deep approach (*+deep*) compared to the others, though the preference is not significant when comparing to its shallow counterpart (*sc-lstm*). Example dialogue acts and their top-5 realisations are shown in Table 5.

5 Conclusion and Future Work

In this paper we have proposed a neural network-based generator that is capable of generating natural linguistically varied responses based on a deep, semantically controlled LSTM architecture which we call SC-LSTM. The generator can be trained on unaligned data by jointly optimising its sentence planning and surface realisation components using a simple cross entropy criterion without any heuristics or handcrafting. We found that the SC-LSTM model achieved the best overall performance on two objective metrics across two different domains. An evaluation by human judges also confirmed that the SC-LSTM approach is strongly preferred to a variety of existing methods.

This work represents a line of research that tries

#	Example Dialogue Acts and Realizations from SF Restaurant Domain
1	inform(name="red door cafe", goodformeal="breakfast", area="cathedral hill", kidsallowed="no") *red door cafe is a good restaurant for breakfast in the area of cathedral hill and does not allow children .* *red door cafe is a good restaurant for breakfast in the cathedral hill area and does not allow children .* *red door cafe is a good restaurant for breakfast in the cathedral hill area and does not allow kids .* *red door cafe is good for breakfast and is in the area of cathedral hill and does not allow children .* *red door cafe does not allow kids and is in the cathedral hill area and is good for breakfast .*
2	informonly(name="dosa on fillmore and kiss seafood", pricerange="expensive", near="lower pacific heights") *there is no place other than dosa on fillmore and kiss seafood that are expensive near to lower pacific heights .* *dosa on fillmore and kiss seafood is the only expensive restaurant near lower pacific heights .* *the only listed restaurant near lower pacific heights in the expensive price range is dosa on fillmore and kiss seafood .* *i apologize , dosa on fillmore and kiss seafood is the only expensive restaurant near lower pacific heights .* *i apologize , dosa on fillmore and kiss seafood are the only expensive restaurants near lower pacific heights .*
#	**Example Dialogue Acts and Realizations from SF Hotel Domain**
3	inform(type="hotel",count="182",dogsallowed="dontcare") *there are 182 hotels if you do not care whether dogs are allowed .* *there are 182 hotels if you do not care whether they allow dogs .* *182 hotels are available if dogs allowed or not is not an issue .* *there are 182 hotels if allowing dogs or not is not an issue .* *there are 182 hotels if whether dogs are allowed does not matter .*
4	informonly(name="red victorian bed breakfast",acceptscreditcards="yes",near="haight",hasinternet="yes") *red victorian bed breakfast is the only hotel near haight and accepts credit cards and has internet .* *red victorian bed breakfast is the only hotel near haight and has internet and accepts credit cards .* *red victorian bed breakfast is the only hotel near haight that accept credit cards and offers internet .* *the red victorian bed breakfast has internet and near haight , it does accept credit cards .* *the red victorian bed breakfast is the only hotel near haight that accepts credit cards , and offers internet .*

Table 5: Samples of top 5 realisations from the deep SC-LSTM (+deep) system output.

to model the NLG problem in a unified architecture, whereby the entire model is end-to-end trainable from data. We contend that this approach can produce more natural responses which are more similar to colloquial styles found in human conversations. Another key potential advantage of neural network based language processing is the implicit use of distributed representations for words and a single compact parameter encoding of the information to be conveyed. This suggests that it should be possible to further condition the generator on some dialogue features such discourse information or social cues during the conversation. Furthermore, adopting a corpus based regime enables domain scalability and multilingual NLG to be achieved with less cost and a shorter lifecycle. These latter aspects will be the focus of our future work in this area.

6 Acknowledgements

Tsung-Hsien Wen and David Vandyke are supported by Toshiba Research Europe Ltd, Cambridge Research Laboratory.

References

Gabor Angeli, Percy Liang, and Dan Klein. 2010. A simple domain-independent probabilistic approach to generation. In *Proceedings of the 2010 Conference on EMNLP*, EMNLP '10. Association for Computational Linguistics.

Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, James Bergstra, Ian J. Goodfellow, Arnaud Bergeron, Nicolas Bouchard, and Yoshua Bengio. 2012. Theano: new features and speed improvements. Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop.

Anja Belz. 2008. Automatic generation of weather forecast texts using comprehensive probabilistic generation-space models. *Natural Language Engineering*.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *Neural Networks, IEEE Transactions on*.

James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. 2010. Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference*.

Adam Cheyer and Didier Guzzoni. 2007. Method and apparatus for building an intelligent automated assistant. US Patent App. 11/518,292.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning*.

Milica Gašić, Dongho Kim, Pirros Tsiakoulis, Catherine Breslin, Matthew Henderson, Martin Szummer, Blaise Thomson, and Steve Young. 2014. Incremental on-line adaptation of pomdp-based dialogue managers to extended domains. In *In Proceedings on InterSpeech*.

Milica Gašić, Dongho Kim, Pirros Tsiakoulis, and Steve Young. 2015. Distributed dialogue policies for multi-domain statistical dialogue management. In *In Proceedings on ICASSP*.

Alex Graves, Marcus Liwicki, Santiago Fernández, Roman Bertolami, Horst Bunke, and Jürgen Schmidhuber. 2009. A novel connectionist system for unconstrained handwriting recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*.

Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013a. Speech recognition with deep recurrent neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*.

Alex Graves, Abdel-rahman Mohamed, and Geoffrey E. Hinton. 2013b. Speech recognition with deep recurrent neural networks. *CoRR*, abs/1303.5778.

Alex Graves, Greg Wayne, and Ivo Danihelka. 2014. Neural turing machines. *CoRR*, abs/1410.5401.

Alex Graves. 2013. Generating sequences with recurrent neural networks. *CoRR*, abs/1308.0850.

Matthew Henderson, Blaise Thomson, and Steve Young. 2014. Robust dialog state tracking using delexicalised recurrent neural networks and unsupervised adaptation. In *Proceedings of IEEE Spoken Language Technology*.

Geoffrey Hinton, Li Deng, Dong Yu, George Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara Sainath, and Brian Kingsbury. 2012. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *Signal Processing Magazine, IEEE*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*.

Andrej Karpathy and Li Fei-Fei. 2014. Deep visual-semantic alignments for generating image descriptions. *CoRR*.

Ravi Kondadadi, Blake Howald, and Frank Schilder. 2013. A statistical nlg framework for aggregated planning and realization. In *Proceedings of the 51st Annual Meeting of the ACL*. Association for Computational Linguistics.

Karen Kukich. 1987. Where do phrases come from: Some preliminary experiments in connectionist phrase generation. In *Natural Language Generation*. Springer Netherlands.

Irene Langkilde and Kevin Knight. 1998. Generation that exploits corpus-based statistical knowledge. In *Proceedings of the 36th Annual Meeting of the ACL*, ACL '98.

François Mairesse and Marilyn A. Walker. 2011. Controlling user perceptions of linguistic style: Trainable generation of personality traits. *Computer Linguistics*.

François Mairesse and Steve Young. 2014. Stochastic language generation in dialogue using factored language models. *Computer Linguistics*.

François Mairesse, Milica Gašić, Filip Jurčíček, Simon Keizer, Blaise Thomson, Kai Yu, and Steve Young. 2010. Phrase-based statistical language generation using graphical models and active learning. In *Proceedings of the 48th ACL*, ACL '10.

William C. Mann and Sandra A. Thompson. 1988. Rhetorical structure theory: Toward a functional theory of text organization. *Text*.

Tomáš Mikolov and Geoffrey Zweig. 2012. Context dependent recurrent neural network language model. In *In Proceedings on IEEE SLT workshop*.

Tomáš Mikolov, Martin Karafit, Lukáš Burget, Jan Černocký, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *In Proceedings on InterSpeech*.

Tomáš Mikolov, Stefan Kombrink, Lukáš Burget, Jan H. Černocký, and Sanjeev Khudanpur. 2011a. Extensions of recurrent neural network language model. In *ICASSP, 2011 IEEE International Conference on*.

Tomáš Mikolov, Stefan Kombrink, Anoop Deoras, Lukáš Burget, and Jan Černocký. 2011b. Rnnlm - recurrent neural network language modeling toolkit. In *In Proceedings on ASRU*.

Danilo Mirkovic and Lawrence Cavedon. 2011. Dialogue management using scripts. EP Patent 1,891,625.

Alice H. Oh and Alexander I. Rudnicky. 2000. Stochastic language generation for spoken dialogue systems. In *Proceedings of the 2000 ANLP/NAACL Workshop on Conversational Systems - Volume 3*, ANLP/NAACL-ConvSyst '00.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on ACL*. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on EMNLP*. Association for Computational Linguistics.

Adwait Ratnaparkhi. 2002. Trainable approaches to surface natural language generation and their application to conversational dialog systems. *Computer Speech and Language*.

Verena Rieser and Oliver Lemon. 2010. Natural language generation as planning under uncertainty for spoken dialogue systems. In *Empirical Methods in Natural Language Generation*. Springer-Verlag.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *Signal Processing, IEEE Transactions on*.

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

Amanda Stent and Martin Molina. 2009. Evaluating automatic extraction of rules for sentence plan construction. In *Proceedings of SIGdial*. Association for Computational Linguistics.

Amanda Stent, Rashmi Prasad, and Marilyn Walker. 2004. Trainable sentence planning for complex information presentation in spoken dialog systems. In *In Proceedings of the Annual Meeting of the ACL*.

Amanda Stent, Matthew Marge, and Mohit Singhai. 2005. Evaluating evaluation methods for generation in the presence of variation. In *in Proceedings of CICLing 2005*.

Martin Sundermeyer, Tamer Alkhouli, Joern Wuebker, and Hermann Ney. 2014. Translation modeling with bidirectional recurrent neural networks. In *Proceedings of the 2014 Conference on EMNLP*. Association for Computational Linguistics.

Ilya Sutskever, James Martens, and Geoffrey E. Hinton. 2011. Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*. ACM.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. *CoRR*.

Marilyn A Walker, Owen C Rambow, and Monica Rogati. 2002. Training a sentence planner for spoken dialogue using boosting. *Computer Speech and Language*.

Marilyn Walker, Amanda Stent, Franois Mairesse, and Rashmi Prasad. 2007. Individual and domain adaptation in sentence planning for dialogue. *Journal of Artificial Intelligence Research (JAIR*.

Tsung-Hsien Wen, Milica Gašić, Dongho Kim, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young. 2015. Stochastic language generation in dialogue using recurrent neural networks with convolutional sentence reranking. In *Proceedings of SIGdial*. Association for Computational Linguistics.

Paul J Werbos. 1990. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*.

Kaisheng Yao, Baolin Peng, Yu Zhang, Dong Yu, Geoffrey Zweig, and Yangyang Shi. 2014. Spoken language understanding using long short-term memory neural networks. In *In Proceedings on IEEE SLT workshop*. IEEE Institute of Electrical and Electronics Engineers.

Steve Young, Milica Gašić, Blaise Thomson, and Jason D. Williams. 2013. Pomdp-based statistical spoken dialog systems: A review. *Proceedings of the IEEE*.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. *CoRR*, abs/1409.2329.

Xingxing Zhang and Mirella Lapata. 2014. Chinese poetry generation with recurrent neural networks. In *Proceedings of the 2014 Conference on EMNLP*. Association for Computational Linguistics, October.

Do Multi-Sense Embeddings Improve Natural Language Understanding?

Jiwei Li
Computer Science Department
Stanford University
Stanford, CA 94305, USA
jiweil@stanford.edu

Dan Jurafsky
Computer Science Department
Stanford University
Stanford, CA 94305, USA
jurafsky@stanford.edu

Abstract

Learning a distinct representation for each sense of an ambiguous word could lead to more powerful and fine-grained models of vector-space representations. Yet while 'multi-sense' methods have been proposed and tested on artificial word-similarity tasks, we don't know if they improve real natural language understanding tasks. In this paper we introduce a multi-sense embedding model based on Chinese Restaurant Processes that achieves state of the art performance on matching human word similarity judgments, and propose a pipelined architecture for incorporating multi-sense embeddings into language understanding.

We then test the performance of our model on part-of-speech tagging, named entity recognition, sentiment analysis, semantic relation identification and semantic relatedness, controlling for embedding dimensionality. We find that multi-sense embeddings do improve performance on some tasks (part-of-speech tagging, semantic relation identification, semantic relatedness) but not on others (named entity recognition, various forms of sentiment analysis). We discuss how these differences may be caused by the different role of word sense information in each of the tasks. The results highlight the importance of testing embedding models in real applications.

1 Introduction

Enriching vector models of word meaning so they can represent multiple word senses per word type seems to offer the potential to improve many language understanding tasks. Most traditional embedding models associate each word type with a single embedding (e.g., Bengio et al. (2006)). Thus the embedding for homonymous words like *bank* (with senses including 'sloping land' and 'financial institution') is forced to represent some uneasy central tendency between the various meanings. More fine-grained embeddings that represent more natural regions in semantic space could thus improve language understanding.

Early research pointed out that embeddings could model aspects of word sense (Kintsch, 2001) and recent research has proposed a number of models that represent each word type by different senses, each sense associated with a sense-specific embedding (Kintsch, 2001; Reisinger and Mooney, 2010; Neelakantan et al., 2014; Huang et al., 2012; Chen et al., 2014; Pina and Johansson, 2014; Wu and Giles, 2015; Liu et al., 2015). Such sense-specific embeddings have shown improved performance on simple artificial tasks like matching human word similarity judgments— WS353 (Rubenstein and Goodenough, 1965) or MC30 (Huang et al., 2012).

Incorporating multisense word embeddings into general NLP tasks requires a pipelined architecture that addresses three major steps:

1. **Sense-specific representation learning**: learn word sense specific embeddings from a large corpus, either unsupervised or aided by external resources like WordNet.

2. **Sense induction**: given a text unit (a phrase, sentence, document, etc.), infer word senses for its tokens and associate them with corresponding sense-specific embeddings.

3. **Representation acquisition for phrases or sentences:** learn representations for text units given sense-specific embeddings and pass them to machine learning classifiers.

Most existing work on multi-sense embeddings emphasizes the first step by learning sense spe-

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1722–1732,
Lisbon, Portugal, 17-21 September 2015.

cific embeddings, but does not explore the next two steps. These are important steps, however, since it isn't clear how existing multi-sense embeddings can be incorporated into and benefit real-world NLU tasks.

We propose a pipelined architecture to address all three steps and apply it to a variety of NLP tasks: part-of-speech tagging, named entity recognition, sentiment analysis, semantic relation identification and semantic relatedness. We find:

- Multi-sense embeddings give improved performance in some tasks (e.g., semantic similarity for words and sentences, semantic relation identification part-of-speech tagging), but not others (e.g., sentiment analysis, named entity extraction). In our analysis we offer some suggested explanations for these differences.
- Some of the improvements for multi-sense embeddings are no longer visible when using more sophisticated neural models like LSTMs which have more flexibility in filtering away the informational chaff from the wheat.
- It is important to carefully compare against embeddings of the same dimensionality.
- When doing so, the most straightforward way to yield better performance on these tasks is just to increase embedding dimensionality.

After describing related work, we introduce the new unsupervised sense-learning model in section 3, give our sense-induction algorithm in section 4, and then in following sections evaluate its performance for word similarity, and then various NLP tasks.

2 Related Work

Neural embedding learning frameworks represent each token with a dense vector representation, optimized through predicting neighboring words or decomposing co-occurrence matrices (Bengio et al., 2006; Collobert and Weston, 2008; Mnih and Hinton, 2007; Mikolov et al., 2013; Mikolov et al., 2010; Pennington et al., 2014). Standard neural models represent each word with a single unique vector representation.

Recent work has begun to augment the neural paradigm to address the multi-sense problem by associating each word with a series of sense specific embeddings. The central idea is to augment standard embedding learning models like skip-grams by disambiguating word senses based on local co-occurrence— e.g., the fruit "apple" tends to co-occur with the words "cider, tree, pear" while the homophonous IT company co-occurs with words like "iphone", "Google" or "ipod".

For example Reisinger and Mooney (2010) and Huang et al. (2012) propose ways to develop multiple embeddings per word type by pre-clustering the contexts of each token to create a fixed number of senses for each word, and then relabeling each word token with the clustered sense before learning embeddings. Neelakantan et al. (2014) extend these models by relaxing the assumption that each word must have a fixed number of senses and using a non-parametric model setting a threshold to decide when a new sense cluster should be split off; Liu et al. (2015) learns sense/topic specific embeddings by combining neural frameworks with LDA topic models. Wu and Giles (2015) disambiguate sense embeddings from Wikipedia by first clustering wiki documents. Chen et al. (2014) turn to external resources and used a predefined inventory of senses, building a distinct representation for every sense defined by the Wordnet dictionary. Other relevant work includes Qiu et al. (2014) who maintains separate representations for different part-of-speech tags of the same word.

Recent work is mostly evaluated on the relatively artificial task of matching human word similarity judgments.

3 Learning Sense-Specific Embeddings

We propose to build on this previous literature, most specifically Huang et al. (2012) and Neelakantan et al. (2014), to develop an algorithm for learning multiple embeddings for each word type, each embedding corresponding to a distinct induced word sense. Such an algorithm should have the property that a word should be associated with a new sense vector just when evidence in the context (e.g., neighboring words, document-level co-occurrence statistics) suggests that it is sufficiently different from its early senses. Such a line of thinking naturally points to Chinese Restaurant Processes (CRP) (Blei et al., 2004; Teh et al., 2006) which have been applied in the related field of word sense induction. In the analogy of

CRP, the current word could either sit at one of the existing tables (belonging to one of the existing senses) or choose a new table (a new sense). The decision is made by measuring semantic relatedness (based on local context information and global document information) and the number of customers already sitting at that table (the popularity of word senses). We propose such a model and show that it improves over the state of the art on a standard word similarity task.

3.1 Chinese Restaurant Processes

We offer a brief overview of Chinese Restaurant Processes in this section; readers interested in more details can consult the original papers (Blei et al., 2004; Teh et al., 2006; Pitman, 1995). CRP can be viewed as a practical interpretation of Dirichlet Processes (Ferguson, 1973) for non-parametric clustering. In the analogy, each data point is compared to a customer in a restaurant. The restaurant has a series of tables t, each of which serves a dish d_t. This dish can be viewed as the index of a cluster or a topic. The next customer w to enter would either choose an existing table, sharing the dish (cluster) already served or choosing a new cluster based on the following probability distribution:

$$Pr(t_w = t) \propto \begin{cases} N_t P(w|d_t) & \text{if t already exists} \\ \gamma P(w|d_{new}) & \text{if t is new} \end{cases} \quad (1)$$

where N_t denotes the number of customers already sitting at table t and $P(w|d_t)$ denotes the probability of assigning the current data point to cluster d_t. γ is the hyper parameter controlling the preference for sitting at a new table.

CRPs exhibit a useful "rich get richer" property because they take into account the popularity of different word senses. They are also more flexible than a simple threshold strategy for setting up new clusters, due to the robustness introduced by adopting the relative ratio of $P(w|d_t)$ and $P(w|d_{new})$.

3.2 Incorporating CRP into Distributed Language Models

We describe how we incorporate CRP into a standard distributed language model[1].

As in the standard vector-space model, each token w is associated with a K dimensional global embedding e_w. Additionally, it is associated with a set of senses $Z_w = \{z_w^1, z_w^2, ..., z_w^{|Z_w|}\}$ where $|Z_w|$ denotes the number of senses discovered for word w. Each sense z is associated with a distinct sense-specific embedding e_w^z. When we encounter a new token w in the text, at the first stage, we maximize the probability of seeing the current token given its context as in standard language models using the global vector e_w:

$$p(e_w|e_{\text{neigh}}) = F(e_w, e_{\text{neigh}}) \quad (2)$$

F() can take different forms in different learning paradigms, e.g., $F = \prod_{w' \in neigh} p(e_w, e_{w'})$ for skip-gram or $F = p(e_w, g(e_w))$ for SENNA (Collobert and Weston, 2008) and CBOW, where $g(e_{neigh})$ denotes a function that projects the concatenation of neighboring vectors to a vector with the same dimension as e_w for SENNA and the bag-or-word averaging for CBOW (Mikolov et al., 2013).

Unlike traditional one-word-one-vector frameworks, e_{neigh} includes sense information in addition to the global vectors for neighbors. e_{neigh} can therefore be written as[2].

$$e_{\text{neigh}} = \{e_{n-k}, , ..., e_{n-1}, e_{n+1}, ..., e_{n-k}\} \quad (3)$$

Next we would use CRP to decide which sense the current occurrence corresponds to, or construct a new sense if it is a new meaning that we have not encountered before. Based on CRP, the probability that assigns the current occurrence to each of the discovered senses or a new sense is given by:

$$Pr(z_w = z) \propto \begin{cases} N_z^w P(e_w^z|\text{context}) \\ \qquad \text{if z already exists} \\ \gamma P(w|z_{new}) \quad \text{if z is new} \end{cases} \quad (4)$$

where N_z^w denotes the number of times already assigned to sense z for token w. $P(e_w^z|\text{context})$ denotes the probability that current occurrence belonging to (or generated by) sense z.

The algorithm for parameter update for the one token predicting procedure is illustrated in Figure

[1] We omit details about training standard distributed models; see Collobert and Weston (2008) and Mikolov et al. (2013).

[2] For models that predict succeeding words, sense labels for preceding words have already been decided. For models that predict words using both left and right contexts, the labels for right-context words have not been decided yet. In such cases we just use its global word vector to fill up the position.

01: **Input** : Token sequence $\{w_n, w_{\text{neigh}}\}$.
02: Update parameters involved in Equ (3)(4) based on current word prediction.
03: Sample sense label z from CRP.
04: If a new sense label z is sampled:
05: - add z to Z_{w_n}
06: - $e^z_{w_n} = \text{argmax}\, p(w_n|z_m)$
07: else: update parameters involved based on sampled sense label z.

Figure 1: Incorporating CRP into Neural Language Models.

1: Line 2 shows parameter updating through predicting the occurrence of current token. Lines 4-6 illustrate the situation when a new word sense is detected, in which case we would add the newly detected sense z into Z_{w_n}. The vector representation e^z_w for the newly detected sense would be obtained by maximizing the function $p(e^z_w|\text{context})$.

As we can see, the model performs word-sense clustering and embedding learning jointly, each one affecting the other. The prediction of the global vector of the current token (line2) is based on both the global and sense-specific embeddings of its neighbors, as will be updated through predicting the current token. Similarly, once the sense label is decided (line7), the model will adjust the embeddings for neighboring words, both global word vectors and sense-specific vectors.

Training We train embeddings using Gigaword5 + Wikipedia2014. The training approach is implemented using skip-grams (SG) (Mikolov et al., 2013). We induced senses for the top 200,000 most frequent words (and used a unified "unknown" token for other less-frequent tokens). The window size is set to 11. We iterate three times over the corpus.

4 Obtaining Word Representations for NLU tasks

Next we describe how we decide sense labels for tokens in context. The scenario is treated as a inference procedure for sense labels where all global word embeddings and sense-specific embeddings are kept fixed.

Given a document or a sentence, we have an objective function with respect to sense labels by multiplying Eq.2 over each containing token. Computing the global optimum sense labeling—in which every word gets an optimal sense label—requires searching over the space of all senses for all words, which can be expensive. We therefore chose two simplified heuristic approaches:

- **Greedy Search**: Assign each token the locally optimum sense label and represent the current token with the embedding associated with that sense.

- **Expectation**: Compute the probability of each possible sense for the current word, and represent the word with the expectation vector:

$$\vec{e}_w = \sum_{z \in Z_w} p(w|z, \text{context}) \cdot e^z_w$$

5 Word Similarity Evaluation

We evaluate our embeddings by comparing with other multi-sense embeddings on the standard artificial task for matching human word similarity judgments.

Early work used similarity datasets like WS353 (Finkelstein et al., 2001) or RG (Rubenstein and Goodenough, 1965), whose context-free nature makes them a poor evaluation. We therefore adopt Stanford's Contextual Word Similarities (SCWS) (Huang et al., 2012), in which human judgments are associated with pairs of words in context. Thus for example "bank" in the context of "river bank" would have low relatedness with "deficit" in the context "financial deficit".

We first use the Greedy or Expectation strategies to obtain word vectors for tokens given their context. These vectors are then used as input to get the value of cosine similarity between two words.

Performances are reported in Table 1. Consistent with earlier work (e.g.., Neelakantan et al. (2014)), we find that multi-sense embeddings result in better performance in the context-dependent SCWS task (SG+Greedy and SG+Expect are better than SG). As expected, performance is not as high when global level information is ignored when choosing word senses (SG+Greedy) as when it is included (SG+Expect), as neighboring words don't provide sufficient information for word sense disambiguation.

To note, the proposed CRF models work a little better than earlier baselines, which gives some evidence that it is sufficiently strong to stand in for

Model	SCWS Correlation
SkipGram	66.4
SG+Greedy	69.1
SG+Expect	69.7
Chen	68.4
Neelakantan	69.3

Table 1: Performances for different set of multi-sense embeddings (300d) evaluated on SCWS by measuring the Spearman correlation between each model's similarity and the human judgments. Baselines performances are reprinted from Neelakantan et al. (2014) and Chen et al. (2014); we report the best performance across all settings mentioned in their paper.

this class of multi-sense models and serves as a promise for being extended to NLU tasks.

Visualization Table 2 shows examples of semantically related words given the local context. Word embeddings for tokens are obtained by using the inferred sense labels from the Greedy model and are then used to search for nearest neighbors in the vector space based on cosine similarity. Like earlier models (e.g., Neelakantan et al. (2014))., the model can disambiguate different word senses (in examples like *bank, rock* and *apple*) based on their local context; although of course the model is also capable of dealing with polysemy—senses that are less distinct.

6 Experiments on NLP Tasks

Having shown that multi-sense embeddings improve word similarity tasks, we turn to ask whether they improve real-world NLU tasks: POS tagging, NER tagging, sentiment analysis at the phrase and sentence level, semantic relationship identification and sentence-level semantic relatedness. For each task, we experimented on the following sets of embeddings, which are trained using the word2vec package on the same corpus:

- Standard one-word-one-vector embeddings from skip-gram (50d).
- Sense disambiguated embeddings from Section 3 and 4 using Greedy Search and Expectation (50d)
- The concatenation of global word embeddings and sense-specific embeddings (100d).
- Standard one-word-one-vector skip-gram embeddings with dimensionality doubled (100d) (100d is the correct corresponding baseline since the concatenation above doubles the dimensionality of word vectors)
- Embeddings with very high dimensionality (300d).

As far as possible we try to perform an apple-to-apple comparison on these tasks, and our goal is an analytic one—to investigate how well semantic information can be encoded in multi-sense embeddings and how they can improve NLU performances—rather than an attempt to create state-of-the-art results. Thus for example, in tagging tasks (e.g., NER, POS), we follow the protocols in (Collobert et al., 2011) using the concatenation of neighboring embeddings as input features rather than treating embeddings as auxiliary features which are fed into a CRF model along with other manually developed features as in Pennington et al. (2014). Or for experiments on sentiment and other tasks where sentence level embeddings are required we only employ standard recurrent or recursive models for sentence embedding rather than models with sophisticated state-of-the-art methods (e.g., Tai et al. (2015; Irsoy and Cardie (2014)).

Significance testing for comparing models is done via the bootstrap test (Efron and Tibshirani, 1994). Unless otherwise noted, significant testing is performed on one-word-one-vector embedding (50d) versus multi-sense embedding using Expectation inference (50d) and one-vector embedding (100d) versus Expectation (100d).

6.1 The Tasks

Named Entity Recognition We use the CoNLL-2003 English benchmark for training, and test on the CoNLL-2003 test data. We follow the protocols in Collobert et al. (2011), using the concatenation of neighboring embeddings as input to a multi-layer neural model. We employ a five-layer neural architecture, comprised of an input layer, three convolutional layers with rectifier linear activation function and a softmax output layer. Training is done by gradient descent with minibatches where each sentence is treated as one batch. Learning rate, window size, number of hidden units of hidden layers, L2 regularizations and number of iterations are tuned on the development set.

Part-of-Speech Tagging We use Sections 0–18 of the Wall Street Journal (WSJ) data for train-

Context	Nearest Neighbors
Apple is a kind of fruit.	pear, cherry, mango, juice, peach, plum, fruit, cider, apples, tomato, orange, bean, pie
Apple releases its new ipads.	microsoft, intel, dell, ipad, macintosh, ipod, iphone, google, computer, imac, hardware
He borrowed the money from **banks**.	banking, credit, investment, finance, citibank, currency, assets, loads, imf, hsbc
along the shores of lakes, **banks** of rivers	land, coast, river, waters, stream, inland, area, coasts, shoreline, shores, peninsula
Basalt is the commonest volcanic **rock**.	boulder, stone, rocks, sand, mud, limestone, volcanic, sedimentary, pelt, lava, basalt
Rock is the music of teenage rebellion.	band, pop, bands, song, rap, album, jazz. blues, singer, hip-pop, songs, guitar, musician

Table 2: Nearest neighbors of words given context. The embeddings from context words are first inferred with the Greedy strategy; nearest neighbors are computed by cosine similarity between word embeddings. Similar phenomena have been observed in earlier work (Neelakantan et al., 2014)

Standard (50) 0.852	Greedy (50) 0.852 (+0)	Expectation(50) 0.854 (+0.02)
Standard (100) 0.867	Global+G (100) 0.866 (-0.01)	Global+E (100) 0.871 (+0.04)
Standard (300) 0.882		

Table 3: Accuracy for Different Models on Name Entity Recognition. Global+E stands for Global+Expectation inference and Global+G stands for Global+Greedy inference. p-value 0.223 for Standard(50) verse Expectation (50) and 0.310 for Standard(100) verse Expectation (100).

ing, sections 19–21 for validation and sections 22–24 for testing. Similar to NER, we trained 5-layer neural models which take the concatenation of neighboring embeddings as inputs. We adopt a similar training and parameter tuning strategy as for POS tagging.

Standard (50) 0.925	Greedy (50) 0.934 (+0.09)	Expectation (50) 0.938 (+0.13)
Standard (100) 0.940	Global+G (100) 0.946 (+0.06)	Global+E (100) 0.952 (+0.12)
Standard (300) 0.954		

Table 4: Accuracy for Different Models on Part of Speech Tagging. P-value 0.033 for 50d and 0.031 for 100d.

Sentence-level Sentiment Classification (Pang) The sentiment dataset of Pang et al. (2002) consists of movie reviews with a sentiment label for each sentence. We divide the original dataset into training(8101)/dev(500)/testing(2000). Word embeddings are initialized using the aforementioned types of embeddings and kept fixed in the learning procedure. Sentence level embeddings are achieved by using standard sequence recurrent neural models (Pearlmutter, 1989) (for details, please refer to Appendix section). The obtained embedding is then fed into a sigmoid classifier. Convolutional matrices at the word level are randomized from [-0.1, 0.1] and learned from sequence models. For training, we adopt AdaGrad with mini-batch. Parameters (i.e., $L2$ penalty, learning rate and mini batch size) are tuned on the development set. Due to space limitations, we omit details of recurrent models and training.

Standard (50) 0.750	Greedy (50) 0.752(+0.02)	Expectation (50) 0.750(+0.00)
Standard (100) 0.768	Global+G (100) 0.765(-0.03)	Global+E (100) 0.763(-0.05)
Standard (300) 0.774		

Table 5: Accuracy for Different Models on Sentiment Analysis (Pang et al.'s dataset). P-value 0.442 for 50d and 0.375 for 100d.

Sentiment Analysis–Stanford Treebank The Stanford Sentiment Treebank (Socher et al., 2013) contains gold-standard labels for each constituent in the parse tree (phrase level), thus allowing us to investigate a sentiment task at a finer granularity than the dataset in Pang et al. (2002) where labels are only found at the top of each sentence, The sentences in the treebank were split into a training(8544)/development(1101)/testing(2210) dataset.

Following Socher et al. (2013) we obtained embeddings for tree nodes by using a recursive neural network model, where the embedding for parent node is obtained in a bottom-up fashion based on its children. The embeddings for each parse tree constituent are output to a softmax layer; see Socher et al. (2013).

We focus on the standard version of recursive neural models. Again we fixed word embeddings to each of the different embedding settings described above[3]. Similarly, we adopted AdaGrad

[3]Note that this is different from the settings used in

with mini-batch. Parameters (i.e., $L2$ penalty, learning rate and mini batch size) are tuned on the development set. The number of iterations is treated as a variable to tune and parameters are harvested based on the best performance on the development set.

Standard (50)	Greedy (50)	Expectation (50)
0.818	0.815 (-0.03)	0.820 (+0.02)
Standard (100)	Global+G (100)	Global+E (100)
0.838	0.840 (+0.02)	0.838 (+0.00)
Standard (300)		
0.854		

Table 6: Accuracy for Different Models on Sentiment Analysis (binary classification on Stanford Sentiment Treebank.). P-value 0.250 for 50d and 0.401 for 100d.

Semantic Relationship Classification SemEval-2010 Task 8 (Hendrickx et al., 2009) is to find semantic relationships between pairs of nominals, e.g., in "My [apartment]$_{e1}$ has a pretty large [kitchen]$_{e2}$" classifying the relation between [apartment] and [kitchen] as *component-whole*. The dataset contains 9 ordered relationships, so the task is formalized as a 19-class classification problem, with directed relations treated as separate labels; see Hendrickx et al. (2009) for details.

We follow the recursive implementations defined in Socher et al. (2012). The path in the parse tree between the two nominals is retrieved, and the embedding is calculated based on recursive models and fed to a softmax classifier. For pure comparison purpose, we only use embeddings as features and do not explore other combination of artificial features. We adopt the same training strategy as for the sentiment task (e.g., Adagrad, mini-batches, etc).

Standard (50)	Greedy (50)	Expectation (50)
0.748	0.760 (+0.12)	0.762 (+0.14)
Standard(100)	Global+G (100)	Global+E (100)
0.770	0.782 (+0.12)	0.778 (+0.18)
Standard(300)		
0.798		

Table 7: Accuracy for Different Models on Semantic Relationship Identification. P-value 0.017 for 50d and 0.020 for 100d.

(Socher et al., 2013) where word vectors were treated as parameters to optimize.

Sentence Semantic Relatedness We use the Sentences Involving Compositional Knowledge (SICK) dataset (Marelli et al., 2014) consisting of 9927 sentence pairs, split into training(4500)/development(500)/Testing(4927). Each sentence pair is associated with a gold-standard label ranging from 1 to 5, indicating how semantically related are the two sentences, from 1 (the two sentences are unrelated) to 5 (the two are very related).

In our setting, the similarity between two sentences is measured based on sentence-level embeddings. Let s_1 and s_2 denote two sentences and e_{s_1} and e_{s_2} denote corresponding embeddings. e_{s_1} and e_{s_2} are achieved through recurrent or recursive models (as illustrated in Appendix section). Again, word embeddings are obtained by simple table look up in one-word-one-vector settings and inferred using the Greedy or Expectation strategy in multi-sense settings. We adopt two different recurrent models for acquiring sentence-level embeddings, a standard recurrent model and an LSTM model (Hochreiter and Schmidhuber, 1997).

The similarity score is predicted using a regression model built on the structure of a three layer convolutional model, with concatenation of e_{s1} and e_{s2} as input, and a regression score from 1-5 as output. We adopted the same training strategy as described earlier. The trained model is then used to predict the relatedness score between two new sentences. Performance is measured using Pearson's r between the predicted score and gold-standard labels.

Standard(50)	Greedy (50)	Expectation (50)
0.824	0.838(+0.14)	0.836(+0.12)
Standard (100)	Global+G (100)	Global+E (100)
0.835	0.840 (+0.05)	0.845 (+0.10)
Standard(300)		
0.850		

Table 8: Pearson's r for Different Models on Semantic Relatedness for Standard Models. P-value 0.028 for 50d and 0.042 for 100d.

6.2 Discussions

Results for different tasks are represented in Tables 3-9.

At first glance it seems that multi-sense embeddings do indeed offer superior performance, since combining global vectors with sense-specific vectors introduces a consistent performance boost

Standard(50)	Greedy(50)	Expectation(50)
0.843	0.848 (+0.05)	0.846 (+0.03)
Standard(100)	Global+G (100)	Global+E (100)
0.850	0.853 (+0.03)	0.854 (+0.04)
Standard(300)		
0.850		

Table 9: Pearson's r for Different Models on Semantic Relatedness for LSTM Models. P-value 0.145 for 50d and 0.170 for 100d.

for every task, when compared with the standard (50d) setting. But of course this is an unfair comparison; combining global vector with sense-specific vector doubles the dimensionality of vector to 100, making comparison with standard dimensionality (50d) unfair. When comparing with standard (100), the conclusions become more nuanced.

For every task, the +Expectation method has performances that often seem to be higher than the simple baseline (both for the 50d case or the 100d case). However, only some of these differences are significant.

(1) Using multi-sense embeddings is significantly helpful for tasks like semantic relatedness (Tables 7-8). This is sensible since sentence meaning here is sensitive to the semantics of one particular word, which could vary with word sense and which would directly be reflected on the relatedness score.

(2) By contrast, for sentiment analysis (Tables 5-6), much of the task depends on correctly identifying a few sentiment words like "good" or "bad", whose senses tend to have similar sentiment values, and hence for which multi-sense embeddings offer little help. Multi-sense embeddings might promise to help sentiment analysis for some cases, like disambiguating the word "sound" in "safe and sound" versus "movie sound". But we suspect that such cases are not common, explaining the non-significance of the improvement. Furthermore, the advantages of neural models in sentiment analysis tasks presumably lie in their capability to capture local composition like negation, and it's not clear how helpful multi-sense embeddings are for that aspect.

(3) Similarly, multi-sense embeddings help for POS tagging, but not for NER tagging (Table 3-4). Word senses have long been known to be related to POS tags. But the largest proportion of NER tags consists of the negative not-a-NER ("O") tag, each of which is likely correctly labelable regardless of whether senses are disambiguated or not (since presumably if a word is not a named entity, most of its senses are not named entities either).

(4) As we apply more sophisticated models like LSTM to semantic relatedness tasks (in Table 9), the advantages caused by multi-sense embeddings disappears.

(5) Doubling the number of dimensions is sufficient to increase performance as much as using the complex multi-sense algorithm. (Of course increasing vector dimensionality (to 300) boosts performance even more, although at the significant cost of exponentially increasing time complexity.) We do larger one-word-one-vector embeddings do so well? We suggest some hypotheses:

- though information about distinct senses is encoded in one-word-one-vector embeddings in a mixed and less structured way, we suspect that the compositional nature of neural models is able to separate the informational chaff from the wheat and choose what information to take up, bridging the gap between single vector and multi-sense paradigms. For models like LSTMs which are better at doing such a job by using gates to control information flow, the difference between two paradigms should thus be further narrowed, as indeed we found.

- The pipeline model proposed in the work requires sense-label inference (i.e., step 2). We proposed two strategies: GREEDY and EXPECTATION, and found that GREEDY models perform worse than EXPECTATION, as we might expect[4]. But even EXPECTATION can be viewed as another form of one-word-one-vector models, just one where different senses are entangled but weighted to emphasize the important ones. Again, this suggests another cause for the strong relative performance of larger-dimensioned one-word-one-vector models.

7 Conclusion

In this paper, we expand ongoing research into multi-sense embeddings by first proposing a new version based on Chinese restaurant processes that achieves state of the art performance on simple

[4]GREEDY models work in a more aggressive way and likely make mistakes due to the non-global-optimum nature and limited context information

word similarity matching tasks. We then introduce a pipeline system for incorporating multi-sense embeddings into NLP applications, and examine multiple NLP tasks to see whether and when multi-sense embeddings can introduce performance boosts. Our results suggest that simply increasing the dimensionality of baseline skip-gram embeddings is sometimes sufficient to achieve the same performance wins that come from using multi-sense embeddings. That is, the most straightforward way to yield better performance on these tasks is just to increase embedding dimensionality.

Our results come with some caveats. In particular, our conclusions are based on the pipelined system that we introduce, and other multi-sense embedding systems (e.g., a more advanced sense learning model or a better sense label model or a completely different pipeline system) may find stronger effects of multi-sense models. Nonetheless we do consistently find improvements for multi-sense embeddings in some tasks (part-of-speech tagging and semantic relation identification), suggesting the benefits of our multi-sense models and those of others. Perhaps the most important implication of our results may be the evidence they provide for the importance of going beyond simple human-matching tasks, and testing embedding models by using them as components in real NLP applications.

8 Appendix

In sentiment classification and sentence semantic relatedness tasks, classification models require embeddings that represent the input at a sentence or phrase level. We adopt recurrent networks (standard ones or LSTMs) and recursive networks in order to map a sequence of tokens with various length to a vector representation.

Recurrent Networks A recurrent network successively takes word w_t at step t, combines its vector representation e_t with the previously built hidden vector h_{t-1} from time $t-1$, calculates the resulting current embedding h_t, and passes it to the next step. The embedding h_t for the current time t is thus:

$$h_t = \tanh(W \cdot h_{t-1} + V \cdot e_t) \tag{5}$$

where W and V denote compositional matrices. If N_s denote the length of the sequence, h_{N_s} represents the whole sequence S.

Recursive Networks Standard recursive models work in a similar way by working on neighboring words by parse tree order rather than sequence order. They compute the representation for each parent node based on its immediate children recursively in a bottom-up fashion until reaching the root of the tree. For a given node η in the tree and its left child η_{left} (with representation e_{left}) and right child η_{right} (with representation e_{right}), the standard recursive network calculates e_η:

$$e_\eta = \tanh(W \cdot e_{\eta_{\text{left}}} + V \cdot e_{\eta_{\text{right}}}) \tag{6}$$

Long Short Term Memory (LSTM) LSTM models (Hochreiter and Schmidhuber, 1997) are defined as follows: given a sequence of inputs $X = \{x_1, x_2, ..., x_{n_X}\}$, an LSTM associates each timestep with an input, memory and output gate, respectively denoted as i_t, f_t and o_t. We notationally disambiguate e and h, where e_t denote the vector for an individual text unit (e.g., word or sentence) at time step t while h_t denotes the vector computed by the LSTM model at time t by combining e_t and h_{t-1}. σ denotes the sigmoid function. $W \in \mathbb{R}^{4K \times 2K}$. The vector representation h_t for each time-step t is given by:

$$\begin{bmatrix} i_t \\ f_t \\ o_t \\ l_t \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{bmatrix} W \cdot \begin{bmatrix} h_{t-1} \\ e_t \end{bmatrix} \tag{7}$$

$$c_t = f_t \cdot c_{t-1} + i_t \cdot l_t \tag{8}$$

$$h_t^s = o_t \cdot c_t \tag{9}$$

9 Acknowledgments

We would like to thank Sam Bowman, Ignacio Cases, Kevin Gu, Gabor Angeli, Sida Wang, Percy Liang and other members of the Stanford NLP group, as well as anonymous reviewers for their helpful advice on various aspects of this work. We gratefully acknowledge the support of the NSF via award IIS-1514268, the Defense Advanced Research Projects Agency (DARPA) Deep Exploration and Filtering of Text (DEFT) Program under Air Force Research Laboratory (AFRL) contract no. FA8750-13-2-0040. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of NSF, DARPA, AFRL, or the US government.

References

Yoshua Bengio, Holger Schwenk, Jean-Sébastien Senécal, Fréderic Morin, and Jean-Luc Gauvain. 2006. Neural probabilistic language models. In *Innovations in Machine Learning*, pages 137–186. Springer.

David M. Blei, Thomas L. Griffiths, Michael I. Jordan, and Joshua B. Tenenbaum. 2004. Hierarchical topic models and the nested chinese restaurant process. *Advances in neural information processing systems*, 16.

Xinxiong Chen, Zhiyuan Liu, and Maosong Sun. 2014. A unified model for word sense representation and disambiguation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1025–1035.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM.

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

Bradley Efron and Robert J Tibshirani. 1994. *An introduction to the bootstrap*. CRC press.

Thomas S Ferguson. 1973. A bayesian analysis of some nonparametric problems. *The annals of statistics*, pages 209–230.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2001. Placing search in context: The concept revisited. In *Proceedings of the 10th international conference on World Wide Web*, pages 406–414. ACM.

Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. 2009. Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In *Proceedings of the Workshop on Semantic Evaluations: Recent Achievements and Future Directions*, pages 94–99. Association for Computational Linguistics.

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

Eric H Huang, Richard Socher, Christopher D Manning, and Andrew Y Ng. 2012. Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 873–882. Association for Computational Linguistics.

Ozan Irsoy and Claire Cardie. 2014. Deep recursive neural networks for compositionality in language. In *Advances in Neural Information Processing Systems*, pages 2096–2104.

Walter Kintsch. 2001. Predication. *Cognitive Science*, 25(2):173–202.

Yang Liu, Zhiyuan Liu, Tat-Seng Chua, and Maosong Sun. 2015. Topical word embeddings. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*.

Marco Marelli, Luisa Bentivogli, Marco Baroni, Raffaella Bernardi, Stefano Menini, and Roberto Zamparelli. 2014. Semeval-2014 task 1: Evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. *SemEval-2014*.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH*, pages 1045–1048.

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

Andriy Mnih and Geoffrey Hinton. 2007. Three new graphical models for statistical language modelling. In *Proceedings of the 24th international conference on Machine learning*, pages 641–648. ACM.

Arvind Neelakantan, Jeevan Shankar, Alexandre Passos, and Andrew McCallum. 2014. Efficient non-parametric estimation of multiple embeddings per word in vector space. In *Proceedings of EMNLP*.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: Sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 79–86. Association for Computational Linguistics.

Barak A Pearlmutter. 1989. Learning state space trajectories in recurrent neural networks. *Neural Computation*, 1(2):263–269.

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

Luis Nieto Pina and Richard Johansson. 2014. A simple and efficient method to generate word sense representations. *arXiv preprint arXiv:1412.6045*.

Jim Pitman. 1995. Exchangeable and partially exchangeable random partitions. *Probability theory and related fields*, 102(2):145–158.

Lin Qiu, Yong Cao, Zaiqing Nie, and Yong Rui. 2014. Learning word representation considering proximity and ambiguity. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*.

Joseph Reisinger and Raymond J Mooney. 2010. Multi-prototype vector-space models of word meaning. In *NAACL*.

Herbert Rubenstein and John B Goodenough. 1965. Contextual correlates of synonymy. *Communications of the ACM*, 8(10):627–633.

Richard Socher, Brody Huval, Christopher D Manning, and Andrew Y Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1201–1211. Association for Computational Linguistics.

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

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*.

Yee Whye Teh, Michael I. Jordan, Matthew J. Beal, and David M. Blei. 2006. Hierarchical dirichlet processes. *Journal of the american statistical association*, 101(476).

Zhaohui Wu and C. Lee Giles. 2015. Sense-aware semantic analysis: A multi-prototype word representation model using wikipedia. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*.

Learning Semantic Composition to Detect Non-compositionality of Multiword Expressions

Majid Yazdani
Computer Science Department
University of Geneva
majid.yazdani@unige.ch

Meghdad Farahmand
Computer Science Department
University of Geneva
meghdad.farahmand@unige.ch

James Henderson
Xerox Research Center Europe
james.henderson@xrce.xerox.com

Abstract

Non-compositionality of multiword expressions is an intriguing problem that can be the source of error in a variety of NLP tasks such as language generation, machine translation and word sense disambiguation. We present methods of non-compositionality detection for English noun compounds using the unsupervised learning of a semantic composition function. Compounds which are not well modeled by the learned semantic composition function are considered non-compositional. We explore a range of distributional vector-space models for semantic composition, empirically evaluate these models, and propose additional methods which improve results further. We show that a complex function such as polynomial projection can learn semantic composition and identify non-compositionality in an unsupervised way, beating all other baselines ranging from simple to complex. We show that enforcing sparsity is a useful regularizer in learning complex composition functions. We show further improvements by training a decomposition function in addition to the composition function. Finally, we propose an EM algorithm over latent compositionality annotations that also improves the performance.

1 Introduction

Multiword Expressions (MWEs) are sequences of words that exhibit some kind of idiosyncrasy. This idiosyncrasy can be semantic, statistical, or syntactic[1]. *Ivory tower*, *speed limit*, and *at large* are examples of semantically, statistically and syntactically idiosyncratic MWEs respectively. Note that an MWE can be idiosyncratic at several levels. In general, semantically idiosyncratic MWEs are commonly referred to as non-compositional (Baldwin and Kim, 2010) and statistically idiosyncratic MWEs are commonly referred to as collocations (Sag et al., 2002). Non-compositional MWEs are those whose meaning can not be readily inferred from the meaning of their constituents and collocations are those MWEs whose constituents co-occur more than expected by chance. Collocations constitute the largest subset of all kinds of MWEs, however, non-compositional ones cause more problems in various NLP tasks, for example word sense disambiguation (McCarthy et al., 2003) and machine translation (Lin, 1999). It may also be more challenging to model non-compositionality than collocational weight as the former has to do with modelling the semantics and the latter can to some extent be modeled by conventional statistical measures such as mutual information. Detecting non-compositionality in an automatic fashion has been the aim of much previous research.

In this paper, we capture non-compositionality of English Noun Compounds (NCs)[2] based on the assumption that the majority of the compounds are compositional, for which a composition function can be learned. This implies that the compounds for which a composition function cannot be learned with a relatively low error are non-compositional.

In previous work on vector-space models of distributional semantics, semantic composition has been commonly assumed to be a trivial predetermined function such as addition, multiplication,

[1] MWEs can have other less significant kinds of idiosyncrasies. For instance lexical idiosyncrasy as in *ad hoc*, and pragmatic idiosyncrasy as in *good morning* (Baldwin and Kim, 2010)

[2] MWEs have various syntactic categories such as noun compounds, verb particle constructions, light verb constructions, etc., with noun compounds and verb particle constructions constituting the most prominent categories of MWEs.

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1733–1742,
Lisbon, Portugal, 17-21 September 2015.

and their weighted variations (Mitchell and Lapata, 2008; Reddy et al., 2011; Salehi et al., 2015). Nevertheless there is some work that regards composition as a more complex function. For instance Widdows (2008) who propose (but doesn't empirically test) the use of Tensor and Convolution products for modelling non-compositionality, Baroni and Zamparelli (2010) who regard adjectives in adjectival-noun compositions as matrices that can be learned by linear regression, and Socher et al. (2012) who present a model that learns phrase composition by means of a recursive neural network. The two latter works show that complex composition models significantly outperform additive and multiplicative functions. In this work, we too assume that composition is arguably a complex function. We believe simplified composition functions, such as additive and multiplicative functions and their weighted variations, while having advantages such as being impervious to overfitting, can not completely capture semantic composition. Nevertheless modelling composition by means of a powerful function can be equally inadequate for our purposes. An overly powerful composition function memorizes all compositional and non-compositional compounds, resulting in overfitting and low learning error that hinders discrimination between compositional and non-compositional compounds. We examine various classes of composition functions, ranging from the least to the most powerful (in terms of learning capacity). We show that complex functions clearly do a better job in modelling semantic composition and in detecting non-compositionality compared to commonly used additive and multiplicative functions.

Compositional compounds are also decomposable; intuitively, their semantics is the union of the semantics of their components. More formally, conditioned on the vector of the compound, vectors of the component words should be *independently* predictable. This principle, together with the assumption that most of the compounds are compositional, leads to the conclusion that a model of composition should be able to be auto-reconstructive: the composition function that maps component-words' vectors to their compound vector should have an associated decomposition function that independently predicts each of the component-words' vectors from this compound vector. An auto-reconstructive model enables us to exploit more data in order to learn semantic composition and predict compositionality. We show that auto-reconstruction can improve the accuracy of composition functions and improve detecting non-compositionality.

To further improve non-compositionality detection, we propose an EM-like detection algorithm based on hidden compositionality annotations. The best composition is the one that is the best fit on all the data points except the non-compositional ones. Since we don't use annotated data at training time, we assume annotations to be hidden variables and iteratively alternate between optimizing the composition function and optimizing the hidden compositionality annotations. We show that this iterative algorithm increases the accuracy of non-compositionality detection compared to the case when training is done on all examples.

We run our experiments on the data set of Farahmand et al. (2015) who provide a set of English NCs which are annotated with non-compositionality judgments. We show that quadratic regression significantly outperforms additive and multiplicative baselines and all other models in modelling semantic composition and identifying the non-compositional NCs. In short, the contributions of our work are: to empirically evaluate various composition functions ranging from simple to overly complex in order to find the most accurate function; to propose, to the best of our knowledge for the first time, a method of identifying non-compositional phrases as phrases for which a composition function cannot be readily learned; to propose learning decomposability as another criterion to detect non-compositionality; and to examine possible ways of improving the accuracy of the models by means of EM on hidden compositionality annotations.

2 Related Work

To the best of our knowledge, attempts to extract non-compositionality in computational linguistics go back to 1998. Tapanainen et al. (1998) propose a method to identify non-compositional verb-object collocations based on the semantic asymmetry of verb-object relation. They assume that in a verb-object idiomatic expression, the object is a more interesting element in the sense that if the object appears with one (or only a few) verbs in a large corpus, it presumably has an id-

iomatic nature. Lin (1999) argues that the mutual information between the constituents of a non-compositional phrase is significantly different from that of a phrase created by substituting the constituents of that phrase with their similar words. Their evaluation reveals a low precision ($16 - 39\%$) and recall ($14 - 21\%$). In any case this method is not able to discriminate non-compositional MWEs from collocational MWEs as they share the same property of non-substitutability (their constituents cannot be replaced with their synonyms). Baldwin et al. (2003) present a method that decides about the non-compositionality of English NCs and verb particle constructions by using latent semantic analysis to calculate the similarity between a MWE and its components. They argue that a higher similarity indicates a higher degree of compositionality. McCarthy et al. (2003) devise a number of measures based on comparison of the neighbors of phrasal verbs and their corresponding simplex verbs. They evaluate these measures by calculating their correlation with human compositionality judgments on a set of phrasal verbs. They show that some of the measures have significant correlations with human judgments. Venkatapathy and Joshi (2005) present a supervised model that benefits from both collocational and contextual information and ranks the MWE candidates based on their non-compositionality. Katz and Giesbrecht (2006) use distributional semantics and LSA as a model of context similarity to test whether the local context of a MWE can distinguish its idiomatic use from literal use. They further compare the context of a MWE with the context of its components and show that this can be used to decide whether the expression is idiomatic or not. Cook et al. (2007) is a relatively different work where the authors propose a syntactic approach to identify semantic non-compositionality of verb-noun MWEs. McCarthy et al. (2007) use various models of selectional preferences for detecting non-compositional verb-object pairs.

Reddy et al. (2011) employ the additive and multiplicative composition functions presented by Mitchell and Lapata (2008)[3] and several similarity based models to measure the compositionality of MWEs. Similarity based models measure the similarity of a MWE vector and sum/product of its constituents' vectors. Their evaluation (which is carried out on a set of 90 annotated NCs) shows that there is a relatively high correlation (Spearman ρ of between 0.51 and 0.71) between their models' predictions and human judgments on non-compositionality of English NCs, with weighted additive function outperforming all the other models. Kiela and Clark (2013) present a model of detecting non-compositionality based on the hypothesis that the average distance between a phrase vector and its substituted phrase vectors is related to its compositionality. In particular compositional phrases are less similar to their neighbors in semantic space. The distributional vectors representing the semantics of words were created using the standard window method and 50,000 most frequent context words. They show that their model slightly ($+0.014$ and $+0.007$) outperforms their baselines (Venkatapathy and Joshi, 2005; McCarthy et al., 2007).

All of the models mentioned so far are based on conventional[4] or count based vector space representation of the words. More recent works however are based on representation learning of word embeddings. Baroni and Zamparelli (2010) regard adjective as matrices and nouns as real-valued vectors for Italian adjective noun composition. They learn the adjective matrices by linear regression. In this work, however, every adjective is presented by a new matrix which leads to a large number of parameters. Socher et al. (2012) suggest that composition function is a matrix that multiplies on the word vectors, and Mikolov et al. (2013b) present a model of learning non-compositional phrases by calculating a data-driven score for certain frequent phrases (up to size two) and learn them as a whole. Salehi et al. (2015) borrow the word embeddings from (Mikolov et al., 2013a) to model the semantics of words and use several composition functions from (Mitchell and Lapata, 2008; Reddy et al., 2011) to predict the non-compositionality of MWEs. They compare the performance of word embeddings with conventional distributional vector representations and discover the superiority of word embeddings in predicting non-compositionality of MWEs.

[3] Mitchell and Lapata (2008) present an analysis of vector-based additive and multiplicative semantic composition models where each words is represented by its distributional vector. They conclude that multiplicative and combined models do a better job in modelling vector-based semantic composition than other models.

[4] Conventional or count based models of distributional similarity as oppose to word embeddings (Salehi et al., 2015; Baroni et al., 2014).

3 Representation of Words and Compounds

In order to represent words and compounds we use word embeddings, which are a form of vector space models. Vector space models represent the semantics of words and phrases with real valued vectors. Word embeddings have proven to be effective models of semantic representation of words and outperform the count-based models in various NLP tasks (Baroni et al., 2014; Collobert et al., 2011; Collobert and Weston, 2008; Yazdani and Popescu-Belis, 2013; Huang et al., 2012; Mikolov et al., 2013c). They have been successfully applied to semantic composition (Mikolov et al., 2013b) and outperformed the conventional count based contextual models in predicting non-compositionality of MWEs (Salehi et al., 2015).

In this work we use word embeddings of Mikolov et al. (2013a) to represent the semantics of words and compounds. We chose an English Wikipedia dump as our corpus. After filtering HTML tags and noise we POS-tagged the corpus and extracted $\approx 70k$ compounds whose frequency of occurrence was above 50. We learn the embeddings of these compounds as single tokens using the word2vec[5] bag-of-word model. We also learn the embeddings of the compounds of the evaluation set, plus the embeddings of all the compounds' component words. Compounds' sizes are restricted to two (i.e. bigrams) for the sake of simplicity and to respect the evaluation set standards. The compounds and word embeddings are then used as supervised signals to learn a composition function.

4 Supervised Models of Composition on Word Embeddings

After the unsupervised learning of word embeddings and candidate compound embeddings (see section 3), we use these embeddings as supervised signals in order to train our composition functions. The term supervised might be misleading as the models do not have any information about the compositionality of the compounds during the training phase, and in that respect it is unsupervised. To describe the models in a formal way, throughout the paper we use the following notations: d represents the size of embeddings, $\phi(w_i)$ represents embedding of w_i, and $\tilde{\phi}(w_i{-}w_j) = f(\phi(w_i), \phi(w_j))$ represents the learned embedding of bigram $w_i{-}w_j$ by the composition function f. The training error of bigram $w_i{-}w_j$ by f is $e_{ij} = \|\tilde{\phi}(w_i{-}w_j) - \phi(w_i, w_j)\|$, and $\|\|$ is norm 2. The composition functions are described in the following sections.

Given unsupervised embeddings for both words and compounds, a composition model is trained to map the word embeddings to the compound embeddings, with norm 2 error e_{ij} defined above. Then this same error for this same task (norm 2 between predicted and unsupervised compound embeddings) is used to measure non-compositionality. In other words, we learn a composition function (with several models) and identify non-compositional expressions as those for which the error of this composition function is high.

We explore various classes of composition functions of word embeddings, ranging from simple to complex, to find the most effective one. We want a composition function that is powerful enough to learn composition for compositional compounds, but simple enough that it fails to learn composition for non-compositional compounds. To this end, we investigate linear projections, polynomial projections, and neural networks. We try these models with and without sparsity regularisation, which reduces the number of non-zero parameters while otherwise keeping the complexity of the function that can be learned.

4.1 Linear Projection

In this model we assume that the embedding of a bigram is a linear projection of its component words' embeddings.

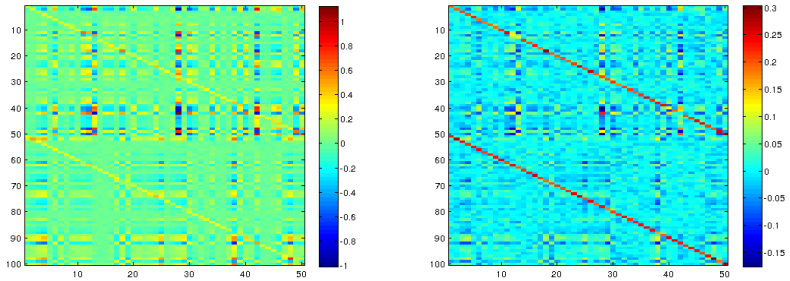
$$f(\phi(w_i), \phi(w_j)) = [\phi(w_i), \phi(w_j)]\theta_{2d\times d}$$

To train this function we optimize the least square error, which gives us a multi-variant linear regression.

$$\min_{\theta} \|[\phi(w_i), \phi(w_j)]\theta_{2d\times d} - \phi(w_i, w_j)\|$$

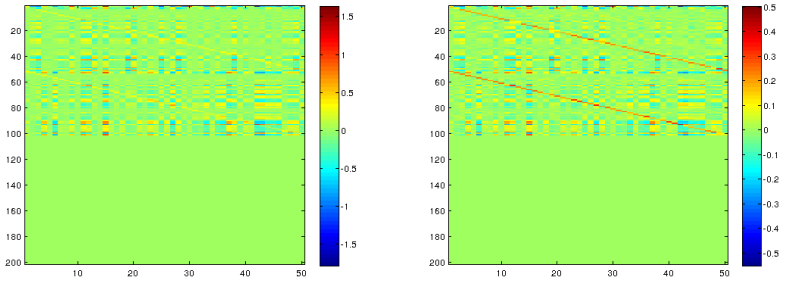
As mentioned earlier, a composition function that doesn't overfit the training data and induces a more meaningful error is more suitable for our purpose. One effective way of reducing overfitting and increasing generalization is by keeping only the important parameters of the model, which is done by enforcing sparsity on model parameters.

[5] https://code.google.com/p/word2vec/



(a) Linear Projection (b) Sparse Linear Projection

Figure 1: Linear transformation matrix of compositionality for embeddings of size 50



(a) Pure Quadratic (b) Sparse Pure Quadratic

Figure 2: Pure quadratic transformation matrix of compositionality for embeddings of size 50

In case of sparse linear projections, only a few elements of the projection matrix θ are non-zero. This means that not all dimensions of the latent space has a role in all dimensions of the compound embedding.

To apply sparsity on θ, we add a norm 1 penalty on it and add that to the least square optimization. This forms a multi-variant lasso regression (Tibshirani, 2011).

$$\min_{\theta} \|[\phi(w_i), \phi(w_j)]\theta - \phi(w_i, w_j)\| + \lambda|\theta|$$

Figure 1 shows the transformation matrices of linear projection and sparse linear projection. The two diagonals of the matrices correspond to the sum of the two embeddings, which we can see are the main component of the sparse function, and play an important role in the non-sparse one. We will see that despite being an important component of these functions, sum alone is not capable of accurately modelling semantic composition.

4.2 Polynomial Projection

Polynomial projection is a non-linear projection that assumes the relation between compound embedding and the component words' embeddings should be a polynomial of degree n. This can be viewed as a form of linear regression where first a polynomial transformation is applied to the input vector and then a linear projection is fitted. If ψ shows the polynomial transformation then we have:

$$f(\phi(w_i), \phi(w_j)) = \psi([\phi(w_i), \phi(w_j)])\theta$$

We couldn't successfully apply any polynomial beyond quadratic transformation without overfitting. The case of a quadratic ψ transformation is:

$$\psi(x) = \underbrace{x_1^2, \cdots x_n^2}_{\text{Pure quadratic}}, \underbrace{x_1x_2, \cdots x_{n-1}x_n}_{\text{interaction terms}}, \underbrace{x_1, \cdots x_n}_{\text{linear terms}}$$

Similar to the linear case we can have sparse version of the polynomial regression in which we allow the presence of only a few non-zero elements in the θ matrix. The sparsity regularizer is more important in the case of polynomial regression as we have many more parameters. The quadratic model is similar to Recursive Neural Tensor compositionality model of Socher et al. (2013). But in our model the tensor is symmetric around the diagonal. Figure 2 shows the pure quadratic transformation matrices.

4.3 Neural Networks

A feed forward neural network is a universal approximator (Cybenko, 1989): feed-forward network with a single hidden layer can approximate any continuous function, provided it has enough hidden units. Therefore we use neural networks as a powerful class of learning models to learn semantic composition. The number of hidden units gives us a measure to control expressiveness of our model.

$$f(\phi(w_i), \phi(w_j)) = \sigma([\phi(w_i), \phi(w_j)]W_{ih})W_{ho}$$

Similar to the previous models, we optionally impose sparsity over weight matrices of the neural network to be able to induce more meaningful learning errors.

4.4 Experimental Results

We evaluated the above models on the data set of Farahmand et al. (2015). They provide a set of 1042 English NCs with four non-compositionality judgments. The judgments are binary decisions taken by four experts about whether or not a compound is non-compositional. We calculate a vote-based non-compositionality score for each of the data set compounds by summing over its non-compositionality judgments. The neural network models are trained using stochastic gradient descent. We use the additive and multiplicative models of modelling composition and detecting non-compositionality presented by Salehi et al. (2015) and (Reddy et al., 2011) as state of the art baselines.

The results are shown in Table 1. The second column shows the correlation between different models' predictions and the annotated data in terms of Spearman ρ. The last three columns show the performance of different models in terms of Normalized Discounted Cumulative Gain (NDCG), F_1 score and and Precision at 100 (P@100). For these three scores we consider the problem of predicting non-compositional NCs a problem with a binary solution where we assume compounds (of the evaluation set) with at least two non-compositionality votes are non-compositional. *NDCG* assigns a higher score to a ranked list of compounds if the non-compositional ones are ranked higher in the list. F_1 column represents the maximum F_1 score on the top-n elements of the ranked list returned by the corresponding model for all n in $[1 - \text{size-of-ranked-list}]$. P@100 shows the precision at the first 100 compounds ranked as non-compositional. The models are listed in the order of complexity of the composition function. The addition-based baseline which was explored in a variety of previous work does not seem to be as powerful as the other models. It is outperformed by almost all learned models. In general, we can see that more complex functions tend to learn compositionality in a more effective way.

Model	Spearman ρ	NDCG	F_1	P@100
Additive model (Salehi et al., 2015); (Reddy et al., 2011)	20.83	81.39	36.95	43
Multiplicative model (Reddy et al., 2011)	9.18	76	35.61	22
Sparse Linear	37.58	84.25	46.40	48
Linear	38.09	84.25	46.41	49
Sparse Pure Quad.	37.85	84.11	47.05	48
Pure Quad.	38.57	84.68	47.01	47
Sparse Interaction	**41.03**	**85.82**	**48.71**	**54**
Interaction	40.25	85.69	48.64	50
Quadratic	40.25	85.59	48.34	49
Sparse NN (H=1000)	37.08	85.04	46.35	52
NN (H=1000)	37.51	84.97	45.47	51

Table 1: Results for each model's ability to predict non-compositionality.

As mentioned earlier, overly powerful learners overfit and do not produce meaningful errors for the detection task. Sparsity seems to address this issue by reducing the number of non-zero parameters while the function can still keep the complex terms if needed. In general sparse models show improvement over their non-sparse counterparts, specifically for more powerful models.

5 Auto-reconstructive Models

In this section, we investigate the hypothesis that we can detect non-compositionality better by not only modelling a composition function, but also modelling a decomposition function. For compositional compounds, given the meaning of the compound, the meaning of the two component words should be conditionally independent. We therefore assume that the decomposition function predicts the component words' vectors independently. Let us illustrate this assumption by examining the non-compositional compound *flag stop*. Given the semantics of this compound (a point at which a vehicle in public transportation stops only

on prearrangement or signal[6]), we can not readily predict one of its component words without knowing the other. Now consider the compositional compound *hip injury*. Given the semantics of this compound it is much easier to predict each of its component words independently.

In the previous section, the training signals came from the embeddings of the candidate compounds and their component words. In this section we extend our model such that it can benefit from more training signals. To this end, we formalize the assumption that a compositional compound is also decomposable as an auto-reconsructive model. We thus add this hypothesis to the learning process: a good composition function not only builds the semantics of the compound from the semantics of its component words, but it also allows the independent prediction of the semantics of its component words from the compound semantics. In the following sections we add this assumption to both linear projection (which encompasses polynomial) and Neural Network models.

5.1 Auto-reconstructive Linear Models

Let $Y_{M\times d}$ be a matrix whose rows are the precomputed compound embeddings, and $X_{M\times 2d}$ be a matrix whose rows are the concatenation of the embeddings for the words of these compounds. Let $A_{N\times 2d}$ be another matrix where every row contains the concatenation of the embeddings for the words of a compound, but this matrix includes many compounds for which we did not precompute compound embeddings. We assume that the rows of matrix A include the rows of matrix X. In linear models the auto-reconstructive objective function is as follows:

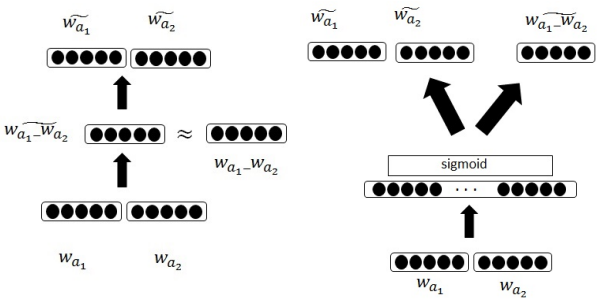
$$\min_{\theta,\theta'} \|X\theta - Y\| + \lambda\|A\theta\theta' - A\|$$

where λ is a meta-parameter for the importance of the auto-reconstruction in the objective. A schematic of this model is shown in Figure 3a.

We can look at this problem as the following weighted least square problem:

$$\min_{\theta,\theta'} \| \begin{pmatrix} X \\ A \end{pmatrix} \theta - \begin{pmatrix} Y \\ A\theta'^T(\theta'\theta'^T)^{-1} \end{pmatrix} \| \begin{pmatrix} 1 \\ \vdots \\ \lambda \end{pmatrix}$$

In the above matrix formula we transformed the auto-reconstruction part of the objective to a pseudo regressand of the least square. To solve this optimization we design an efficient alternating least squares algorithm.



(a) Linear auto-reconstructive (b) NN Auto-reconstructive

Figure 3: Auto-reconstructive linear and neural network models

First we initialize θ_0 to be the answer of the original multi-variant linear regression, $\theta_0 = X \setminus Y$ where $X \setminus = (X^T X)^{-1} X^T$ is the pseudoinverse of X. Let us assume W is the diagonal matrix with first M elements of the diagonal being 1 and the remaining N being λ. We alternate between the following formulas until the algorithm converges. First we approximate the next θ' based on the current approximation of θ, then we use this value of θ' to calculate the pseudo regressand part of the least square. In the final step we solve the weighted least square for this new regressand matrix and continue iterating these stages until the algorithm converges.

$$\theta'_t = (A\theta_t) \setminus A \tag{1}$$

$$X_2 = \begin{pmatrix} X \\ A \end{pmatrix}$$

$$Y_2 = \begin{pmatrix} Y \\ A\theta'^T_{t-1}(\theta'_{t-1}\theta'^T_{t-1})^{-1} \end{pmatrix} \tag{2}$$

$$\theta_t = (X_2^T W X_2)^{-1}(X_2)^T W Y_2 \tag{3}$$

The above algorithm can also be used in the case of polynomial regression. The only thing that needs to be done is to replace X and A by their polynomial transformations.

5.2 Auto-reconstructive Neural Networks

The auto-reconstructive neural network follows the same idea. The objective function changes to:

$$\min_{W_{ih},W_{hi},W_{oh}} \|\sigma(XW_{ih})W_{ho} - Y\| + \lambda\|\sigma(AW_{ih})W_{hi} - A\| \tag{4}$$

[6]Definition taken from Merriam-Webster Dictionary

Composition	Spearman ρ	NDCG	F_1	P@100
Linear	38.09	84.25	46.41	49
Linear+ auto	37.52	84.55	46.55	49
Interaction	**40.25**	85.69	48.64	50
Interaction+ auto	39.29	**85.71**	48.95	**56**
NN (H=1000)	37.40	84.50	46.34	49
NN (H=1000) + auto	39.98	85.17	**49.12**	55

Table 2: Results comparing the auto-reconstructive models' ability to predict non-compositionality.

Composition	Spearman ρ	NDCG	F_1	P@100
Linear	38.09	84.25	46.41	49
Linear+ LA	37.80	84.60	46.29	48
Interaction	40.25	85.69	**48.64**	50
Interaction+ LA	**40.56**	86.30	48.34	51
NN (H=1000)	37.40	84.50	46.34	49
NN (H=1000) + LA	39.23	**85.36**	48.15	**55**

Table 3: Results comparing the latent annotation models' ability to predict non-compositionality.

Figure 3b shows the schematic of this model. We optimize this objective using stochastic gradient descent with early stopping. The results are shown in Table 2. We choose the first $300K$ frequent noun-noun compounds from the corpus in order to build matrix A. Each row of A created by concatenating the component words vectors. The results show that the auto-reconstructive models generally improve over their counterparts. As mentioned earlier, the improvement comes from two facts. On the one hand we increase the training signals by implementing the decomposability hypothesis. On the other hand, the auto-reconstructive model enables us to exploit more data in addition to the candidate compounds. There is almost no improvement in the case of linear model because this model does not have enough learning capacity to benefit from a higher number of training signals.

6 Non-compositionality Detection Using Latent Annotations

All the models discussed in this paper are unsupervised since they don't have any access to labels specifying compositionality of compounds. The above models simply assume that most compounds are compositional, and therefore train their composition and decomposition functions on all compounds. In this section we incorporate in the models an intrinsic uncertainty about the compositionality annotation of the training set.

The best (optimum) composition function is the one that fits well all the compositional compounds and does not fit the non-compositional ones. But we assume that we do not have training labels indicating compositionality. To overcome this uncertainty and improve the learning process, we introduce latent compositionality labels to the model. We assume each candidate compound has a latent annotation, 1 or 0, showing whether or not it is compositional. Let us assume a non-compositionality detection system that returns B non-compositional candidates that should have their own lexical unit and parameters. The objective of this composition function training is to minimize the error of compositional compounds and not the error of non-compositional ones. In order to implement this objective we use the following loss function:

$$\min_{\lambda_{ij},\theta} \sum_{ij} \lambda_{ij} e_{ij}^2$$

$$\text{s.t } \lambda_{ij} \in \{0,1\},$$

$$\sum_{ij} \lambda_{ij} = N - B$$

where λ_{ij} represents the hidden compositionality annotation and e_{ij} is again the learning error for the pair $w_i{-}w_j$. We want to find the B points such that annotating them as non-compositional results in the minimum error of this objective. The algorithm that alternates between optimizing the composition learning and the hidden annotations eventually converges to this solution.

If the errors are fixed, the B compounds with the biggest errors are the answers to the non-compositional annotation optimization of that iteration. Therefore to solve this optimization we follow an EM-like algorithm: First we set all λ_{ij} to 1 and perform the optimization on the composition function. Then we sort the compounds by their error and set the λ_{ij} of the biggest B elements to 0, and the rest to 1. In other words we assume the compounds with big error are presumably non-compositional according to what we know until that iteration. We continue alternating between training the composition function and annotating high error points until the algorithm reaches convergence. The results are shown in Table 3. Models that use latent annotations clearly outperform their counterparts, especially in terms

of precision at 100. This is expected since at training time we consider a model that returns B non-compositional compounds and therefore precision at 100 is optimized. The latent annotations do not improve the linear model since the model is simple and there is not much room to improve its learning.

7 Conclusions

We proposed a framework to detect non-compositional compounds as the compounds that stand out as outliers in the process of learning compositionality of English noun compounds. We proposed and evaluated a range of functions with a variety of complexities that model semantic composition. We showed that learners such as polynomial projection and neural networks which are distinctly more complex than commonly used additive and multiplicative functions can model semantic composition more effectively. We showed that a function as complex as quadratic projection is a better learner of compositionality than simpler models. We further showed that enforcing sparsity is an effective way of learning a complex composition function while avoiding overfitting and producing meaningful learning errors. Furthermore, we improved our models by incorporating an auto-reconstructive loss function that enables us to benefit from more training signals and cover more data. Finally, we addressed the intrinsic label uncertainty in training data by considering latent annotations, and showed that it can further improve the results.

Acknowledgments

This research was partially funded by Hasler foundation project no. 15019, "Deep Neural Network Dependency Parser for Context-aware Representation Learning".

References

Timothy Baldwin and Su Nam Kim. 2010. Multiword expressions. *Handbook of Natural Language Processing, second edition. Morgan and Claypool*.

Timothy Baldwin, Colin Bannard, Takaaki Tanaka, and Dominic Widdows. 2003. An empirical model of multiword expression decomposability. In *Proceedings of the ACL 2003 workshop on Multiword expressions: analysis, acquisition and treatment-Volume 18*, pages 89–96. Association for Computational Linguistics.

Marco Baroni and Roberto Zamparelli. 2010. Nouns are vectors, adjectives are matrices: Representing adjective-noun constructions in semantic space. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1183–1193. Association for Computational Linguistics.

Marco Baroni, Georgiana Dinu, and Germán Kruszewski. 2014. Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 238–247.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 160–167, New York, NY, USA. ACM.

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

Paul Cook, Afsaneh Fazly, and Suzanne Stevenson. 2007. Pulling their weight: Exploiting syntactic forms for the automatic identification of idiomatic expressions in context. In *Proceedings of the workshop on a broader perspective on multiword expressions*, pages 41–48. Association for Computational Linguistics.

G. Cybenko. 1989. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314.

Meghdad Farahmand, Aaron Smith, and Joakim Nivre. 2015. A multiword expression data set: Annotating non-compositionality and conventionalization for english noun compounds. In *Proceedings of the 11th Workshop on Multiword Expressions (MWE-NAACL 2015)*. Association for Computational Linguistics.

Eric H. Huang, Richard Socher, Christopher D. Manning, and Andrew Y. Ng. 2012. Improving Word Representations via Global Context and Multiple Word Prototypes. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Graham Katz and Eugenie Giesbrecht. 2006. Automatic identification of non-compositional multi-word expressions using latent semantic analysis. In *Proceedings of the Workshop on Multiword Expressions: Identifying and Exploiting Underlying Properties*, MWE '06, pages 12–19, Stroudsburg, PA, USA. Association for Computational Linguistics.

Douwe Kiela and Stephen Clark. 2013. Detecting compositionality of multi-word expressions using nearest neighbours in vector space models. In *EMNLP*, pages 1427–1432.

Dekang Lin. 1999. Automatic identification of non-compositional phrases. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics on Computational Linguistics*, ACL '99, pages 317–324, Stroudsburg, PA, USA. Association for Computational Linguistics.

Diana McCarthy, Bill Keller, and John Carroll. 2003. Detecting a continuum of compositionality in phrasal verbs. In *Proceedings of the ACL-SIGLEX Workshop on Multiword Expressions: Analysis, Acqusation and Treatment*, pages 73–80.

Diana McCarthy, Sriram Venkatapathy, and Aravind K Joshi. 2007. Detecting compositionality of verb-object combinations using selectional preferences. In *EMNLP-CoNLL*, pages 369–379.

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

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

Tomas Mikolov, Wen tau Yih, and Geoffrey Zweig. 2013c. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT-2013)*. Association for Computational Linguistics, May.

Jeff Mitchell and Mirella Lapata. 2008. Vector-based models of semantic composition. In *Proceedings of ACL-08: HLT*, pages 236–244.

Siva Reddy, Diana McCarthy, and Suresh Manandhar. 2011. An empirical study on compositionality in compound nouns. In *IJCNLP*, pages 210–218.

Ivan A Sag, Timothy Baldwin, Francis Bond, Ann Copestake, and Dan Flickinger. 2002. Multiword expressions: A pain in the neck for nlp. In *Computational Linguistics and Intelligent Text Processing*, pages 1–15. Springer.

Bahar Salehi, Paul Cook, and Timothy Baldwin. 2015. A word embedding approach to predicting the compositionality of multiword expressions. In *Proceedings of NAACL HLT*. Association for Computational Linguistics.

Richard Socher, Brody Huval, Christopher D Manning, and Andrew Y Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1201–1211. Association for Computational Linguistics.

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

Pasi Tapanainen, Jussi Piitulainen, and Timo Järvinen. 1998. Idiomatic object usage and support verbs. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics - Volume 2*, ACL '98, pages 1289–1293, Stroudsburg, PA, USA. Association for Computational Linguistics.

Robert Tibshirani. 2011. Regression shrinkage and selection via the lasso: a retrospective. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(3):273–282.

Sriram Venkatapathy and Aravind K. Joshi. 2005. Measuring the relative compositionality of verb-noun (v-n) collocations by integrating features. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pages 899–906, Stroudsburg, PA, USA. Association for Computational Linguistics.

Dominic Widdows. 2008. Semantic vector products: Some initial investigations. In *Second AAAI Symposium on Quantum Interaction*, volume 26, page 28th.

Majid Yazdani and Andrei Popescu-Belis. 2013. Computing text semantic relatedness using the contents and links of a hypertext encyclopedia. *Artif. Intell.*, 194:176–202.

Solving General Arithmetic Word Problems

Subhro Roy
University of Illinois,
Urbana Champaign
sroy9@illinois.edu

Dan Roth
University of Illinois,
Urbana Champaign
danr@illinois.edu

Abstract

This paper presents a novel approach to automatically solving arithmetic word problems. This is the first algorithmic approach that can handle arithmetic problems with multiple steps and operations, without depending on additional annotations or predefined templates. We develop a theory for expression trees that can be used to represent and evaluate the target arithmetic expressions; we use it to uniquely decompose the target arithmetic problem to multiple classification problems; we then compose an expression tree, combining these with world knowledge through a constrained inference framework. Our classifiers gain from the use of *quantity schemas* that supports better extraction of features. Experimental results show that our method outperforms existing systems, achieving state of the art performance on benchmark datasets of arithmetic word problems.

1 Introduction

In recent years there is growing interest in understanding natural language text for the purpose of answering science related questions from text as well as quantitative problems of various kinds. In this context, understanding and solving arithmetic word problems is of specific interest. Word problems arise naturally when reading the financial section of a newspaper, following election coverage, or when studying elementary school arithmetic word problems. These problems pose an interesting challenge to the NLP community, due to its concise and relatively straightforward text, and seemingly simple semantics. Arithmetic word problems are usually directed towards elementary school students, and can be solved by combining the numbers mentioned in text with basic operations (addition, subtraction, multiplication, division). They are simpler than algebra word problems which require students to identify variables, and form equations with these variables to solve the problem.

Initial methods to address arithmetic word problems have mostly focussed on subsets of problems, restricting the number or the type of operations used (Roy et al., 2015; Hosseini et al., 2014) but could not deal with multi-step arithmetic problems involving all four basic operations. The template based method of (Kushman et al., 2014), on the other hand, can deal with all types of problems, but implicitly assumes that the solution is generated from a set of predefined equation templates.

In this paper, we present a novel approach which can solve a general class of arithmetic problems without predefined equation templates. In particular, it can handle multiple step arithmetic problems as shown in Example 1.

Example 1
Gwen was organizing her book case making sure each of the shelves had exactly 9 *books on it. She has* 2 *types of books - mystery books and picture books. If she had* 3 *shelves of mystery books and* 5 *shelves of picture books, how many books did she have in total?*

The solution involves understanding that the number of shelves needs to be summed up, and that the total number of shelves needs to be multiplied by the number of books each shelf can hold. In addition, one has to understand that the number "2" is not a direct part of the solution of the problem.

While a solution to these problems eventually requires composing multi-step numeric expressions from text, we believe that directly predicting this complex expression from text is not feasible.

At the heart of our technical approach is the novel notion of an *Expression Tree*. We show that the arithmetic expressions we are interested in can always be represented using an Expression Tree that has some unique decomposition properties. This allows us to decompose the problem of mapping the text to the arithmetic expression to a collection of simple prediction problems, each determining the lowest common ancestor operation between a pair of quantities mentioned in the problem. We then formulate the decision problem of composing the final expression tree as a joint inference problem, via an objective function that consists of all these decomposed prediction problems, along with legitimacy and background knowledge constraints.

Learning to generate the simpler decomposed expressions allows us to support generalization across problems types. In particular, our system could solve Example 1 even though it has never seen a problem that requires both addition and multiplication operations.

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1743–1752,
Lisbon, Portugal, 17-21 September 2015.

We also introduce a second concept, that of *quantity schema*, that allows us to focus on the information relevant to each quantity mentioned in the text. We show that features extracted from quantity schemas help reasoning effectively about the solution. Moreover, quantity schemas help identify unnecessary text snippets in the problem text. For instance, in Example 2, the information that "Tom washed cars over the weekend" is irrelevant; he could have performed any activity to earn money. In order to solve the problem, we only need to know that he had $76 last week, and now he has $86.

Example 2
Last week Tom had $74. He washed cars over the weekend and now has $86. How much money did he make from the job?

We combine the classifiers' decisions using a constrained inference framework that allows for incorporating world knowledge as constraints. For example, we deliberatively incorporate the information that, if the problems asks about an "amount", the answer must be positive, and if the question starts with "how many", the answer will most likely be an integer.

Our system is evaluated on two existing datasets of arithmetic word problems, achieving state of the art performance on both. We also create a new dataset of multistep arithmetic problems, and show that our system achieves competitive performance in this challenging evaluation setting.

The next section describes the related work in the area of automated math word problem solving. We then present the theory of expression trees and our decomposition strategy that is based on it. Sec. 4 presents the overall computational approach, including the way we use quantity schemas to learn the mapping from text to expression tree components. Finally, we discuss our experimental study and conclude.

2 Related Work

Previous work in automated arithmetic problem solvers has focussed on a restricted subset of problems. The system described in (Hosseini et al., 2014) handles only addition and subtraction problems, and requires additional annotated data for verb categories. In contrast, our system does not require any additional annotations and can handle a more general category of problems. The approach in (Roy et al., 2015) supports all four basic operations, and uses a pipeline of classifiers to predict different properties of the problem. However, it makes assumptions on the number of quantities mentioned in the problem text, as well as the number of arithmetic steps required to solve the problem. In contrast, our system does not have any such restrictions, effectively handling problems mentioning multiple quantities and requiring multiple steps. Kushman's approach to automatically solving algebra word problems (Kushman et al., 2014) might be the most related to ours. It tries to map numbers from the problem text to predefined equation templates. However, they implicitly assume that similar equation forms have been seen in the training data. In contrast, our system can perform competitively, even when it has never seen similar expressions in training.

There is a recent interest in understanding text for the purpose of solving scientific and quantitative problems of various kinds. Our approach is related to work in understanding and solving elementary school standardized tests (Clark, 2015). The system described in (Berant et al., 2014) attempts to automatically answer biology questions, by extracting the structure of biological processes from text. There has also been efforts to solve geometry questions by jointly understanding diagrams and associated text (Seo et al., 2014). A recent work (Sadeghi et al., 2015) tries to answer science questions by visually verifying relations from images.

Our constrained inference module falls under the general framework of Constrained Conditional Models (CCM) (Chang et al., 2012). In particular, we use the $L + I$ scheme of CCMs, which predicts structured output by independently learning several simple components, combining them at inference time. This has been successfully used to incorporate world knowledge at inference time, as well as getting around the need for large amounts of jointly annotated data for structured prediction (Roth and Yih, 2005; Punyakanok et al., 2005; Punyakanok et al., 2008; Clarke and Lapata, 2006; Barzilay and Lapata, 2006; Roy et al., 2015).

3 Expression Tree and Problem Decomposition

We address the problem of automatically solving arithmetic word problems. The input to our system is the problem text P, which mentions n quantities $q_1, q_2, \ldots, q_n$. Our goal is to map this problem to a read-once arithmetic expression E that, when evaluated, provides the problem's solution. We define a read-once arithmetic expression as one that makes use of each quantity at most once. We say that E is a *valid* expression, if it is such a Read-Once arithmetic expression, and we only consider in this work problems that *can be* solved using valid expressions (it's possible that they can be solved also with invalid expressions).

An expression tree $\mathcal{T}$ for a valid expression E is a binary tree whose leaves represent quantities, and each internal node represents one of the four basic operations. For a non-leaf node n, we represent the operation associated with it as $\odot(n)$, and its left and right child as $lc(n)$ and $rc(n)$ respectively. The numeric value of the quantity associated with a leaf node n is denoted as $\mathrm{Q}(n)$. Each node n also has a value associated with it, represented as $\textsc{Val}(n)$, which can be computed in a

recursive way as follows:

$$\textsc{Val}(n) = \begin{cases} \mathrm{Q}(n) & \text{if } n \text{ is a leaf} \\ \textsc{Val}(lc(n)) \odot(n) \textsc{Val}(rc(n)) & \text{otherwise} \end{cases} \quad (1)$$

For any expression tree $\mathcal{T}$ for expression E with root node n_{root}, the value of $\textsc{Val}(n_{root})$ is exactly equal to the numeric value of the expression E. Therefore, this gives a natural representation of numeric expressions, providing a natural parenthesization of the numeric expression. Fig 1 shows an example of an arithmetic problem with solution expression and an expression tree for the solution expression.

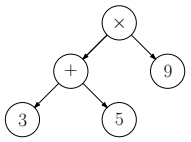
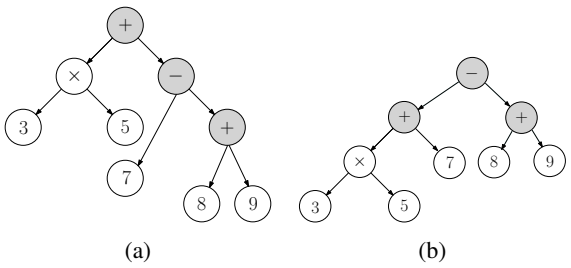
Problem	
Gwen was organizing her book case making sure each of the shelves had exactly 9 *books on it. She has* 2 *types of books - mystery books and picture books. If she had* 3 *shelves of mystery books and* 5 *shelves of picture books, how many books did she have total?*	
Solution	Expression Tree of Solution
$(3 + 5) \times 9 = 72$	

Figure 1: An arithmetic word problem, solution expression and the corresponding expression tree

Definition An expression tree $\mathcal{T}$ for a valid expression E is called ***monotonic*** if it satisfies the following conditions:

1. If an addition node is connected to a subtraction node, then the subtraction node is the parent.
2. If a multiplication node is connected to a division node, then the division node is the parent.

Fig 2 shows two different expression trees for the same expression. Fig 2b is monotonic whereas fig 2a is not.



(a) (b)

Figure 2: Two different expression trees for the numeric expression $(3 \times 5) + 7 - 8 - 9$. The right one is monotonic, whereas the left one is not.

Our decomposition relies on the idea of monotonic expression trees. We try to predict for each pair of quantities q_i, q_j, the operation at the lowest common ancestor (LCA) node of the monotonic expression tree for the solution expression. We also predict for each quantity, whether it is relevant to the solution. Finally, an inference module combines all these predictions.

In the rest of the section, we show that for any pair of quantities q_i, q_j in the solution expression, any monotonic tree for the solution expression has the same LCA operation. Therefore, predicting the LCA operation becomes a multiclass classification problem.

The reason that we consider the monotonic representation of the expression tree is that different trees could otherwise give different LCA operation for a given pair of quantities. For example, in Fig 2, the LCA operation for quantities 5 and 8 can be $+$ or $-$, depending on which tree is considered.

Definition We define an ***addition-subtraction chain*** of an expression tree to be the maximal connected set of nodes labeled with addition or subtraction.

The nodes of an addition-subtraction (AS) chain C represent a set of terms being added or subtracted. These terms are sub-expressions created by subtrees rooted at neighboring nodes of the chain. We call these terms the ***chain terms*** of C, and the whole expression, after node operations have been applied to the chain terms, the ***chain expression*** of C. For example, in fig 2, the shaded nodes form an addition-subtraction chain. The chain expression is $(3\times5)+7-8-9$, and the chain terms are 3×5, 7, 8 and 9. We define a ***multiplication-division (MD) chain*** in a similar way.

Theorem 3.1. *Every valid expression can be represented by a monotonic expression tree.*

Proof. The proof is procedural, that is, we provide a method to convert any expression tree to a monotonic expression tree for the same expression. Consider a non-monotonic expression tree E, and without loss of generality, assume that the first condition for monotonicity is not valid. Therefore, there exists an addition node n_i and a subtraction node n_j, and n_i is the parent of n_j. Consider an addition-subtraction chain C which includes n_i, n_j. We now replace the nodes of C and its subtrees in the following way. We add a single subtraction node n_-. The left subtree of n_- has all the addition chain terms connected by addition nodes, and the right subtree of n_- has all the subtraction chain terms connected by addition nodes. Both subtrees of n_- only require addition nodes, hence monotonicity condition is satisfied. We can construct the monotonic tree in Fig 2b from the non-monotonic tree of Fig 2a using this procedure. The addition chain terms are 3×5 and 7, and the subtraction chain terms are 8 and 9. As as was described above, we introduce the root subtraction node in Fig 2b and attach the addition chain terms to the left and the subtraction chain terms to the right. The same line of reasoning can be used to handle the second condition with multiplication and division replacing addition and subtraction, respectively.

□

Theorem 3.2. *Consider two valid expression trees $\mathcal{T}_1$ and $\mathcal{T}_2$ for the same expression E. Let C_1, C_2 be the chain containing the root nodes of $\mathcal{T}_1$ and T_2 respectively. The chain type (addition-subtraction or multiplication-division) as well as the the set of chain terms of C_1 and C_2 are identical.*

Proof. We first prove that the chains containing the roots are both AS or both MD, and then show that the chain terms are also identical.

We prove by contradiction that the chain type is same. Let C_1's type be "addition-subtraction" and C_2's type be "multiplication-division" (without loss of generality). Since both C_1 and C_2 generate the same expression E, we have that E can be represented as sum (or difference) of two expressions as well as product(or division) of two expressions. Transforming a sum (or difference) of expressions to a product (or division) requires taking common terms from the expressions, which imply that the sum (or difference) had duplicate quantities. The opposite transformation adds same term to various expressions leading to multiple uses of the same quantity. Therefore, this will force at least one of C_1 and C_2 to use the same quantity more than once, violating validity.

We now need to show that individual chain terms are also identical. Without loss of generality, let us assume that both C_1 and C_2 are "addition-subtraction" chains. Suppose the chain terms of C_1 and C_2 are not identical. The chain expression for both the chains will be the same (since they are root chains, the chain expressions has to be the same as E). Let the chain expression for C_1 be $\sum_i t_i - \sum_i t'_i$, where t_i's are the addition chain terms and t'_i are the subtraction chain terms. Similarly, let the chain expression for C_2 be $\sum_i s_i - \sum_i s'_i$. We know that $\sum_i t_i - \sum_i t'_i = \sum_i s_i - \sum_i s'_i$, but the set of t_i's and t'_i's is not the same as the set of s_i and s'_i's. However it should be possible to transform one form to the other using mathematical manipulations. This transformation will involve taking common terms, or multiplying two terms, or both. Following previous explanation, this will force one of the expressions to have duplicate quantities, violating validity. Hence, the chain terms of C_1 and C_2 are identical.

□

Consider an expression tree $\mathcal{T}$ for a valid expression E. For a distinct pair of quantities q_i, q_j participating in expression E, we denote by n_i, n_j the leaves of the expression tree $\mathcal{T}$ representing q_i, q_j, respectively. Let $n_{LCA}(q_i, q_j; \mathcal{T})$ to be the lowest common ancestor node of n_i and n_j. We also define $order(q_i, q_j; \mathcal{T})$ to be true if n_i appears in the left subtree of $n_{LCA}(q_i, q_j; \mathcal{T})$ and n_j appears in the right subtree of $n_{LCA}(q_i, q_j; \mathcal{T})$ and set $order(q_i, q_j; \mathcal{T})$ to false otherwise. Finally we define $\odot_{LCA}(q_i, q_j; \mathcal{T})$ for a pair of quantities q_i, q_j as follows :

$$\odot_{LCA}(q_i, q_j, \mathcal{T}) = \begin{cases} + & \text{if } \odot(n_{LCA}(q_i, q_j; \mathcal{T})) = + \\ \times & \text{if } \odot(n_{LCA}(q_i, q_j; \mathcal{T})) = \times \\ - & \text{if } \odot(n_{LCA}(q_i, q_j; \mathcal{T})) = - \text{ and } order(q_i, q_j; \mathcal{T}) = true \\ -_{reverse} & \text{if } \odot(n_{LCA}(q_i, q_j; \mathcal{T})) = - \text{ and } order(q_i, q_j; \mathcal{T}) = false \\ \div & \text{if } \odot(n_{LCA}(q_i, q_j; \mathcal{T})) = \div \text{ and } order(q_i, q_j; \mathcal{T}) = true \\ \div_{reverse} & \text{if } \odot(n_{LCA}(q_i, q_j; \mathcal{T})) = \div \text{ and } order(q_i, q_j; \mathcal{T}) = false \end{cases} \quad (2)$$

Definition Given two expression trees $\mathcal{T}_1$ and $\mathcal{T}_2$ for the same expression E, $\mathcal{T}_1$ is *LCA-equivalent* to $\mathcal{T}_2$ if for every pair quantities q_i, q_j in the expression E, we have $\odot_{LCA}(q_i, q_j, \mathcal{T}_1) = \odot_{LCA}(q_i, q_j, \mathcal{T}_2)$.

Theorem 3.3. *All monotonic expression trees for an expression are LCA-equivalent to each other.*

Proof. We prove by induction on the number of quantities used in an expression. For all expressions E with 2 quantities, there exists only one monotonic expression tree, and hence, the statement is trivially true. This satisfies our base case.

For the inductive case, we assume that for all expressions with $k < n$ quantities, the theorem is true. Now, we need to prove that any expression with n nodes will also satisfy the property.

Consider a valid (as in all cases) expression E, with monotonic expression trees $\mathcal{T}_1$ and $\mathcal{T}_2$. From theorem 3.2, we know that the chains containing the roots of $\mathcal{T}_1$ and $\mathcal{T}_2$ have identical type and terms. Given two quantities q_i, q_j of E, the lowest common ancestor of both $\mathcal{T}_1$ and $\mathcal{T}_2$ will either both belong to the chain containing the root, or both belong to one of the chain terms. If the LCA node is part of the chain for both $\mathcal{T}_1$ and $\mathcal{T}_2$, monotonic property ensures that the LCA operation will be identical. If the LCA node is part of a chain term (which is an expression tree of size less than n), the property is satisfied by induction hypothesis.

□

The theory just presented suggests that it is possible to uniquely decompose the overall problem to simpler steps and this will be exploited in the next section.

4 Mapping Problems to Expression Trees

Given the uniqueness properties proved in Sec. 3, it is sufficient to identify the operation between any two relevant quantities in the text, in order to determine the unique valid expression. In fact, identifying the operation between *any* pair of quantities provides much needed redundancy given the uncertainty in identifying the operation from text, and we exploit it in our final joint inference.

Consequently, our overall method proceeds as follows: given the problem text P, we detect quantities $q_1, q_2, \ldots, q_n$. We then use two classifiers, one for relevance and other to predict the LCA operations for a monotonic expression tree of the solution. Our training makes use of the notion of quantity schemas, which we describe in Section 4.2. The distributional output of these classifiers is then used in a joint inference procedure that determines the final expression tree.

Our training data consists of problem text paired with a monotonic expression tree for the solution expression. Both the relevance and LCA operation classifiers are trained on gold annotations.

4.1 Global Inference for Expression Trees

In this subsection, we define the scoring functions corresponding to the decomposed problems, and show how we combine these scores to perform global inference. For a problem P with quantities $q_1, q_2, \ldots, q_n$, we define the following scoring functions:

1. $\textsc{Pair}(q_i, q_j, op)$: Scores the likelihood of $\odot_{LCA}(q_i, q_j, \mathcal{T}) = op$, where $\mathcal{T}$ is a monotone expression tree of the solution expression of P. A multiclass classifier trained to predict LCA operations (Section 4.4) can provide these scores.

2. $\textsc{Irr}(q)$: Scores the likelihood of quantity q being an irrelevant quantity in P, that is, q is not used in creating the solution. A binary classifier trained to predict whether a quantity q is relevant or not (Section 4.3), can provide these scores.

For an expression E, let $\mathcal{I}(E)$ be the set of all quantities in P which are not used in expression E. Let $\mathcal{T}$ be a monotonic expression tree for E. We define $\text{Score}(E)$ of an expression E in terms of the above scoring functions and a scaling parameter w_{IRR} as follows:

$$\text{Score}(E) = w_{\text{IRR}} \sum_{q \in \mathcal{I}(E)} \textsc{Irr}(q) + \sum_{q_i, q_j \notin \mathcal{I}(E)} \textsc{Pair}(q_i, q_j, \odot_{LCA}(q_i, q_j, \mathcal{T})) \quad (3)$$

Our final expression tree is an outcome of a constrained optimization process, following (Roth and Yih, 2004; Chang et al., 2012). Our objective function makes use of the scores returned by $\textsc{Irr}(\cdot)$ and $\textsc{Pair}(\cdot)$ to determine the expression tree and is constrained by legitimacy and background knowledge constraints, detailed below.

1. **Positive Answer**: Most arithmetic problems asking for amounts or number of objects usually have a positive number as an answer. Therefore, while searching for the best scoring expression, we reject expressions generating negative answer.

2. **Integral Answer**: Problems with questions such as 'how many" usually expect integral solutions. We only consider integral solutions as legitimate outputs for such problems.

Let $\mathcal{C}$ be the set of valid expressions that can be formed using the quantities in a problem P, and which satisfy the above constraints. The inference algorithm now becomes the following:

$$\arg\max_{E \in \mathcal{C}} \text{Score}(E) \quad (4)$$

The space of possible expressions is large, and we employ a beam search strategy to find the highest scoring constraint satisfying expression (Chang et al., 2012). We construct an expression tree using a bottom up approach, first enumerating all possible sets of irrelevant quantities, and next over all possible expressions, keeping the top k at each step. We give details below.

1. **Enumerating Irrelevant Quantities**: We generate a state for all possible sets of irrelevant quantities, ensuring that there is at least two relevant quantities in each state. We refer to each of the relevant quantities in each state as a term. Therefore, each state can be represented as a set of terms.

2. **Enumerating Expressions**: For generating a next state S' from S, we choose a pair of terms t_i and t_j in S and one of the four basic operations, and form a new term by combining terms t_i and t_j with the operation. Since we do not know which of the possible next states will lead to the optimal goal state, we enumerate all possible next states (that is, enumerate all possible pairs of terms and all possible operations); we prune the beam to keep only the top k candidates. We terminate when all the states in the beam have exactly one term.

Once we have a top k list of candidate expression trees, we choose the highest scoring tree which satisfies the constraints. However, there might not be any tree in the beam which satisfies the constraints, in which case, we choose the top candidate in the beam. We use $k = 200$ in our experiments.

In order to choose the value for the w_{IRR}, we search over the set $\{10^{-6}, 10^{-4}, 10^{-2}, 1, 10^2, 10^4, 10^6\}$, and choose the parameter setting which gives the highest accuracy on the training data.

4.2 Quantity Schema

In order to generalize across problem types as well as over simple manipulations of the text, it is necessary to train our system only with relevant information from the problem text. E.g., for the problem in example 2, we do not want to take decisions based on how Tom earned money. Therefore, there is a need to extract the relevant information from the problem text. To this end, we introduce the concept of a *quantity schema* which we extract for each quantity in the problem's text. Along with the question asked, the quantity

schemas provides all the information needed to solve most arithmetic problems.

A quantity schema for a quantity q in problem P consists of the following components.

1. **Associated Verb** For each quantity q, we detect the verb associated with it. We traverse up the dependency tree starting from the quantity mention, and choose the first verb we reach. We used the easy first dependency parser (Goldberg and Elhadad, 2010).

2. **Subject of Associated Verb** We detect the noun phrase, which acts as subject of the associated verb (if one exists).

3. **Unit** We use a shallow parser to detect the phrase p in which the quantity q is mentioned. All tokens of the phrase (other than the number itself) are considered as unit tokens. Also, if p is followed by the prepositional phrase "of" and a noun phrase (according to the shallow parser annotations), we also consider tokens from this second noun phrase as unit tokens. Finally, if no unit token can be extracted, we assign the unit of the neighboring quantities as the unit of q (following previous work (Hosseini et al., 2014)).

4. **Related Noun Phrases** We consider all noun phrases which are connected to the phrase p containing quantity q, with NP-PP-NP attachment. If only one quantity is mentioned in a sentence, we consider all noun phrases in it as related.

5. **Rate** We determine whether quantity q refers to a *rate* in the text, as well as extract two unit components defining the rate. For example, "7 kilometers per hour" has two components "kilometers" and "hour". Similarly, for sentences describing unit cost like "Each egg costs 2 dollars", "2" is a rate, with units "dollars" and "egg".

In addition to extracting the quantity schemas for each quantity, we extract the surface form text which poses the question. For example, in the question sentence, "How much will John have to pay if he wants to buy 7 oranges?", our extractor outputs "How much will John have to pay" as the question.

4.3 Relevance Classifier

We train a binary SVM classifier to determine, given problem text P and a quantity q in it, whether q is needed in the numeric expression generating the solution. We train on gold annotations and use the score of the classifier as the scoring function $\textsc{Irr}(\cdot)$.

4.3.1 Features

The features are extracted from the quantity schemas and can be broadly categorized into three groups:

1. **Unit features**: Most questions specifically mention the object whose amount needs to be computed, and hence questions provide valuable clue as to which quantities can be irrelevant. We add a feature for whether the unit of quantity q is present in the question tokens. Also, we add a feature based on whether the units of other quantities have better matches with question tokens (based on the number of tokens matched), and one based on the number of quantities which have the maximum number of matches with the question tokens.

2. **Related NP features**: Often units are not enough to differentiate between relevant and irrelevant quantities. Consider the following:

Example 3
Problem : *There are 8 apples in a pile on the desk. Each apple comes in a package of 11. 5 apples are added to the pile. How many apples are there in the pile?*
Solution : $(8 + 5) = 13$

   The relevance decision depends on the noun phrase "the pile", which is absent in the second sentence. We add a feature indicating whether a related noun phrase is present in the question. Also, we add a feature based on whether the related noun phrases of other quantities have better match with the question. Extraction of related noun phrases is described in Section 4.2.

3. **Miscellaneous Features**: When a problem mentions only two quantities, both of them are usually relevant. Hence, we also add a feature based on the number of quantities mentioned in text.

We include pairwise conjunction of the above features.

4.4 LCA Operation Classifier

In order to predict LCA operations, we train a multiclass SVM classifier. Given problem text P and a pair of quantities p_i and p_j, the classifier predicts one of the six labels described in Eq. 2. We consider the confidence scores for each label supplied by the classifier as the scoring function $\textsc{Pair}(\cdot)$.

4.4.1 Features

We use the following categories of features:

1. **Individual Quantity features**: Dependent verbs have been shown to play significant role in solving addition and subtraction problems (Hosseini et al., 2014). Hence, we add the dependent verb of the quantity as a feature. Multiplication and division problems are largely dependent on rates described in text. To capture that, we add a feature based on whether the quantity is a rate, and whether any component of rate unit is present in

the question. In addition to these quantity schema features, we add selected tokens from the neighborhood of the quantity mention. Neighborhood of quantities are often highly informative of LCA operations, for example, "He got 80 more marbles", the term "more" usually indicates addition. We add as features adverbs and comparative adjectives mentioned in a window of size 5 around the quantity mention.

2. **Quantity Pair features**: For a pair (q_i, q_j) we add features to indicate whether they have the same dependent verbs, to indicate whether both dependent verbs refer to the same verb mention, whether the units of q_i and q_j are the same and, if one of them is a rate, which component of the unit matches with the other quantity's unit. Finally, we add a feature indicating whether the value of q_i is greater than the value of q_j.

3. **Question Features**: Finally, we add a few features based on the question asked. In particular, for arithmetic problems where only one operation is needed, the question contains signals for the required operation. Specifically, we add indicator features based on whether the question mentions comparison-related tokens (e.g., "more", "less" or "than"), or whether the question asks for a rate (indicated by tokens such as "each" or "one").

We include pairwise conjunction of the above features. For both classifiers, we use the Illinois-SL package [1] under default settings.

5 Experimental Results

In this section, we evaluate the proposed method on publicly available datasets of arithmetic word problems. We evaluate separately the relevance and LCA operation classifiers, and show the contribution of various features. Lastly, we evaluate the performance of the full system, and quantify the gains achieved by the constraints.

5.1 Datasets

We evaluate our system on three datasets, each of which comprise a different category of arithmetic word problems.

1. **AI2 Dataset**: This is a collection of 395 addition and subtraction problems, released by (Hosseini et al., 2014). They performed a 3-fold cross validation, with every fold containing problems from different sources. This helped them evaluate robustness to domain diversity. We follow the same evaluation setting.

2. **IL Dataset**: This is a collection of arithmetic problems released by (Roy et al., 2015). Each of these problems can be solved by performing one operation. However, there are multiple problems having the same template. To counter this, we perform a few modifications to the dataset. First, for each problem, we replace the numbers and nouns with the part of speech tags, and then we cluster the problems based on unigrams and bigrams from this modified problem text. In particular, we cluster problems together whose unigram-bigram similarity is over 90%. We next prune each cluster to keep at most 5 problems in each cluster. Finally we create the folds ensuring all problems in a cluster are assigned to the same fold, and each fold has similar distribution of all operations. We have a final set of 562 problems, and we use a 5-fold cross validation to evaluate on this dataset.

3. **Commoncore Dataset**: In order to test our system's ability to handle multi-step problems, we create a new dataset of multi-step arithmetic problems. The problems were extracted from *www.commoncoresheets.com*. In total, there were 600 problems, 100 for each of the following types:

   (a) Addition followed by Subtraction
   (b) Subtraction followed by Addition
   (c) Addition and Multiplication
   (d) Addition and Division
   (e) Subtraction and Multiplication
   (f) Subtraction and Division

   This dataset had no irrelevant quantities. Therefore, we did not use the relevance classifier in our evaluations.

   In order to test our system's ability to generalize across problem types, we perform a 6-fold cross validation, with each fold containing all the problems from one of the aforementioned categories. This is a more challenging setting relative to the individual data sets mentioned above, since we are evaluating on multi-step problems, without ever looking at problems which require the same set of operations.

5.2 Relevance Classifier

Table 2 evaluates the performance of the relevance classifier on the AI2 and IL datasets. We report two accuracy values: Relax - fraction of quantities which the classifier got correct, and Strict - fraction of math problems, for which all quantities were correctly classified. We report accuracy using all features and then removing each feature group, one at a time.

[1] http://cogcomp.cs.illinois.edu/page/software_view/Illinois-SL

	AI2		IL		CC	
	Relax	Strict	Relax	Strict	Relax	Strict
All features	**88.7**	**85.1**	**75.7**	**75.7**	60.0	25.8
No Individual Quantity features	73.6	67.6	52.0	52.0	29.2	0.0
No Quantity Pair features	83.2	79.8	63.6	63.6	49.3	16.5
No Question features	86.8	83.9	73.3	73.3	**60.5**	**28.3**

Table 1: Performance of LCA Operation classifier on the datasets AI2, IL and CC.

	AI2		IL	
	Relax	Strict	Relax	Strict
All features	94.7	89.1	**95.4**	**93.2**
No Unit features	88.9	71.5	92.8	91.0
No NP features	**94.9**	**89.6**	95.0	91.2
No Misc. features	92.0	85.9	93.7	89.8

Table 2: Performance of Relevance classifier on the datasets AI2 and IL.

We see that features related to units of quantities play the most significant role in determining relevance of quantities. Also, the related NP features are not helpful for the AI2 dataset.

5.3 LCA Operation Classifier

Table 1 evaluates the performance of the LCA Operation classifier on the AI2, IL and CC datasets. As before, we report two accuracies - Relax - fraction of quantity pairs for which the classifier correctly predicted the LCA operation, and Strict - fraction of math problems, for which all quantity pairs were correctly classified. We report accuracy using all features and then removing each feature group, one at a time.

The strict and relaxed accuracies for IL dataset are identical, since each problem in IL dataset only requires one operation. The features related to individual quantities are most significant; in particular, the accuracy goes to 0.0 in the CC dataset, without using individual quantity features. The question features are not helpful for classification in the CC dataset. This can be attributed to the fact that all problems in CC dataset require multiple operations, and questions in multi-step problems usually do not contain information for each of the required operations.

5.4 Global Inference Module

Table 3 shows the performance of our system in correctly solving arithmetic word problems. We show the impact of various contraints, and also compare against previously best known results on the AI2 and IL datasets. We also show results using each of the two constraints separately, and using no constraints at all.

	AI2	IL	CC
All constraints	72.0	**73.9**	**45.2**
Positive constraint	**78.0**	72.5	36.5
Integral constraint	71.8	73.4	39.0
No constraint	77.7	71.9	29.6
(Hosseini et al., 2014)	77.7	-	-
(Roy et al., 2015)	-	52.7	-
(Kushman et al., 2014)	64.0	73.7	2.3

Table 3: Accuracy in correctly solving arithmetic problems. First four rows represent various configurations of our system. We achieve state of the art results in both AI2 and IL datasets.

The previously known best result in the AI2 dataset is reported in (Hosseini et al., 2014). Since we follow the exact same evaluation settings, our results are directly comparable. We achieve state of the art results, without having access to any additional annotated data, unlike (Hosseini et al., 2014), who use labeled data for verb categorization. For the IL dataset, we acquired the system of (Roy et al., 2015) from the authors, and ran it with the same fold information. We outperform their system by an absolute gain of over 20%. We believe that the improvement was mainly due to the dependence of the system of (Roy et al., 2015) on lexical and neighborhood of quantity features. In contrast, features from quantity schemas help us generalize across problem types. Finally, we also compare against the template based system of (Kushman et al., 2014). (Hosseini et al., 2014) mentions the result of running the system of (Kushman et al., 2014) on AI2 dataset, and we report their result here. For IL and CC datasets, we used the system released by (Kushman et al., 2014).

The integrality constraint is particularly helpful when division is involved, since it can lead to fractional answers. It does not help in case of the AI2 dataset, which involves only addition and subtraction problems. The role of the constraints becomes more significant in case of multi-step problems and, in particular, they contribute an absolute improvement of over 15% over the system without constraints on the CC dataset. The template based system of (Kushman et al., 2014) performs on par with our system on the IL dataset. We believe that it is due to the small number of equation templates in the IL dataset. It performs poorly on the CC dataset, since we evaluate on unseen problem types, which do not ensure that equation templates in the test data will be seen in the training data.

5.5 Discussion

The leading source of errors for the classifiers are erroneous quantity schema extraction and lack of understanding of unknown or rare verbs. For the relevance classifier on the AI2 dataset, 25% of the errors were due to mistakes in extracting the quantity schemas and 20% could be attributed to rare verbs. For the LCA operation classifier on the same dataset, 16% of the errors were due to unknown verbs and 15% were due to mistakes in extracting the schemas. The erroneous extraction of accurate quantity schemas is very significant for the IL dataset, contributing 57% of the errors for the relevance classifier and 39% of the errors for the LCA operation classifier. For the operation classifier on the CC dataset, 8% of the errors were due to verbs and 16% were due to faulty quantity schema extraction. Quantity Schema extraction is challenging due to parsing issues as well as some non-standard rate patterns, and it will be one of the future work targets. For example, in the sentence, "How many 4-dollar toys can he buy?", we fail to extract the rate component of the quantity 4.

6 Conclusion

This paper presents a novel method for understanding and solving a general class of arithmetic word problems. Our approach can solve all problems whose solution can be expressed by a read-once arithmetic expression, where each quantity from the problem text appears at most once in the expression. We develop a novel theoretical framework, centered around the notion of monotone expression trees, and showed how this representation can be used to get a unique decomposition of the problem. This theory naturally leads to a computational solution that we have shown to uniquely determine the solution - determine the arithmetic operation between any two quantities identified in the text. This theory underlies our algorithmic solution - we develop classifiers and a constrained inference approach that exploits redundancy in the information, and show that this yields strong performance on several benchmark collections. In particular, our approach achieves state of the art performance on two publicly available arithmetic problem datasets and can support natural generalizations. Specifically, our approach performs competitively on multistep problems, even when it has never observed the particular problem type before.

Although we develop and use the notion of expression trees in the context of numerical expressions, the concept is more general. In particular, if we allow leaves of expression trees to represent variables, we can express algebraic expressions and equations in this framework. Hence a similar approach can be targeted towards algebra word problems, a direction we wish to investigate in the future.

The datasets used in the paper are available for download at *http://cogcomp.cs.illinois.edu/page/resource_view/98*.

Acknowledgments

This research was sponsored by DARPA (under agreement number FA8750-13-2-0008), and a grant from AI2. Any opinions, findings, conclusions or recommendations are those of the authors and do not necessarily reflect the view of the agencies.

References

R. Barzilay and M. Lapata. 2006. Aggregation via Set Partitioning for Natural Language Generation. In *Human Language Technologies - North American Chapter of the Association for Computational Linguistics*, June.

J. Berant, V. Srikumar, P. Chen, A. V. Linden, B. Harding, B. Huang, P. Clark, and C. D. Manning. 2014. Modeling biological processes for reading comprehension. In *Proceedings of EMNLP*.

M. Chang, L. Ratinov, and D. Roth. 2012. Structured learning with constrained conditional models. *Machine Learning*, 88(3):399–431, 6.

P. Clark. 2015. Elementary School Science and Math Tests as a Driver for AI: Take the Aristo Challenge! In *Proceedings of IAAI*.

J. Clarke and M. Lapata. 2006. Constraint-based sentence compression: An integer programming approach. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 144–151, Sydney, Australia, July. ACL.

Y. Goldberg and M. Elhadad. 2010. An efficient algorithm for easy-first non-directional dependency parsing. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 742–750, Los Angeles, California, June.

M. J. Hosseini, H. Hajishirzi, O. Etzioni, and N. Kushman. 2014. Learning to solve arithmetic word problems with verb categorization. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar*, pages 523–533.

N. Kushman, L. Zettlemoyer, R. Barzilay, and Y. Artzi. 2014. Learning to automatically solve algebra word problems. In *ACL*, pages 271–281.

V. Punyakanok, D. Roth, and W. Yih. 2005. The necessity of syntactic parsing for semantic role labeling. In *Proc. of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1117–1123.

V. Punyakanok, D. Roth, and W. Yih. 2008. The importance of syntactic parsing and inference in semantic role labeling. *Computational Linguistics*, 34(2).

D. Roth and W. Yih. 2004. A linear programming formulation for global inference in natural language tasks. In Hwee Tou Ng and Ellen Riloff, editors,

Proc. of the Conference on Computational Natural Language Learning (CoNLL), pages 1–8. Association for Computational Linguistics.

D. Roth and W. Yih. 2005. Integer linear programming inference for conditional random fields. In *Proc. of the International Conference on Machine Learning (ICML)*, pages 737–744.

S. Roy, T. Vieira, and D. Roth. 2015. Reasoning about quantities in natural language. *Transactions of the Association for Computational Linguistics*, 3.

F. Sadeghi, S. K. Divvala, and A. Farhadi. 2015. Viske: Visual knowledge extraction and question answering by visual verification of relation phrases. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June.

M. J. Seo, H. Hajishirzi, A. Farhadi, and O. Etzioni. 2014. Diagram understanding in geometry questions. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada.*, pages 2831–2838.

Distant Supervision for Relation Extraction via Piecewise Convolutional Neural Networks

Daojian Zeng, Kang Liu, Yubo Chen and Jun Zhao
National Laboratory of Pattern Recognition
Institute of Automation, Chinese Academy of Sciences, Beijing, 100190, China
{djzeng,kliu,yubo.chen,jzhao}@nlpr.ia.ac.cn

Abstract

Two problems arise when using distant supervision for relation extraction. First, in this method, an already existing knowledge base is heuristically aligned to texts, and the alignment results are treated as labeled data. However, the heuristic alignment can fail, resulting in wrong label problem. In addition, in previous approaches, statistical models have typically been applied to ad hoc features. The noise that originates from the feature extraction process can cause poor performance.

In this paper, we propose a novel model dubbed the Piecewise Convolutional Neural Networks (PCNNs) with multi-instance learning to address these two problems. To solve the first problem, distant supervised relation extraction is treated as a multi-instance problem in which the uncertainty of instance labels is taken into account. To address the latter problem, we avoid feature engineering and instead adopt convolutional architecture with piecewise max pooling to automatically learn relevant features. Experiments show that our method is effective and outperforms several competitive baseline methods.

1 Introduction

In relation extraction, one challenge that is faced when building a machine learning system is the generation of training examples. One common technique for coping with this difficulty is distant supervision (Mintz et al., 2009) which assumes that if two entities have a relationship in a known knowledge base, then all sentences that mention these two entities will express that relationship in some way. Figure 1 shows an example of the automatic labeling of data through distant supervision. In this example, *Apple* and *Steve Jobs* are two related entities in Freebase[1]. All sentences that contain these two entities are selected as training instances. The distant supervision strategy is an effective method of automatically labeling training data. However, it has two major shortcomings when used for relation extraction.

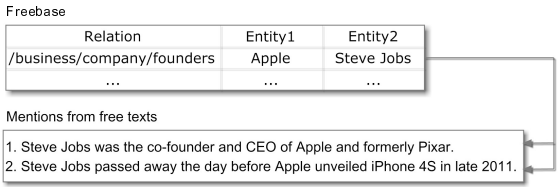


Figure 1: Training instances generated through distant supervision. Upper sentence: correct labeling; lower sentence: incorrect labeling.

First, the distant supervision assumption is too strong and causes the wrong label problem. A sentence that mentions two entities does not necessarily express their relation in a knowledge base. It is possible that these two entities may simply share the same topic. For instance, the upper sentence indeed expresses the "company/founders" relation in Figure 1. The lower sentence, however, does not express this relation but is still selected as a training instance. This will hinder the performance of a model trained on such noisy data.

Second, previous methods (Mintz et al., 2009; Riedel et al., 2010; Hoffmann et al., 2011) have typically applied supervised models to elaborately designed features when obtained the labeled data through distant supervision. These features are often derived from preexisting Natural Language Processing (NLP) tools. Since errors inevitably exist in NLP tools, the use of traditional features leads to error propagation or accumulation. Distant supervised relation extraction generally ad-

[1]http://www.freebase.com/

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1753–1762,
Lisbon, Portugal, 17-21 September 2015.

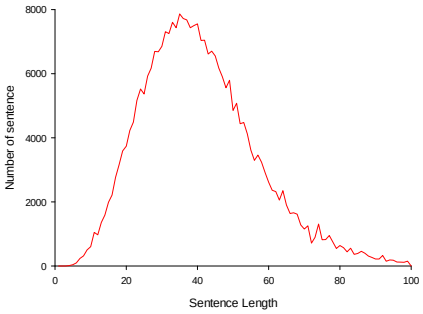


Figure 2: The sentence length distribution of Riedel's dataset.

dresses corpora from the Web, including many informal texts. Figure 2 shows the sentence length distribution of a benchmark distant supervision dataset that was developed by Riedel et al. (2010). Approximately half of the sentences are longer than 40 words. McDonald and Nivre (2007) showed that the accuracy of syntactic parsing decreases significantly with increasing sentence length. Therefore, when using traditional features, the problem of error propagation or accumulation will not only exist, it will grow more serious.

In this paper, we propose a novel model dubbed Piecewise Convolutional Neural Networks (PCNNs) with multi-instance learning to address the two problems described above. To address the first problem, distant supervised relation extraction is treated as a multi-instance problem similar to previous studies (Riedel et al., 2010; Hoffmann et al., 2011; Surdeanu et al., 2012). In multi-instance problem, the training set consists of many bags, and each contains many instances. The labels of the bags are known; however, the labels of the instances in the bags are unknown. We design an objective function at the bag level. In the learning process, the uncertainty of instance labels can be taken into account; this alleviates the wrong label problem.

To address the second problem, we adopt convolutional architecture to automatically learn relevant features without complicated NLP preprocessing inspired by Zeng et al. (2014). Our proposal is an extension of Zeng et al. (2014), in which a single max pooling operation is utilized to determine the most significant features. Although this operation has been shown to be effective for textual feature representation (Collobert et al., 2011; Kim, 2014), it reduces the size of the hidden layers too rapidly and cannot capture the structural information between two entities (Graham, 2014). For example, to identify the relation between *Steve Jobs* and *Apple* in Figure 1, we need to specify the entities and extract the structural features between them. Several approaches have employed manually crafted features that attempt to model such structural information. These approaches usually consider both internal and external contexts. A sentence is inherently divided into three segments according to the two given entities. The internal context includes the characters inside the two entities, and the external context involves the characters around the two entities (Zhang et al., 2006). Clearly, single max pooling is not sufficient to capture such structural information. To capture structural and other latent information, we divide the convolution results into three segments based on the positions of the two given entities and devise a piecewise max pooling layer instead of the single max pooling layer. The piecewise max pooling procedure returns the maximum value in each segment instead of a single maximum value over the entire sentence. Thus, it is expected to exhibit superior performance compared with traditional methods.

The contributions of this paper can be summarized as follows.

- We explore the feasibility of performing distant supervised relation extraction without hand-designed features. PCNNS are proposed to automatically learn features without complicated NLP preprocessing.
- To address the wrong label problem, we develop innovative solutions that incorporate multi-instance learning into the PCNNS for distant supervised relation extraction.
- In the proposed network, we devise a piecewise max pooling layer, which aims to capture structural information between two entities.

2 Related Work

Relation extraction is one of the most important topics in NLP. Many approaches to relation extraction have been developed, such as bootstrapping, unsupervised relation discovery and supervised classification. Supervised approaches are the most commonly used methods for relation

extraction and yield relatively high performance (Bunescu and Mooney, 2006; Zelenko et al., 2003; Zhou et al., 2005). In the supervised paradigm, relation extraction is considered to be a multi-class classification problem and may suffer from a lack of labeled data for training. To address this problem, Mintz et al. (2009) adopted Freebase to perform distant supervision. As described in Section 1, the algorithm for training data generation is sometimes faced with the wrong label problem. To address this shortcoming, (Riedel et al., 2010; Hoffmann et al., 2011; Surdeanu et al., 2012) developed the relaxed distant supervision assumption for multi-instance learning. The term 'multi-instance learning was coined by (Dietterich et al., 1997) while investigating the problem of predicting drug activity. In multi-instance learning, the uncertainty of instance labels can be taken into account. The focus of multi-instance learning is to discriminate among the bags.

These methods have been shown to be effective for relation extraction. However, their performance depends strongly on the quality of the designed features. Most existing studies have concentrated on extracting features to identify the relations between two entities. Previous methods can be generally categorized into two types: feature-based methods and kernel-based methods. In feature-based methods, a diverse set of strategies is exploited to convert classification clues (e.g., sequences, parse trees) into feature vectors (Kambhatla, 2004; Suchanek et al., 2006). Feature-based methods suffer from the necessity of selecting a suitable feature set when converting structured representations into feature vectors. Kernel-based methods provide a natural alternative to exploit rich representations of input classification clues, such as syntactic parse trees. Kernel-based methods enable the use of a large set of features without needing to extract them explicitly. Several kernels have been proposed, such as the convolution tree kernel (Qian et al., 2008), the subsequence kernel (Bunescu and Mooney, 2006) and the dependency tree kernel (Bunescu and Mooney, 2005).

Nevertheless, as mentioned in Section 1, it is difficult to design high-quality features using existing NLP tools. With the recent revival of interest in neural networks, many researchers have investigated the possibility of using neural networks to automatically learn features (Socher et al., 2012; Zeng et al., 2014). Inspired by Zeng et al. (2014), we propose the use of PCNNs with multi-instance learning to automatically learn features for distant supervised relation extraction. Dietterich et al. (1997) suggested that the design of multi-instance modifications for neural networks is a particularly interesting topic. Zhang and Zhou (2006) successfully incorporated multi-instance learning into traditional Backpropagation (BP) and Radial Basis Function (RBF) networks and optimized these networks by minimizing a sum-of-squares error function. In contrast to their method, we define the objective function based on the cross-entropy principle.

3 Methodology

Distant supervised relation extraction is formulated as multi-instance problem. In this section, we present innovative solutions that incorporate multi-instance learning into a convolutional neural network to fulfill this task. PCNNs are proposed for the automatic learning of features without complicated NLP preprocessing. Figure 3 shows our neural network architecture for distant supervised relation extraction. It illustrates the procedure that handles one instance of a bag. This procedure includes four main parts: *Vector Representation*, *Convolution*, *Piecewise Max Pooling* and *Softmax Output*. We describe these parts in detail below.

3.1 Vector Representation

The inputs of our network are raw word tokens. When using neural networks, we typically transform word tokens into low-dimensional vectors. In our method, each input word token is transformed into a vector by looking up pre-trained word embeddings. Moreover, we use position features (PFs) to specify entity pairs, which are also transformed into vectors by looking up position embeddings.

3.1.1 Word Embeddings

Word embeddings are distributed representations of words that map each word in a text to a 'k'-dimensional real-valued vector. They have recently been shown to capture both semantic and syntactic information about words very well, setting performance records in several word similarity tasks (Mikolov et al., 2013; Pennington et al., 2014). Using word embeddings that have been trained a priori has become common practice for

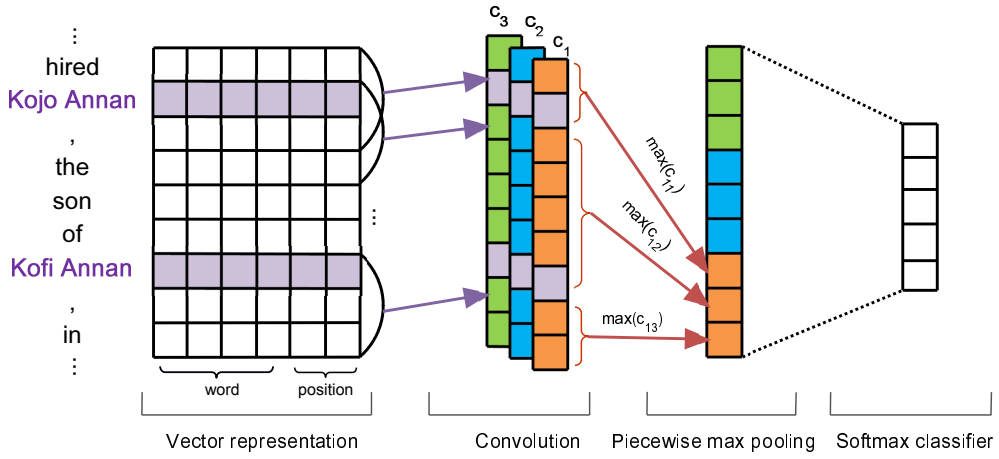


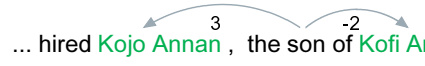
Figure 3: The architecture of PCNNs (better viewed in color) used for distant supervised relation extraction, illustrating the procedure for handling one instance of a bag and predicting the relation between *Kojo Annan* and *Kofi Annan*.

enhancing many other NLP tasks (Parikh et al., 2014; Huang et al., 2014).

A common method of training a neural network is to randomly initialize all parameters and then optimize them using an optimization algorithm. Recent research (Erhan et al., 2010) has shown that neural networks can converge to better local minima when they are initialized with word embeddings. Word embeddings are typically learned in an entirely unsupervised manner by exploiting the co-occurrence structure of words in unlabeled text. Researchers have proposed several methods of training word embeddings (Bengio et al., 2003; Collobert et al., 2011; Mikolov et al., 2013). In this paper, we use the Skip-gram model (Mikolov et al., 2013) to train word embeddings.

3.1.2 Position Embeddings

In relation extraction, we focus on assigning labels to entity pairs. Similar to Zeng et al. (2014), we use PFs to specify entity pairs. A PF is defined as the combination of the relative distances from the current word to e_1 and e_2. For instance, in the following example, the relative distances from *son* to e_1 (*Kojo Annan*) and e_2 (*Kofi Annan*) are 3 and -2, respectively.



Two position embedding matrixes ($\mathbf{PF}_1$ and $\mathbf{PF}_2$) are randomly initialized. We then transform the relative distances into real valued vectors by looking up the position embedding matrixes. In the example shown in Figure 3, it is assumed that the size of the word embedding is $d_w = 4$ and that the size of the position embedding is $d_p = 1$. In combined word embeddings and position embeddings, the vector representation part transforms an instance into a matrix $\mathbf{S} \in \mathbb{R}^{s \times d}$, where s is the sentence length and $d = d_w + d_p * 2$. The matrix $\mathbf{S}$ is subsequently fed into the convolution part.

3.2 Convolution

In relation extraction, an input sentence that is marked as containing the target entities corresponds only to a relation type; it does not predict labels for each word. Thus, it might be necessary to utilize all local features and perform this prediction globally. When using a neural network, the convolution approach is a natural means of merging all these features (Collobert et al., 2011).

Convolution is an operation between a vector of weights, $\mathbf{w}$, and a vector of inputs that is treated as a sequence $\mathbf{q}$. The weights matrix $\mathbf{w}$ is regarded as the filter for the convolution. In the example shown in Figure 3, we assume that the length of the filter is w ($w = 3$); thus, $\mathbf{w} \in \mathbb{R}^m$ ($m = w * d$). We consider $\mathbf{S}$ to be a sequence $\{\mathbf{q}_1, \mathbf{q}_2, \cdots, \mathbf{q}_s\}$, where $\mathbf{q}_i \in \mathbb{R}^d$. In general, let $\mathbf{q}_{i:j}$ refer to the concatenation of $\mathbf{q}_i$ to $\mathbf{q}_j$. The convolution operation involves taking the dot product of $\mathbf{w}$ with each w-gram in the sequence $\mathbf{q}$ to obtain another sequence $\mathbf{c} \in \mathbb{R}^{s+w-1}$:

$$c_j = \mathbf{w}\mathbf{q}_{j-w+1:j} \quad (1)$$

where the index j ranges from 1 to $s+w-1$. Out-of-range input values $\mathbf{q}_i$, where $i < 1$ or $i > s$, are

taken to be zero.

The ability to capture different features typically requires the use of multiple filters (or feature maps) in the convolution. Under the assumption that we use n filters ($\mathbf{W} = \{\mathbf{w}_1, \mathbf{w}_2, \cdots, \mathbf{w}_n\}$), the convolution operation can be expressed as follows:

$$c_{ij} = \mathbf{w}_i \mathbf{q}_{j-w+1:j} \quad 1 \leq i \leq n \tag{2}$$

The convolution result is a matrix $\mathbf{C} = \{\mathbf{c}_1, \mathbf{c}_2, \cdots, \mathbf{c}_n\} \in \mathbb{R}^{n \times (s+w-1)}$. Figure 3 shows an example in which we use 3 different filters in the convolution procedure.

3.3 Piecewise Max Pooling

The size of the convolution output matrix $\mathbf{C} \in \mathbb{R}^{n \times (s+w-1)}$ depends on the number of tokens s in the sentence that is fed into the network. To apply subsequent layers, the features that are extracted by the convolution layer must be combined such that they are independent of the sentence length. In traditional Convolution Neural Networks (CNNs), max pooling operations are often applied for this purpose (Collobert et al., 2011; Zeng et al., 2014). This type of pooling scheme naturally addresses variable sentence lengths. The idea is to capture the most significant features (with the highest values) in each feature map.

However, despite the widespread use of single max pooling, this approach is insufficient for relation extraction. As described in the first section, single max pooling reduces the size of the hidden layers too rapidly and is too coarse to capture fine-grained features for relation extraction. In addition, single max pooling is not sufficient to capture the structural information between two entities. In relation extraction, an input sentence can be divided into three segments based on the two selected entities. Therefore, we propose a piecewise max pooling procedure that returns the maximum value in each segment instead of a single maximum value. As shown in Figure 3, the output of each convolutional filter $\mathbf{c}_i$ is divided into three segments $\{\mathbf{c}_{i1}, \mathbf{c}_{i2}, \mathbf{c}_{i3}\}$ by *Kojo Annan* and *Kofi Annan*. The piecewise max pooling procedure can be expressed as follows:

$$p_{ij} = \max(\mathbf{c}_{ij}) \quad 1 \leq i \leq n, \ 1 \leq j \leq 3 \tag{3}$$

For the output of each convolutional filter, we can obtain a 3-dimensional vector $\mathbf{p}_i = \{p_{i1}, p_{i2}, p_{i3}\}$. We then concatenate all vectors $\mathbf{p}_{1:n}$ and apply a non-linear function, such as the hyperbolic tangent. Finally, the piecewise max pooling procedure outputs a vector:

$$\mathbf{g} = \tanh(\mathbf{p}_{1:n}) \tag{4}$$

where $\mathbf{g} \in \mathbb{R}^{3n}$. The size of $\mathbf{g}$ is fixed and is no longer related to the sentence length.

3.4 Softmax Output

To compute the confidence of each relation, the feature vector $\mathbf{g}$ is fed into a softmax classifier.

$$\mathbf{o} = \mathbf{W}_1 \mathbf{g} + b \tag{5}$$

$\mathbf{W_1} \in \mathbb{R}^{n_1 \times 3n}$ is the transformation matrix, and $\mathbf{o} \in \mathbb{R}^{n_1}$ is the final output of the network, where n_1 is equal to the number of possible relation types for the relation extraction system.

We employ dropout (Hinton et al., 2012) on the penultimate layer for regularization. Dropout prevents the co-adaptation of hidden units by randomly dropping out a proportion p of the hidden units during forward computing. We first apply a "masking" operation $(\mathbf{g} \circ \mathbf{r})$ on $\mathbf{g}$, where $\mathbf{r}$ is a vector of Bernoulli random variables with probability p of being 1. Eq.(5) becomes:

$$\mathbf{o} = \mathbf{W}_1(\mathbf{g} \circ \mathbf{r}) + b \tag{6}$$

Each output can then be interpreted as the confidence score of the corresponding relation. This score can be interpreted as a conditional probability by applying a softmax operation (see Section 3.5). In the test procedure, the learned weight vectors are scaled by p such that $\hat{\mathbf{W}}_1 = p\mathbf{W}_1$ and are used (without dropout) to score unseen instances.

3.5 Multi-instance Learning

In order to alleviate the wrong label problem, we use multi-instance learning for PCNNs. The PCNNs-based relation extraction can be stated as a quintuple $\theta = (\mathbf{E}, \mathbf{PF}_1, \mathbf{PF}_2, \mathbf{W}, \mathbf{W}_1)$[2]. The input to the network is a bag. Suppose that there are T bags $\{M_1, M_2, \cdots, M_T\}$ and that the i-th bag contains q_i instances $M_i = \{m_i^1, m_i^2, \cdots, m_i^{q_i}\}$. The objective of multi-instance learning is to predict the labels of the unseen bags. In this paper, all instances in a bag are considered independently. Given an input instance m_i^j, the network with the parameter θ outputs a vector $\mathbf{o}$, where the r-th component o_r corresponds to the score associated

[2] $\mathbf{E}$ represents the word embeddings.

Algorithm 1 Multi-instance learning

1: Initialize θ. Partition the bags into mini-batches of size b_s.
2: Randomly choose a mini-batch, and feed the bags into the network one by one.
3: Find the j-th instance m_i^j $(1 \leq i \leq b_s)$ in each bag according to Eq. (9).
4: Update θ based on the gradients of m_i^j $(1 \leq i \leq b_s)$ via Adadelta.
5: Repeat steps 2-4 until either convergence or the maximum number of epochs is reached.

with relation r. To obtain the conditional probability $p(r|m, \theta)$, we apply a softmax operation over all relation types:

$$p(r|m_i^j;\theta) = \frac{e^{o_r}}{\sum_{k=1}^{n_1} e^{o_k}} \quad (7)$$

The objective of multi-instance learning is to discriminate bags rather than instances. To do so, we must define the objective function on the bags. Given all (T) training bags (M_i, y_i), we can define the objective function using cross-entropy at the bag level as follows:

$$J(\theta) = \sum_{i=1}^{T} \log p(y_i|m_i^j;\theta) \quad (8)$$

where j is constrained as follows:

$$j^* = \arg\max_j p(y_i|m_i^j;\theta) \;\; 1 \leq j \leq q_i \quad (9)$$

Using this defined objective function, we maximize $J(\theta)$ through stochastic gradient descent over shuffled mini-batches with the Adadelta (Zeiler, 2012) update rule. The entire training procedure is described in Algorithm 1.

From the introduction presented above, we know that the traditional backpropagation algorithm modifies a network in accordance with all training instances, whereas backpropagation with multi-instance learning modifies a network based on bags. Thus, our method captures the nature of distant supervised relation extraction, in which some training instances will inevitably be incorrectly labeled. When a trained PCNN is used for prediction, a bag is positively labeled if and only if the output of the network on at least one of its instances is assigned a positive label.

4 Experiments

Our experiments are intended to provide evidence that supports the following hypothesis: automatically learning features using PCNNs with multi-instance learning can lead to an increase in performance. To this end, we first introduce the dataset and evaluation metrics used. Next, we test several variants via cross-validation to determine the parameters to be used in our experiments. We then compare the performance of our method to those of several traditional methods. Finally, we evaluate the effects of piecewise max pooling and multi-instance learning[3].

4.1 Dataset and Evaluation Metrics

We evaluate our method on a widely used dataset[4] that was developed by (Riedel et al., 2010) and has also been used by (Hoffmann et al., 2011; Surdeanu et al., 2012). This dataset was generated by aligning Freebase relations with the NYT corpus, with sentences from the years 2005-2006 used as the training corpus and sentences from 2007 used as the testing corpus.

Following previous work (Mintz et al., 2009), we evaluate our method in two ways: the held-out evaluation and the manual evaluation. The held-out evaluation only compares the extracted relation instances against Freebase relation data and reports the precision/recall curves of the experiments. In the manual evaluation, we manually check the newly discovered relation instances that are not in Freebase.

4.2 Experimental Settings

4.2.1 Pre-trained Word Embeddings

In this paper, we use the Skip-gram model (word2vec)[5] to train the word embeddings on the NYT corpus. Word2vec first constructs a vocabulary from the training text data and then learns vector representations of the words. To obtain the embeddings of the entities, we concatenate the tokens of a entity using the ## operator when the entity has multiple word tokens. Since a comparison of the word embeddings is beyond the scope

[3] With regard to the position feature, our experiments yield the same positive results described in Zeng et al. (2014). Because the position feature is not the main contribution of this paper, we do not present the results without the position feature.

[4] http://iesl.cs.umass.edu/riedel/ecml/

[5] https://code.google.com/p/word2vec/

Window size	*Feature maps*	*Word dimension*	*Position dimension*	*Batch size*	*Adadelta parameter*	*Dropout probability*
$w = 3$	$n = 230$	$d_w = 50$	$d_p = 5$	$b_s = 50$	$\rho = 0.95, \varepsilon = 1e^{-6}$	$p = 0.5$

Table 1: Parameters used in our experiments.

of this paper, our experiments directly utilize 50-dimensional vectors.

4.2.2 Parameter Settings

In this section, we experimentally study the effects of two parameters on our models: the window size, w, and the number of feature maps, n. Following (Surdeanu et al., 2012), we tune all of the models using three-fold validation on the training set. We use a grid search to determine the optimal parameters and manually specify subsets of the parameter spaces: $w \in \{1, 2, 3, \cdots, 7\}$ and $n \in \{50, 60, \cdots, 300\}$. Table 1 shows all parameters used in the experiments. Because the position dimension has little effect on the result, we heuristically choose $d_p = 5$. The batch size is fixed to 50. We use Adadelta (Zeiler, 2012) in the update procedure; it relies on two main parameters, ρ and ε, which do not significantly affect the performance (Zeiler, 2012). Following (Zeiler, 2012), we choose 0.95 and $1e^{-6}$, respectively, as the values of these parameters. In the dropout operation, we randomly set the hidden unit activities to zero with a probability of 0.5 during training.

4.3 Comparison with Traditional Approaches

4.3.1 Held-out Evaluation

The held-out evaluation provides an approximate measure of precision without requiring costly human evaluation. Half of the Freebase relations are used for testing. The relation instances discovered from the test articles are automatically compared with those in Freebase.

To evaluate the proposed method, we select the following three traditional methods for comparison. *Mintz* represents a traditional distant-supervision-based model that was proposed by (Mintz et al., 2009). *MultiR* is a multi-instance learning method that was proposed by (Hoffmann et al., 2011). *MIML* is a multi-instance multi-label model that was proposed by (Surdeanu et al., 2012). Figure 4 shows the precision-recall curves for each method, where *PCNNs+MIL* denotes our method, and demonstrates that *PCNNs+MIL* achieves higher precision over the entire range of recall. *PCNNs+MIL* enhances the recall to approximately 34% without any loss of precision. In terms of both precision and recall, *PCNNs+MIL* outperforms all other evaluated approaches. Notably, the results of the methods evaluated for comparison were obtained using manually crafted features. By contrast, our result is obtained by automatically learning features from original words. The results demonstrate that the proposed method is an effective technique for distant supervised relation extraction. Automatically learning features via PCNNs can alleviate the error propagation that occurs in traditional feature extraction. Incorporating multi-instance learning into a convolutional neural network is an effective means of addressing the wrong label problem.

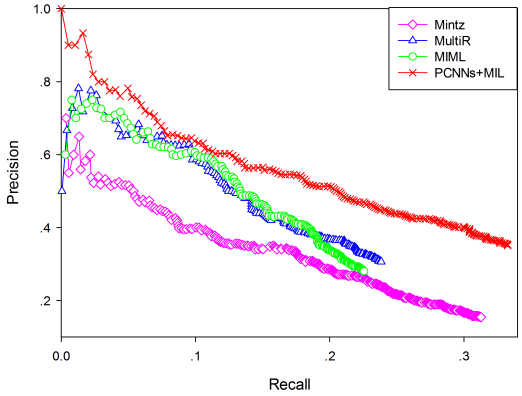


Figure 4: Performance comparison of the proposed method with traditional approaches.

Top N	Mintz	MultiR	MIML	PCNNs+MIL
Top 100	0.77	0.83	0.85	**0.86**
Top 200	0.71	0.74	0.75	**0.80**
Top 500	0.55	0.59	0.61	**0.69**
Average	0.676	0.720	0.737	**0.783**

Table 2: Precision values for the top 100, top 200, and top 500 extracted relation instances upon manual evaluation.

4.3.2 Manual Evaluation

It is worth emphasizing that there is a sharp decline in the held-out precision-recall curves of *PCNNs+MIL* at very low recall (Figure 4). A manual check of the misclassified examples that were produced with high confidence reveals that the ma-

jorities of these examples are false negatives and are actually true relation instances that were misclassified due to the incomplete nature of Freebase.

Thus, the held-out evaluation suffers from false negatives in Freebase. We perform a manual evaluation to eliminate these problems. For the manual evaluation, we choose the entity pairs for which at least one participating entity is not present in Freebase as a candidate. This means that there is no overlap between the held-out and manual candidates. Because the number of relation instances that are expressed in the test data is unknown, we cannot calculate the recall in this case. Instead, we calculate the precision of the top N extracted relation instances. Table 2 presents the manually evaluated precisions for the top 100, top 200, and top 500 extracted instances. The results show that *PCNNs+MIL* achieves the best performance; moreover, the precision is higher than in the held-out evaluation. This finding indicates that many of the false negatives that we predict are, in fact, true relational facts. The sharp decline observed in the held-out precision-recall curves is therefore reasonable.

4.4 Effect of Piecewise Max Pooling and Multi-instance Learning

In this paper, we develop a method of piecewise max pooling and incorporate multi-instance learning into convolutional neural networks for distant supervised relation extraction. To demonstrate the effects of these two techniques, we empirically study the performance of systems in which these techniques are not implemented through held-out evaluations (Figure 5). *CNNs* represents convolutional neural networks to which single max pooling is applied. Figure 5 shows that when piecewise max pooling is used (*PCNNs*), better results are produced than those achieved using *CNNs*. Moreover, compared with *CNNs+MIL*, *PCNNs* achieve slightly higher precision when the recall is greater than 0.08. Since the parameters for all the model are determined by grid search, we can observe that *CNNs* cannot achieve competitive results compared to *PCNNs* when increasing the size of the hidden layer of convolutional neural networks. It means that we cannot capture more useful information by simply increasing the network parameter. These results demonstrate that the proposed piecewise max pooling technique is beneficial and

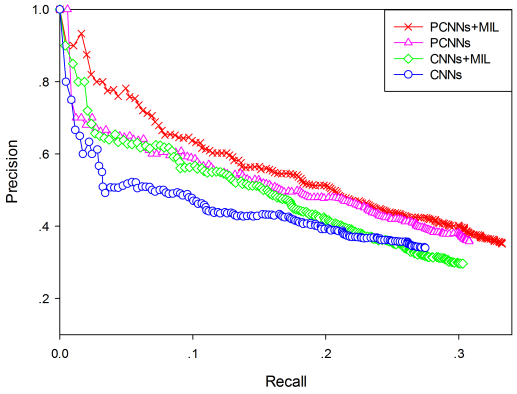


Figure 5: Effect of piecewise max pooling and multi-instance learning.

can effectively capture structural information for relation extraction. A similar phenomenon is also observed when multi-instance learning is added to the network. Both *CNNs+MIL* and *PCNNs+MIL* outperform their counterparts *CNNs* and *PCNNs*, respectively, thereby demonstrating that incorporation of multi-instance learning into our neural network was successful in solving the wrong label problem. As expected, *PCNNs+MIL* obtains the best results because the advantages of both techniques are achieved simultaneously.

5 Conclusion

In this paper, we exploit Piecewise Convolutional Neural Networks (PCNNs) with multi-instance learning for distant supervised relation extraction. In our method, features are automatically learned without complicated NLP preprocessing. We also successfully devise a piecewise max pooling layer in the proposed network to capture structural information and incorporate multi-instance learning to address the wrong label problem. Experimental results show that the proposed approach offers significant improvements over comparable methods.

Acknowledgments

This work was sponsored by the National Basic Research Program of China (no. 2014CB340503) and the National Natural Science Foundation of China (no. 61272332 and no. 61202329). We thank the anonymous reviewers for their insightful comments.

References

Yoshua Bengio, Rjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155.

Razvan C. Bunescu and Raymond J. Mooney. 2005. A shortest path dependency kernel for relation extraction. In *Proceedings of HLT/EMNLP*, pages 724–731.

Razvan Bunescu and Raymond Mooney. 2006. Subsequence kernels for relation extraction. *Proceedings of NIPS*, 18:171–178.

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

Thomas G Dietterich, Richard H Lathrop, and Tomás Lozano-Pérez. 1997. Solving the multiple instance problem with axis-parallel rectangles. *Journal of Artificial intelligence*, 89(1):31–71.

Dumitru Erhan, Aaron Courville, Yoshua Bengio, and Pascal Vincent. 2010. Why does unsupervised pretraining help deep learning? *Journal of Machine Learning Research*, 11:625–660.

Benjamin Graham. 2014. Fractional max-pooling. *arXiv preprint arXiv:1412.6071*.

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

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S. Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of ACL*, pages 541–550.

Fei Huang, Arun Ahuja, Doug Downey, Yi Yang, Yuhong Guo, and Alexander Yates. 2014. Learning representations for weakly supervised natural language processing tasks. *Journal of Computational Linguistics*, 40(1):85–120, March.

Nanda Kambhatla. 2004. Combining lexical, syntactic, and semantic features with maximum entropy models for extracting relations. In *Proceedings of ACLdemo*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of EMNLP*, pages 1746–1751.

Ryan T McDonald and Joakim Nivre. 2007. Characterizing the errors of data-driven dependency parsing models. In *Proceedings of EMNLP-CoNLL*, pages 122–131.

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

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

Ankur P Parikh, Shay B Cohen, and Eric P Xing. 2014. Spectral unsupervised parsing with additive tree metrics. In *Proceedings of ACL*, pages 1062–1072.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of EMNLP 2014*, pages 1746–1751.

Longhua Qian, Guodong Zhou, Fang Kong, Qiaoming Zhu, and Peide Qian. 2008. Exploiting constituent dependencies for tree kernel-based semantic relation extraction. In *Proceedings of COLING*, pages 697–704.

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

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

Fabian M. Suchanek, Georgiana Ifrim, and Gerhard Weikum. 2006. Combining linguistic and statistical analysis to extract relations from web documents. In *Proceedings of KDD*, pages 712–717.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D. Manning. 2012. Multi-instance multi-label learning for relation extraction. In *Proceedings of EMNLP-CoNLL*, pages 455–465.

Matthew D. Zeiler. 2012. Adadelta: An adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, abs/1212.5701.

Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. 2003. Kernel methods for relation extraction. *Journal of Machine Learning Research*, 3:1083–1106.

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

Minling Zhang and Zhihua Zhou. 2006. Adapting rbf neural networks to multi-instance learning. *Neural Processing Letters*, 23(1):1–26.

Min Zhang, Jie Zhang, Jian Su, and Guodong Zhou. 2006. A composite kernel to extract relations between entities with both flat and structured features. In *Proceedings of ACL*, pages 825–832.

Guodong Zhou, Jian Su, Jie Zhang, and Min Zhang. 2005. Exploring various knowledge in relation extraction. In *Proceedings of ACL*, pages 427–434.

CORE: Context-Aware Open Relation Extraction with Factorization Machines

Fabio Petroni
Sapienza University of Rome
Rome, Italy
petroni@dis.uniroma1.it

Luciano Del Corro
Max Planck Institute for Informatics
Saarbrücken, Germany
delcorro@mpi-inf.mpg.de

Rainer Gemulla
University of Mannheim
Mannheim, Germany
rgemulla@uni-mannheim.de

Abstract

We propose CORE, a novel matrix factorization model that leverages contextual information for open relation extraction. Our model is based on factorization machines and integrates facts from various sources, such as knowledge bases or open information extractors, as well as the context in which these facts have been observed. We argue that integrating contextual information—such as metadata about extraction sources, lexical context, or type information—significantly improves prediction performance. Open information extractors, for example, may produce extractions that are unspecific or ambiguous when taken out of context. Our experimental study on a large real-world dataset indicates that CORE has significantly better prediction performance than state-of-the-art approaches when contextual information is available.

1 Introduction

Open relation extraction (open RE) is the task of extracting new facts for a potentially unbounded set of relations from various sources such as knowledge bases or natural language text. The task is closely related to *targeted information extraction* (IE), which aims to populate a knowledge base (KB) with new facts for the KB's relations, such as wasBornIn(Sepp Herberger, Mannheim). Existing methods either reason within the KB itself (Franz et al., 2009; Nickel et al., 2011; Drumond et al., 2012) or leverage large text corpora to learn patterns that are indicative of KB relations (Mintz et al., 2009; Surdeanu et al., 2012; Min et al., 2013). In both cases, targeted IE methods are inherently limited to an (often small) set of predefined relations, i.e., they are not "open". The open RE task is also related to *open information extraction* (open IE) (Banko et al., 2007; Del Corro and Gemulla, 2013), which extracts large amounts of surface relations and their arguments from natural language text; e.g., "critizices"("Dante", "Catholic Church").[1] Although open IE is a domain-independent approach, the extracted surface relations are purely syntactic and often ambiguous or noisy. Moreover, open IE methods usually do not "predict" facts that have not been explicitly observed in the input data. Open RE combines the above tasks by predicting new facts for an open set of relations. The key challenge in open RE is to reason jointly over the *universal schema* consisting of KB relations and surface relations (Riedel et al., 2013).

A number of matrix or tensor factorization models have recently been proposed in the context of relation extraction (Nickel et al., 2012; Riedel et al., 2013; Huang et al., 2014; Chang et al., 2014). These models use the available data to learn latent semantic representations of entities (or entity pairs) and relations in a domain-independent way; the latent representations are subsequently used to predict new facts. Existing models often focus on either targeted IE or open RE. Targeted models are used for within-KB reasoning; they rely on the closed-world assumption and often do not scale with the number of relations. Open RE models use the open-world assumption, which is more suitable for the open RE task because the available data is often highly incomplete. In this paper, we propose CORE, a novel open RE factorization model that incorporates and exploits contextual information to improve prediction performance.

Consider for example the sentence "Tom Peloso joined Modest Mouse to record their fifth studio album". Open IE systems may extract the *surface fact* "join"(TP, MM) from this sentence. Note

[1] We mark (non-disambiguated) mentions of entities and relations in quotation marks throughout this paper.

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1763–1773,
Lisbon, Portugal, 17-21 September 2015.

that *surface relation* "join" is unspecific; in this case, it refers to becoming a member of a music band (as opposed to, say, an employee of a company). Most existing open RE systems use the extracted surface fact for further reasoning, but they ignore the context from which the fact was extracted. We argue in this paper that exploiting contextual information is beneficial for open RE. For our example, we may use standard NLP tools like a named entity recognizer to detect that TP is a person and MM an organization. These coarse-grained types give us hints about the domain and range of the "join" relation for the surface fact, although the actual meaning of "join" still remains opaque. Now imagine that the above sentence was extracted from a newspaper article published in the music section. This information can help to infer that "join" indeed refers to joining a band. Other contextual information, such as the words "record" and "album" that occur in the sentence, further strengthen this interpretation. A context-aware open RE system should leverage such information to accurately predict facts like "is band member of"(TP, MM) and "plays with"(TP, MM).

Note that the prediction of the fact "is band member of"(TP, MM) is facilitated if we make use of a KB that knows that TP is a musician and MM is a music band. If TP and/or MM are not present in the knowledge base, however, such a reasoning does not apply. In our work, we consider both linked *entities* (in-KB) and non-linked *entity mentions* (out of-KB). Since KB are often incomplete, this open approach to handle named entities allows us to extract facts for all entities, even if they do not appear in the KB.

In this paper, we propose CORE, a flexible open RE model that leverages contextual information. CORE is inspired by the combined factorization and entity model (FE) of Riedel et al. (2013). As FE, CORE associates latent semantic representations with entities, relations, and arguments. In contrast to FE, CORE uses factorization machines (Rendle, 2012) as its underlying framework, which allows us to incorporate context in a flexible way. CORE is able to leverage and integrate arbitrary contextual information associated with the input facts into its open RE factorization model. To support reasoning under the open-world assumption, we propose an efficient method for parameter estimation in factorization machines based on Bayesian personalized ranking (Rendle et al., 2009).

We conducted an experimental study on a real-world dataset using contextual information along the lines mentioned above. Our model is extensible, i.e., additional contextual information can be integrated when available. Even with limited amount of contextual information used in our experiments, our CORE model provided higher prediction performance than previous models. Our findings validate the usefulness of contextual information for the open RE task.

2 Related Work

There is a large body of related work on relation extraction; we restrict attention to methods that are most similar to our work.

Targeted IE. Targeted IE methods aim to extract from natural-language text new instances of a set of predefined relations, usually taken from a KB. Most existing methods make use of distant supervision, i.e., they start with a set of seed instances (pairs of entities) for the relations of interest, search for these seed instances in text, learn a relation extractor from the so-obtained training data, and optionally iterate (Mintz et al., 2009; Surdeanu et al., 2012; Min et al., 2013). Open RE models are more general then targeted IE methods in that they additionally reason about surface relations that do not correspond to KB relations. For this reason, Riedel et al. (2013) argued and experimentally validated that open RE models can outperform targeted IE methods.

Open IE. In contrast to targeted IE, the goal of open IE is to extract all (or most) relations expressed in natural-language text, whether or not these relations are defined in a KB (Banko et al., 2007; Fader et al., 2011; Del Corro and Gemulla, 2013). The facts obtained by open IE methods are often not disambiguated, i.e., the entities and/or the relation are not linked to a knowledge base; e.g., "criticizes"("Dante", "Catholic Church"). The goal of our work is to reason about extracted open-IE facts and their contextual information. Our method is oblivious to the actual open IE method being used.

Relation clustering. One way to reason about KB and surface relations is to cluster the relations: whenever two relations appear in the same cluster, they are treated as synonymous (Hasegawa et al., 2004; Shinyama and Sekine, 2006; Yao

et al., 2011; Takamatsu et al., 2011; Min et al., 2012; Akbik et al., 2012; de Lacalle and Lapata, 2013). For example, if "criticizes" and "hates" are clustered together, then we may predict "hates"("Dante", "Catholic Church") from the above fact (which is actually not true). The general problem with relation clustering is its "black and white" approach to relations: either two relations are the same or they are different. This assumption generally does not hold for the surface relations extracted by open IE systems (Riedel et al., 2013); examples of other types of relationships between relations include implication or mutual exclusion.

Tensor factorization. Matrix or tensor factorization approaches try to address the above problem: instead of clustering relations, they directly predict facts. Both matrix and tensor models learn and make use of semantic representations of relations and their arguments. The semantic representations ideally captures all the information present in the data; it does not, however, establish a direct relationship (such as synonymy) between different KB or surface relations.

Tensor factorization models conceptually model the input data as a subject×relation×object tensor, in which non-zero values correspond to input facts. The tensor is factored to construct a new tensor in which predicted facts take large non-zero values. Examples of such tensor factorization models are TripleRank (Franz et al., 2009), RESCAL (Nickel et al., 2011; Nickel et al., 2012), or PITF (Drumond et al., 2012). Tensor factorization models are generally well-suited to reason within a KB because they are able to predict relations between arbitrary pairs of subjects and objects. In the context of open RE, however, these methods suffer from limited scalability with the number of relations as well as from their large prediction space (Chang et al., 2014).

Matrix factorization. The key difference between matrix and tensor factorization models is that the former restrict the prediction space, i.e., these models generally cannot predict arbitrary facts. Similar to distant supervision approaches, matrix factorization models focus on predicting facts for which some direct evidence exists. In more detail, most methods restrict the prediction space to the set of facts for which the subject and the object share at least some relation in the input data. For this reason, matrix factorization models are not suited for in-KB reasoning; an individual pair of entities usually does not occur in more than one KB relation. In the open RE context, however, input relations are semantically related so that many subject-object pairs belong to multiple relations. The key advantage of matrix methods is (1) that this restriction allows them to use additional features—such as features for each subject-object pair—and (2) that they scale much better with the number of relations. Examples of such matrix factorization models include (Tresp et al., 2009; Jiang et al., 2012; Fan et al., 2014; Huang et al., 2014). Chang et al. (2014) have also shown that a combination of matrix and tensor factorization models can be fruitful. Closest to our work is the "universal schema" matrix factorization approach of Riedel et al. (2013), which combines a latent features model, a neighborhood model and an entity model but does not incorporate context. Our CORE model follows the universal schema idea, but uses a more general factorization model, which includes the information captured by the latent features and entity model (but not the neighborhood model), and incorporates contextual information.

Using contextual information. It is well known that contextual information can improve IE methods. Information such as bag-of-words, part-of-speech tags, entity types, or parse trees have been integrated into many existing systems (Mintz et al., 2009; Zhang et al., 2012; Takamatsu et al., 2011; Zhou et al., 2007; de Lacalle and Lapata, 2013; Akbik et al., 2012). Our work differs in that we integrate contextual information into an open RE system. To do so, we leverage factorization machines (Rendle et al., 2011; Rendle, 2012), which have been successfully applied to exploit contextual information in the context of recommender systems. We show how to model open RE data and context with factorization machines and provide a method for parameter estimation under the open-world assumption.

3 The CORE Model

Input data. We model the input data as a set of *observations* of the form (r, t, c), where r refer to a KB or surface relation, t refer to a subject-object pair of entities (or entity mentions) and c to contextual information. An observation obtained from the example of the introduction may be ("join", (TP, MM), { types:(person,org),

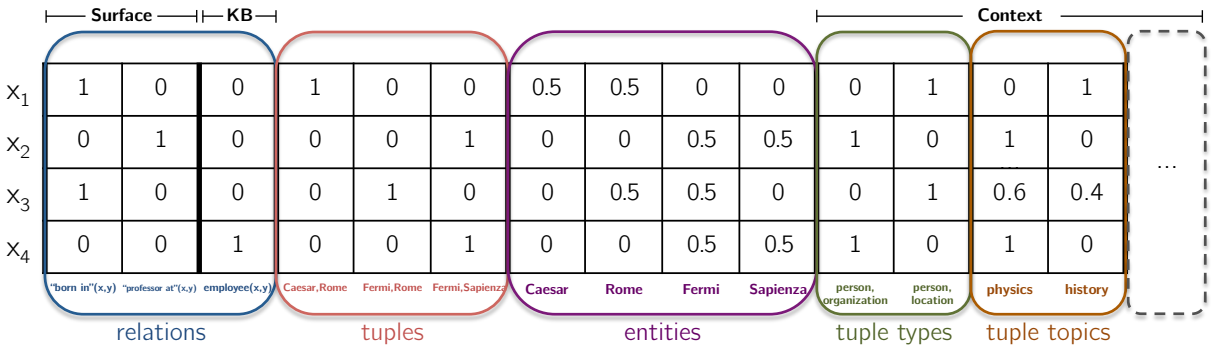


Figure 1: Example for representing a context-aware open RE problem with CORE

topic:music, word:record, word:album, ...}). Denote by R the set of all observed *relations*, by E the set of all observed *entities*, and by $T \subseteq E \times E$ the set of all observed entity pairs, which we refer to as *tuples*. A *fact* takes form $r(t)$ and is composed of a relation $r \in R$ and a tuple $t \in T$; e.g., "join"(TP, MM). Note that there may be multiple observations for a fact. Finally, denote by C the set of all *contextual variables*; each observation is associated with a set $c \subseteq C$ of context variables. In this paper we restrict attention to categorical context variables; our model can potentially handle continuous context as in (Rendle, 2012).

Problem definition. The open RE task is to produce a ranked list of tuples $T_r \subseteq T$ for each relation $r \in R$; the list is restricted to new tuples, i.e., tuples $t \in T$ for which $r(t)$ has not been observed in the input. The rank of each tuple reflects the model's prediction of the likelihood that the corresponding fact is indeed true. A good model thus ranks correct facts higher than incorrect ones.

Modeling facts. Denote by $V = R \cup T \cup E \cup C$ the set of all observed relations, tuples, entities, and contextual variables. For ease of exposition, we refer to the elements of V as *variables*. We model the input data in terms of a matrix in which each row corresponds to a fact (i.e., not an observation) and each column to a variable. We group columns according to the type of the variables; e.g, there are relation columns, tuple columns, entity columns, and a group of columns for each type of contextual information. The matrix is populated such that in each row the values of each column group sum up to unity, i.e., we normalize values within column groups. In particular, we set to 1 the values of the variable of the relation and the tuple of the corresponding fact. We set to 0.5 the variables corresponding to the two entities referred to by the fact. An example is shown in Fig. 1. Here the first row, for instance, corresponds to the fact "born in"(Caesar, Rome). Note that we model tuples and entities separately: the entity variables expose which arguments belong to the fact, the tuple variables expose their order.

Modeling context. As described above, we model the data in terms of a matrix in which rows corresponds to facts (instead of observations). The reasoning behind this approach is as follows. First, we may see a fact in multiple observations; our goal is to leverage all the available context. Second, facts but not observations are the target of our predictions. Finally, we are interested in predicting new facts, i.e., facts that we have not seen in the input data. For these facts, there is no corresponding observation so that we cannot directly obtain contextual information. To address these points, our model aggregates the context of relevant observations for each fact; this approach allows us to provide comprehensive contextual information for both observed and unobserved facts.

We group contextual information by the type of information: examples include metadata about the extraction sources (e.g., from an article on music), types of the entities of a tuple (e.g., (person, location)), or the bag-of-words in the sentence from which an extraction has been obtained. We aggregate the contextual information for each tuple $t \in T$; this tuple-level approach allows us to provide contextual information for unobserved facts. In more detail, we count in how many observations each contextual variable has been associated with the tuple, and then normalize the count values to 1 within each group of columns. The so-obtained values can be interpreted as the relative frequencies with which each contextual variable is associated with the tuple. The contextual information as-

sociated with each fact is given by the aggregated, normalized context of its tuple.

Fig. 1 shows context information arranged in two groups: tuple types and tuple topics. We capture information such as that the tuple (Caesar, Rome) has only been seen in articles on history or that tuple (Fermi, Rome) is mentioned in both physics and history articles (slightly more often in the former). Since context is associated with tuples, facts 2 and 4 on (Fermi, Sapienza) share contextual information. This form of context sharing (as well as entity sharing) allows us to propagate information about tuples across various relations.

Factorization model. CORE employs a matrix factorization model based on factorization machines and the open-world assumption to capture latent semantic information about the individual variables. In particular, we associate with each variable $v \in V$ a *bias term* $b_v \in \mathbb{R}$ and a *latent feature vector* $\boldsymbol{f}_v \in \mathbb{R}^d$, where the *dimensionality* d of the latent feature space is a hyperparameter of our model. Denote by X the set of rows in the input matrix, which we henceforth refer to as *training points*. For each training point $x \in X$, denote by x_v the value of variable $v \in V$ in the corresponding row of the matrix. Our model associates with training point $x \in X$ a *score* $s(x)$ computed as follows:

$$s(x) = \sum_{v \in V} x_v b_v + \sum_{v_1 \in V} \sum_{v_2 \in V \setminus \{v_1\}} x_{v_1} x_{v_2} \boldsymbol{f}_{v_1}^T \boldsymbol{f}_{v_2} \quad (1)$$

Here the bias terms models the contribution of each individual variable to the final score, whereas the latent feature vectors model the contribution of all pairwise interactions between variables. Note that only bias terms and feature vectors corresponding to non-zero entries in x affect the score and that x is often sparse. Since we can compute $s(x)$ in time linear to both the number of nonzero entries in x and the dimensionality d (Rendle, 2012), score computation is fast. As discussed below, we (roughly) estimate bias terms and feature vectors such that observed facts achieve high scores. We may thus think of each feature vector as a low-dimensional representation of the global information contained in the corresponding variable.

Prediction. Given estimates for bias terms and latent feature vectors, we rank unobserved facts as follows. Fix a relation $r \in R$ and a tuple $t \in T$ such that $r(t)$ has not been observed. As indicated above, our model overcomes the key problem that there is no observation, and thus no context, for $r(t)$ by context aggregation and sharing. In particular, we create an *test point* $\hat{x}$ for tuple $r(t)$ in a way similar to creating data points, i.e., we set the relation, tuple, and entity variables accordingly and add the aggregated, normalized context of t. Once test point $\hat{x}$ has been created, we can predict its score $s(\hat{x})$ using Eq. (1). We then rank each unobserved tuple by its so-obtained score, i.e., tuples with higher scores are ranked higher. The resulting ranking constitutes the list T_r of predicted facts for relation r.

Bayesian personalized ranking. The parameters of our model are given by $\Theta = \{\, b_v, \boldsymbol{f}_v \mid v \in V \,\}$. In approaches based on the closed-world assumption, Θ is estimated by minimizing the error between model predictions and target values (e.g., 1 for true facts, 0 for false facts). In our setting of open RE, all our observations are positive, i.e., we do not have negative training data. One way to handle the absence of negative training data is to associate a target value of 0 to all unobserved facts. This closed-world approach essentially assumes that all unobserved facts are false, which may not be a suitable assumption for the sparsely observed relations of open RE. Following Riedel et al. (2013), we adopt the open-world assumption instead, i.e., we treat each unobserved facts as unknown. Since factorization machines originally require explicit target values (e.g., feedback in recommender systems), we need to adapt parameter estimation to the open-world setting.

In more detail, we employ a variant of the Bayesian personalized ranking (BPR) optimization criterion (Rendle et al., 2009). We associate with each training point x a set of *negative samples* X_x^-. Each negative sample $x^- \in X_x^-$ is an unobserved fact with its associated context (constructed as described in the prediction section above). Generally, the negative samples x^- should be chosen such that they are "less likely" to be true than fact x. We maximize the following optimization criterion:

$$\frac{1}{|X|} \sum_{x \in X} \left(\sum_{x^- \in X_x^-} \frac{\ln \sigma(\delta(x, x^-))}{|X_x^-|} - \lambda \|\Theta_x\|^2 \right) \quad (2)$$

where $\sigma(x) = \frac{1}{1+e^{-x}}$ denotes the logistic function, $\delta(x, x^-) = s(x) - s(x^-)$ denotes the differ-

ence of scores, and $\Theta_x = \{ b_v, \boldsymbol{f}_v \mid x_v \neq 0 \}$ the subset of the model parameters relevant for training point x. Here we use L2 regularization controlled by a single hyperparameter λ. In essence, the BPR criterion aims to maximize the average "difference" $\ln \sigma(\delta(x, x^-))$ between the score of fact x and each of its negative samples x^-, averaged over all facts. In other words, we aim to score x higher than each x^-. (Note that under the closed-world assumption, we would instead consider x^- as being false.) For a more in-depth discussion of BPR, see (Rendle et al., 2009).

Sampling negative evidence. To make BPR effective, the set of negative samples needs to be chosen carefully. A naive approach is to take the set of all unobserved facts between each relation $r \in R$ and each tuple $t \in T$ (or $E \times E$) as the set X_x^-. The reasoning is that, after all, we expect "random" unobserved facts to be less likely to be true than observed facts. This naive approach is problematic, however, because the set of negative samples is independent of x and thus not sufficiently informative (i.e., it contains many irrelevant samples).

To overcome this problem, the negative sample set needs to be related to x in some way. Since we ultimately use our model to rank tuples for each relation individually, we consider as negative evidence for x only unobserved facts from the same relation (Riedel et al., 2013). In more detail, we (conceptually) build a negative sample set X_r^- for each relation $r \in R$. We include into X_r^- all facts $r(t)$—again, along with their context—such that $t \in T$ is an observed tuple but $r(t)$ is an unobserved fact. Thus the subject-object pair t of entities is not observed with relation r in the input data (but with some other relation). The set of negative samples associated with each training point x is defined by the relation r of the fact contained in x, that is $X_x^- = X_r^-$. Note that we do not actually construct the negative sample sets; see below.

Parameter estimation. We maximize Eq. (2) using stochastic gradient ascent. This allows us to avoid constructing the sets X_x^-, which are often infeasibly large, and worked well in our experiments. In particular, in each stochastic gradient step, we randomly sample a training point $x \in X$, and subsequently randomly sample a negative sample $x^- \in X_x^-$. This sampling procedure can be implemented very efficiently. We then perform the following ascent step with learning rate η:

$$\Theta \leftarrow \Theta + \eta \nabla_\Theta \left(\ln \sigma(d(x, x^-)) - \lambda \|\Theta_x\|^2 \right)$$

One can show that the stochastic gradient used in the formula above is an unbiased estimate of the gradient of Eq. (2). To speed up parameter estimation, we use a parallel lock-free version of stochastic gradient ascent as in Recht et al. (2011). This allows our model to handle (reasonably) large datasets.

	size	info
facts	453.9k	14.7k Freebase, 174.1k surface linked, 184.5k surface partially-linked, 80.6k surface non-linked.
relations	4.7k	94 Freebase, 4.6k surface.
tuples	178.5k	69.5k linked, 71.5k partially-linked, 37.5k non-linked.
entities	114.2k	36.8k linked, 77.4k non-linked.

Table 1: Dataset statistics.

4 Experiments

We conducted an experimental study on real-world data to compare our CORE model with other state-of-the-art approaches.[2] Our experimental study closely follows the one of Riedel et al. (2013).

4.1 Experimental Setup

Dataset. We made use of the dataset of Riedel et al. (2013), but extended it with contextual information. The dataset consisted of 2.5M surface facts extracted from the New York Times corpus (Sandhaus, 2008), as well as 16k facts from Freebase. Surface facts have been obtained by using a named-entity recognizer, which additionally labeled each named entity mention with its coarse-grained type (i.e., person, organization, location, miscellaneous). For each pair of entities found within a sentence, the shortest dependency path between these pairs was taken as surface relation. The entity mentions in each surface fact were linked to Freebase using a simple string matching

[2] Source code, datasets, and supporting material are available at `http://dws.informatik.uni-mannheim.de/en/resources/software/core/`

Relation	#	PITF	NFE	CORE	CORE+m	CORE+t	CORE+w	CORE+mt	CORE+mtw
person/company	208	70 (0.47)	92 (0.81)	91 (0.83)	90 (0.84)	91 (0.87)	92 (0.87)	95 (0.93)	**96** (**0.94**)
person/place_of_birth	117	1 (0.0)	92 (*0.9*)	90 (0.88)	92 (*0.9*)	92 (*0.9*)	89 (0.87)	**93** (*0.9*)	92 (*0.9*)
location/containedby	102	7 (0.0)	63 (0.47)	62 (0.47)	63 (0.46)	61 (0.47)	61 (0.44)	62 (0.49)	**68** (**0.55**)
parent/child	88	9 (0.01)	64 (0.6)	64 (0.56)	64 (0.59)	64 (0.62)	64 (0.57)	67 (**0.67**)	**68** (0.63)
person/place_of_death	71	1 (0.0)	67 (0.93)	67 (0.92)	*69* (*0.94*)	67 (0.93)	67 (0.92)	*69* (*0.94*)	67 (0.92)
person/parents	67	20 (0.1)	51 (0.64)	52 (0.62)	51 (0.61)	49 (0.64)	47 (0.6)	*53* (**0.67**)	*53* (0.65)
author/works_written	65	24 (0.08)	45 (0.59)	49 (0.62)	51 (0.69)	50 (0.68)	50 (0.68)	51 (**0.7**)	**52** (0.67)
person/nationality	61	21 (0.08)	25 (0.19)	27 (0.17)	28 (0.2)	26 (0.2)	**29** (0.19)	27 (0.18)	27 (**0.21**)
neighbor./neighborhood_of	39	3 (0.0)	24 (0.44)	23 (0.45)	26 (0.5)	27 (0.47)	27 (0.49)	*30* (0.51)	*30* (**0.52**)
film/directed_by	15	7 (0.06)	7 (0.15)	11 (0.22)	9 (0.25)	10 (0.27)	**15** (**0.52**)	11 (0.28)	12 (0.31)
company/founders	11	0 (0.0)	*10* (*0.34*)	*10* (*0.34*)	*10* (0.26)	*10* (0.21)	*10* (0.22)	*10* (0.22)	*10* (0.24)
sports_team/league	11	1 (0.0)	7 (0.24)	*10* (0.23)	*10* (*0.3*)	7 (0.22)	*10* (0.27)	8 (0.29)	9 (*0.3*)
structure/architect	11	7 (0.63)	7 (0.63)	9 (0.7)	*11* (0.84)	*11* (0.73)	*11* (**0.9**)	*11* (0.8)	10 (0.77)
team/arena_stadium	9	2 (0.01)	6 (0.14)	6 (0.19)	6 (0.18)	6 (0.15)	6 (0.18)	*7* (**0.29**)	*7* (0.2)
team_owner/teams_owned	9	4 (0.05)	6 (0.17)	7 (0.18)	7 (0.33)	6 (0.27)	7 (0.19)	6 (0.22)	**8** (**0.34**)
film/produced_by	8	1 (0.03)	4 (0.06)	3 (0.13)	2 (0.12)	3 (0.03)	*6* (0.09)	3 (0.13)	*6* (**0.15**)
roadcast/area_served	5	0 (0.0)	4 (0.71)	4 (**0.73**)	4 (0.65)	4 (0.66)	4 (0.66)	*5* (0.64)	*5* (0.72)
person/religion	5	2 (0.0)	*3* (0.21)	2 (0.22)	1 (0.2)	*3* (0.22)	*3* (**0.25**)	2 (0.21)	*3* (0.21)
composer/compositions	3	*2* (0.1)	*2* (0.34)	*2* (0.35)	*2* (0.34)	*2* (0.35)	1 (0.33)	*2* (0.22)	*2* (**0.36**)
Average $MAP^{100}_{\#}$		0.09	0.46	0.47	0.49	0.47	0.49	0.49	**0.51**
Weighted Average $MAP^{100}_{\#}$		0.14	0.64	0.64	0.66	0.67	0.66	*0.70*	*0.70*

Table 2: True facts and $MAP^{100}_{\#}$ (in parentheses) in the top-100 evaluation-set tuples for Freebase relations. We consider as context the article metadata (m), the tuple types (t) and the bag-of-words (w). Best value per relation in bold (unique winner) or italic (multiple winners). Average weighs are # column values.

method. If no match was found, the entity mention was kept as is. There were around 2.2M tuples (distinct entity pairs) in this dataset, out of which 580k were fully linked to Freebase. For each of these tuples, the dataset additionally included all of the corresponding facts from Freebase. Using the metadata[3] of each New York Times article, we enriched each surface fact by the following contextual information: news desk (e.g., sports desk, foreign desk), descriptors (e.g., finances, elections), online section (e.g., sports, business), section (e.g., a, d), publication year, and bag-of-words of the sentence from which the surface fact has been extracted.

Training data. From the raw dataset described above, we filtered out all surface relations with less than 10 instances, and all tuples with less than two instances, as in Riedel et al. (2013). Tab. 1 summarizes statistics of the resulting dataset. Here we considered a fact or tuple as *linked* if both of its entities were linked to Freebase, as *partially-linked* if only one of its entities was linked, and as *non-linked* otherwise. In contrast to previous work (Riedel et al., 2013; Chang et al., 2014), we retain partially-linked and non-linked facts in our dataset.

[3]Further information can be found at `https://catalog.ldc.upenn.edu/LDC2008T19`.

Evaluation set. Open RE models produce predictions for all relations and all tuples. To keep the experimental study feasible and comparable to previous studies, we use the full training data but evaluate each model's predictions on only the subsample of 10k tuples ($\approx 6\%$ of all tuples) of Riedel et al. (2013). The subsample consisted of 20% linked, 40% partially-linked and 40% non-linked tuples. For each (surface) relation and method, we predicted the top-100 new facts (not in training) for the tuples in the subsample.

Considered methods. We compared various forms of our CORE model with PITF and the matrix factorization model NFE. Our study focused on these two factorization models because they outperformed other models (including non-factorization models) in previous studies (Riedel et al., 2013; Chang et al., 2014). All models were trained with the full training data described above.

PITF (Drumond et al., 2012). PITF is a recent tensor factorization method designed for within-KB reasoning. PITF is based on factorization machines so that we used our scalable CORE implementation for training the model.

NFE (Riedel et al., 2013). NFE is the full model proposed in the "universal schema" work of Riedel et al. (2013). It uses a linear combination of three component models: a neighborhood

Relation	#	PITF	NFE	CORE	CORE+m	CORE+t	CORE+w	CORE+mt	CORE+mtw
head	162	34 (0.18)	80 (0.66)	83 (0.66)	82 (0.63)	76 (0.57)	77 (0.57)	83 (0.69)	**88** (**0.73**)
scientist	144	44 (0.17)	76 (0.6)	74 (0.55)	73 (0.56)	74 (0.6)	73 (0.59)	*78* (0.66)	*78* (**0.69**)
base	133	10 (0.01)	85 (0.71)	86 (0.71)	86 (0.78)	88 (0.79)	85 (0.75)	83 (0.76)	**89** (**0.8**)
visit	118	4 (0.0)	73 (0.6)	75 (0.61)	76 (0.64)	80 (0.68)	74 (0.64)	75 (0.66)	**82** (**0.74**)
attend	92	11 (0.02)	65 (0.58)	64 (0.59)	65 (0.63)	62 (0.6)	66 (0.63)	62 (0.58)	**69** (**0.64**)
adviser	56	2 (0.0)	42 (0.56)	**47** (0.58)	44 (0.58)	43 (0.59)	45 (*0.63*)	43 (0.53)	44 (*0.63*)
criticize	40	5 (0.0)	31 (0.66)	33 (0.62)	33 (**0.7**)	33 (0.67)	33 (0.61)	35 (0.69)	**37** (0.69)
support	33	3 (0.0)	19 (0.27)	22 (*0.28*)	18 (0.21)	19 (*0.28*)	22 (0.27)	**23** (0.27)	21 (0.27)
praise	5	0 (0.0)	2 (0.0)	2 (0.01)	4 (*0.03*)	3 (0.01)	3 (0.02)	**5** (*0.03*)	2 (0.01)
vote	3	2 (0.01)	*3* (0.63)	*3* (0.63)	*3* (0.32)	*3* (0.49)	*3* (0.51)	*3* (0.59)	*3* (**0.64**)
Average $\text{MAP}^{100}_{\#}$		0.04	0.53	0.53	0.51	0.53	0.53	0.55	**0.59**
Weighted Average $\text{MAP}^{100}_{\#}$		0.08	0.62	0.61	0.63	0.63	0.61	0.65	**0.70**

Table 3: True facts and $\text{MAP}^{100}_{\#}$ (in parentheses) in the top-100 evaluation-set tuples for surface relations. We consider as context the article metadata (m), the tuple types (t) and the bag-of-words (w). Best value per relation in bold (unique winner) or italic (multiple winners). Average weighs are # column values.

model (N), a matrix factorization model (F), and an entity model (E). The F and E models together are similar (but not equal) to our CORE model without context. The NFE model outperformed tensor models (Chang et al., 2014) as well as clustering methods and distantly supervised methods in the experimental study of Riedel et al. (2013) for open RE tasks. We use the original source code of Riedel et al. (2013) for training.

CORE. We include multiple variants of our model in the experimental study, each differing by the amount of context being used. We consider as context the article metadata (m), the tuple types (t) and the bag-of-words (w). Each tuple type is a pair of subject-object types of (e.g. (person, location)). The basic CORE model uses relations, tuples and entities as variables. We additionally consider the CORE+t, CORE+w, CORE+mt, and CORE+mtw models, where the suffix indicates which contextual information has been included. The total number of variables in the resulting models varied between 300k (CORE) to 350k (CORE+mtw). We used a modified version of *libfm* for training.[4] Our version adds support for BPR and parallelizes the training algorithm.

Methodology. To evaluate the prediction performance of each method, we followed Riedel et al. (2013). We considered a collection of 19 Freebase relations (Tab. 2) and 10 surface relations (Tab. 3) and restrict predictions to tuples in the evaluation set.

Evaluation metrics. For each relation and method, we computed the top-100 evaluation set predictions and labeled them manually. We used as evaluation metrics the mean average precision defined as:

$$\text{MAP}^{100}_{\#} = \frac{\sum_{k=1}^{100} I_k \cdot P@k}{\min\{100, \#\}} \tag{3}$$

where indicator I_k takes value 1 if the k-th prediction is true and 0 otherwise, and $\#$ denotes the number of true tuples for the relation in the top-100 predictions of all models. The denominator is included to account for the fact that the evaluation set may include less than 100 true facts. $\text{MAP}^{100}_{\#}$ reflects how many true facts are found by each method as well as their ranking. If all $\#$ facts are found and ranked top, then $\text{MAP}^{100}_{\#} = 1$. Note that our definition of $\text{MAP}^{100}_{\#}$ differs slightly from Riedel et al. (2013); our metric is more robust because it is based on completely labeled evaluation data. To compare the prediction performance of each system across multiple relations, we averaged $\text{MAP}^{100}_{\#}$ values, in both an unweighted and a weighted (by $\#$) fashion.

Parameters. For all systems, we used $d = 100$ latent factors, $\lambda = 0.01$ for all variables, a constant learning rate of $\eta = 0.05$, and ran 1000 epochs of stochastic gradient ascent. These choices correspond to the ones of Riedel et al. (2013); no further tuning was performed.

4.2 Results.

Prediction performance. The results of our experimental study are summarized in Tab. 2 (Freebase relations) and Tab. 3 (surface relations). As mentioned before, all reported numbers are with respect to our evaluation set. Each entry shows the number of true facts in the top-100 predictions and, in parentheses, the $\text{MAP}^{100}_{\#}$ value. The # col-

[4] http://www.libfm.org

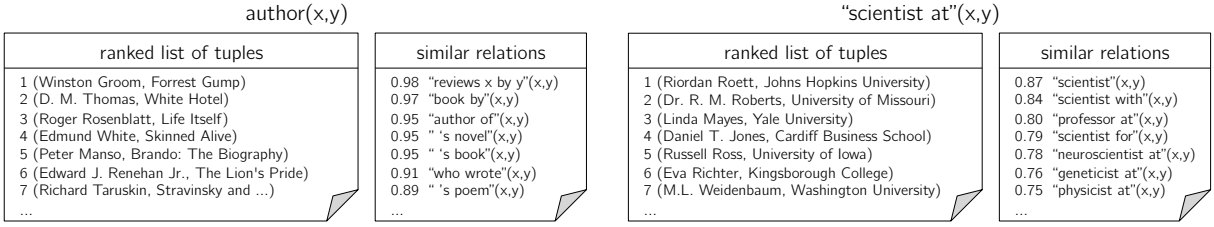


Figure 2: Some facts predicted by our model for the Freebase relation author(x,y) and the surface relation "scientist at"(x,y). Most similar relations also reported, using cosine similarity between the corresponding latent feature vectors as distance.

umn list the total number of true facts found by at least one method. The last two lines show the aggregated $\text{MAP}^{100}_{\#}$ scores.

We start our discussion with the results for Freebase relations (Tab. 2). First note that the PITF model generally did not perform well; as discussed before, tensor factorization models such as PITF suffer from a large prediction space and cannot incorporate tuple-level information. NFE and CORE, both matrix factorization models, performed better and were on par with each other. This indicates that our use of factorization machines does not affect performance in the absence of context; after all, both methods essentially make use of the same amount of information. The key advantage of our model over NFE is that we can incorporate contextual information. Our results indicate that using such information indeed improves prediction performance. The CORE+mtw model performed best overall; it increased the average $\text{MAP}^{100}_{\#}$ by four points (six points weighted) compared to the best context-unware model. Note that for some relations, including only subsets of the contextual information produced better results than using all contextual information (e.g., film/directed_by). We thus conjecture that extending our model by variable-specific regularization terms may be beneficial.

Tab. 3 summarizes our results for surface relations. In general, the relative performance of the models agreed with the one on Freebase relations. One difference is that using bag-of-word context significantly boosted prediction performance. One reason for this boost is that related surface relations often share semantically related words (e.g., "professor at" and "scientist at") and may occur in similar sentences (e.g., mentioning "university", "research", ...).

Anecdotal results. Fig. 2 shows the top test-set predictions of CORE+mtw for the author and "scientist at" relations. In both cases, we also list relations that have a similar semantic representation in our model (highest cosine similarity). Note that semantic similarity of relations is one aspect of our model; predictions incorporate other aspects such as context (i.e., two "similar" relations in different contexts are treated differently).

Training time. We used a machine with 16-cores Intel Xeon processor and 128GB of memory. Training CORE took roughly one hour, NFE roughly six hours (single core only), and training CORE+mtw took roughly 20 hours. Our implementation can handle reasonably large data, but an investigation of faster, more scalable training methods appears worthwhile.

5 Conclusion

We proposed CORE, a matrix factorization model for open RE that incorporates contextual information. Our model is based on factorization machines and the open-world assumption, integrates various forms of contextual information, and is extensible. Our experimental study suggests that exploiting context can significantly improve prediction performance.

References

Alan Akbik, Larysa Visengeriyeva, Priska Herger, Holmer Hemsen, and Alexander Löser. 2012. Unsupervised discovery of relations and discriminative extraction patterns. In *Proceedings of the 24th International Conference on Computational Linguistics (COLING 2012)*.

Michele Banko, Michael J Cafarella, Stephen Soderl, Matt Broadhead, and Oren Etzioni. 2007. Open information extraction from the web. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI)*.

Kai-Wei Chang, Wen-tau Yih, Bishan Yang, and Christopher Meek. 2014. Typed tensor decomposition of knowledge bases for relation extraction. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Oier Lopez de Lacalle and Mirella Lapata. 2013. Unsupervised relation extraction with general domain knowledge. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Luciano Del Corro and Rainer Gemulla. 2013. Clausie: clause-based open information extraction. In *Proceedings of the 22nd International Conference on World Wide Web (WWW)*.

Lucas Drumond, Steffen Rendle, and Lars Schmidt-Thieme. 2012. Predicting rdf triples in incomplete knowledge bases with tensor factorization. In *Proceedings of the 27th Annual ACM Symposium on Applied Computing (SAC)*.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Miao Fan, Deli Zhao, Qiang Zhou, Zhiyuan Liu, Thomas Fang Zheng, and Edward Y Chang. 2014. Distant supervision for relation extraction with matrix completion. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*.

Thomas Franz, Antje Schultz, Sergej Sizov, and Steffen Staab. 2009. Triplerank: Ranking semantic web data by tensor decomposition. In *Proceedings of the 8th International Semantic Web Conference (ISWC)*.

Takaaki Hasegawa, Satoshi Sekine, and Ralph Grishman. 2004. Discovering relations among named entities from large corpora. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics (ACL)*.

Yi Huang, Volker Tresp, Maximilian Nickel, Achim Rettinger, and Hans-Peter Kriegel. 2014. A scalable approach for statistical learning in semantic graphs. *Semantic Web*, 5(1):5–22.

Xueyan Jiang, Volker Tresp, Yi Huang, and Maximilian Nickel. 2012. Link prediction in multi-relational graphs using additive models. In *Proceedings of the 2012 International Workshop on Semantic Technologies meet Recommender Systems & Big Data (SeRSy)*.

Bonan Min, Shuming Shi, Ralph Grishman, and Chin-Yew Lin. 2012. Ensemble semantics for large-scale unsupervised relation extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*.

Bonan Min, Ralph Grishman, Li Wan, Chang Wang, and David Gondek. 2013. Distant supervision for relation extraction with an incomplete knowledge base. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (HLT-NAACL)*.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint conference of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing (ACL-IJCNLP)*.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data. In *Proceedings of the 28th international conference on machine learning (ICML)*.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2012. Factorizing yago: scalable machine learning for linked data. In *Proceedings of the 21st International Conference on World Wide Web (WWW)*.

Benjamin Recht, Christopher Re, Stephen Wright, and Feng Niu. 2011. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *Proceedings of the 25th Annual Conference on Neural Information Processing Systems (NIPS)*.

Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. Bpr: Bayesian personalized ranking from implicit feedback. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence (UAI)*.

Steffen Rendle, Zeno Gantner, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2011. Fast context-aware recommendations with factorization machines. In *Proceedings of the 34th international ACM conference on Research and development in Information Retrieval (SIGIR)*.

Steffen Rendle. 2012. Factorization machines with libfm. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 3(3):57.

Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (HLT-NAACL)*.

E. Sandhaus. 2008. The New York Times Annotated Corpus. *Linguistic Data Consortium, Philadelphia*, 6(12).

Yusuke Shinyama and Satoshi Sekine. 2006. Preemptive information extraction using unrestricted relation discovery. In *Proceedings of the 2006 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (HLT-NAACL)*.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D Manning. 2012. Multi-instance multi-label learning for relation extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*.

Shingo Takamatsu, Issei Sato, and Hiroshi Nakagawa. 2011. Probabilistic matrix factorization leveraging contexts for unsupervised relation extraction. In *Advances in Knowledge Discovery and Data Mining*, pages 87–99. Springer.

Volker Tresp, Yi Huang, Markus Bundschus, and Achim Rettinger. 2009. Materializing and querying learned knowledge. In *Proceedings of the 2009 International Workshop on Inductive Reasoning and Machine Learning for the Semantic Web (IRMLeS)*.

Limin Yao, Aria Haghighi, Sebastian Riedel, and Andrew McCallum. 2011. Structured relation discovery using generative models. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Congle Zhang, Raphael Hoffmann, and Daniel S. Weld. 2012. Ontological smoothing for relation extraction with minimal supervision. In *Proceedings of the 26th Conference on Artificial Intelligence (AAAI)*.

GuoDong Zhou, Min Zhang, DongHong Ji, and QiaoMing Zhu. 2007. Tree kernel-based relation extraction with context-sensitive structured parse tree information. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*.

Improved Relation Extraction with Feature-Rich Compositional Embedding Models

Matthew R. Gormley[1*] and **Mo Yu**[2*] and **Mark Dredze**[1]
[1]Human Language Technology Center of Excellence
Center for Language and Speech Processing
Johns Hopkins University, Baltimore, MD, 21218
[2]Machine Intelligence and Translation Lab
Harbin Institute of Technology, Harbin, China
gflfof@gmail.com, {mrg, mdredze}@cs.jhu.edu

Abstract

Compositional embedding models build a representation (or embedding) for a linguistic structure based on its component word embeddings. We propose a Feature-rich Compositional Embedding Model (FCM) for relation extraction that is expressive, generalizes to new domains, and is easy-to-implement. The key idea is to combine both (unlexicalized) hand-crafted features with learned word embeddings. The model is able to directly tackle the difficulties met by traditional compositional embeddings models, such as handling arbitrary types of sentence annotations and utilizing global information for composition. We test the proposed model on two relation extraction tasks, and demonstrate that our model outperforms both previous compositional models and traditional feature rich models on the ACE 2005 relation extraction task, and the SemEval 2010 relation classification task. The combination of our model and a log-linear classifier with hand-crafted features gives state-of-the-art results. We made our implementation available for general use[1].

1 Introduction

Two common NLP feature types are lexical properties of words and unlexicalized linguistic/structural interactions between words. Prior work on relation extraction has extensively studied how to design such features by combining *discrete* lexical properties (e.g. the identity of a word, its lemma, its morphological features) with aspects of a word's linguistic context (e.g. whether it lies between two entities or on a dependency path between them). While these help learning, they make generalization to unseen words difficult. An alternative approach to capturing lexical information relies on *continuous* word embeddings[2] as representative of words but generalizable to new words. Embedding features have improved many tasks, including NER, chunking, dependency parsing, semantic role labeling, and relation extraction (Miller et al., 2004; Turian et al., 2010; Koo et al., 2008; Roth and Woodsend, 2014; Sun et al., 2011; Plank and Moschitti, 2013; Nguyen and Grishman, 2014). Embeddings can capture lexical information, but alone they are insufficient: in state-of-the-art systems, they are used alongside features of the broader linguistic context.

In this paper, we introduce a compositional model that combines unlexicalized linguistic context and word embeddings for relation extraction, a task in which contextual feature construction plays a major role in generalizing to unseen data. Our model allows for the composition of embeddings with arbitrary linguistic structure, as expressed by hand crafted features. In the following sections, we begin with a precise construction of compositional embeddings using word embeddings in conjunction with unlexicalized features. Various feature sets used in prior work (Turian et al., 2010; Nguyen and Grishman, 2014; Hermann et al., 2014; Roth and Woodsend, 2014) are cap-

*Gormley and Yu contributed equally.
[1]https://github.com/mgormley/pacaya

[2]Such embeddings have a long history in NLP, including term-document frequency matrices and their low-dimensional counterparts obtained by linear algebra tools (LSA, PCA, CCA, NNMF), Brown clusters, random projections and vector space models. Recently, neural networks / deep learning have provided several popular methods for obtaining such embeddings.

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1774–1784,
Lisbon, Portugal, 17-21 September 2015.

Class	M_1	M_2	Sentence Snippet
(1) ART(M_1,M_2)	a man	a taxicab	*A man driving what appeared to be a taxicab*
(2) PART-WHOLE(M_1,M_2)	the southern suburbs	Baghdad	*direction of the southern suburbs of Baghdad*
(3) PHYSICAL(M_2,M_1)	the united states	284 people	*in the united states , 284 people died*

Table 1: Examples from ACE 2005. In (1) the word "driving" is a strong indicator of the relation *ART*[3] between M_1 and M_2. A feature that depends on the embedding for this context word could generalize to other lexical indicators of the same relation (e.g. "operating") that don't appear with *ART* during training. But lexical information alone is insufficient; relation extraction requires the identification of lexical roles: *where* a word appears structurally in the sentence. In (2), the word "of" between "suburbs" and "Baghdad" suggests that the first entity is part of the second, yet the earlier occurrence after "direction" is of no significance to the relation. Even finer information can be expressed by a word's role on the dependency path between entities. In (3) we can distinguish the word "died" from other irrelevant words that don't appear between the entities.

tured as special cases of this construction. Adding these compositional embeddings directly to a standard log-linear model yields a special case of our full model. We then treat the word embeddings as parameters giving rise to our powerful, efficient, and easy-to-implement *log-bilinear model*. The model capitalizes on arbitrary types of linguistic annotations by better utilizing features associated with substructures of those annotations, including global information. We choose features to promote different properties and to distinguish different functions of the input words.

The full model involves three stages. First, it decomposes the annotated sentence into substructures (i.e. a word and associated annotations). Second, it extracts features for each substructure (word), and combines them with the word's embedding to form a *substructure embedding*. Third, we sum over substructure embeddings to form a composed *annotated sentence embedding*, which is used by a final softmax layer to predict the output label (relation).

The result is a state-of-the-art relation extractor for unseen domains from ACE 2005 (Walker et al., 2006) and the relation classification dataset from SemEval-2010 Task 8 (Hendrickx et al., 2010).

Contributions This paper makes several contributions, including:

1. We introduce the FCM, a new compositional embedding model for relation extraction.
2. We obtain the best reported results on ACE-2005 for coarse-grained relation extraction in the cross-domain setting, by combining FCM with a log-linear model.
3. We obtain results on on SemEval-2010 Task 8 competitive with the best reported results.

Note that other work has already been published that builds on the FCM, such as Hashimoto et al. (2015), Nguyen and Grishman (2015), dos Santos et al. (2015), Yu and Dredze (2015) and Yu et al. (2015). Additionally, we have extended FCM to incorporate a low-rank embedding of the features (Yu et al., 2015), which focuses on fine-grained relation extraction for ACE and ERE. This paper obtains better results than the low-rank extension on ACE coarse-grained relation extraction.

[3]In ACE 2005, *ART* refers to a relation between a person and an artifact; such as a user, owner, inventor, or manufacturer relationship

2 Relation Extraction

In relation extraction we are given a sentence as input with the goal of identifying, for all pairs of entity mentions, what relation exists between them, if any. For each pair of entity mentions in a sentence S, we construct an instance $(y, \mathbf{x})$, where $\mathbf{x} = (M_1, M_2, S, A)$. $S = \{w_1, w_2, ..., w_n\}$ is a sentence of length n that expresses a relation of type y between two entity mentions M_1 and M_2, where M_1 and M_2 are sequences of words in S. A is the associated annotations of sentence S, such as part-of-speech tags, a dependency parse, and named entities. We consider directed relations: for a relation type Rel, $y{=}Rel(M_1, M_2)$ and $y'{=}Rel(M_2, M_1)$ are different relations. Table 1 shows ACE 2005 relations, and has a strong label bias towards negative examples. We also consider the task of relation *classification* (SemEval), where the number of negative examples is artificially reduced.

Embedding Models Word embeddings and compositional embedding models have been successfully applied to a range of NLP tasks, however the applications of these embedding models to relation extraction are still limited. Prior work on relation classification (e.g. SemEval 2010 Task 8) has focused on short sentences with at most one relation per sentence (Socher et al., 2012; Zeng et al., 2014). For relation extraction, where negative examples abound, prior work has assumed that only the named entity boundaries and not their types were available (Plank and Moschitti, 2013; Nguyen et al., 2015). Other work has as-

sumed that the order of two entities in a relation are given while the relation type itself is unknown (Nguyen and Grishman, 2014; Nguyen and Grishman, 2015). The standard relation extraction task, as adopted by ACE 2005 (Walker et al., 2006), uses long sentences containing multiple named entities with known types[4] and unknown relation directions. We are the first to apply neural language model embeddings to this task.

Motivation and Examples Whether a word is indicative of a relation depends on multiple properties, which may relate to its context within the sentence. For example, whether the word is in-between the entities, on the dependency path between them, or to their left or right may provide additional complementary information. Illustrative examples are given in Table 1 and provide the motivation for our model. In the next section, we will show how we develop informative representations capturing both the semantic information in word embeddings and the contextual information expressing a word's role relative to the entity mentions. We are the first to incorporate all of this information at once. The closest work is that of Nguyen and Grishman (2014), who use a log-linear model for relation extraction with embeddings as features for only the entity heads. Such embedding features are insensitive to the broader contextual information and, as we show, are not sufficient to elicit the word's role in a relation.

3 A Feature-rich Compositional Embedding Model for Relations

We propose a general framework to construct an embedding of a sentence with annotations on its component words. While we focus on the relation extraction task, the framework applies to any task that benefits from both embeddings and typical hand-engineered lexical features.

3.1 Combining Features with Embeddings

We begin by describing a precise method for constructing substructure embeddings and annotated sentence embeddings from existing (usually unlexicalized) features and embeddings. Note that these embeddings can be included directly in a log-linear model as features—doing so results in a special case of our full model presented in the next subsection.

An annotated sentence is first decomposed into substructures. The type of substructures can vary by task; for relation extraction we consider one substructure per word[5]. For each substructure in the sentence we have a hand-crafted feature vector f_{w_i} and a dense embedding vector e_{w_i}. We represent each substructure as the outer product $\otimes$ between these two vectors to produce a matrix, herein called a **substructure embedding**: $h_{w_i} = f_{w_i} \otimes e_{w_i}$. The features f_{w_i} are based on the local context in S and annotations in A, which can include global information about the annotated sentence. These features allow the model to promote different properties and to distinguish different functions of the words. Feature engineering can be task specific, as relevant annotations can change with regards to each task. In this work we utilize unlexicalized binary features common in relation extraction. Figure 1 depicts the construction of a sentence's substructure embeddings.

We further sum over the substructure embeddings to form an **annotated sentence embedding**:

$$e_x = \sum_{i=1}^{n} f_{w_i} \otimes e_{w_i} \quad (1)$$

When both the hand-crafted features and word embeddings are treated as inputs, as has previously been the case in relation extraction, this annotated sentence embedding can be used directly as the features of a log-linear model. In fact, we find that the feature sets used in prior work for many other NLP tasks are special cases of this simple construction (Turian et al., 2010; Nguyen and Grishman, 2014; Hermann et al., 2014; Roth and Woodsend, 2014). This highlights an important connection: when the word embeddings are constant, our constructions of substructure and annotated sentence embeddings are just specific forms of polynomial (specifically quadratic) feature combination—hence their commonality in the literature. Our experimental results suggest that such a construction is more powerful than directly including embeddings into the model.

3.2 The Log-Bilinear Model

Our full log-bilinear model first forms the substructure and annotated sentence embeddings from

[4] Since the focus of this paper is relation extraction, we adopt the evaluation setting of prior work which uses gold named entities to better facilitate comparison.

[5] We use words as substructures for relation extraction, but use the general terminology to maintain model generality.

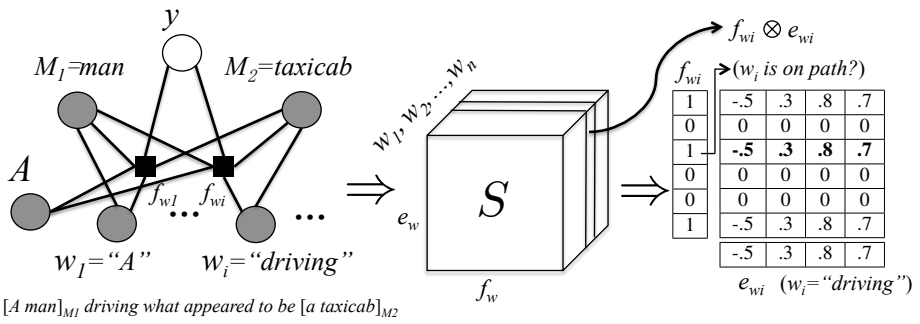


Figure 1: Example construction of substructure embeddings. Each substructure is a word w_i in S, augmented by the target entity information and related information from annotation A (e.g. a dependency tree). We show the factorization of the annotated sentence into substructures (left), the concatenation of the substructure embeddings for the sentence (middle), and a single substructure embedding from that concatenation (right). The annotated sentence embedding (not shown) would be the sum of the substructure embeddings, as opposed to their concatenation.

the previous subsection. The model uses its parameters to score the annotated sentence embedding and uses a softmax to produce an output label. We call the entire model the **Feature-rich Compositional Embedding Model (FCM)**.

Our task is to determine the label y (relation) given the instance $\mathbf{x} = (M_1, M_2, S, A)$. We formulate this as a probability.

$$P(y|\mathbf{x}; T, \mathbf{e}) = \frac{\exp\left(\sum_{i=1}^{n} T_y \odot (f_{w_i} \otimes e_{w_i})\right)}{Z(\mathbf{x})} \quad (2)$$

where $\odot$ is the 'matrix dot product' or Frobenious inner product of the two matrices. The normalizing constant which sums over all possible output labels $y' \in L$ is given by $Z(\mathbf{x}) = \sum_{y' \in L} \exp\left(\sum_{i=1}^{n} T_{y'} \odot (f_{w_i} \otimes e_{w_i})\right)$. The parameters of the model are the word embeddings $\mathbf{e}$ for each word type and a list of weight matrix $T = [T_y]_{y \in L}$ which is used to score each label y. The model is log-bilinear [6] (i.e. log-quadratic) since we recover a log-*linear* model by fixing either $\mathbf{e}$ or T. We study both the full log-bilinear and the log-linear model obtained by fixing the word embeddings.

3.3 Discussion of the Model

Substructure Embeddings Similar words (i.e. those with similar embeddings) with similar functions in the sentence (i.e. those with similar features) will have similar matrix representations. To understand our selection of the outer product, consider the example in Fig. 1. The word "driving" can indicate the *ART* relation if it appears on the dependency path between M_1 and M_2. Suppose the third feature in f_{w_i} indicates this `on-path` feature. Our model can now learn parameters which give the third row a high weight for the *ART* label. Other words with embeddings similar to "driving" that appear on the dependency path between the mentions will similarly receive high weight for the *ART* label. On the other hand, if the embedding is similar but is not on the dependency path, it will have 0 weight. Thus, our model generalizes its model parameters across words with similar embeddings only when they share similar functions in the sentence.

Smoothed Lexical Features Another intuition about the selection of outer product is that it is actually a smoothed version of traditional lexical features used in classical NLP systems. Consider a lexical feature $f = u \wedge w$, which is a conjunction (logic-and) between non-lexical property u and lexical part (word) w. If we represent w as a one-hot vector, then the outer product exactly recovers the original feature f. Then if we replace the one-hot representation with its word embedding, we get the current form of our FCM. Therefore, our model can be viewed as a smoothed version of lexical features, which keeps the expressive strength, and uses embeddings to generalize to low frequency features.

Time Complexity Inference in FCM is much faster than both CNNs (Collobert et al., 2011) and RNNs (Socher et al., 2013b; Bordes et al., 2012). FCM requires $O(snd)$ products on average with sparse features, where s is the average number of per-word non-zero feature values, n is the length of the sentence, and d is the dimension of word embedding. In contrast, CNNs and RNNs usually

[6] Other popular log-bilinear models are the log-bilinear language models (Mnih and Hinton, 2007; Mikolov et al., 2013).

have complexity $O(C \cdot nd^2)$, where C is a model dependent constant.

4 Hybrid Model

We present a hybrid model which combines the FCM with an existing log-linear model. We do so by defining a new model:

$$p_{\text{FCM+loglin}}(y|x) = \frac{1}{Z} p_{\text{FCM}}(y|x) p_{\text{loglin}}(y|x) \quad (3)$$

The log-linear model has the usual form: $p_{\text{loglin}}(y|x) \propto \exp(\boldsymbol{\theta} \cdot \boldsymbol{f}(x, y))$, where $\boldsymbol{\theta}$ are the model parameters and $\boldsymbol{f}(x, y)$ is a vector of features. The integration treats each model as a providing a score which we multiply together. The constant Z ensures a normalized distribution.

5 Training

FCM training optimizes a cross-entropy objective:

$$\ell(D; T, \mathbf{e}) = \sum_{(\mathbf{x}, y) \in D} \log P(y|\mathbf{x}; T, \mathbf{e})$$

where D is the set of all training data and $\mathbf{e}$ is the set of word embeddings. To optimize the objective, for each instance $(y, \mathbf{x})$ we perform stochastic training on the loss function $\ell = \ell(y, \mathbf{x}; T, \mathbf{e}) = \log P(y|\mathbf{x}; T, \mathbf{e})$. The gradients of the model parameters are obtained by backpropagation (i.e. repeated application of the chain rule). We define the vector $\mathbf{s} = [\sum_i T_y \odot (f_{w_i} \otimes e_{w_i})]_{1 \le y \le L}$, which yields

$$\frac{\partial \ell}{\partial \mathbf{s}} = \left[\left(I[y' = y] - P(y'|\mathbf{x}; T, \mathbf{e}) \right)_{1 \le y \le L} \right]^T,$$

where the indicator function $I[x]$ equals 1 if x is true and 0 otherwise. We have the following gradients: $\frac{\partial \ell}{\partial T} = \frac{\partial \ell}{\partial \mathbf{s}} \otimes \sum_{i=1}^{n} f_{w_i} \otimes e_{w_i}$, which is equivalent to:

$$\frac{\partial \ell}{\partial T_{y'}} = \left(I[y = y'] - P(y'|\mathbf{x}; T, \mathbf{e}) \right) \cdot \sum_{i=1}^{n} f_{w_i} \otimes e_{w_i}.$$

When we treat the word embeddings as parameters (i.e. the log-bilinear model), we also fine-tune the word embeddings with the FCM model:

$$\frac{\partial \ell}{\partial e_w} = \sum_{i=1}^{n} \left(\left(\sum_y \frac{\partial \ell}{\partial \mathbf{s}_y} T_y \right) \cdot f_i \right) \cdot I[w_i = w].$$

As is common in deep learning, we initialize these embeddings from an neural language model and then *fine-tune* them for our supervised task. The training process for the hybrid model (§ 4) is also easily done by backpropagation since each sub-model has separate parameters.

Set	Template
`HeadEmb`	$\{I[i = h_1], I[i = h_2]\}$ (w_i is head of M_1/M_2) $\times \{\phi, t_{h_1}, t_{h_2}, t_{h_1} \oplus t_{h_2}\}$
`Context`	$I[i = h_1 \pm 1]$ (left/right token of w_{h_1}) $I[i = h_2 \pm 1]$ (left/right token of w_{h_2})
`In-between`	$I[i > h_1] \& I[i < h_2]$ (in between) $\times \{\phi, t_{h_1}, t_{h_2}, t_{h_1} \oplus t_{h_2}\}$
`On-path`	$I[w_i \in P]$ (on path) $\times \{\phi, t_{h_1}, t_{h_2}, t_{h_1} \oplus t_{h_2}\}$

Table 2: Feature sets used in FCM.

6 Experimental Settings

Features Our FCM features (Table 2) use a feature vector f_{w_i} over the word w_i, the two target entities M_1, M_2, and their dependency path. Here h_1, h_2 are the indices of the two head words of M_1, M_2, $\times$ refers to the Cartesian product between two sets, t_{h_1} and t_{h_2} are entity types (named entity tags for ACE 2005 or WordNet supertags for SemEval 2010) of the head words of two entities, and ϕ stands for the empty feature. $\oplus$ refers to the conjunction of two elements. The `In-between` features indicate whether a word w_i is in between two target entities, and the `On-path` features indicate whether the word is on the dependency path, on which there is a set of words P, between the two entities.

We also use the target entity type as a feature. Combining this with the basic features results in more powerful compound features, which can help us better distinguish the functions of word embeddings for predicting certain relations. For example, if we have a person and a vehicle, we know it will be more likely that they have an *ART* relation. For the *ART* relation, we introduce a corresponding weight vector, which is closer to lexical embeddings similar to the embedding of "drive".

All linguistic annotations needed for features (POS, chunks[7], parses) are from Stanford CoreNLP (Manning et al., 2014). Since SemEval does not have gold entity types we obtained WordNet and named entity tags using Ciaramita and Altun (2006). For all experiments we use 200-d word embeddings trained on the NYT portion of the Gigaword 5.0 corpus (Parker et al., 2011), with word2vec (Mikolov et al., 2013). We use the CBOW model with negative sampling (15 negative words). We set a window size c=5, and remove types occurring less than 5 times.

Models We consider several methods. (1) FCM in isolation without fine-tuning. (2) FCM in isolation with fine-tuning (i.e. trained as a log-bilinear

[7] Obtained from the constituency parse using the CONLL 2000 chunking converter (Perl script).

model). (3) A log-linear model with a rich binary feature set from Sun et al. (2011) (Baseline)—this consists of all the baseline features of Zhou et al. (2005) plus several additional carefully-chosen features that have been highly tuned for ACE-style relation extraction over years of research. We exclude the Country gazetteer and WordNet features from Zhou et al. (2005). The two remaining methods are hybrid models that integrate FCM as a submodel within the log-linear model (§ 4). We consider two combinations. (4) The feature set of Nguyen and Grishman (2014) obtained by using the embeddings of heads of two entity mentions (+HeadOnly). (5) Our full FCM model (+FCM). All models use L2 regularization tuned on dev data.

6.1 Datasets and Evaluation

ACE 2005 We evaluate our relation extraction system on the English portion of the ACE 2005 corpus (Walker et al., 2006).[8] There are 6 domains: Newswire (`nw`), Broadcast Conversation (`bc`), Broadcast News (`bn`), Telephone Speech (`cts`), Usenet Newsgroups (`un`), and Weblogs (`wl`). Following prior work we focus on the domain adaptation setting, where we train on one set (the union of the news domains (`bn+nw`), tune hyperparameters on a dev domain (half of `bc`) and evaluate on the remainder (`cts`, `wl`, and the remainder of `bc`) (Plank and Moschitti, 2013; Nguyen and Grishman, 2014). We assume that gold entity spans *and* types are available for train and test. We use all pairs of entity mentions to yield 43,518 total relations in the training set. We report precision, recall, and F1 for relation extraction. While it is not our focus, for completeness we include results with unknown entity types following Plank and Moschitti (2013) (Appendix 1).

SemEval 2010 Task 8 We evaluate on the SemEval 2010 Task 8 dataset[9] (Hendrickx et al., 2010) to compare with other compositional models and highlight the advantages of FCM. This task is to determine the relation type (or no relation) between two entities in a sentence. We adopt the setting of Socher et al. (2012). We use 10-fold cross validation on the training data to select hyperparameters and do regularization by early stopping. The learning rates for FCM with/without fine-tuning are 5e-3 and 5e-2 respectively. We report macro-F1 and compare to previously published results.

[8] Many relation extraction systems evaluate on the ACE 2004 corpus (Mitchell et al., 2005). Unfortunately, the most common convention is to use 5-fold cross validation, treating the entirety of the dataset as both train *and* evaluation data. Rather than continuing to overfit this data by perpetuating the cross-validation convention, we instead focus on ACE 2005.

[9] `http://docs.google.com/View?docid=dfvxd49s_36c28v9pmw`

7 Results

ACE 2005 Despite FCM's (1) simple feature set, it is competitive with the log-linear baseline (3) on out-of-domain test sets (Table 3). In the typical gold entity spans and types setting, both Plank and Moschitti (2013) and Nguyen and Grishman (2014) found that they were unable to obtain improvements by adding embeddings to baseline feature sets. By contrast, we find that on all domains the combination baseline + FCM (5) obtains the highest F1 and significantly outperforms the other baselines, *yielding the best reported results for this task.* We found that fine-tuning of embeddings (2) did not yield improvements on our out-of-domain development set, in contrast to our results below for SemEval. We suspect this is because fine-tuning allows the model to overfit the training domain, which then hurts performance on the unseen ACE test domains. Accordingly, Table 3 shows only the log-linear model.

Finally, we highlight an important contrast between FCM (1) and the log-linear model (3): the latter uses over 50 feature templates based on a POS tagger, dependency parser, chunker, and constituency parser. FCM uses only a dependency parse but still obtains better results (Avg. F1).

SemEval 2010 Task 8 Table 4 shows FCM compared to the best reported results from the SemEval-2010 Task 8 shared task and several other compositional models.

For the FCM we considered two feature sets. We found that using NE tags instead of WordNet tags helps with fine-tuning but hurts without. This may be because the set of WordNet tags is larger making the model more expressive, but also introduces more parameters. When the embeddings are fixed, they can help to better distinguish different functions of embeddings. But when fine-tuning, it becomes easier to over-fit. Alleviating over-fitting is a subject for future work (§ 9).

With either WordNet or NER features, FCM achieves better performance than the RNN and MVRNN. With NER features and fine-tuning, it outperforms a CNN (Zeng et al., 2014) and also

Model	bc			cts			wl			Avg.
	P	R	F1	P	R	F1	P	R	F1	F1
(1) FCM only (ST)	66.56	**57.86**	61.90	65.62	44.35	52.93	57.80	44.62	50.36	55.06
(3) Baseline (ST)	**74.89**	48.54	58.90	74.32	40.26	52.23	63.41	43.20	51.39	54.17
(4) + HeadOnly (ST)	70.87	50.76	59.16	71.16	43.21	53.77	57.71	42.92	49.23	54.05
(5) + FCM (ST)	74.39	55.35	**63.48**	**74.53**	**45.01**	**56.12**	**65.63**	**47.59**	**55.17**	**58.26**

Table 3: Comparison of models on ACE 2005 out-of-domain test sets. Baseline + HeadOnly is our reimplementation of the features of Nguyen and Grishman (2014).

Classifier	Features	F1
SVM (Rink and Harabagiu, 2010) (Best in SemEval2010)	POS, prefixes, morphological, WordNet, dependency parse, Levin classed, ProBank, FrameNet, NomLex-Plus, Google n-gram, paraphrases, TextRunner	82.2
RNN	word embedding, syntactic parse	74.8
RNN + linear	word embedding, syntactic parse, POS, NER, WordNet	77.6
MVRNN	word embedding, syntactic parse	79.1
MVRNN + linear	word embedding, syntactic parse, POS, NER, WordNet	82.4
CNN (Zeng et al., 2014)	word embedding, WordNet	82.7
CR-CNN (log-loss)	word embedding	82.7
CR-CNN (ranking-loss)	word embedding	**84.1**
RelEmb (word2vec embedding)	word embedding	81.8
RelEmb (task-spec embedding)	word embedding	82.8
RelEmb (task-spec embedding) + linear	word embedding, dependency paths, WordNet, NE	83.5
DepNN	word embedding, dependency paths	82.8
DepNN + linear	word embedding, dependency paths, WordNet, NER	83.6
(1) FCM (log-linear)	word embedding, dependency parse, WordNet	82.0
	word embedding, dependency parse, NER	81.4
(2) FCM (log-bilinear)	word embedding, dependency parse, WordNet	82.5
	word embedding, dependency parse, NER	83.0
(5) FCM (log-linear) + linear (Hybrid)	word embedding, dependency parse, WordNet	83.1
	word embedding, dependency parse, NER	**83.4**

Table 4: Comparison of FCM with previously published results for SemEval 2010 Task 8.

the combination of an embedding model and a traditional log-linear model (RNN/MVRNN + linear) (Socher et al., 2012). As with ACE, FCM uses less linguistic resources than many close competitors (Rink and Harabagiu, 2010).

We also compared to concurrent work on enhancing the compositional models with task-specific information for relation classification, including Hashimoto et al. (2015) (RelEmb), which trained task-specific word embeddings, and dos Santos et al. (2015) (CR-CNN), which proposed a task-specific ranking-based loss function. Our Hybrid methods (FCM + linear) get comparable results to theirs. Note that their base compositional model results without any task-specific enhancements, i.e. RelEmb with word2vec embeddings and CR-CNN with log-loss, are still lower than the best FCM result. We believe that FCM can be also improved with these task-specific enhancements, e.g. replacing the word embeddings to the task-specific ones from (Hashimoto et al., 2015) increases the result to 83.7% (see §7.2 for details). We leave the application of ranking-based loss to future work.

Finally, a concurrent work (Liu et al., 2015) proposes DepNN, which builds representations for the dependency path (and its attached subtrees) between two entities by applying recursive and convolutional neural networks successively. Compared to their model, our FCM achieves comparable results. Of note, our FCM and the RelEmb are also the most efficient models among all above compositional models since they have linear time complexity with respect to the dimension of embeddings.

7.1 Effects of the embedding sub-models

We next investigate the effects of different types of features on FCM using ablation tests on ACE 2005 (Table 5.) We focus on FCM alone with the feature templates of Table 2. Additionally, we show results of using *only* the head embedding features from Nguyen and Grishman (2014) (HeadOnly). Not surprisingly, the HeadOnly model performs poorly (F1 score = 14.30%), showing the importance of our rich binary feature set. Among all the features templates, removing `HeadEmb` results in the largest degradation. The second most im-

Feature Set	Prec	Rec	F1
HeadOnly	31.67	9.24	14.30
FCM	69.17	**56.73**	**62.33**
-HeadEmb	66.06	47.00	54.92
-Context	70.89	55.27	62.11
-In-between	66.39	51.86	58.23
-On-path	69.23	53.97	60.66
FCM-EntityTypes	**71.33**	34.68	46.67

Table 5: Ablation test of FCM on development set.

portant feature template is In-between, while Context features have little impact. Removing all entity type features (t_{h_i}) does significantly worse than the full model, showing the value of our entity type features.

7.2 Effects of the word embeddings

Good word embeddings are critical for both FCM and other compositional models. In this section, we show the results of FCM with embeddings used to initialize other recent state-of-the-art models. Those embeddings include the 300-d baseline embeddings trained on English Wikipedia (w2v-enwiki-d300) and the 100-d task-specific embeddings (task-specific-d100)[10] from the RelEmb paper (Hashimoto et al., 2015), the 400-d embeddings from the CR-CNN paper (dos Santos et al., 2015). Moreover, we list the best result (DepNN) in Liu et al. (2015), which uses the same embeddings as ours. Table 6 shows the effects of word embeddings on FCM and provides relative comparisons between FCM and the other state-of-the-art models. We use the same hyperparameters and number of iterations in Table 4.

The results show that using different embeddings to initialize FCM can improve F1 beyond our previous results. We also find that increasing the dimension of the word embeddings does not necessarily lead to better results due to the problem of over-fitting (e.g.w2v-enwiki-d400 vs. w2v-enwiki-d300). With the same initial embeddings, FCM usually gets better results without any changes to the hyperparameters than the competing model, further confirming the advantage of FCM at the model-level as discussed under Table 4. The only exception is the DepNN model, which gets better result than FCM on the same embeddings. The task-specific embeddings from (Hashimoto et al., 2015) leads to the best performance (an improvement of 0.7%). This observation suggests that the other compositional models may also benefit from the work of Hashimoto et al. (2015).

[10]In the task-specific setting, FCM will represent entity words and context words with separate sets of embeddings.

Embeddings	Model	F1
w2v-enwiki-d300	RelEmb	81.8
	(2) FCM (log-bilinear)	83.4
task-specific-d100	RelEmb	82.8
	RelEmb+linear	83.5
	(2) FCM (log-bilinear)	**83.7**
w2v-enwiki-d400	CR-CNN	82.7
	(2) FCM (log-bilinear)	83.0
w2v-nyt-d200	DepNN	83.6
	(2) FCM (log-bilinear)	83.0

Table 6: Evaluation of FCMs with different word embeddings on SemEval 2010 Task 8.

8 Related Work

Compositional Models for Sentences In order to build a representation (embedding) for a sentence based on its component word embeddings and structural information, recent work on compositional models (stemming from the deep learning community) has designed model structures that mimic the structure of the input. For example, these models could take into account the order of the words (as in Convolutional Neural Networks (CNNs)) (Collobert et al., 2011) or build off of an input tree (as in Recursive Neural Networks (RNNs) or the Semantic Matching Energy Function) (Socher et al., 2013b; Bordes et al., 2012).

While these models work well on sentence-level representations, the nature of their designs also limits them to fixed types of substructures from the annotated sentence, such as chains for CNNs and trees for RNNs. Such models cannot capture arbitrary combinations of linguistic annotations available for a given task, such as word order, dependency tree, and named entities used for relation extraction. Moreover, these approaches ignore the differences in functions between words appearing in different roles. This does not suit more general substructure labeling tasks in NLP, e.g. these models cannot be directly applied to relation extraction since they will output the same result for any pair of entities in a same sentence.

Compositional Models with Annotation Features To tackle the problem of traditional compositional models, Socher et al. (2012) made the RNN model specific to relation extraction tasks by working on the minimal sub-tree which spans the two target entities. However, these specializations

to relation extraction does not generalize easily to other tasks in NLP. There are two ways to achieve such specialization in a more general fashion:

1. Enhancing Compositional Models with Features. A recent trend enhances compositional models with annotation features. Such an approach has been shown to significantly improve over pure compositional models. For example, Hermann et al. (2014) and Nguyen and Grishman (2014) gave different weights to words with different syntactic context types or to entity head words with different argument IDs. Zeng et al. (2014) use concatenations of embeddings as features in a CNN model, according to their positions relative to the target entity mentions. Belinkov et al. (2014) enrich embeddings with linguistic features before feeding them forward to a RNN model. Socher et al. (2013a) and Hermann and Blunsom (2013) enhanced RNN models by refining the transformation matrices with phrase types and CCG super tags.

2. Engineering of Embedding Features. A different approach to combining traditional linguistic features and embeddings is hand-engineering features with word embeddings and adding them to log-linear models. Such approaches have achieved state-of-the-art results in many tasks including NER, chunking, dependency parsing, semantic role labeling, and relation extraction (Miller et al., 2004; Turian et al., 2010; Koo et al., 2008; Roth and Woodsend, 2014; Sun et al., 2011; Plank and Moschitti, 2013). Roth and Woodsend (2014) considered features similar to ours for semantic role labeling.

However, in prior work both of above approaches are only able to utilize limited information, usually one property for each word. Yet there may be different useful properties of a word which can contribute to the performances of the task. By contrast, our FCM can easily utilize these features without changing the model structures.

In order to better utilize the dependency annotations, recently work built their models according to the dependency paths (Ma et al., 2015; Liu et al., 2015), which share similar motivations to the usage of `On-path` features in our work.

Task-Specific Enhancements for Relation Classification An orthogonal direction of improving compositional models for relation classification is to enhance the models with task-specific information. For example, Hashimoto et al. (2015) trained task-specific word embeddings, and dos Santos et al. (2015) proposed a ranking-based loss function for relation classification.

9 Conclusion

We have presented FCM, a new compositional model for deriving sentence-level and substructure embeddings from word embeddings. Compared to existing compositional models, FCM can easily handle arbitrary types of input and handle global information for composition, while remaining easy to implement. We have demonstrated that FCM alone attains near state-of-the-art performances on several relation extraction tasks, and in combination with traditional feature based log-linear models it obtains state-of-the-art results.

Our next steps in improving FCM focus on enhancements based on task-specific embeddings or loss functions as in Hashimoto et al. (2015; dos Santos et al. (2015). Moreover, as the model provides a general idea for representing both sentences and sub-structures in language, it has the potential to contribute useful components to various tasks, such as dependency parsing, SRL and paraphrasing. Also as kindly pointed out by one anonymous reviewer, our FCM can be applied to the TAC-KBP (Ji et al., 2010) tasks, by replacing the training objective to a multi-instance multi-label one (e.g. Surdeanu et al. (2012)). We plan to explore the above applications of FCM in the future.

Acknowledgments

We thank the anonymous reviewers for their comments, and Nicholas Andrews, Francis Ferraro, and Benjamin Van Durme for their input. We thank Kazuma Hashimoto, Cícero Nogueira dos Santos, Bing Xiang and Bowen Zhou for sharing their word embeddings and many helpful discussions. Mo Yu is supported by the China Scholarship Council and by NSFC 61173073.

References

Yonatan Belinkov, Tao Lei, Regina Barzilay, and Amir Globerson. 2014. Exploring compositional architectures and word vector representations for prepositional phrase attachment. *Transactions of the Association for Computational Linguistics*, 2:561–572.

Antoine Bordes, Xavier Glorot, Jason Weston, and Yoshua Bengio. 2012. A semantic matching en-

ergy function for learning with multi-relational data. *Machine Learning*, pages 1–27.

Massimiliano Ciaramita and Yasemin Altun. 2006. Broad-coverage sense disambiguation and information extraction with a supersense sequence tagger. In *EMNLP2006*, pages 594–602, July.

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

Cicero dos Santos, Bing Xiang, and Bowen Zhou. 2015. Classifying relations by ranking with convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 626–634, Beijing, China, July. Association for Computational Linguistics.

Kazuma Hashimoto, Pontus Stenetorp, Makoto Miwa, and Yoshimasa Tsuruoka. 2015. Task-oriented learning of word embeddings for semantic relation classification. *arXiv preprint arXiv:1503.00095*.

Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. 2010. Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In *Proceedings of SemEval-2 Workshop*.

Karl Moritz Hermann and Phil Blunsom. 2013. The role of syntax in vector space models of compositional semantics. In *Association for Computational Linguistics*, pages 894–904.

Karl Moritz Hermann, Dipanjan Das, Jason Weston, and Kuzman Ganchev. 2014. Semantic frame identification with distributed word representations. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1448–1458, Baltimore, Maryland, June. Association for Computational Linguistics.

Heng Ji, Ralph Grishman, Hoa Trang Dang, Kira Griffitt, and Joe Ellis. 2010. Overview of the tac 2010 knowledge base population track. In *Third Text Analysis Conference (TAC 2010)*.

Terry Koo, Xavier Carreras, and Michael Collins. 2008. Simple semi-supervised dependency parsing. In *Proceedings of ACL-08: HLT*, pages 595–603, Columbus, Ohio, June. Association for Computational Linguistics.

Qi Li and Heng Ji. 2014. Incremental joint extraction of entity mentions and relations. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 402–412, Baltimore, Maryland, June. Association for Computational Linguistics.

Yang Liu, Furu Wei, Sujian Li, Heng Ji, Ming Zhou, and Houfeng WANG. 2015. A dependency-based neural network for relation classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 285–290, Beijing, China, July. Association for Computational Linguistics.

Mingbo Ma, Liang Huang, Bowen Zhou, and Bing Xiang. 2015. Dependency-based convolutional neural networks for sentence embedding. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 174–179, Beijing, China, July. Association for Computational Linguistics.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. *arXiv preprint arXiv:1310.4546*.

Scott Miller, Jethran Guinness, and Alex Zamanian. 2004. Name tagging with word clusters and discriminative training. In Susan Dumais, Daniel Marcu, and Salim Roukos, editors, *HLT-NAACL 2004: Main Proceedings*. Association for Computational Linguistics.

Alexis Mitchell, Stephanie Strassel, Shudong Huang, and Ramez Zakhary. 2005. Ace 2004 multilingual training corpus. *Linguistic Data Consortium, Philadelphia*.

Andriy Mnih and Geoffrey Hinton. 2007. Three new graphical models for statistical language modelling. In *Proceedings of the 24th international conference on Machine learning*, pages 641–648. ACM.

Thien Huu Nguyen and Ralph Grishman. 2014. Employing word representations and regularization for domain adaptation of relation extraction. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 68–74, Baltimore, Maryland, June. Association for Computational Linguistics.

Thien Huu Nguyen and Ralph Grishman. 2015. Relation extraction: Perspective from convolutional neural networks. In *Proceedings of NAACL Workshop on Vector Space Modeling for NLP*.

Thien Huu Nguyen, Barbara Plank, and Ralph Grishman. 2015. Semantic representations for do-

main adaptation: A case study on the tree kernel-based method for relation extraction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 635–644, Beijing, China, July. Association for Computational Linguistics.

Robert Parker, David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2011. English gigaword fifth edition, june. *Linguistic Data Consortium, LDC2011T07*.

Barbara Plank and Alessandro Moschitti. 2013. Embedding semantic similarity in tree kernels for domain adaptation of relation extraction. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1498–1507, Sofia, Bulgaria, August. Association for Computational Linguistics.

Bryan Rink and Sanda Harabagiu. 2010. Utd: Classifying semantic relations by combining lexical and semantic resources. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 256–259, Uppsala, Sweden, July. Association for Computational Linguistics.

Michael Roth and Kristian Woodsend. 2014. Composition of word representations improves semantic role labelling. In *EMNLP*.

Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1201–1211, Jeju Island, Korea, July. Association for Computational Linguistics.

Richard Socher, John Bauer, Christopher D Manning, and Andrew Y Ng. 2013a. Parsing with compositional vector grammars. In *In Proceedings of the ACL conference*. Citeseer.

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

Ang Sun, Ralph Grishman, and Satoshi Sekine. 2011. Semi-supervised relation extraction with large-scale word clustering. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 521–529, Portland, Oregon, USA, June. Association for Computational Linguistics.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D Manning. 2012. Multi-instance multi-label learning for relation extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 455–465. Association for Computational Linguistics.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Association for Computational Linguistics*, pages 384–394.

Christopher Walker, Stephanie Strassel, Julie Medero, and Kazuaki Maeda. 2006. ACE 2005 multilingual training corpus. *Linguistic Data Consortium, Philadelphia*.

Mo Yu and Mark Dredze. 2015. Learning composition models for phrase embeddings. *Transactions of the Association for Computational Linguistics*, 3:227–242.

Mo Yu, Matthew R. Gormley, and Mark Dredze. 2015. Combining word embeddings and feature embeddings for fine-grained relation extraction. In *Proceedings of NAACL*.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation classification via convolutional deep neural network. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 2335–2344, Dublin, Ireland, August. Dublin City University and Association for Computational Linguistics.

GuoDong Zhou, Jian Su, Jie Zhang, and Min Zhang. 2005. Exploring various knowledge in relation extraction. In *Association for Computational Linguistics*, pages 427–434.

Classifying Relations via Long Short Term Memory Networks along Shortest Dependency Paths

Yan Xu[†], Lili Mou[†], Ge Li[†*], Yunchuan Chen[‡], Hao Peng[†], Zhi Jin[†*]
[†]Software Institute, Peking University, 100871, P. R. China
{xuyan14,lige,zhijin}@sei.pku.edu.cn,{doublepower.mou,penghao.pku}@gmail.com
[‡]University of Chinese Academy of Sciences, chenyunchuan11@mails.ucas.ac.cn

Abstract

Relation classification is an important research arena in the field of natural language processing (NLP). In this paper, we present SDP-LSTM, a novel neural network to classify the relation of two entities in a sentence. Our neural architecture leverages the shortest dependency path (SDP) between two entities; multichannel recurrent neural networks, with long short term memory (LSTM) units, pick up heterogeneous information along the SDP. Our proposed model has several distinct features: (1) The shortest dependency paths retain most relevant information (to relation classification), while eliminating irrelevant words in the sentence. (2) The multichannel LSTM networks allow effective information integration from heterogeneous sources over the dependency paths. (3) A customized dropout strategy regularizes the neural network to alleviate overfitting. We test our model on the SemEval 2010 relation classification task, and achieve an F_1-score of 83.7%, higher than competing methods in the literature.

1 Introduction

Relation classification is an important NLP task. It plays a key role in various scenarios, e.g., information extraction (Wu and Weld, 2010), question answering (Yao and Van Durme, 2014), medical informatics (Wang and Fan, 2014), ontology learning (Xu et al., 2014), etc. The aim of relation classification is to categorize into predefined classes the relations between pairs of marked entities in given texts. For instance, in the sentence "A trillion gallons of [water]$_{e_1}$ have been poured into an empty [region]$_{e_2}$ of outer space," the entities *water* and *region* are of relation `Entity-Destination`(e_1, e_2).

Traditional relation classification approaches rely largely on feature representation (Kambhatla, 2004), or kernel design (Zelenko et al., 2003; Bunescu and Mooney, 2005). The former method usually incorporates a large set of features; it is difficult to improve the model performance if the feature set is not very well chosen. The latter approach, on the other hand, depends largely on the designed kernel, which summarizes all data information. Deep neural networks, emerging recently, provide a way of highly automatic feature learning (Bengio et al., 2013), and have exhibited considerable potential (Zeng et al., 2014; dos Santos et al., 2015). However, human engineering—that is, incorporating human knowledge to the network's architecture—is still important and beneficial.

This paper proposes a new neural network, SDP-LSTM, for relation classification. Our model utilizes the shortest dependency path (SDP) between two entities in a sentence; we also design a long short term memory (LSTM)-based recurrent neural network for information processing. The neural architecture is mainly inspired by the following observations.

- Shortest dependency paths are informative (Fundel et al., 2007; Chen et al., 2014). To determine the two entities' relation, we find it mostly sufficient to use only the words along the SDP: they concentrate on most relevant information while diminishing less relevant noise. Figure 1 depicts the dependency parse tree of the aforementioned sentence. Words along the SDP form a trimmed phrase (*gallons of water poured into region*) of the original sentence, which conveys much information about the target relation. Other words, such as *a*, *trillion*, *outer space*, are less informative and may bring noise if not dealt with properly.

*Corresponding authors.

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1785–1794,
Lisbon, Portugal, 17-21 September 2015.

- Direction matters. Dependency trees are a kind of directed graph. The dependency relation between *into* and *region* is `PREP`; such relation hardly makes any sense if the directed edge is reversed. Moreover, the entities' relation distinguishes its directionality, that is, $r(a, b)$ differs from $r(b, a)$, for a same given relation r and two entities a, b. Therefore, we think it necessary to let the neural model process information in a direction-sensitive manner. Out of this consideration, we separate an SDP into two sub-paths, each from an entity to the common ancestor node. The extracted features along the two sub-paths are concatenated to make final classification.

- Linguistic information helps. For example, with prior knowledge of hyponymy, we know "water is a kind of substance." This is a hint that the entities, *water* and *region*, are more of `Entity-Destination` relation than, say, `Communication-Topic`. To gather heterogeneous information along SDP, we design a multichannel recurrent neural network. It makes use of information from various sources, including words themselves, POS tags, WordNet hypernyms, and the grammatical relations between governing words and their children.

For effective information propagation and integration, our model leverages LSTM units during recurrent propagation. We also customize a new dropout strategy for our SDP-LSTM network to alleviate the problem of overfitting. To the best of our knowledge, we are the first to use LSTM-based recurrent neural networks for the relation classification task.

We evaluate our proposed method on the SemEval 2010 relation classification task, and achieve an F_1-score of 83.7%, higher than competing methods in the literature.

In the rest of this paper, we review related work in Section 2. In Section 3, we describe our SDP-LSTM model in detail. Section 4 presents quantitative experimental results. Finally, we have our conclusion in Section 5.

2 Related Work

Relation classification is a widely studied task in the NLP community. Various existing methods mainly fall into three classes: feature-based, kernel-based, and neural network-based.

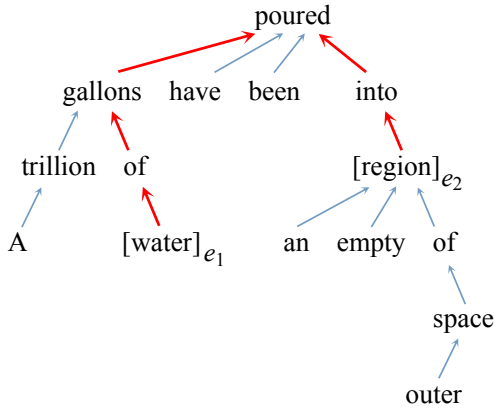


Figure 1: The dependency parse tree corresponding to the sentence "A trillion gallons of water have been poured into an empty region of outer space." Red lines indicate the shortest dependency path between entities *water* and *region*. An edge $a \rightarrow b$ refers to a being governed by b. Dependency types are labeled by the parser, but not presented in the figure for clarity.

In feature-based approaches, different sets of features are extracted and fed to a chosen classifier (e.g., logistic regression). Generally, three types of features are often used. Lexical features concentrate on the entities of interest, e.g., entities *per se*, entity POS, entity neighboring information. Syntactic features include chunking, parse trees, etc. Semantic features are exemplified by the concept hierarchy, entity class, entity mention. Kambhatla (2004) uses a maximum entropy model to combine these features for relation classification. However, different sets of handcrafted features are largely complementary to each other (e.g., hypernyms versus named-entity tags), and thus it is hard to improve performance in this way (GuoDong et al., 2005).

Kernel-based approaches specify some measure of similarity between two data samples, without explicit feature representation. Zelenko et al. (2003) compute the similarity of two trees by utilizing their common subtrees. Bunescu and Mooney (2005) propose a shortest path dependency kernel for relation classification. Its main idea is that the relation strongly relies on the dependency path between two given entities. Wang (2008) provides a systematic analysis of several kernels and show that relation extraction can bene-

fit from combining convolution kernel and syntactic features. Plank and Moschitti (2013) introduce semantic information into kernel methods in addition to considering structural information only. One potential difficulty of kernel methods is that all data information is completely summarized by the kernel function (similarity measure), and thus designing an effective kernel becomes crucial.

Deep neural networks, emerging recently, can learn underlying features automatically, and have attracted growing interest in the literature. Socher et al. (2011) propose a recursive neural network (RNN) along sentences' parse trees for sentiment analysis; such model can also be used to classify relations (Socher et al., 2012). Hashimoto et al. (2013) explicitly weight phrases' importance in RNNs to improve performance. Ebrahimi and Dou (2015) rebuild an RNN on the dependency path between two marked entities. Zeng et al. (2014) explore convolutional neural networks, by which they utilize sequential information of sentences. dos Santos et al. (2015) also use the convolutional network; besides, they propose a ranking loss function with data cleaning, and achieve the state-of-the-art result in SemEval-2010 Task 8.

In addition to the above studies, which mainly focus on relation classification approaches and models, other related research trends include information extraction from Web documents in a semi-supervised manner (Bunescu and Mooney, 2007; Banko et al., 2007), dealing with small datasets without enough labels by distant supervision techniques (Mintz et al., 2009), etc.

3 The Proposed SDP-LSTM Model

In this section, we describe our SDP-LSTM model in detail. Subsection 3.1 delineates the overall architecture of our model. Subsection 3.2 presents the rationale of using SDPs. Four different information channels along the SDP are explained in Subsection 3.3. Subsection 3.4 introduces the recurrent neural network with long short term memory, which is built upon the dependency path. Subsection 3.5 customizes a dropout strategy for our network to alleviate overfitting. We finally present our training objective in Subsection 3.6.

3.1 Overview

Figure 2 depicts the overall architecture of our SDP-LSTM network.

First, a sentence is parsed to a dependency tree by the Stanford parser;[1] the shortest dependency path (SDP) is extracted as the input of our network. Along the SDP, four different types of information—referred to as *channels*—are used, including the words, POS tags, grammatical relations, and WordNet hypernyms. (See Figure 2a.) In each channel, discrete inputs, e.g., words, are mapped to real-valued vectors, called *embeddings*, which capture the underlying meanings of the inputs.

Two recurrent neural networks (Figure 2b) pick up information along the left and right sub-paths of the SDP, respecitvely. (The path is separated by the common ancestor node of two entities.) Long short term memory (LSTM) units are used in the recurrent networks for effective information propagation. A max pooling layer thereafter gathers information from LSTM nodes in each path.

The pooling layers from different channels are concatenated, and then connected to a hidden layer. Finally, we have a $\mathrm{softmax}$ output layer for classification. (See again Figure 2a.)

3.2 The Shortest Dependency Path

The dependency parse tree is naturally suitable for relation classification because it focuses on the action and agents in a sentence (Socher et al., 2014). Moreover, the shortest path between entities, as discussed in Section 1, condenses most illuminating information for entities' relation.

We also observe that the sub-paths, separated by the common ancestor node of two entities, provide strong hints for the relation's directionality. Take Figure 1 as an example. Two entities *water* and *region* have their common ancestor node, *poured*, which separates the SDP into two parts:

$$[\text{water}]_{e_1} \rightarrow \text{of} \rightarrow \text{gallons} \rightarrow \text{poured}$$

and

$$\text{poured} \leftarrow \text{into} \leftarrow [\text{region}]_{e_2}$$

The first sub-path captures information of e_1, whereas the second sub-path is mainly about e_2. By examining the two sub-paths separately, we know e_1 and e_2 are of relation $\texttt{Entity-Destination}(e_1, e_2)$, rather than $\texttt{Entity-Destination}(e_2, e_1)$.

Following the above intuition, we design two recurrent neural networks, which propagate

[1] http://nlp.stanford.edu/software/lex-parser.shtml

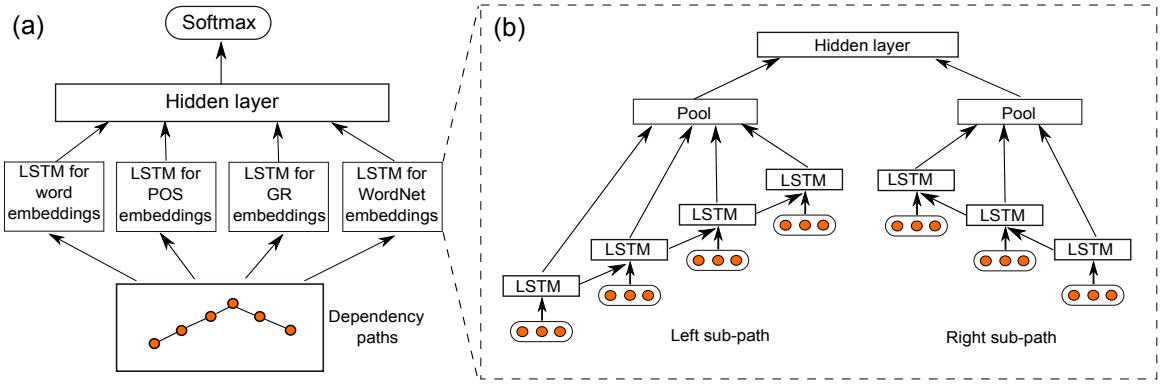


Figure 2: (a) The overall architecture of SDP-LSTM. (b) One channel of the recurrent neural networks built upon the shortest dependency path. The channels are words, part-of-speech (POS) tags, grammatical relations (abbreviated as *GR* in the figure), and WordNet hypernyms.

bottom-up from the entities to their common ancestor. In this way, our model is direction-sensitive.

3.3 Channels

We make use of four types of information along the SDP for relation classification. We call them *channels* as these information sources do not interact during recurrent propagation. Detailed channel descriptions are as follows.

- **Word representations**. Each word in a given sentence is mapped to a real-valued vector by looking up in a word embedding table. Unsupervisedly trained on a large corpus, word embeddings are thought to be able to well capture words' syntactic and semantic information (Mikolov et al., 2013b).

- **Part-of-speech tags**. Since word embeddings are obtained on a generic corpus of a large scale, the information they contain may not agree with a specific sentence. We deal with this problem by allying each input word with its POS tag, e.g., `noun`, `verb`, etc. In our experiment, we only take into use a coarse-grained POS category, containing 15 different tags.

- **Grammatical relations**. The dependency relations between a governing word and its children makes a difference in meaning. A same word pair may have different dependency relation types. For example, "*beats* $\xrightarrow{\text{nsubj}}$ *it*" is distinct from "*beats* $\xrightarrow{\text{dobj}}$ *it*." Thus, it is necessary to capture such grammatical relations in SDPs. In our experiment, grammatical relations are grouped into 19 classes, mainly based on a coarse-grained classification (De Marneffe et al., 2006).

- **WordNet hypernyms**. As illustrated in Section 1, hyponymy information is also useful for relation classification. (Details are not repeated here.) To leverage WordNet hypernyms, we use a tool developed by Ciaramita and Altun (2006).[2] The tool assigns a hypernym to each word, from 41 predefined concepts in WordNet, e.g., `noun.food`, `verb.motion`, etc. Given its hypernym, each word gains a more abstract concept, which helps to build a linkage between different but conceptual similar words.

As we can see, POS tags, grammatical relations, and WordNet hypernyms are also discrete (like words *per se*). However, no prevailing embedding learning method exists for POS tags, say. Hence, we randomly initialize their embeddings, and tune them in a supervised fashion during training. We notice that these information sources contain much fewer symbols, 15, 19, and 41, than the vocabulary size (greater than 25,000). Hence, we believe our strategy of random initialization is feasible, because they can be adequately tuned during supervised training.

3.4 Recurrent Neural Network with Long Short Term Memory Units

The recurrent neural network is suitable for modeling sequential data by nature, as it keeps a hid-

[2] http://sourceforge.net/projects/supersensetag

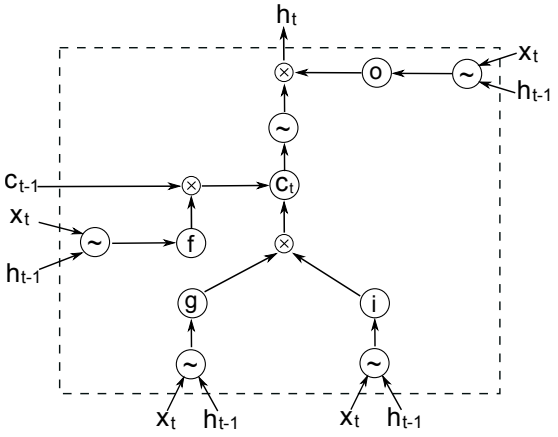


Figure 3: A long short term memory unit. h: hidden unit. c: memory cell. i: input gate. f: forget gate. o: output gate. g: candidate cell. $\otimes$: element-wise multiplication. $\sim$: activation function.

den state vector $\boldsymbol{h}$, which changes with input data at each step accordingly. We use the recurrent network to gather information along each sub-path in the SDP (Figure 2b).

The hidden state $\boldsymbol{h}_t$, for the t-th word in the sub-path, is a function of its previous state $\boldsymbol{h}_{t-1}$ and the current word $\boldsymbol{x}_t$. Traditional recurrent networks have a basic interaction, that is, the input is linearly transformed by a weight matrix and nonlinearly squashed by an activation function. Formally, we have

$$\boldsymbol{h}_t = f(W_{in}\boldsymbol{x}_t + W_{rec}\boldsymbol{h}_{t-1} + \boldsymbol{b}_h)$$

where W_{in} and W_{rec} are weight matrices for the input and recurrent connections, respectively. $\boldsymbol{b}_h$ is a bias term for the hidden state vector, and f_h a non-linear activation function (e.g., $\tanh$).

One problem of the above model is known as *gradient vanishing or exploding*. The training of neural networks requires gradient back-propagation. If the propagation sequence (path) is too long, the gradient may probably either grow, or decay, exponentially, depending on the magnitude of W_{rec}. This leads to the difficulty of training.

Long short term memory (LSTM) units are proposed in Hochreiter (1998) to overcome this problem. The main idea is to introduce an adaptive gating mechanism, which decides the degree to which LSTM units keep the previous state and memorize the extracted features of the current data input. Many LSTM variants have been proposed in the literature. We adopt in our method a variant introduced by Zaremba and Sutskever (2014), also used in Zhu et al. (2015).

Concretely, the LSTM-based recurrent neural network comprises four components: an input gate $\boldsymbol{i}_t$, a forget gate $\boldsymbol{f}_t$, an output gate $\boldsymbol{o}_t$, and a memory cell $\boldsymbol{c}_t$ (depicted in Figure 3 and formalized through Equations 1–6 as bellow).

The three adaptive gates $\boldsymbol{i}_t$, $\boldsymbol{f}_t$, and $\boldsymbol{o}_t$ depend on the previous state $\boldsymbol{h}_{t-1}$ and the current input $\boldsymbol{x}_t$ (Equations 1–3). An extracted feature vector $\boldsymbol{g}_t$ is also computed, by Equation 4, serving as the candidate memory cell.

$$\boldsymbol{i}_t = \sigma(W_i{\cdot}\boldsymbol{x}_t + U_i{\cdot}\boldsymbol{h}_{t-1} + \boldsymbol{b}_i) \quad (1)$$
$$\boldsymbol{f}_t = \sigma(W_f{\cdot}\boldsymbol{x}_t + U_f{\cdot}\boldsymbol{h}_{t-1} + \boldsymbol{b}_f) \quad (2)$$
$$\boldsymbol{o}_t = \sigma(W_o{\cdot}\boldsymbol{x}_t + U_o{\cdot}\boldsymbol{h}_{t-1} + \boldsymbol{b}_o) \quad (3)$$
$$\boldsymbol{g}_t = \tanh(W_g{\cdot}\boldsymbol{x}_t + U_g{\cdot}\boldsymbol{h}_{t-1} + \boldsymbol{b}_g) \quad (4)$$

The current memory cell $\boldsymbol{c}_t$ is a combination of the previous cell content $\boldsymbol{c}_{t-1}$ and the candidate content $\boldsymbol{g}_t$, weighted by the input gate $\boldsymbol{i}_t$ and forget gate $\boldsymbol{f}_t$, respectively. (See Equation 5 below.)

$$\boldsymbol{c}_t = \boldsymbol{i}_t \otimes \boldsymbol{g}_t + \boldsymbol{f}_t \otimes \boldsymbol{c}_{t-1} \quad (5)$$

The output of LSTM units is the the recurrent network's hidden state, which is computed by Equation 6 as follows.

$$\boldsymbol{h}_t = \boldsymbol{o}_t \otimes \tanh(\boldsymbol{c}_t) \quad (6)$$

In the above equations, σ denotes a $\mathrm{sigmoid}$ function; $\otimes$ denotes element-wise multiplication.

3.5 Dropout Strategies

A good regularization approach is needed to alleviate overfitting. Dropout, proposed recently by Hinton et al. (2012), has been very successful on feed-forward networks. By randomly omitting feature detectors from the network during training, it can obtain less interdependent network units and achieve better performance. However, the conventional dropout does not work well with recurrent neural networks with LSTM units, since dropout may hurt the valuable memorization ability of memory units.

As there is no consensus on how to drop out LSTM units in the literature, we try several dropout strategies for our SDP-LSTM network:

- Dropout embeddings;
- Dropout inner cells in memory units, including i_t, g_t, o_t, c_t, and h_t; and

- Dropout the penultimate layer.

As we shall see in Section 4.2, dropping out LSTM units turns out to be inimical to our model, whereas the other two strategies boost in performance.

The following equations formalize the dropout operations on the embedding layers, where D denotes the dropout operator. Each dimension in the embedding vector, $\boldsymbol{x}_t$, is set to zero with a predefined dropout rate.

$$\boldsymbol{i}_t = \sigma(W_i \cdot D(\boldsymbol{x}_t) + U_i \cdot \boldsymbol{h}_{t-1} + \boldsymbol{b}_i) \quad (7)$$

$$\boldsymbol{f}_t = \sigma(W_f \cdot D(\boldsymbol{x}_t) + U_f \cdot \boldsymbol{h}_{t-1} + \boldsymbol{b}_f) \quad (8)$$

$$\boldsymbol{o}_t = \sigma(W_o \cdot D(\boldsymbol{x}_t) + U_o \cdot \boldsymbol{h}_{t-1} + \boldsymbol{b}_o) \quad (9)$$

$$\boldsymbol{g}_t = \tanh\Big(W_g \cdot D(\boldsymbol{x}_t) + U_g \cdot \boldsymbol{h}_{t-1} + \boldsymbol{b}_g\Big) \quad (10)$$

3.6 Training Objective

The SDP-LSTM described above propagates information along a sub-path from an entity to the common ancestor node (of the two entities). A max pooling layer packs, for each sub-path, the recurrent network's states, $\boldsymbol{h}$'s, to a fixed vector by taking the maximum value in each dimension.

Such architecture applies to all channels, namely, words, POS tags, grammatical relations, and WordNet hypernyms. The pooling vectors in these channels are concatenated, and fed to a fully connected hidden layer. Finally, we add a $\mathrm{softmax}$ output layer for classification. The training objective is the penalized cross-entropy error, given by

$$J = -\sum_{i=1}^{n_c} t_i \log y_i + \lambda \left(\sum_{i=1}^{\omega} \|W_i\|_F^2 + \sum_{i=1}^{\upsilon} \|U_i\|_F^2 \right)$$

where $\boldsymbol{t} \in \mathbb{R}^{n_c}$ is the one-hot represented ground truth and $\boldsymbol{y} \in \mathbb{R}^{n_c}$ is the estimated probability for each class by $\mathrm{softmax}$. (n_c is the number of target classes.) $\|\cdot\|_F$ denotes the Frobenius norm of a matrix; ω and υ are the numbers of weight matrices (for W's and U's, respectively). λ is a hyperparameter that specifies the magnitude of penalty on weights. Note that we do not add ℓ_2 penalty to biase parameters.

We pretrained word embeddings by `word2vec` (Mikolov et al., 2013a) on the English Wikipedia corpus; other parameters are initialized randomly. We apply stochastic gradient descent (with mini-batch 10) for optimization; gradients are computed by standard back-propagation. Training details are further introduced in Section 4.2.

4 Experiments

In this section, we present our experiments in detail. Our implementation is built upon Mou et al. (2015). Section 4.1 introduces the dataset; Section 4.2 describes hyperparameter settings. In Section 4.3, we compare SDP-LSTM's performance with other methods in the literature. We also analyze the effect of different channels in Section 4.4.

4.1 Dataset

The SemEval-2010 Task 8 dataset is a widely used benchmark for relation classification (Hendrickx et al., 2009). The dataset contains 8,000 sentences for training, and 2,717 for testing. We split 1/10 samples out of the training set for validation.

The target contains 19 labels: 9 directed relations, and an undirected `Other` class. The directed relations are list as below.

- `Cause-Effect`
- `Component-Whole`
- `Content-Container`
- `Entity-Destination`
- `Entity-Origin`
- `Message-Topic`
- `Member-Collection`
- `Instrument-Agency`
- `Product-Producer`

In the following are illustrated two sample sentences with directed relations.

> [People]$_{e_1}$ have been moving back into [downtown]$_{e_2}$.

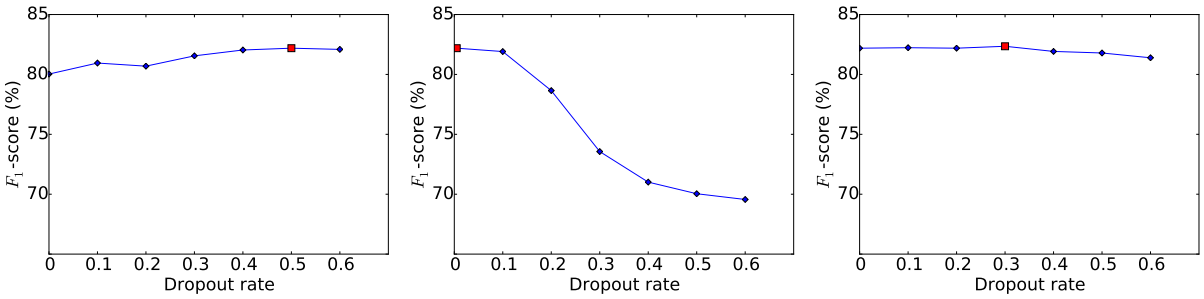
> Financial [stress]$_{e_1}$ is one of the main causes of [divorce]$_{e_2}$.

The target labels are `Entity-Destination` (e_1, e_2), and `Cause-Effect`(e_1, e_2), respectively.

The dataset also contains an undirected `Other` class. Hence, there are 19 target labels in total. The undirected `Other` class takes in entities that do not fit into the above categories, illustrated by the following example.

> A misty [ridge]$_{e_1}$ uprises from the [surge]$_{e_2}$.

We use the official macro-averaged F_1-score to evaluate model performance. This official measurement excludes the `Other` relation. Nonetheless, we have no special treatment of `Other` class in our experiments, which is typical in other studies.



(a) Dropout word embeddings (b) Dropout inner cells of memory units (c) Dropout the penultimate layer

Figure 4: F_1-scores versus dropout rates. We first evaluate the effect of dropout embeddings (a). Then the dropout of the inner cells (b) and the penultimate layer (c) is tested with word embeddings being dropped out by 0.5.

4.2 Hyperparameters and Training Details

This subsection presents hyperparameter tuning for our model. We set word-embeddings to be 200-dimensional; POS, WordNet hyponymy, and grammatical relation embeddings are 50-dimensional. Each channel of the LSTM network contains the same number of units as its source embeddings (either 200 or 50). The penultimate hidden layer is 100-dimensional. As it is not feasible to perform full grid search for all hyperparameters, the above values are chosen empirically.

We add ℓ_2 penalty for weights with coefficient 10^{-5}, which was chosen by validation from the set $\{10^{-2}, 10^{-3}, \cdots, 10^{-7}\}$.

We thereafter validate the proposed dropout strategies in Section 3.5. Since network units in different channels do not interact with each other during information propagation, we herein take one channel of LSTM networks to assess the efficacy. Taking the word channel as an example, we first drop out word embeddings. Then with a fixed dropout rate of word embeddings, we test the effect of dropping out LSTM inner cells and the penultimate units, respectively.

We find that, dropout of LSTM units hurts the model, even if the dropout rate is small, 0.1, say (Figure 4b). Dropout of embeddings improves model performance by 2.16% (Figure 4a); dropout of the penultimate layer further improves by 0.16% (Figure 4c). This analysis also provides, for other studies, some clues for dropout in LSTM networks.

4.3 Results

Table 4 compares our SDT-LSTM with other state-of-the-art methods. The first entry in the table presents the highest performance achieved by traditional feature engineering. Hendrickx et al. (2009) leverage a variety of handcrafted features, and use SVM for classification; they achieve an F_1-score of 82.2%.

Neural networks are first used in this task in Socher et al. (2012). They build a recursive neural network (RNN) along a constituency tree for relation classification. They extend the basic RNN with matrix-vector interaction and achieve an F_1-score of 82.4%.

Zeng et al. (2014) treat a sentence as sequential data and exploit the convolutional neural network (CNN); they also integrate word position information into their model. dos Santos et al. (2015) design a model called CR-CNN; they propose a ranking-based cost function and elaborately diminish the impact of the `Other` class, which is not counted in the official F_1-measure. In this way, they achieve the state-of-the-art result with the F_1-score of 84.1%. Without such special treatment, their F_1-score is 82.7%.

Yu et al. (2014) propose a Feature-rich Compositional Embedding Model (FCM) for relation classification, which combines unlexicalized linguistic contexts and word embeddings. They achieve an F_1-score of 83.0%.

Our proposed SDT-LSTM model yields an F_1-score of 83.7%. It outperforms existing competing approaches, in a fair condition of softmax with cross-entropy error.

It is worth to note that we have also conducted two controlled experiments: (1) Traditional RNN without LSTM units, achieving an F_1-score of 82.8%; (2) LSTM network over the entire dependency path (instead of two sub-paths), achieving an F_1-score of 82.2%. These results demonstrate

Classifier	Feature set	F_1
SVM	POS, WordNet, prefixes and other morphological features, depdency parse, Levin classes, PropBank, FanmeNet, NomLex-Plus, Google n-gram, paraphrases, TextRunner	82.2
RNN	Word embeddings	74.8
	Word embeddings, POS, NER, WordNet	77.6
MVRNN	Word embeddings	79.1
	Word embeddings, POS, NER, WordNet	82.4
CNN	Word embeddings	69.7
	Word embeddings, word position embeddings, WordNet	82.7
Chain CNN	Word embeddings, POS, NER, WordNet	82.7
FCM	Word embeddings	80.6
	Word embeddings, depedency parsing, NER	83.0
CR-CNN	Word embeddings	82.8†
	Word embeddings, position embeddings	82.7
	Word embeddings, position embeddings	**84.1**†
SDP-LSTM	Word embeddings	82.4
	Word embeddings, POS embeddings, WordNet embeddings, grammar relation embeddings	**83.7**

Table 1: Comparison of relation classification systems. The "†" remark refers to special treatment for the `Other` class.

the effectiveness of LSTM and directionality in relation classification.

4.4 Effect of Different Channels

This subsection analyzes how different channels affect our model. We first used word embeddings only as a baseline; then we added POS tags, grammatical relations, and WordNet hypernyms, respectively; we also combined all these channels into our models. Note that we did not try the latter three channels alone, because each single of them (e.g., POS) does not carry much information.

We see from Table 2 that word embeddings alone in SDP-LSTM yield a remarkable performance of 82.35%, compared with CNNs 69.7%, RNNs 74.9–79.1%, and FCM 80.6%.

Adding either grammatical relations or WordNet hypernyms outperforms other existing methods (data cleaning not considered here). POS tagging is comparatively less informative, but still boosts the F_1-score by 0.63%.

We notice that, the boosts are not simply added when channels are combined. This suggests that these information sources are complementary to each other in some linguistic aspects. Nonetheless, incorporating all four channels further pushes the F_1-score to 83.70%.

Channels	F_1
Word embeddings	82.35
+ POS embeddings (only)	82.98
+ GR embeddings (only)	83.21
+ WordNet embeddings (only)	83.03
+ POS + GR + WordNet embeddings	83.70

Table 2: Effect of different channels.

5 Conclusion

In this paper, we propose a novel neural network model, named SDP-LSTM, for relation classification. It learns features for relation classification iteratively along the shortest dependency path. Several types of information (word themselves, POS tags, grammatical relations and WordNet hypernyms) along the path are used. Meanwhile, we leverage LSTM units for long-range information propagation and integration. We demonstrate the effectiveness of SDP-LSTM by evaluating the model on SemEval-2010 relation classification task, outperforming existing state-of-art methods (in a fair condition without data cleaning). Our result sheds some light in the relation classification task as follows.

- The shortest dependency path can be a valuable resource for relation classification, covering mostly sufficient information of target

relations.

- Classifying relation is a challenging task due to the inherent ambiguity of natural languages and the diversity of sentence expression. Thus, integrating heterogeneous linguistic knowledge is beneficial to the task.
- Treating the shortest dependency path as two sub-paths, mapping two different neural networks, helps to capture the directionality of relations.
- LSTM units are effective in feature detection and propagation along the shortest dependency path.

Acknowledgments

This research is supported by the National Basic Research Program of China (the 973 Program) under Grant No. 2015CB352201 and the National Natural Science Foundation of China under Grant Nos. 61232015 and 91318301.

References

Michele Banko, Michael J. Cafarella, Stephen Soderland, Matthew Broadhead, and Oren Etzioni. 2007. Open information extraction for the web. In *Proceedings of twentieth International Joint Conference on Artificial Intelligence*, volume 7, pages 2670–2676.

Yoshua Bengio, Aaron Courville, and Pierre Vincent. 2013. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828.

Razvan C. Bunescu and Raymond J. Mooney. 2005. A shortest path dependency kernel for relation extraction. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 724–731.

Razvan Bunescu and Raymond Mooney. 2007. Learning to extract relations from the web using minimal supervision. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, volume 45, pages 576–583.

Yun-Nung Chen, Dilek Hakkani-Tur, and Gokan Tur. 2014. Deriving local relational surface forms from dependency-based entity embeddings for unsupervised spoken language understanding. In *Spoken Language Technology Workshop (SLT), 2014 IEEE*, pages 242–247.

Massimiliano Ciaramita and Yasemin Altun. 2006. Broad-coverage sense disambiguation and information extraction with a supersense sequence tagger. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 594–602.

Marie-Catherine De Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of the International Conference on Language Resources and Evaluation*, volume 6, pages 449–454.

Cıcero Nogueira dos Santos, Bing Xiang, and Bowen Zhou. 2015. Classifying relations by ranking with convolutional neural networks. In *Proceedings of 53rd Annual Meeting of the Association for Computational Linguistics*, pages 626–634.

Javid Ebrahimi and Dejing Dou. 2015. Chain based rnn for relation classification. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1244–1249.

Katrin Fundel, Robert Küffner, and Ralf Zimmer. 2007. Relexłrelation extraction using dependency parse trees. *Bioinformatics*, 23(3):365–371.

Zhou GuoDong, Su Jian, Zhang Jie, and Zhang Min. 2005. Exploring various knowledge in relation extraction. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 427–434.

Kazuma Hashimoto, Makoto Miwa, Yoshimasa Tsuruoka, and Takashi Chikayama. 2013. Simple customization of recursive neural networks for semantic relation classification. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1372–1376.

Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. 2009. Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In *Proceedings of the Workshop on Semantic Evaluations: Recent Achievements and Future Directions)*, pages 94–99.

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

Sepp Hochreiter. 1998. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116.

Nanda Kambhatla. 2004. Combining lexical, syntactic, and semantic features with maximum entropy models for extracting relations. In *Proceedings of the ACL 2004 on Interactive Poster and Demonstration Sessions*, page 22. Association for Computational Linguistics.

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

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013b. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 1003–1011. Association for Computational Linguistics.

Lili Mou, Hao Peng, Ge Li, Yan Xu, Lu Zhang, and Zhi Jin. 2015. Discriminative neural sentence modeling by tree-based convolution. *arXiv preprint arXiv:1504.01106*.

Barbara Plank and Alessandro Moschitti. 2013. Embedding semantic similarity in tree kernels for domain adaptation of relation extraction. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 1498–1507.

Richard Socher, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher D. Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 151–161.

Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1201–1211.

Richard Socher, Andrej Karpathy, Quoc V. Le, Christopher D. Manning, and Andrew Y. Ng. 2014. Grounded compositional semantics for finding and describing images with sentences. *Transactions of the Association for Computational Linguistics*, 2:207–218.

Chang Wang and James Fan. 2014. Medical relation extraction with manifold models. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 828–838.

Mengqiu Wang. 2008. A re-examination of dependency path kernels for relation extraction. In *Proceedings of the Third International Joint Conference on Natural Language Processing*, pages 841–846.

Fei Wu and Daniel S. Weld. 2010. Open information extraction using wikipedia. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 118–127.

Yan Xu, Ge Li, Lili Mou, and Yangyang Lu. 2014. Learning non-taxonomic relations on demand for ontology extension. *International Journal of Software Engineering and Knowledge Engineering*, 24(08):1159–1175.

Xuchen Yao and Benjamin Van Durme. 2014. Information extraction over structured data: Question answering with freebase. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 956–966.

Mo Yu, Matthew Gormley, and Mark Dredze. 2014. Factor-based compositional embedding models. In *NIPS Workshop on Learning Semantics*.

Wojciech Zaremba and Ilya Sutskever. 2014. Learning to execute. *arXiv preprint arXiv:1410.4615*.

Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. 2003. Kernel methods for relation extraction. *The Journal of Machine Learning Research*, 3:1083–1106.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation classification via convolutional deep neural network. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 2335–2344.

Xiaodan Zhu, Parinaz Sobhani, and Hongyu Guo. 2015. Long short-term memory over tree structures. In *Proceedings of The 32nd International Conference on Machine Learning*, pages 1604–1612.

A Computational Cognitive Model of Novel Word Generalization

Aida Nematzadeh, Erin Grant, and Suzanne Stevenson
Department of Computer Science
University of Toronto
{aida,eringrant,suzanne}@cs.toronto.edu

Abstract

A key challenge in vocabulary acquisition is learning which of the many possible meanings is appropriate for a word. The word generalization problem refers to how children associate a word such as *dog* with a meaning at the appropriate category level in a taxonomy of objects, such as Dalmatians, dogs, or animals. We present the first computational study of word generalization integrated within a word-learning model. The model simulates child and adult patterns of word generalization in a word-learning task. These patterns arise due to the interaction of type and token frequencies in the input data, an influence often observed in people's generalization of linguistic categories.

1 Introduction

Learning word meanings is a challenging early step in child language acquisition. Imagine a child hears the word *dax* for the first time while observing a white rabbit jumping around – *dax* might mean WHITE RABBIT, RABBIT, ANIMAL, CUTE, LOOK, etc. (Quine, 1960). How does the child learn the correct meaning of a word from a large pool of potential meanings? A possible explanation is that children infer a word's meaning by identifying the commonalities across the situations in which the word occurs (Pinker, 1989). One mechanism for achieving this is *cross-situational learning* (*e.g.*, Siskind, 1996; Frank et al., 2007; Fazly et al., 2010; Kachergis et al., 2012). Recent word learning experiments confirm that both adults and children infer the correct word-meaning mappings by keeping track of cross-situational statistics across individually ambiguous learning trials (Yu and Smith, 2007; Smith and Yu, 2008; Yurovsky et al., 2014).

Although cross-situational learning is a general mechanism for narrowing down the meaning of a word, it does not explain how children overcome an interesting challenge in word learning: determining the correct level of a hierarchical taxonomy that a word refers to. For example, children learn that the word *dog* refers to all kinds of dogs, and not to a specific breed, such as Dalmatians, or to a more general category, such as animals – even though some of these choices (*e.g.*, animals) are compatible with all the cross-situational evidence available for *dog* (because all dogs are also animals). We use the term "word generalization" to refer to this problem of associating a word with the meaning at an appropriate category level, given some sample of experiences with the word.

Previous research has argued that children use a specific bias or constraint – the *basic-level assumption* – to focus their word generalizations appropriately (Markman, 1991; Golinkoff et al., 1994). According to this bias, children prefer to associate a word to a set of objects that form a *basic-level* category, such as dogs or trucks, and that share a significant number of attributes. It is less preferred to associate a new word to much more specific *subordinate* categories, such as Dalmatians or bulldozers, or to more general *superordinate* ones, like animals or vehicles, whose members share fewer attributes (Rosch, 1973; Rosch et al., 1976). It remains an important open question of whether a word learner requires such a bias to acquire appropriate mappings.

Xu and Tenenbaum (2007) (X&T henceforth) studied the word generalization problem in a set of experiments in which children and adults were asked to determine which level of a taxonomy a novel word referred to. X&T further examined this behavioral data through computational modelling. They proposed a Bayesian model that, given a few exemplars of a novel word, matches human behaviour in how it maps the word to its

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1795–1804,
Lisbon, Portugal, 17-21 September 2015.

meanings in a taxonomic category.

The Bayesian model of X&T is important in providing insight into how people might reason about samples of data that exemplify categories. However, it relies on having complete, built-in knowledge about the taxonomic hierarchy, including both the detailed composition of categories and the values for between-object similarities, drawn from adult similarity judgments. Furthermore, the X&T model does not address the issue of word generalization in the broader context of word learning: While their model reasons over samples of data associated with a word label, it does not develop a meaning representation of the word over time, as a child must do. It is important to understand how word generalization occurs when embedded in the natural process of learning a word meaning and in the context of more limited category knowledge.

We address these issues by providing a unified account of word learning and word generalization within a computational model of cross-situational learning. Unlike the X&T model, our model is an incremental learner that gradually acquires the meaning of words, and uses these developing meanings in determining the appropriate extension of a word to elements of a taxonomy. Our model has general knowledge of category structure without having an elaborated taxonomy encoding known object similarities. Moreover, in the absence of any bias toward generalization to particular kinds of categories, the model exhibits the observed "basic-level bias" due to general mechanisms of productivity that have been proposed to apply to many aspects of linguistic knowledge (*e.g.*, Bybee, 1985; Croft and Cruse, 2004).[1]

In what follows, we first describe the human experiments of X&T, and then present our computational model and the experiments that simulate the X&T data.

2 Novel Word Generalization in People

X&T perform a set of empirical studies to investigate how children and adults generalize novel words learned from a few examples to the appropriate level of meaning in a taxonomy. In each training trial of an experiment, participants hear a novel word (such as *fep*) and observe one or more instances exemplifying the word (in the form of pictures for adults and toy objects for children). The conditions vary in that the make-up of the set of training instances is representative of different levels of a taxonomy (*e.g.*, all Dalmatians vs. various kinds of dogs vs. various kinds of animals). In the testing phase, participants are asked to select all objects that they think are *feps* from a set of test items. Both children and adults make various inferences about what a *fep* is depending on the levels of the taxonomy from which the training instances are drawn.

Specifically, X&T use a taxonomy with animals, vehicles, and vegetables, from which instances are drawn to produce the training conditions in Fig. 1(a). For example, in one training condition, participants are shown a Dalmatian, a poodle, and a beagle in three consecutive trials, hearing the word *fep* to refer to each object. After training, participants are asked to select all *feps* from the set of test objects, which includes items from all 3 superordinate categories. As illustrated in Fig. 1(b), each test object is assessed as one of the following types of match to the training data:

- a **subordinate match**: an object of the same subordinate category as a training object (*e.g.*, Dalmatians in Fig. 1)
- a **basic-level match**: an object of the same basic-level category as a training object (*e.g.*, a dog, but not the same breed as one in training [which would be a subordinate match])
- a **superordinate match**: an object of the same superordinate category as the training objects (*e.g.*, another kind of animal, but not one seen in training [which would be a subordinate or basic-level match])

X&T report the percentage of test objects of each type of match that are selected by participants within each training condition; see Fig. 2. (For example, the reported value for "super. match" would be 75% if participants on average chose 3 of the 4 superordinate matches in the test set.)

Consider first the data from adults. After seeing a single object (1-example condition – *e.g.*, a Dalmatian), adults show a strong basic-level bias – i.e., they tend to generalize the word *fep* to refer to *both* Dalmatians (subordinate matches) and to

[1] Computational cognitive models are often categorized with respect to Marr's levels of analysis, *i.e.*, their degree of abstraction (Marr, 1982). The model of X&T is at the computational level, providing a Bayesian framework for the problem of word generalization. In contrast, our model investigates more detailed mechanisms and thus lies between the algorithmic and computational levels of analysis.

Training condition:	Example Trials: Trial 1	Trial 2	Trial 3
1 example	Dalmatian	∅	∅
3 subord.	Dalmatian	Dalmatian	Dalmatian
3 basic	Dalmatian	poodle	beagle
3 super.	Dalmatian	penguin	sheep

(a) An example of the training conditions.

Example test object selected:	Type of test match:
Dalmatian	subordinate match
bulldog	basic match
cat	superordinate match

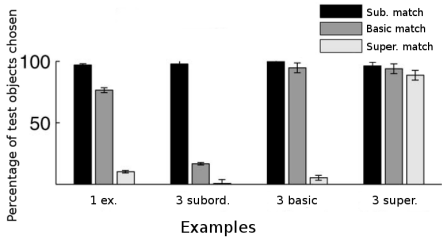
(b) An example of the test responses.

Figure 1: The training and testing conditions of X&T; see text in Section 2.

other dogs (basic-level matches), but not to other animals (superordinate matches). But with 3 instances of a Dalmatian (3-subordinate condition), this behaviour is attenuated – the number of basic-level matches is much lower. For the 3-basic-level and 3-superordinate conditions, the adults show generalization up to categories consistent with the evidence – i.e., at the basic and superordinate levels, respectively.

Interestingly, children also show a basic-level bias, but differ from adults in that it is less pronounced – e.g., they are less likely than adults to select basic-level matches (other dogs) having seen a single Dalmatian or having seen 3 Dalmatians. In the other conditions, children's behaviour is similar to adults, but shows somewhat less generalization to unseen types of objects (*e.g.*, other kinds of dogs/animals than those in training).

(a) **Adult data:**



(b) **Child data:**

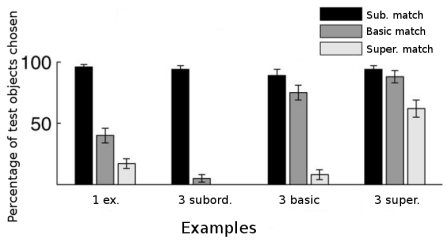


Figure 2: X&T data for (a) adults and (b) children. Each bar is the percentage of chosen test objects of a type of test match: *i.e.*, subord(inate), basic(-level), or super(ordinate).

3 The Word Learning Framework

Our computational model is based on the cross-situational word learner of Fazly et al. (2010) (henceforth, FAS), which accounts for a range of observed patterns in child and adult vocabulary acquisition. Here we give an overview of the FAS model; the next section explains extensions to handle the novel word generalization task.

A naturalistic language learning scenario consists of both linguistic data (what a child a hears) and non-linguistic data (what a child perceives). This input is modeled as a sequence of utterance–scene (U–S) pairs, where an utterance is a group of words and a scene is a set of semantic features representing the meaning of those words:

U: { *look*, *a*, *fep*, ... }
S: { PERCEPTION, LOOK, ..., DALMATIAN, DOG, ... }

Given such input, for each word w, the model of FAS learns a probability distribution over all semantic features, $P_t(.|w)$, which represents the word's meaning at time t. Initially, at time $t = 0$, $P_0(.|w)$ is a uniform distribution. The word meanings are incrementally learned using an algorithm that implements cross-situational learning: for each pair of a word w and a semantic feature f, the model learns $P_t(f|w)$ from co-occurrences of w and f across all the utterance–scene pairs seen up to time t, as follows.

Given an utterance–scene pair U–S at time t, and drawing on its learned knowledge of word meanings up to time $t-1$, the model of FAS calculates an *alignment* probability for each w_j–f_i pair. This probability reflects how strongly the feature f_i is associated with w_j compared to its association with other words in U:

$$P_t(a_{ij}|U, f_i) = \frac{P_{t-1}(f_i|w_j)}{\sum_{w' \in U} P_{t-1}(f_i|w')} \qquad (1)$$

where a_{ij} indicates the mapping between the word w_j and the semantic feature f_i.

These probabilities are incrementally accumulated for each w_j–f_i pair, capturing the overall strength of association of w_j and f_i at time t:

$$\begin{aligned} \text{assoc}_t(f_i, w_j) = \text{assoc}_{t-1}(f_i, w_j) \\ + P_t(a_{ij}|U, f_i) \end{aligned} \qquad (2)$$

The (normalized) association scores then serve as the basis for the incremental adjustment of the meaning probabilities of all features f_i for each word w_j seen in the input at time t:

$$P_t(f_i|w_j) = \frac{\text{assoc}_t(f_i, w_j) + \gamma}{\sum_{f_m \in \mathcal{M}} \text{assoc}_t(f_m, w_j) + k\,\gamma} \quad (3)$$

Here $\mathcal{M}$ is the group of all features that the model has observed, k is the expected number of such features, and γ is a small smoothing parameter, which determines the prior probability of observing a new feature.

Smoothing entails that features previously unseen with a word (all f_i such that $\text{assoc}_t(f_i, w_j) = 0$) have a small but non-zero probability. That is, when f_i is unseen with w_j, Eqn. (3) reduces to:

$$P^u_t(f_i|w_j) = \frac{\gamma}{\sum_{f_m \in \mathcal{M}} \text{assoc}_t(f_m, w_j) + k\,\gamma} \quad (4)$$

This unseen probability, P^u_t, reflects the learner's "openness" to the word being associated with new features (Nematzadeh et al., 2011): a higher or lower $P^u_t(f_i|w_j)$ will affect how strongly a previously unseen f_i can be associated with w_j in the alignment process (Eqn. (1)). We return to this property of the model below, as it relates to the behaviour of our model in making generalizations.

4 Extensions to the Model

We assume that the representation of meaning can be abstracted to features that correspond to different levels of categorization. For example, a Dalmatian in an input scene is represented as {DALMATIAN, DOG, ANIMAL} and a Bulldog as {BULLDOG, DOG, ANIMAL}, where we use FEATURENAME to refer to all the features that are specific to that level of object category. (Note that we could replace each of these features with the appropriate "true" set of features, but use the more compact representation for simplicity.) To acquire the meaning of the word *Dalmatian*, the model must learn a probability distribution in which $P(f|$*Dalmatian*$)$ is relatively high for the features DALMATIAN, DOG, and ANIMAL, and low for features such as BULLDOG, CAT, and VEGETABLE.

Introducing Feature Groups. In the FAS model, all the features for a word are dependent: increasing the probability of any feature results in decreasing the probability of others. However, this interaction is not always desirable, as many features regularly co-occur in the world. This is especially an issue for features from a category hierarchy, where features of a subordinate category should not compete with features of the parent. That is, while a higher probability of DALMATIAN features (*e.g.*, black spotted coat) may lessen the likelihood of BULLDOG features (*e.g.*, wrinkles), it should *not* decrease the probability of DOG features (*e.g.*, having 4 legs).

To address this, we extend the model by using *feature groups* that collect together sets of features that sensibly compete. Each feature group is comprised of all features at the same level of specificity in the category hierarchy, which are therefore mutually exclusive, such as DOG, CAT, and BIRD (*i.e.*, different kinds of animals). Instead of learning a single probability distribution over all features as the meaning of a word, the extended model learns *a set of* probability distributions for a word, one for each feature group (*i.e.*, one per level of the hierarchy). Features within a group thereby compete for the probability mass associated with a word, but those from across groups (*e.g.*, DALMATIAN and DOG) can freely co-occur without competing.

The model does not know a priori all the features in a group, but when presented with a newly observed feature, it can identify the appropriate group for it. In taking this approach, we assume the learner can distinguish the level of specificity of features perceived in the scene. For example, in the scene representations {DALMATIAN, DOG, ANIMAL} and {SIAMESE, CAT, ANIMAL}, the learner can recognize that DOG and CAT are at the same level of the hierarchy (kinds of animals) and that DALMATIAN and SIAMESE are at the same, *more specific* level in the hierarchy (finer-grained breeds of animals). Our assumption is that children (at this stage in their development) can identify a degree of similarity among concepts that enables them to recognize that Dalmatians and Siamese are distinguished by similar properties (such as fur color), which differ from more distinguishing properties at higher taxonomic levels (such as number of legs). The model has no other prior knowledge of the category structure. For example, it is not built into the model that DALMATIAN is a type of DOG, only that it is more specific than DOG; any association between them would be learned from their pattern of co-occurrence with a word over time.

Note that, in contrast to the model of X&T, our model does not start with a full taxonomy (it does not know, for example, that Dalmatians and poodles are hyponyms of dogs) and it does not have built-in knowledge of similarities among concepts. Still, it encodes some taxonomic knowledge in the feature groups, and an important future direction will be to show that this knowledge is learnable from the input.

Calculating Feature Group Probabilities. To appropriately split the probability mass within a feature group $\mathcal{G}$ (but not across feature groups), we use a new formulation of Eqn. (3) to update the meaning probabilities for $f_i \in \mathcal{G}$ as follows:

$$P_t(f_i|w_j) = \frac{\text{assoc}_t(f_i, w_j) + \gamma_{\mathcal{G}}}{\sum_{f_m \in \mathcal{G}} \text{assoc}_t(f_m, w_j) + k_{\mathcal{G}}\gamma_{\mathcal{G}}} \quad (5)$$

where $k_{\mathcal{G}}$ is the expected number of features in $\mathcal{G}$, and the smoothing factor $\gamma_{\mathcal{G}}$ reflects the prior belief in observing a feature f in $\mathcal{G}$.[2]

With this new formulation, the probability of a feature f_i previously unseen with word w_j now reduces to (cf. Eqn. (4)):

$$P^u_t(f_i|w_j) = \frac{\gamma_{\mathcal{G}}}{\sum_{f_m \in \mathcal{G}} \text{assoc}_t(f_m, w_j) + k_{\mathcal{G}}\gamma_{\mathcal{G}}} \quad (6)$$

for $f_i \in \mathcal{G}$. Note that the smoothing factor $\gamma_{\mathcal{G}}$ depends on $\mathcal{G}$, and thus the openness of the word to be associated with new (previously unseen) features can vary depending on the feature group.

This unseen probability is very important to the model's generalization behaviour. Generalization involves the model accepting that a learned word can refer to objects not seen with it before: e.g., in the experiments here, we would expect that the learned meaning for *fep* after seeing three animals such as a dog, a penguin, and a sheep could also accommodate the meaning of a different animal such as a cat. This ability of the model to associate new meaning features with a word depends precisely on the unseen probability formulation: the higher the unseen probability for a feature and a word, the more the feature will be acceptable as a generalization of the word.

Type-Token Effects on Generalization. The unseen probability is sensitive to how many instances of features from a group have already been seen with a word w_j: As the model observes more instances (tokens) of features from $\mathcal{G}$ with w_j, the corresponding $assoc_t$ score(s) increase, thereby increasing the denominator in Eqn. (6) and decreasing P^u_t. Thus the tendency to generalize w_j to more features in $\mathcal{G}$ – i.e., to accept additional features as part of the meaning of w_j – will decrease as the model has more evidence of (observed) features in that group occurring with w_j.

Generalization of a category to include new kinds of items is typically a function of both token *and* type frequency (*e.g.*, Bybee, 1985; Croft and Cruse, 2004): a category with more diverse types is more easily extended to new cases. While the evolving association scores capture the effect of observing more feature *tokens*, our model as given does not distinguish the number of different *types* of features seen within a group (*e.g.*, two DOGs vs. one DOG and one CAT).

We address this issue by having $\gamma_{\mathcal{G}}$ depend on the number of observed types of features in the group:

$$\gamma^t_{\mathcal{G}} = \gamma^0_{\mathcal{G}} \times \text{type}(\mathcal{G}, t)^2 \quad (7)$$

where $\text{type}(\mathcal{G}, t)$ is the number of different kinds of features seen in that group (e.g., DOG and CAT are two different feature types from the same group) up through time t. In this way, the P^u_t of a feature that occurs in a group with more observed feature types is higher than the P^u_t in a group with fewer observed types.

Thus both the type frequency of features in $\mathcal{G}$ and their token frequency of co-occurrence with word w_j will influence – the first positively and the second negatively – how readily w_j can refer to objects with previously unseen features from $\mathcal{G}$.

5 Experimental Set-up

We model X&T's behavioural experiments with our computational word learner as extended above.[3] Following X&T, we use a three-tiered category hierarchy, and the four training conditions and assessment of three types of test matches as described in Figure 1.

Training the model. In each condition, the model processes a sequence of 3 utterance-scene pairs, and updates $P_t(f_i|w_j)$ after each pair using Eqns. (5) and (6). The utterance-scene pair in each trial consists of the novel word coupled with the scene representation of a training object from the

[2] Each feature group forms a Categorical distribution with $k_{\mathcal{G}}$ categories ($\text{Cat}(\theta_1, ..., \theta_{k_{\mathcal{G}}})$), where the θ_i are drawn from a prior Dirichlet distribution $\text{Dir}(\gamma_1, ..., \gamma_{k_{\mathcal{G}}})$ at time $t = 0$, and the θ_i are updated at time t to be the expected value of the posterior Dirichlet distribution, given in Eqn. (5) or Eqn. (6).

[3] Link to our code/data: `github.com/eringrant/word_learning/tree/hypothesis-space`.

category hierarchy. The object's scene representation is given as a set of four features, each taken from one of four feature groups: one feature corresponding to each of the subordinate, basic, and superordinate levels of the hierarchy, and a unique "instance" feature, as shown in Table 1. (The "instance" feature is added to simulate the variations in the different objects of the same subordinate category in the X&T experiments.)

1	**U:**	{ *fep* }
	S:	{ INSTANCE$_1$, DALMATIAN, DOG, ANIMAL }
2	**U:**	{ *fep* }
	S:	{ INSTANCE$_2$, TABBY, CAT, ANIMAL }
3	**U:**	{ *fep* }
	S:	{ INSTANCE$_3$, POLAR BEAR, BEAR, ANIMAL }

Table 1: An example of a sequence of utterance-scene pair trials in the 3-super. condition.

Testing the model. After training on a novel word, in order to assess its level of generalization within the category hierarchy, we compare the model's learned meaning of the word to test objects that constitute various types of matches to the training conditions: *i.e.*, subordinate matches, basic-level matches, and superordinate matches. Table 2 gives an example of each type of match:

subord.:	{ INSTANCE$_4$, DALMATIAN, DOG, ANIMAL }
basic:	{ INSTANCE$_5$, POODLE, DOG, ANIMAL }
super.:	{ INSTANCE$_6$, TOUCAN, BIRD, ANIMAL }

Table 2: An example of each level match from the test objects, given the training condition in Table 1.

To assess whether the model generalizes the learned meaning of a word w to the various types of test matches, we first consider the probability of a test object Y at time t given the learned meaning of w:

$$P_t(Y|w) = \prod_{y_i \in Y} P_t(y_i|w) \qquad (8)$$

where y_i are the features in Y, and $P_t(y_i|w)$ is calculated using Eqn. (5) for features y_i observed with w during training, and using Eqn. (6) for y_i not observed with w. (Recall that Eqn. (5) reduces to Eqn. (6) when a feature has not been seen with the word.) From $P_t(Y|w)$, we subtract the predictive probability of the test object before the model has observed any data, $P_0(Y|w)$, which gives us its increase in preference attributable to the word learning trials.[4]

Calculating $P_t(Y|w) - P_0(Y|w)$ is informative about one test object, but we need to measure generalization of the learned word to all the objects of a certain type of match – *i.e.*, subordinate, basic-level, or superordinate. We formulate the probability of generalization to a type of test match as the relative average increase in preference for test items of that type of match, using the Shepard-Luce choice rule (Shepard, 1958; Luce, 1959):

$$P_{gen}(m|w) = \frac{\text{avg}_{Y \in m}\,[P_t(Y|w)\text{-}P_0(Y|w)]}{\sum_{m'} \text{avg}_{Y' \in m'}\,[P_t(Y'|w)\text{-}P_0(Y'|w)]}$$

where m is the set of test objects at a certain level of match, and m' ranges over subordinate matches, basic-level matches, and superordinate matches.

Using $P_{gen}(m|w)$ to communicate our models results has the advantage of using the learned word meanings in a very direct way to assess the preference for the various types of test matches in the X&T experiments. However, the disadvantage is that this measure is not directly comparable to the reported figures from the human data, which are the percentage of test objects selected of a particular type of match. Hence, in presenting our results below, we focus on the general patterns of preferences indicated by the different measures.

Parameters. To model children, whom we assume to have no bias towards generalization to specific category levels, we equate all parameters $k_{\mathcal{G}}$ and $\gamma_{\mathcal{G}}$ across all feature groups, reflecting that all category levels are treated equivalently. Here we use values of $k_{\mathcal{G}} = 100$ and initial values of $\gamma_{\mathcal{G}} = 0.5$ for all $\mathcal{G}$ as the "child" parameter settings.[5] In contrast, we assume that adults, through word learning experience, have accumulated biases that reflect observed differences in feature groups. More specifically, we assume that the probability of observing a new feature for a group $\mathcal{G}$ depends on the degree of specificity of that group: That is, over time, it is less likely to observe a completely new kind of animal, e.g., than a new breed of dog. We simulate these biases by us-

[4] $P_0(Y|w) = \prod_{\mathcal{G}} \frac{1}{k_{\mathcal{G}}}$ is the prior probability of any object instance, given parameters drawn from the Dirichlet prior, because Eqn. (6) yields the value $\frac{1}{k_{\mathcal{G}}}$ when all assoc_t scores are 0 – *i.e.*, no features from $\mathcal{G}$ have been observed with the word.

[5] To determine the parameters for the "child" learner, we examined a number of settings with equal parameter values for all the feature groups, and observed similar results in these settings. (We did not perform an exhaustive search over the parameter space.)

ing various values for the parameter $\gamma_{\mathcal{G}}$, which determines the prior probability of a word being observed with new (previously unseen) features in $\mathcal{G}$ (cf. Eqn. (6)). We assume that the expected number of features ($k_{\mathcal{G}}$) is the same across groups. We perform a non-exhaustive search on the parameter space of $\gamma_{\mathcal{G}}$ to select a set of values that yield the patterns of X&T's adult experiments. The "adult" parameter values are given in Table 3:[6]

$\gamma_{\text{inst}} = 1.2$	$\gamma_{\text{subord}} = 1.0$	$\gamma_{\text{basic}} = 0.5$	$\gamma_{\text{super}} = 0.2$
$k_{\text{inst}} = 100$	$k_{\text{subord}} = 100$	$k_{\text{basic}} = 100$	$k_{\text{super}} = 100$

Table 3: "Adult" parameter settings.

6 Experimental Results

We present results of the model using both child settings (Figure 3b) and adult settings (Figure 3a). Recall that these values do not correspond to the percentages reported in the human data; to evaluate the patterns of generalization, we look at the relative preference for the various types of test match. Note also that since the generalization probabilities sum to 1.0 within each of the 4 training conditions, we can only compare the *pattern* of generalization across conditions (and not the actual value of the probabilities).

We discuss each of the child and adult sets of results in detail below.

6.1 The Child Learner

Recall that in the simulations of a child, we use equal values across all feature groups for the $k_{\mathcal{G}}$ and initial $\gamma_{\mathcal{G}}$ parameter settings, to reflect that the learner has no bias towards generalization to specific category levels.

Looking at the results in Figure 3b, we can see that the child learner generally replicates the patterns of results observed in X&T's experiment on children (cf. Figure 2b). Given multiple training items (the 3-subord., 3-basic, and 3-super. conditions), the model, like children, generalizes to the lowest level category in the hierarchy that is consistent with the training items, roughly equally preferring items from that category or lower, with slight preference for the lower categories. In contrast, after seeing a single training example (the

[6]For a certain range of such parameter settings – i.e., with gradually decreasing $\gamma_{\mathcal{G}}$, which determines the prior probability of a word being observed with new (previously unseen) features in $\mathcal{G}$ (cf. Eqn. (6)). for feature groups at successively higher levels in the hierarchy — the model produces similar results to the presented adult learner.

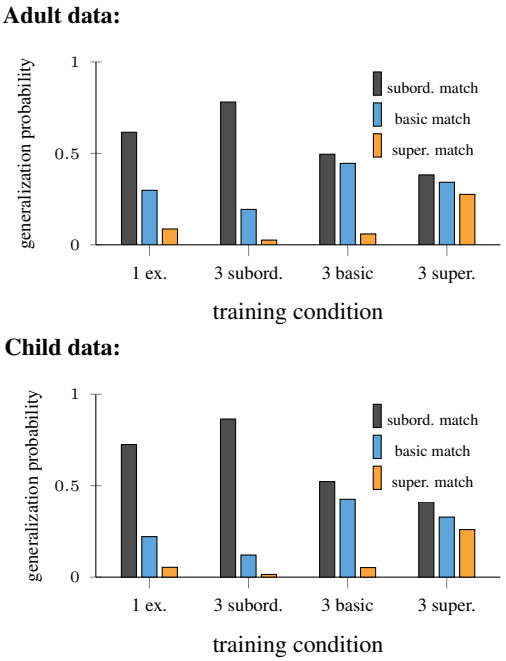


Figure 3: Our model data for (a) adults and (b) children. Each bar is the probability of a type of test match: *i.e.*, subord(inate), basic(-level), or super(ordinate).

1-ex. condition), the model shows some tendency to generalize to the basic-level, demonstrating a small but notable basic-level bias — e.g., the tendency to consider the word as referring to any dogs (but less so to other animals) after seeing just a single example of a particular kind of dog. As in children, the difference in the model between the preference for subordinate vs. basic-level matches is much smaller when trained on 1 instance as opposed to 3 subordinates. (In Figure 3b, compare the difference between the 1st bar [subord. match] and 2nd bar [basic match] of the 1-ex. training condition to that of the 3-subord. training condition.)

Interestingly, our child learner exhibits the observed basic-level bias in the absence of any difference in the model in how it treats different category levels. The observed pattern arises from a type/token frequency interaction of the kind often noted to influence generalization of linguistic categories (*e.g.*, Bybee, 1985; Croft and Cruse, 2004): here, the interaction between the token frequency of word–feature pairs in the input and the type frequency of different features within a group of dependent features. For example, having seen 3 types of animals ("3 super." condition), the model can readily accommodate that *fep* refers to another kind of animal, in contrast to the "3 basic" condition, where it has seen the same number of tokens but only a single feature type from the feature

group at that level (3 dogs). We can also clearly see the inverse impact of token frequencies on generalization: the more examples of a single subordinate type are seen, the less the model accepts that *fep* refers to a different kind of subordinate (the "3-subord." vs. "1-ex." conditions). That is, with only 1 token of DALMATIAN, the model can generalize to other types of dogs more readily than when it has seen 3 tokens of DALMATIAN.

In general, interactions between the type and token frequencies of the different feature groups interact to yield the observed patterns in the model. These results indicate that properties of the input data coupled with the model's handling of feature groups can account for children's word generalization behaviour, without the need for an explicit basic-level bias.

6.2 The Adult Learner

Adult participants in X&T exhibited a stronger tendency than children to generalize to the basic-level category, especially after seeing a single exemplar. We explore whether the model can simulate an adult learner as well. As discussed in Section 5, by varying the parameters $\gamma_{\mathcal{G}}$, we can incorporate biases towards different category levels that we assume an adult has learned. More specifically, we set $\gamma_{\mathcal{G}}$ to successively larger values for more specific feature groups $\mathcal{G}$, to ensure successively greater generalization in lower levels of the hierarchy (see Table 3). As shown in Figure 3a, our model (using such settings of the parameters) replicates the patterns of X&T's adult experiments (cf. Figure 2a), including a stronger basic-level bias than that shown by children. That is, in the 1-ex. and 3-subord. conditions, the difference between the 1st bar [subord. match] and 2nd bar [basic match] is smaller for the adult settings of the model (Figure 3a) than for the child settings (Figure 3b), mimicking the stronger basic-level bias found in adults.

6.3 Variations in Basic-level Generalization

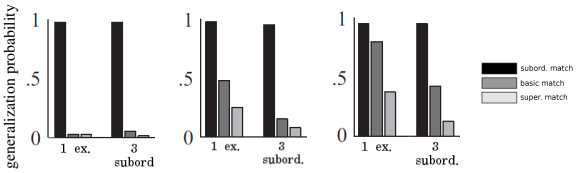
Research shows that people's degree of basic-level generalization depends on the overall category of the objects. Specifically, Abbott et al. (2012) perform the same set of experiments as X&T on adults, exploring three additional superordinate categories (clothing, containers, and seats). Their results are shown in Figure 4; for space reasons, we focus here on the training conditions with 1-example or 3-subordinates, which are the locus of the basic-level effect. The results show that people exhibit no basic-level generalization for containers, moderate generalization for clothing, and strong generalization for seats (compare Figures 4a, 4b and 4c).

Interestingly, the computational experiments of Abbott et al. (2012) also reveal that the Bayesian model of X&T mimics varying levels of basic-level generalization in the 1-example cases, but does not capture the differences that people exhibit across the categories in the 3-subordinate condition (compare "3 subord." in Figures 4 and 5): unlike people, here the X&T model does not exhibit basic-level generalization for any of the categories.

Abbott et al. (2012) note that a domain like containers may not follow a "natural taxonomy" in having a clear basic-level category. This suggestion is compatible with our view that a basic-level bias arises in response to the particular pattern of co-occurrence of features across the category hierarchy. We looked more closely at the training stimuli of their experiment, and observe that the examples of the category "containers" (with the least basic-level generalization) vary greatly, while those of "clothing" and "seats" are less differentiated. Examples from "containers" include a cigar box, trash can, and mailbox, whereas "seats" are restricted to different types of chair (such as a dining chair and an armchair; see Table 1 in Abbott et al. (2012)).

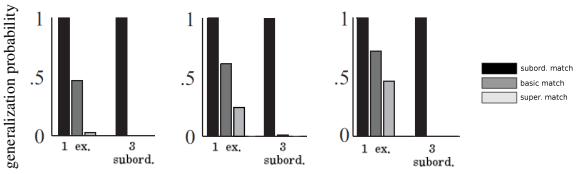
Based on this observation, we hypothesize that people generalize less to a basic-level category when their mental representations for that category's instances have more distinguishing features. Specifically, we assume that people differentiate the given instances of the category "containers" more than those for "clothing" and "seats". We model this difference in the granularity of representations by varying the number of feature groups used in representing an object. Recall that in our earlier experiments, each object was represented as a set of features drawn from 4 different feature groups. We take this representation as the least fine-grained representation and use it for the category "seats". We assume that the objects from the categories "clothing" and "containers" (that exhibit less basic-level generalization) are represented with more feature groups (8 and 12, respectively).

Figure 6 shows the results of running our model



(a) Containers (b) Clothing (c) Seats

Figure 4: Abbott et al. (2012) subject response data.



(a) Containers (b) Clothing (c) Seats

Figure 5: Abbott et al. (2012) model data.

on these three categories using the "adult" parameter settings. As expected, the generalization to the basic-level category is high for the least distinguished category "seats", moderate for the category "clothing", and low for the most distinguished category "containers".[7]

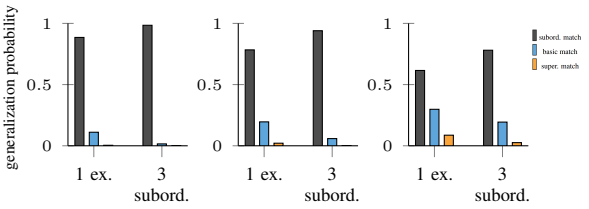
Our results suggest that the observed variation across categories in basic-level generalization could arise from differences in the granularity of representations of categories. This is particularly interesting since the model of X&T, despite encoding an elaborated taxonomy, does not capture the observed behaviour across all training conditions.

7 Conclusions

A key challenge faced by children in vocabulary acquisition is learning which of many possible meanings is appropriate for a word, based largely on ambiguous situational evidence. One aspect of this is what we term the "word generalization" problem, which refers to how children associate a word such as *dog* with a meaning at the appropriate category level in a taxonomy of objects, such as Dalmatians, dogs, or animals.

We present extensions to a cross-situational learner that enable the first computational study of word generalization that is integrated within a word learning model. The model mimics child behavior found by Xu and Tenenbaum (2007): it shows a "basic-level" bias – a preference for word meanings that refer to basic-level objects (like dogs), in contrast to higher-level (animals) or lower-level (Dalmatians) categories – and does so

[7]Similar results obtain using "child" parameter settings, but (as expected) the basic-level generalization is lower.



(a) Containers (b) Clothing (c) Seats

Figure 6: Data from our model.

under parameter settings that treat all levels of category the same in the model (*i.e.*, with no built-in basic-level bias). Other (unequal) parameter settings, which could reflect learned knowledge leading to differential treatment of categories, yield behavior that mimics that of adults, who show a stronger basic-level bias. Moreover, similarly to people (Abbott et al., 2012), our model exhibits variations in generalization to the basic level for different types of objects, a behavior that the model of Xu and Tenenbaum (2007) does not fully replicate.

Overall, the results of our model arise from the interaction of type and token frequencies of features in the input data, which impact the model's evolving word representations. This mechanism in the model captures the type-token influence often observed to underlie people's generalization of linguistic categories – *i.e.*, their linguistic productivity (*e.g.*, Bybee, 1985; Croft and Cruse, 2004).

One shortcoming of the current model is its built-in ability to "detect" in the input that DOG and CAT features are more specific than ANIMAL features. The next step is to consider how the model might learn these relationships from its evolving knowledge of co-occurring features.

Finally, a similar problem to that of word generalization in humans arises in computational linguistics: how to appropriately generalize a set of concepts to an overarching concept that subsumes the set. For example, this problem underlies one way to determine the selectional preferences of a verb: extract the set of nouns that occur as objects of the verb, map them to the concept nodes in a hierarchy such as WordNet, and then determine the best overarching WordNet category for capturing the salient properties of the object nouns overall (*e.g.*, Li and Abe, 1998; Clark and Weir, 2001). An interesting future direction is to explore how an extension of our work can be applied to such problems in computational linguistics.

8 Acknowledgements

We would like to thank Jackie Chi Kit Cheung and the anonymous reviewers for their valuable feedback. We gratefully acknowledge the support of NSERC of Canada, and of an Ontario Graduate Scholarship to the first author.

References

Joshua T. Abbott, Joseph L. Austerweil, and Thomas L. Griffiths. 2012. Constructing a hypothesis space from the web for large-scale bayesian word learning. *Proceedings of the 34th Annual Conference of the Cognitive Science Society*.

Joan L. Bybee. 1985. *Morphology: A study of the relation between meaning and form*. Benjamins, Philadelphia.

Stephen Clark and David Weir. 2001. Class-based probability estimation using a semantic hierarchy. In *Proceedings of the second meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies*, pages 1–8. Association for Computational Linguistics.

William Croft and Alan Cruse. 2004. *Cognitive linguistics*. Cambridge University Press.

Afsaneh Fazly, Afra Alishahi, and Suzanne Stevenson. 2010. A probabilistic computational model of cross-situational word learning. *Cognitive Science*, 34(6):1017–1063.

Michael C. Frank, Noah D. Goodman, and Joshua B. Tenenbaum. 2007. A Bayesian framework for cross-situational word-learning. In *NIPS'07*, volume 20.

Roberta M. Golinkoff, Carolyn B. Mervis, and Kathryn Hirsh-Pasek. 1994. Early object labels: The case for a developmental lexical principles framework. *Journal of child language*, 21(01):125–155.

George Kachergis, Chen Yu, and Richard M. Shiffrin. 2012. An associative model of adaptive inference for learning word–referent mappings. *Psychonomic Bulletin & Review*, pages 1–8.

Hang Li and Naoki Abe. 1998. Word clustering and disambiguation based on co-occurrence data. In *Proceedings of the 17th International Conference on Computational Linguistics - Volume 2*, COLING '98, pages 749–755. Association for Computational Linguistics.

Robert D. Luce. 1959. *Individual choice behaviour*. Wiley, NY.

Ellen M. Markman. 1991. *Categorization and naming in children: Problems of induction*. Mit Press.

David Marr. 1982. Vision: A computational investigation into the human representation and processing of visual information.

Aida Nematzadeh, Afsaneh Fazly, and Suzanne Stevenson. 2011. A computational study of late talking in word-meaning acquisition. In *Proceedings of the 33th Annual Conference of the Cognitive Science Society*, pages 705–710.

Steven Pinker. 1989. *Learnability and Cognition: The acquisition of Argument Structure*. Cambridge, Mass.: MIT Press.

Willard Van Orman Quine. 1960. *Word and Object*. MIT Press.

Eleanor Rosch, Carolyn B. Mervis, Wayne D. Gray, David M, and Penny Boyes-Braem. 1976. Basic objects in natural categories. *Cognitive Psychology*.

Eleanor Rosch, 1973. *On the Internal Structure of Perceptual and Semantic Categories*, pages 111–144. Academic Press, New York, NY.

Roger N. Shepard. 1958. Stimulus and response generalization: Tests of a model relating generalization to distance in psychological space. *Journal of Experimental Psychology*, 55(6):509.

Jeffery M. Siskind. 1996. A computational study of cross-situational techniques for learning word-to-meaning mappings. *Cognition*, 61:39–91.

Linda B. Smith and Chen Yu. 2008. Infants rapidly learn word-referent mappings via cross-situational statistics. *Cognition*, 106(3):1558–1568.

Fei Xu and Joshua B. Tenenbaum. 2007. Word learning as Bayesian inference. *Psychological Review*, 114(2):245–272.

Chen Yu and Linda B. Smith. 2007. Rapid word learning under uncertainty via cross-situational statistics. *Psychological Science*, 18(5):414–420.

Daniel Yurovsky, Damian C. Fricker, Chen Yu, and Linda B. Smith. 2014. The role of partial knowledge in statistical word learning. *Psychonomic bulletin & review*, 21(1):1–22.

Personality Profiling of Fictional Characters using Sense-Level Links between Lexical Resources

Lucie Flekova[†] **and Iryna Gurevych**[†‡]
† Ubiquitous Knowledge Processing Lab (UKP-TUDA)
Department of Computer Science, Technische Universität Darmstadt
‡ Ubiquitous Knowledge Processing Lab (UKP-DIPF)
German Institute for Educational Research
www.ukp.tu-darmstadt.de

"Always be yourself, unless you can be Batman. Then always be Batman."
– Bill Murray

Abstract

This study focuses on personality prediction of protagonists in novels based on the Five-Factor Model of personality. We present and publish a novel collaboratively built dataset of fictional character personality and design our task as a text classification problem. We incorporate a range of semantic features, including WordNet and VerbNet sense-level information and word vector representations. We evaluate three machine learning models based on the speech, actions and predicatives of the main characters, and show that especially the lexical-semantic features significantly outperform the baselines. The most predictive features correspond to reported findings in personality psychology.

1 Introduction

Recent progress in NLP has given rise to the field of personality profiling - automated classification of personality traits based on written, verbal and multimodal behavior of an individual. This research builds upon findings from classical personality psychology and has applications in a wide range of areas from medicine (suicide prevention) across security (forensics, paedophile detection, cyberbullying) to marketing and sales (recommendation systems, target group profiles). The gold standard labels for an objective evaluation of personality are mostly obtained by means of personality tests of the Five Factor Model (FFM) (McCrae and Costa, 1987; Goldberg, 1990), which is well-known and widely accepted in psychology and other research fields. The FFM defines personality along five bipolar scales: Extraversion (sociable vs. reserved), Emotional stability (secure vs. neurotic), Agreeableness (friendly vs. unsympathic), Conscientiousness (organized vs. careless) and Openness to experience (insightful vs. unimaginative). Psychologists have shown that these five personality traits are stable across individual lifespan, demographical and cultural differences (John and Srivastava, 1999) and affect many life aspects. (Terracciano et al., 2008; Rentfrow et al., 2011).

It has been shown that the personality traits of readers impact their literature preferences (Tirre and Dixit, 1995; Mar et al., 2009). Psychology researchers also found that perceived similarity is predictive of interpersonal attraction (Montoya et al., 2008; Byrne, 1961; Chartrand and Bargh, 1999). More explicitly, recent research (Kaufman and Libby, 2012) shows that readers of a narrative develop more favorable attitudes and less stereotype application towards a character, if his difference (e.g. racial) is revealed only later in the story. We therefore hypothesize that readers might have a preference for reading novels depicting fictional characters that are similar to themselves. Finding a direct link between reader's and protagonist's personality traits would advance the development of content-based recommendation systems. As a first step to explore this hypothesis further, it needs to be determined if we are able to construct a personality profile of a fictional character in a similar way as it is done for humans, and which aspects of personality profiling can be exploited to automatize such procedure.

In this paper, we open this research topic by presenting a novel collaboratively built dataset of fictional character personality in Section 3, which we make available on our website.[1] Framing the personality prediction as a text classification task, we incorporate features of both lexical-

[1] https://www.ukp.tu-darmstadt.de/data/personality-profiling/

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1805–1816,
Lisbon, Portugal, 17-21 September 2015.

resource-based and vector space semantics, including WordNet and VerbNet sense-level information and vectorial word representations. We evaluate three machine learning models based on the speech (Section 4), actions (Section 5) and predicatives (Section 6) of the protagonists, and show that especially on the direct speech and action data the lexical-semantic features significantly outperform the baselines. Qualitative analysis reveals that the most predictive features correspond to reported findings in psychology and NLP.

2 Related work

Research in the the area of content-based recommendation systems have shown that incorporating semantic information is valuable for the user and leads to measurable improvements (Passant, 2010; Di Noia et al., 2012; Heitmann and Hayes, 2010). De Clercq et al. (2014) incorporated semantic frames from FrameNet into the recommendation system for books. They represent the plot of each book with a sequence of ca. 200 semantic frames and has shown that the frame information (such as Killing - Revenge - Death) outperforms the bag-of-words approach.Recent NLP experiments begin to reveal the importance of entity-centric models in a variety of tasks. Chambers (2013) show improvement in event schema induction by learning entity-centric rules (e.g., a victim is likely to be a person). Bamman et al. (2014) and Smith et al. (2013) present latent variable models for unsupervised learning of latent character types in movie plot summaries and in English novels, taking authorial style into account. However, even the state-of-the-art NLP work rather describes personas of fictional characters by their role in the story - e.g., action hero, valley girl, best friend, villain etc. - or by their relations to other characters, such as mother or daughter (Elson et al., 2010; Kokkinakis and Malm, 2011), rather than by their inner preferences and motivations. It is important to note here that determining a personality of a character is a very different task from determining its role in the story. Psychological understanding of personality, in contrast to role attribution requires a certain detached objectivity - even outright villains may have traits considered desirable in real life. For example, the devil has in many tales a very high aspiration level, appearing highly conscientious and agreeable. We hypothesize that these deeper personality aspects are those which drive reader's affiliation to the character, thus deserve to be examined closer.

Also literary scholars formulate ad hoc personality descriptions for their experiments, for example to test hypotheses from evolutionary psychology (Johnson et al., 2011) or examine fictional portrayals of physicists (Dotson, 2009). These descriptions are usually adjusted to the experiment focus (e.g. emotions, relationships, ambitions). As McCrae et al. () point out, *a standard set of personality traits, that encompass the full range of characteristics found in all characters in literature* (p.77), is needed for a better comparison.

Hence we base our present study primarily on the previous NLP research on personality prediction of human individuals. Correlations between lexical and stylistic aspects of text and the five FFM personality traits of the author have been found in numerous experiments, with extraversion receiving the most attention (Pennebaker and King, 1999; Dewaele and Furnham, 1999; Gill and Oberlander, 2002; Mehl et al., 2006; Aran and Gatica-Perez, 2013; Lepri et al., 2010). The LIWC lexicon (Pennebaker et al., 2001) established its position as a powerful mean of such analysis.

The first machine learning experiments in this area were conducted by Argamon et al. (2005), Oberlander and Nowson (2006) and Mairesse et al. (2007). Researchers predicted the five personality traits of the authors of stream-of-conscientiousness essays, blog posts and recorded conversation snippets. Given balanced data sets, Mairesse et al. (2007) report binary classification accuracy of 50-56% on extraversion in text and 47-57% in speech, using word ngrams, LIWC, MRC psycholinguistic database (Coltheart, 1981) and prosodic features. Additional improvement is reported when the extraversion was labeled by external judges rather than by self-testing. Extended studies on larger datasets achieve accuracies around 55% (Nowson, 2007; Estival et al., 2007). More recent work in this area focuses on the personality prediction in social networks (Kosinski et al., 2013; Kosinski et al., 2014) and multimodal personality prediction (Biel and Gatica-Perez, 2013; Aran and Gatica-Perez, 2013). These trends emphasized the correlation of network features and audiovisual features with extraversion, giving rise to the Workshop on Computational Personality Recognition (for an overview see (Celli et al., 2013; Celli et al., 2014).

3 Data set construction

Traditionally, the gold standard for this supervised classification task is obtained by the means of personality questionnaires, used for the Five-Factor Model, taken by each of the individuals assessed. This poses a challenge for fictional characters. However, strong correlations have been found between the self-reported and perceived personality traits (Mehl et al., 2006). Our gold standard benefits from the fact that readers enjoy discussing the personality of their favourite book character online. A popular layman instrument for personality classification is the Myers-Brigggs Type Indicator (Myers et al., 1985), shortly MBTI, which sorts personal preferences into four opposite pairs, or dichotomies, such as Thinking vs. Feeling or Judging vs. Perceiving. While the MBTI validity has been questioned by the research community (Pittenger, 2005), the Extraversion scale is showing rather strong validity and correlation to similar trait in the Five-Factor Model (McCrae and Costa, 1989; MacDonald et al., 1994). Our study hence focuses on the Extraversion scale.

Our data was collected from the collaboratively constructed Personality Databank[2] where the readers can vote if a book character is, among other aspects, introverted or extraverted. While the readers used codes based on the MBTI typology, they did not apply the MBTI assessment strategies. There was no explicit annotation guideline and the interpretation was left to readers' intuition and knowledge.[3] This approach of gold standard collection has several obvious drawbacks. First, the question is posed as dichotomic, while in reality the extraversion is a normally distributed trait in human population (Goldberg, 1990). Second, users can view the vote of previous participants, which may influence their decision. While we address both of these issues in our ongoing data collection project based on the Five-Factor Model, we consider them acceptable for this study due to the exploratory character of our pilot research.

We have collected extraversion ratings for 298 book characters, of which 129 (43%) are rather extraverted and 166 (56%) rather introverted. Rated characters come from a wide range of novels that the online users are familiar with, often covering classical literature which is part of the high school syllabus, as well as the most popular modern fiction, such as the Harry Potter series, Twilight, Star Wars or A Game of Thrones. A sample of the most rated introverts and extraverts is given in table 1. The rating distribution in our data is strongly U-shaped. The percentage agreement of voters in our data is 84.9%, calculated as:

$$P = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{k}\frac{n_{ij}(n_{ij}-1)}{n(n-1)}$$

where $k = 2$ (introvert, extravert), N is the number of book characters and n the number of votes per character. Voters on the website were anonymous and cannot be uniquely identified for additional corrections. There is no correlation between the extraversion and the gender of the character.

Character	Book	E	I
Tyrion Lannister	Game of Thrones	52	1
Cersei Lannister	Game of Thrones	48	7
Joffrey Baratheon	Game of Thrones	41	1
Ron Weasley	Harry Potter series	37	4
Jamie Lannister	Game of Thrones	38	9
Draco Malfoy	Harry Potter series	33	4
Anakin Skywalker	Star Wars series	30	6
Robert Baratheon	Game of Thrones	28	2
Gimli	Lord of the Rings	19	2
Jar Jar Binks	Star Wars series	12	2
Harry Potter	Harry Potter series	1	71
Severus Snape	Harry Potter series	1	65
Gandalf	Lord of the Rings	1	59
Yoda	Star Wars series	0	58
Jon Snow	Game of Thrones	1	47
Albus Dumbledore	Harry Potter series	4	46
Ned Stark	Game of Thrones	0	41
Aragorn	Lord of the Rings	1	41
Frodo	Lord of the Rings	1	40
Bran Stark	Game of Thrones	1	36

Table 1: Extraverts (E) and introverts (I) with the highest number of user votes.

Our set of English e-books covered 220 of the characters from our gold standard. We have built three systems to assess the following:

1. **Direct speech**: Does the style and content of character's utterances predict his extraversion in a similar way as it was shown for living individuals?
2. **Actions**: Is the behavior, of which a character is an agent, predictive for extraversion?
3. **Predicatives and adverbs**: Are the explicit (*John was an **exhibitionist***) or implicit (*John shouted **abruptly***) descriptions of the character in the book predictive for extraversion?

[2] http://www.mbti-databank.com/

[3] MBTI defines extraversion as "getting energy from active involvement in events, having a lot of different activities, enjoying being around people." In the NEO Five-Factor Inventory (Costa and McCrae, 1992), underlying facets of extraversion are warmth, gregariousness, assertiveness, activity, excitement seeking and positive emotion.

In the next three sections we present the experimental settings and results for each of the systems.

4 Direct speech of fictional characters

The system for the direct speech resembles the most to the previous systems developed for author personality profiling, e.g. on stream of consciousness essays (Mairesse et al., 2007) or social media posts (Celli et al., 2013) and therefore provides the best opportunity for comparison between human individuals and fictional characters. On top of the comparison to previous research, we exploit the sense links between WordNet and VerbNet to extract additional features - an approach which is novel for this type of task.

4.1 Extraction and assignment of speech

We process the book text using freely available components of the DKPro framework (Gurevych et al., 2007). The most challenging task in building the direct speech data set is assigning to the direct speech utterance the correct speaker. We benefit from the `epub` format of the e-books which defines a paragraph structure in such a way, that only the indirect speech chunk immediately surrounding the direct speech can be considered:

```
<p> John turned to Harry.
 "Let's go," he said.</p>
```

Given the large amount of text available in the books we focus on precision and discard all utterances with no explicit speaker (i.e., 30-70% of the utterances, dependent on the book), as the performance of current systems on such utterance types is still fairly low (O'Keefe et al., 2012; He et al., 2013; Iosif and Mishra, 2014). Similarly, conventional coreference resolution systems did not perform well on this type of data and were therefore not used in the final setup. We adapt the Stanford Named Entity Recognizer(Finkel et al., 2005) to consider titles (Mr., Mrs., Sir...) as a part of the name and to treat the first person I as a named entity. However, identifying only the named entity PERSON in this way is not sufficient. On our evaluation sample consisting of *A Game of Thrones* and *Pride and Prejudice* books (the former annotated by us, the latter by He et al. (2013)), 20% of utterances with explicit named speaker were not recognized. Of those correctly identified as a Person in the adjacent indirect speech, 17% were not the speakers. Therefore we implemented a custom heuristics (Algorithm 1), which additionally benefits from the WordNet semantic classes of verbs, enriching the speaker detection by grabbing the nouns . With this method we retrieve 89% of known speakers, of which 92% is assigned correctly. Retrieved names are grouped based on string overlap (e.g. *Ser Jaime* and *Jaime Lannister*), excluding the match on last name, and corrected for non-obvious groupings (such as *Margaret* and *Peggy*).

Algorithm 1 Assign speaker

1: *nsubj* ← subjects in adjacent indirect speech
2: **if** $count(nsubj(i) = \text{PERSON}) = 1$ **then** $speaker \leftarrow nsubj$
3: **else if** $count(nsubj(i) = \text{PERSON}) \geq 1$ **then** $speaker \leftarrow$ the nearest one to $directSpeech$
4: **else if** $directSpeech$ preceded by VERB.COMMUNICATION **then** $speaker \leftarrow$ the preceding noun(s)
5: **else if** $directSpeech$ followed by VERB.COMMUNICATION **then** $speaker \leftarrow$ the following noun(s)
6: **else if** $directSpeech$ followed by gap & VERB.COMMUNICATION **then** $speaker \leftarrow$ the noun(s) in gap
7: **else if** $directSpeech$ preceded by gap & VERB.COMMUNICATION **then** $speaker \leftarrow$ the noun(s) in gap
return $speaker$

Our experimental data consists of usable direct speech sets of 175 characters - 80 extraverts (E) and 95 introverts (I) - containing 289 274 words in 21 857 utterances (on average 111 utterances for E and 136 for I, as I are often central in books).[4]

4.2 Classification approach for direct speech

All speech utterances of one book character are represented as one instance in our system. We use the leave-one-out classification setup due to the relatively small dataset size, using the support vector machines (SVM-SMO) classifier, which performs well on comparable tasks (Celli et al., 2013). The classification is performed through the DKPro TC Framework (Daxenberger et al., 2014).

Lexical features As a bottom-up approach we use the 1000 most frequent word uni-, bi- and trigrams, 1000 dependency word pairs, 1000 character trigrams and 500 most frequent verbs, adverbs, adjectives and interjections as binary features.

Semantic features Since the top-down approach, i.e. not focusing on individual words, has

[4]The data set size is comparable to ongoing personality profiling challenges - see `http://pan.webis.de`

been found more suitable for the personality profiling task on smaller data sets (Celli et al., 2013), we aim on capturing additional phenomena on a higher level of abstraction. The main part of our features is extracted on sense level. We use the most frequent sense of WordNet (Miller, 1995) to annotate all verbs in the direct speech (a simple but well performing approach for books). We then label the disambiguated verbs with their semantic field given in WordNet (WordNet defines 14 semantic classes of verbs which group verbs by their semantic field) and we measure frequency and occurence of each of these classes (e.g. cognition, communication, motion, perception)[5]. Additionally, we use the lexical-semantic resource UBY (Gurevych et al., 2012) to access the WordNet and VerbNet information, and to exploit the VerbNet sense-level links which connects WordNet senses with the corresponding 273 main VerbNet classes (Kipper-Schuler, 2005). These are more fine-grained (e.g. pay, conspire, neglect, discover) than the WordNet semantic fields. WordNet covered 90% and VerbNet 86% of all the verb occurences.

On word level, we extract 81 additional features using the Linguistic Inquiry and Word Count (LIWC) tools (Pennebaker et al., 2001), which consists of lexicons related to psychological processes (cognitive, perceptual, social, biological, affective) and personal concerns (achievement, religion, death...) and other categories such as fillers, disfluencies or swear words[6]. Additionally, since emotion detection has been found predictive in previous personality work (Mohammad and Kiritchenko, 2013), we measure overall positive and negative sentiment expressed per character, using SentiWordNet (Esuli and Sebastiani, 2006) and NRC Emotion Lexicon (Mohammad and Turney, 2010) for the word lookup, inverting sentiment scores for negated dependency sub-tree given by the Stanford Parser.

Stylistic features Features of this group capture the syntactic and stylistic properties of the utterances of a character, disregarding the content. Starting from the surfacial properties, we measure the sentence, utterance and word length, including the proportion of words shorter than 4 or longer than 6 letters, frequency of each punctuation mark, and endings of each adjective as per Corney et al. (2002). On the syntax level we measure the frequency of each part of speech as well as the 500 most frequent part-of-speech bi-, tri- and quadrigrams, and the frequency of each dependency obtained from the Stanford Parser. We additionally capture the frequency of superlatives, comparatives and modal verbs, the proportion of verbs in present, past and future tense, and the formality of the language as per the part-of-speech-based formality coefficient (Heylighen and Dewaele, 2002), and measure the average depth of the parse trees.

Word embeddings as features Since vector space semantics has been beneficial for predicting author's personality in previous work (Neuman and Cohen, 2014), we use a pre-trained word vector model created by the GloVe algorithm (Pennington et al., 2014) on English Wikipedia. GloVe employs a global log-bilinear regression model that combines the advantages of the global matrix factorization and local context window methods. We assign the resulting 300-dimensional vectors to the words in character's direct speech, excluding stopwords, and calculate an average vector for each character. We calculate for each test character the cosine similarity to the mean vector of extravert, resp. introvert, in the training data, and to each character in the training set individually using the DL4J NLP package[7]. We consider both the final scalar outcome and the difference of each of the individual vector dimensions as features.

4.3 Classification results on direct speech

Table 2 shows the precision, recall, F_1-score and accuracy for extraversion and introversion as a weighted average of the two class values.

ID	Feature set	P	R	F	A
1	- (baseline)	.295	.543	.382	.543
2	Ngrams	.519	.514	.515	.514
3	LIWC	.555	.560	**.552**	.560
4	WordNet	.527	.548	.528	.548
5	VerbNet	.649	.617	**.572**	.617
6	Style	.560	.581	**.558**	.581
7	Sentiment	.524	.543	.419	.543
8	Vectors	.295	.543	.382	.543
9	All	.550	632.	**.588**	.632
Percentage human agreement:					.849

Table 2: Weighted precision (P), recall (R), F-score (F) and accuracy (A) for a direct speech system, in each line using only the given group of features. WordNet stands for WordNet semantic labels, VerbNet setup uses the WordNet-VerbNet links to retrieve VerbNet labels. Highlighted F-scores differ from the majority baseline significantly ($p < 0.05$), using an approximate randomization test.

[5] https://wordnet.princeton.edu/man/lexnames.5WN.html

[6] For complete overview refer to www.liwc.net

[7] http://deeplearning4j.org/

Introvert		
Feat.group	Features	Merit
unigrams	*reason, trouble, strange, indeed*	.24-.19
bigrams	*this time, tell me, I hope*	.19-.16
LIWC	Negate, Discrepancy, Insight, Exclusion	.18-.13
WordNet	stative, creation, cognition	.15-.09
VerbNet	lodge, hunt, defend	.23-.19
Style	modal verbs, neg, sbar, articles	.19-.14
Extravert		
Feat.group	Features	Merit
ngrams	*we, hurry, fat, dirty*	.24-.19
LIWC	We, Inclusion, Pronoun, Body	.18-.09
WordNet	motion, contact, communication, body, perception, change	.14-.07
VerbNet	get, talk, substance emission	.18-.15
Style	pronoun We, whadjp, type-token ratio., interjections	.20-.14

Table 3: The most predictive features for each group for speaker's extraversion and introversion. Correlation merit, as per the correlation feature selection in WEKA, evaluates Pearson's correlation between the feature and the class

Similarly to previous research (Mairesse et al., 2007; Celli et al., 2013), the bottom-up word based approach is outperformed by top-down semantic approaches which employ a more abstract feature representation. As in previous work, LIWC features exhibit good performance. However, the highest performance is achieved employing the VerbNet verb classes with WordNet word-sense disambiguation. Also stylistic features contribute substantially to the classification despite the mixture of genres in our book corpus - especially frequencies of modal verbs and part-of-speech ratios were particularly informative. The most predictive features from each group are listed in Table 3 together with their correlation merit (Hall, 1999), and compared with previous work in Table 4.

Feature	**I/E**	**Ref**	**Feature**	**I/E**	**Ref**
Predictive also in our data:			No effect in our data:		
Pronoun 'we'	-/+	[3]	Neg. emot.	+/-	[1]
Tentative, unsure	+/-	[1]	Pos. emot.	-/+	[1]
Exclusive	+/-	[1]	Self-ref.	-/+	[1]
Inclusive	-/+	[1]	Formality	+/-	[2]
Insight	+/-	[1]	Elaborated	+/-	[3]
Nouns, articles	+/-	[2]	Long sent.	+/-	[3]
Lexical richness	+/-	[2]	Social	-/+	[3]
Negations	+/-	[2]			
Body functions	-/+	[2]			
Interjections	-/+	[3]			
Source ID		**Author**			
[1]		Pennebaker and King (1999)			
[2]		Dewaele and Furnham (1999)			
[3]		Mairesse et al. (2007)			

Table 4: Comparison of our results to previously reported predictive features for speaker's extraversion (E), resp. introversion (I). We list publications where these features were, to our knowledge, reported as novel.

In accordance with the experiments of Pennebaker and King (1999), we observe more frequent exclusions (e.g. *without, but*), hedging and negation expressed by introverts, and inclusion (e.g. *with, and*) by extraverts. Extraverts talk more in first person plural, use more back-channels and interjections, and talk more about aspects related to their body. Introverts show more rationalization through insight words and more factual speech using less pronouns.

Additionally, the semantic features in Table 3 confirm the broad psychological characteristics of both types in general, i.e., for introverts the rationalization, uncertainty and preference for individual or rather static activities, and for extraverts their spontaneity, talkativeness and preference for motion. Furthermore, we observe certain directness in extraverts' speech - note the predictive words *fat* and *dirty* and frequent descriptions of body functions.

Discussion Exploiting the links between lexical-semantic resources (performing WordNet word-sense disambiguation and using VerbNet verb classes linked to the disambiguated senses) was particularly beneficial for this task. WordNet semantic fields for verbs alone are too coarse-grained to capture the nuances in direct speech, and experiments with fine-grained VerbNet classes without WSD resulted in noisy labels. We did not confirm the previously reported findings on emotional polarity - we observe that the genre of the books (e.g. love romance vs horror story) have blurred the subtle differences between individual characters, unfortunately the dataset size did not allow for genre distinctions. Furthermore, a perceived extravert in our case can be a pure villain (Draco Malfoy, Joffrey Baratheon...) as well as a friendly companion (Gimli, Ron Weasley...), while the evil extravert types are possibly rarer in the experiments on human writing, or are more likely to fit under the MBTI definition of extraversion than FFM facets. Another potential cause, based on the error analysis, is the different target of the same sentiment for extraverts and introverts. For example, the ngram "I fear" is highly predictive for an introvert in our data while extraverts would rather use formulations to imply that others should fear. Similarly to Nowson et al. (2005), we did not find any difference in the formality measure of Heylighen and Dewaele (2002). Neither we did in the complexity of sentences as per the parse tree depth

and sentence length. It is probable that these aspects were also impacted by our broad variety of author style (F. Dostoyevsky vs J. K. Rowling). Our basic vector-based features carried no useful information in our case, in contrast to the personality research of Neuman and Cohen (2014). We observed that the factual content of the stories contributed to the character similarity measure more than the subtle personality differences.

5 Actions of fictional characters

While psycholinguists and consequenlty NLP researchers analyzed the relation between speech, resp. writing, and personality of an individual, psychologists often evaluate extraversion through behavioral personality questionnaries (Costa and McCrae, 1992; Goldberg et al., 2006). We hypothesize that similar behavior shall be predictive for extraversion of fictional characters as perceived by the readers.

5.1 Action extraction

For our purpose we define actions as the subject, verb and context of a sentence, where the subject is a named entity Person and the context is either a direct object in relation *dobj* to the verb or a first child of the adjacent verb phrase in a parse tree. After grouping the actions per character, the subject name is removed. For example, a sample of actions of the character Eddard Stark of Game of Thrones would be: *X paused a moment, X studied his face, X changed his mind, X unrolled the paper, X said* etc., visualized in Figure 1. We obtained 22 030 actions for 205 characters (102 E, 116 I), with on average 100 actions for E and 101 for I. Note that also actions for those characters who do not talk enough in the books (often first-person perspectives) could be used.

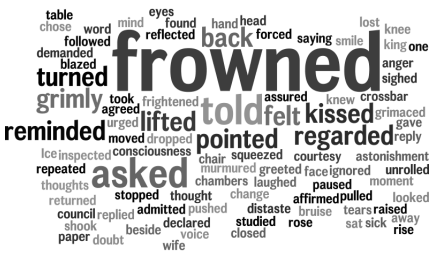


Figure 1: A revealing word cloud of the most frequent words from the actions of which Eddard Stark (Game of Thrones) is a subject. Size is proportional to the frequency of a word.

5.2 Action classification setup

In the system based on actions we use only a subset of the features described in 4.2. From the lexical features we focus on the 500 most frequent verbs and dependency word pairs. Semantic features are used the same way as in 4.2, profiting from LIWC, WordNet, Verbnet and the sentiment lexicons. Word embedding vectors for book characters are in this case computed by taking only the verbs into account rather than all content words. From the stylistic features we use the part-of-speech bigrams and trigrams, verb modality and verb tense.

5.3 Classification results on actions

Table 5 shows the performance of the classification models based on the protagonists' actions, using different feature groups. The overall performance is higher than for the direct speech model.

ID	Feature set	P	R	F	A
1	- (baseline)	.267	.517	.352	.517
2	Ngrams	.539	.506	.505	.507
3	LIWC	.600	.577	**.567**	.577
4	WordNet	.517	.518	.517	.518
5	VerbNet	.599	.583	**.578**	.583
6	Style	.573	.601	**.553**	.601
7	Sentiment	.357	.453	.382	.453
8	Vectors	.504	.497	.451	.497
9	All	.600	.623	**.598**	.623
Percentage human agreement:					.849

Table 5: Weighted precision (P), recall (R), F-score (F) and accuracy (A) for actions - in each line for a system using only the given group of features. WordNet stands for WordNet semantic labels, VerbNet setup uses the WordNet-VerbNet links. Highlighted F-scores differ from the majority baseline significantly ($p < 0.05$), using an approx. randomization test.

Due to the lack of previous NLP experiments on this task, we compare our features to the actions measured in the International Personality Item Pool (Goldberg et al., 2006), frequently used personality assesment questionnaire (Table 6).

The most predictive features of this model capture the activity and excitement seeking facets of extraversion. Stylistic features reflect the complexity difference of the verb phrases (*John jumped vs. John thought about it*), extraverts being characterized by plain verbs. Semantic features exhibit higher precision than stylistic ones. Sense-linked semantic classes of VerbNet demonstrate the preference of extraverts for being active and expressing themselves - they *jump*, *fight, shout, run* in and run out, *eat* and *drink*, *see* and *hear* and get easily *bored*. Extraverts in books also

Extravert International Personality Item Pool:
likes to party, feels comfortable around people, starts conversations, talks to many people, enjoys being a center of attention, makes friends easily, takes charge, captivates people, feels at ease with a company, is skilled in handling social situations
Our experiment:
bring (VN), consume (VN), contiguous location(VN), holding (VN), social (WN), motion (WN), emotion (WN) Leisure (LIWC), Home (LIWC), Family (LIWC), fight, march, care, take, jump, shriek, clear throat, bore, get to, come in, agree, hold, hear, inform, sell, come forward
Introvert International Personality Item Pool:
Doesn't talk much, stays in the background, has little to say, does not draw attention, has difficulties to approach others, is quiet around strangers, feels uncomfortable around others,does not show feelings, is a private person, waits to be lead
Our experiment:
snooze (VN), conceal (VN), wish (VN), stative (WN), creation (WN), walk, sleep, lay, know, maintain, expect, hope, find out, might, help, explain

Table 6: Characteristic actions for extraverts and introverts as assessed in the IPIP personality questionaire, compared to our most informative features

often *bring* or *hold* something. Introverts, on the other hand, seem to favor slow movements - while they are *thinking, reflecting, creating*, looking for *explanations* and *find out* solutions, they tend to *lie down, sit* or *walk*, eventually even *sleep* or *snooze*. The uncertainty typical for introverts is also notable in their actions, as they often *hope* or *wish* for something they *might* like to do. Additionally, semantic classes Social and Family, reported as correlated to extraversion by Pennebaker and King (1999) and not confirmed in our first model, became predictive in protaonists' actions.

5.4 Discussion

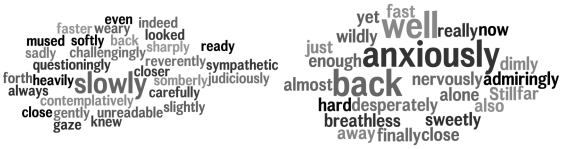
Also in this task, the VerbNet classes brought significant improvement in performance. The classification model based on actions outperforms not only the direct speech model, but also the state-of-the-art systems predicting authors' extraversion from the stream-of-consciousness essays (Mairesse et al., 2007; Celli et al., 2013; Neuman and Cohen, 2014). While surely not directly comparable, this result hints to the fact that the personality is easier to detect from behavior than from person's verbal expression. This would correspond to the findings of Mairesse et al. (2007), Biel and Gatica-Perez (2013) and Aran and Gatica-Perez (2013) on multimodal data sets.

6 Predicatives of fictional characters

Our third extraversion prediction system is subordinate to how fictional characters are described and to the manners in which they behave. We are not aware of a previous NLP work predicting extraversion using descriptive adjectives of the persons in question. We thus juxtapose the most predictive features of our system to the adjectival extraversion markers developed by Goldberg (1992).

6.1 Extraction of descriptive properties

In this setup we extract predicatives of the named entities PERSON in the books - relations *amod* (***angry*** *John*) and *cop* (*John was* ***smart***). As these explicit statements are very sparse in modern novels, we additionally include adverbial modifiers (*advmod*) related to person's actions (*John said* ***angrily***). We extract data for 205 characters, with on average 43 words per character.



(a) Master Yoda (Star Wars)

(b) Sansa Stark (Game of Thrones)

Figure 2: Frequency word clouds for character descriptions

6.2 Classification setup

This system uses similar set of lexical, semantic and vectorial features similarly as in 5.2, this time with the focus on adjectives, nouns and adverbs instead of verbs. Stylistic and VerbNet features are hence not included, word vectors are as in 4.2.

6.3 Classification results on descriptions

Table 7 reports on the performance of individual feature groups. With only few words per character semantic lexicons are less powerful than ngrams.

ID	Feature set	P	R	F	A
1	- (baseline)	.267	.517	.352	.517
2	Ngrams	.686	.657	**.648**	.657
3	LIWC	.645	.601	**.586**	.601
4	WordNet	.518	.545	.528	.545
5	Sentiment	.375	.463	.384	.463
6	Vectors	.267	.517	.352	.517
7	All	.692	.698	**.693**	.698
Percentage human agreement:					.849

Table 7: Weighted precision, recall, F-score and accuracy. Highlighted F-scores differ from the majority baseline significantly ($p < 0.05$).

Table 8 displays the most predictive features in our system contrasted to the adjectival markers.

Extravert Goldberg (1992) :
adventurous,mischievous, playful, rambunctious, dominant, forceful, demonstrative, exhibitionistic, flamboyant, brave, courageous, daring, assured,...
Our experiment :
excited, restlessly, stubbornly, restless, beloved, eager, abruptly, defiantly, darkly, eagerly, loudly, reluctant, stubborn, unwise, ruthless, quickly, abruptly, right, change (WN), social (WN)
Introvert Goldberg (1992) :
bashful, shy, timid, inhibited, restrained unadventurous, unaggressive, uncompetitive bitter, joyless, melancholic, moody, morose,...
Our experiment :
anxious, patiently, hesitantly, backward, softly, warily, coldly, helplessly, respectfully, slowly, politely, thoughtfully, nervously, silent, carefully, gratefully, dryly, sheepishly, politely, weary, calm, gently, sadly, sideways, stative (WN)

Table 8: Characteristic adjectives for extraverts and introverts as reported by L. Goldberg, compared to our most informative features as per the correlation merit

6.4 Discussion on errors

All our systems had issues with characters rated by less than five readers and with protagonists with low agreement. Other challenges arise from authorial style, age of the novel and speech individuality of characters (e.g. *Yoda*). Varied length of information for different characters poses issues in measuring normally distributed features (e.g. ratio of jumping verbs), being in shorter texts less reliable. Ongoing and future work on this task addresses the limitations of these initial experiments, especially the data set size and the gold standard quality. Extending the data will also enable us to examine different book genres as variables for the personality distribution and feature impact. It will be worth examining the relations between characters, since we observed certain patterns in our data, such as the main introvert character supported by his best friend extravert. Additionally, we want to verify if the system in Section 6 is overly optimistic due to the data size.

7 Conclusion and future work

Automated personality profiling of fictional characters, based on rigorous models from personality psychology, has a potential to impact numerous domains. We framed it as a text classification problem and presented a novel collaboratively built dataset of fictional personality. We incorporate features of both lexical resource-based and vectorial semantics, including WordNet and VerbNet sense-level information and vectorial word representations. In models based on the speech and actions of the protagonists, we demonstrated that the sense-linked lexical-semantic features significantly outperform the baselines. The most predictive features correspond to the reported findings in personality psychology and NLP experiments on human personality. Our systems based on actions and appearance of characters demonstrate higher performance than systems based on direct speech, which is in accordance with recent research on personality in social networks (Kosinski et al., 2014; Biel and Gatica-Perez, 2013), revealing the importance of the metadata. We have shown that exploiting the links between lexical resources to leverage more accurate semantic information can be beneficial for this type of tasks, oriented to actions performed by the entity. However, the human annotator agreement in our task stays high above the performance achieved. Considering that most of the sucessful novels were produced as movies, we cannot exclude that our annotators based their decision on the multimodal representation of the protagonists. In the future we aim on collecting a more detail and rigorous gold standard through gamification and expanding our work on all five personality traits from the Five-Factor Model and their facets, and ultimately extend our system to a semi-supervised model dealing with notably larger amount of data. We also plan to examine closer the differences between perceived human and fictional personality, and the relationship between the personality of the reader and the characters.

Acknowledgments

This work has been supported by the Volkswagen Foundation as part of the Lichtenberg Professorship Program under grant No. I/82806 and by the German Research Foundation under grant No. GU 798/14-1. Additional support was provided by the German Federal Ministry of Education and Research (BMBF) as a part of the Software Campus program under the promotional reference 01-S12054 and by the German Institute for Educational Research (DIPF). We also warmly thank Holtzbrinck Digital GmbH for providing a substantial part of the e-book resources, and the EMNLP reviewers for their helpful comments.

References

Oya Aran and Daniel Gatica-Perez. 2013. Cross-domain personality prediction: from video blogs to small group meetings. In *Proceedings of the 15th ACM on International conference on multimodal interaction*.

Shlomo Argamon, Sushant Dhawle, Moshe Koppel, and James W. Pennebaker. 2005. Lexical predictors of personality type. In *Proceedings of the Joint Annual Meeting of the Interface and the Classification Society of North America*.

David Bamman, Ted Underwood, and Noah A. Smith. 2014. A bayesian mixed effects model of literary character. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*.

Joan-Isaac Biel and Daniel Gatica-Perez. 2013. The youtube lens: Crowdsourced personality impressions and audiovisual analysis of vlogs. *Multimedia, IEEE Transactions on*, 15(1):41–55.

Donn Byrne. 1961. Interpersonal attraction and attitude similarity. *The Journal of Abnormal and Social Psychology*, 62(3):713.

Fabio Celli, Fabio Pianesi, David Stillwell, and Michal Kosinski. 2013. Workshop on computational personality recognition (shared task). In *Proceedings of the Workshop on Computational Personality Recognition*.

Fabio Celli, Bruno Lepri, Joan-Isaac Biel, Daniel Gatica-Perez, Giuseppe Riccardi, and Fabio Pianesi. 2014. The workshop on computational personality recognition 2014. In *Proceedings of the ACM International Conference on Multimedia*. ACM.

Nathanael Chambers. 2013. Event schema induction with a probabilistic entity-driven model. In *EMNLP*, volume 13, pages 1797–1807.

Tanya L. Chartrand and John A. Bargh. 1999. The chameleon effect: the perception–behavior link and social interaction. *Journal of personality and social psychology*, 76(6):893.

Max Coltheart. 1981. The mrc psycholinguistic database. *The Quarterly Journal of Experimental Psychology*, 33(4):497–505.

Malcolm Corney, Olivier de Vel, Alison Anderson, and George Mohay. 2002. Gender-preferential text mining of e-mail discourse. In *Proceedings of 18th Annual Computer Security Applications Conference*. IEEE.

Paul T. Costa and Robert R. McCrae. 1992. Professional manual: revised NEO personality inventory (NEO-PI-R) and NEO five-factor inventory (NEO-FFI). *Odessa, FL: Psychological Assessment Resources*.

Johannes Daxenberger, Oliver Ferschke, Iryna Gurevych, and Torsten Zesch. 2014. Dkpro tc: A java-based framework for supervised learning experiments on textual data. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics. System Demonstrations*, pages 61–66.

Orphée De Clercq, Michael Schuhmacher, Simone Paolo Ponzetto, and Veronique Hoste. 2014. Exploiting framenet for content-based book recommendation. In *CBRecSys at ACM RecSys*, number 1613-0073, pages 14–21. CEUR-WS.

Jean-Marc Dewaele and Adrian Furnham. 1999. Extraversion: The unloved variable in applied linguistic research. *Language Learning*, 49(3):509–544.

Tommaso Di Noia, Roberto Mirizzi, Vito Claudio Ostuni, Davide Romito, and Markus Zanker. 2012. Linked open data to support content-based recommender systems. In *Proceedings of the 8th International Conference on Semantic Systems*, I-SEMANTICS, pages 1–8.

Daniel Dotson. 2009. Portrayal of physicists in fictional works. *CLCWeb: Comparative Literature and Culture*, 11(2):5.

David K. Elson, Nicholas Dames, and Kathleen R. McKeown. 2010. Extracting social networks from literary fiction. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*.

Dominique Estival, Tanja Gaustad, Son Bao Pham, Will Radford, and Ben Hutchinson. 2007. Author profiling for english emails. In *Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics*.

Andrea Esuli and Fabrizio Sebastiani. 2006. SentiWordNet: a publicly available lexical resource for opinion mining. In *Proceedings of LREC*, volume 6.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 363–370. Association for Computational Linguistics.

Alastair J. Gill and Jon Oberlander. 2002. Taking care of the linguistic features of extraversion. In *Proceedings of the 24th Annual Conference of the Cognitive Science Society*.

Lewis R. Goldberg, John A. Johnson, Herbert W. Eber, Robert Hogan, Michael C. Ashton, C. Robert Cloninger, and Harrison G. Gough. 2006. The international personality item pool and the future of public-domain personality measures. *Journal of Research in personality*, 40(1):84–96.

Lewis R Goldberg. 1990. An alternative description of personality: the Big-Five factor structure. *Journal of personality and social psychology*, 59(6):1216.

Lewis R Goldberg. 1992. The development of markers for the Big-Five factor structure. *Psychological assessment*, 4(1):26.

Iryna Gurevych, Max Mühlhäuser, Christof Müller, Jürgen Steimle, Markus Weimer, and Torsten Zesch. 2007. Darmstadt Knowledge Processing Repository Based on UIMA. In *Proceedings of the First Workshop on Unstructured Information Management Architecture at Biannual Conference of the Society for Computational Linguistics and Language Technology*, Tübingen, Germany.

Iryna Gurevych, Judith Eckle-Kohler, Silvana Hartmann, Michael Matuschek, Christian M. Meyer, and Christian Wirth. 2012. Uby: A large-scale unified lexical-semantic resource based on LMF. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics.

Mark A. Hall. 1999. *Correlation-based feature selection for machine learning*. Ph.D. thesis, The University of Waikato.

Hua He, Denilson Barbosa, and Grzegorz Kondrak. 2013. Identification of speakers in novels. In *Proceedings of the 51st Annual Meeting on Association for Computational Linguistics*, pages 1312–1320.

Benjamin Heitmann and Conor Hayes. 2010. Using linked data to build open, collaborative recommender systems. In *AAAI symposium Linked data meets artificial intelligence*.

Francis Heylighen and Jean-Marc Dewaele. 2002. Variation in the Contextuality of Language: An Empirical Measure. *Foundations of Science*, 7(3):293–340.

Elias Iosif and Taniya Mishra. 2014. From speaker identification to affective analysis: A multi-step system for analyzing children stories. *EACL 2014*, pages 40–49.

Oliver P. John and Sanjay Srivastava. 1999. The Big Five trait taxonomy: History, measurement, and theoretical perspectives. *Handbook of personality: Theory and research*, 2(1999):102–138.

John A. Johnson, Joseph Carroll, Jonathan Gottschall, and Daniel Kruger. 2011. Portrayal of personality in victorian novels reflects modern research findings but amplifies the significance of agreeableness. *Journal of Research in Personality*, 45(1):50–58.

Geoff F. Kaufman and Lisa K. Libby. 2012. Changing beliefs and behavior through experience-taking. *Journal of personality and social psychology*, 103(1):1.

Karin Kipper-Schuler. 2005. *VerbNet: A broad-coverage, comprehensive verb lexicon.* Ph.D. thesis, University of Pennsylvania.

Dimitrios Kokkinakis and Mats Malm. 2011. Character profiling in 19th century fiction. *Language Technologies for Digital Humanities and Cultural Heritage*.

Michal Kosinski, David Stillwell, and Thore Graepel. 2013. Private traits and attributes are predictable from digital records of human behavior. *Proceedings of the National Academy of Sciences*, 110(15):5802–5805.

Michal Kosinski, Yoram Bachrach, Pushmeet Kohli, David Stillwell, and Thore Graepel. 2014. Manifestations of user personality in website choice and behaviour on online social networks. *Machine learning*, 95(3):357–380.

Bruno Lepri, Ramanathan Subramanian, Kyriaki Kalimeri, Jacopo Staiano, Fabio Pianesi, and Nicu Sebe. 2010. Employing social gaze and speaking activity for automatic determination of the extraversion trait. In *International Conference on Multimodal Interfaces and the Workshop on Machine Learning for Multimodal Interaction*, ICMI-MLMI '10, New York, NY, USA. ACM.

Douglas A. MacDonald, Peter E. Anderson, Catherine I. Tsagarakis, and Cornelius J. Holland. 1994. Examination of the relationship between the Myers-Briggs Type Indicator and the NEO personality inventory. *Psychological Reports*, 74(1):339–344.

François Mairesse, Marilyn A. Walker, Matthias R. Mehl, and Roger K. Moore. 2007. Using linguistic cues for the automatic recognition of personality in conversation and text. *Journal of artificial intelligence research*.

Raymond A. Mar, Keith Oatley, and Jordan B. Peterson. 2009. Exploring the link between reading fiction and empathy: Ruling out individual differences and examining outcomes. *Communications Journal*, 34(4):407–428.

Robert R. McCrae and Paul T. Costa. 1987. Validation of the five-factor model of personality across instruments and observers. *Journal of personality and social psychology*, 52(1):81.

Robert R. McCrae and Paul T. Costa. 1989. Reinterpreting the myers-briggs type indicator from the perspective of the five-factor model of personality. *Journal of personality*, 57(1):17–40.

Robert R. McCrae, James F. Gaines, and Marie A. Wellington. The five-factor model in fact and fiction. *Handbook of Psychology*.

Matthias R. Mehl, Samuel D. Gosling, and James W. Pennebaker. 2006. Personality in its natural habitat: manifestations and implicit folk theories of personality in daily life. *Journal of personality and social psychology*, 90(5):862.

George A. Miller. 1995. WordNet: a lexical database for English. *Communications of the ACM*, 38(11):39–41.

Saif M. Mohammad and Svetlana Kiritchenko. 2013. Using nuances of emotion to identify personality. *AAAI Technical Report WS-13-01 Computational Personality Recognition (Shared Task)*.

Saif M. Mohammad and Peter D. Turney. 2010. Emotions evoked by common words and phrases: Using Mechanical Turk to create an emotion lexicon. In *Proceedings of the NAACL HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text*, pages 26–34.

R. Matthew Montoya, Robert S. Horton, and Jeffrey Kirchner. 2008. Is actual similarity necessary for attraction? a meta-analysis of actual and perceived similarity. *Journal of Social and Personal Relationships*, 25(6):889–922.

Isabel Briggs Myers, Mary H. McCaulley, and Robert Most. 1985. *Manual, a guide to the development and use of the Myers-Briggs type indicator*. Consulting Psychologists Press.

Yair Neuman and Yochai Cohen. 2014. A vectorial semantics approach to personality assessment. *Scientific reports*, 4.

Scott Nowson, Jon Oberlander, and Alastair J. Gill. 2005. Weblogs, genres and individual differences. In *Proceedings of the 27th Annual Conference of the Cognitive Science Society*, volume 1666, page 1671. Citeseer.

Scott Nowson. 2007. Identifying more bloggers: Towards large scale personality classification of personal weblogs. In *In Proceedings of the International Conference on Weblogs and Social*.

Jon Oberlander and Scott Nowson. 2006. Whose thumb is it anyway? classifying author personality from weblog text. In *Proceedings of the COLING/ACL on Main conference poster sessions*.

Tim O'Keefe, Silvia Pareti, James R. Curran, Irena Koprinska, and Matthew Honnibal. 2012. A sequence labelling approach to quote attribution. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 790–799. Association for Computational Linguistics.

Alexandre Passant. 2010. dbrec music recommendations using DBpedia. In *The Semantic Web–ISWC 2010*, pages 209–224.

James W. Pennebaker and Laura A. King. 1999. Linguistic styles: language use as an individual difference. *Journal of personality and social psychology*, 77(6):1296.

James W. Pennebaker, Martha E. Francis, and Roger J. Booth. 2001. Linguistic inquiry and word count: LIWC 2001. *Mahway: Lawrence Erlbaum Associates*, 71.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. *Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014)*, 12.

David J. Pittenger. 2005. Cautionary comments regarding the myers-briggs type indicator. *Consulting Psychology Journal: Practice and Research*, 57(3):210.

Peter J. Rentfrow, Lewis R. Goldberg, and Daniel J. Levitin. 2011. The structure of musical preferences: a five-factor model. *Journal of personality and social psychology*, 100(6):1139.

Noah A. Smith, David Bamman, and Brendan OConnor. 2013. Learning latent personas of film characters. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*.

Antonio Terracciano, Corinna E. Löckenhoff, Rosa M. Crum, O .Joseph Bienvenu, and Paul T. Costa. 2008. Five-factor model personality profiles of drug users. *BMC Psychiatry*, 8(1):22.

William C. Tirre and Sharvari Dixit. 1995. Reading interests: Their dimensionality and correlation with personality and cognitive factors. *Personality and Individual Differences*, 18(6):731–738.

Leave-one-out Word Alignment without Garbage Collector Effects

Xiaolin Wang **Masao Utiyama** **Andrew Finch**
Taro Watanabe* **Eiichiro Sumita**
Advanced Translation Research and Development Promotion Center
National Institute of Information and Communications Technology, Japan
{xiaolin.wang,mutiyama,andrew.finch,eiichiro.sumita}@nict.go.jp
tarow@google.com

Abstract

Expectation-maximization algorithms, such as those implemented in GIZA++ pervade the field of unsupervised word alignment. However, these algorithms have a problem of over-fitting, leading to "garbage collector effects," where rare words tend to be erroneously aligned to untranslated words. This paper proposes a leave-one-out expectation-maximization algorithm for unsupervised word alignment to address this problem. The proposed method excludes information derived from the alignment of a sentence pair from the alignment models used to align it. This prevents erroneous alignments within a sentence pair from supporting themselves. Experimental results on Chinese-English and Japanese-English corpora show that the F_1, precision and recall of alignment were consistently increased by 5.0% – 17.2%, and BLEU scores of end-to-end translation were raised by 0.03 – 1.30. The proposed method also outperformed l_0-normalized GIZA++ and Kneser-Ney smoothed GIZA++.

1 Introduction

Unsupervised word alignment (WA) on bilingual sentence pairs serves as an essential foundation for building most statistical machine translation (SMT) systems. A lot of methods have been proposed to raise the accuracy of WA in an effort to improve end-to-end translation quality. This paper contributes to this effort through refining the widely used expectation-maximization (EM) algorithm for WA (Dempster et al., 1977; Brown et al., 1993b; Och and Ney, 2000).

The EM algorithm for WA has a great influence in SMT. Many well-known toolkits including GIZA++ (Och and Ney, 2003), the Berkeley Aligner (Liang et al., 2006; DeNero and Klein, 2007), Fast Align (Dyer et al., 2013) and SyMGIZA++ (Junczys-Dowmunt and Sza, 2012), all employ this algorithm. GIZA++ in particular is frequently used in systems participating in many shared tasks (Goto et al., 2011; Cettolo et al., 2013; Bojar et al., 2013).

However, the EM algorithm for WA is well-known for introducing "garbage collector effects." Rare words have a tendency to collect garbage, that is they have a tendency to be erroneously aligned to untranslated words (Brown et al., 1993a; Moore, 2004; Ganchev et al., 2008; V Graça et al., 2010). Figure 1(a) shows a real sentence pair, denoted $\mathbf{s}$, from the GALE Chinese-English Word Alignment and Tagging Training corpus (GALE WA corpus)[1] with it's human-annotated word alignment. The Chinese word "HE ZHANG," denoted w_r, which means river custodian, only occurs once in the whole corpus. We performed EM training using GIZA++ on this corpus concatenated with 442,967 training sentence pairs from the NIST Open Machine Translation (OpenMT) 2006 evaluation[2]. The resulting alignment is shown in Figure 1(b). It can be seen that w_r is erroneously aligned to multiple English words.

To find the cause of this, we checked the alignments in each iteration i of $\mathbf{s}$, denoted $\mathbf{a_s^i}$. We found that in $\mathbf{a_s^1}$, w_r together with the other source-side words were aligned with uniform probability to all the target-side words since the alignment models provided no prior information. However, in $\mathbf{a_s^2}$, w_r became erroneously aligned,

* The author now is affiliated with Google, Japan.

[1] Released by Linguistic Data Consortium, catalog number LDC2012T16, LDC2012T20, LDC2012T24 and LDC2013T05.

[2] http://www.itl.nist.gov/iad/mig/tests/mt/2006/

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1817–1827,
Lisbon, Portugal, 17-21 September 2015.

because the alignment distribution[3] of w_r was only learned from $\mathbf{a_s^1}$, thus consisted of non-zero values only for generating the target-side words in $\mathbf{s}$. Therefore, the alignment probabilities from the rare word w_r to the unaligned words in $\mathbf{s}$ were extraordinarily high, since almost all of the probability mass was distributed among them. In other words, the story behind these garbage collector effects is that erroneous alignments are able to provide support for themselves; the probability distribution learned only from $\mathbf{s}$ is re-applied to $\mathbf{s}$. In this way, these "garbage collector effects" are a form of over-fitting.

Motivated by this observation, we propose a leave-one-out EM algorithm for WA in this paper. Recently this technique has been applied to avoid over-fitting in kernel density estimation (Roux and Bach, 2011); instead of performing maximum likelihood estimation, maximum leave-one-out likelihood estimation is performed. Figure 1(c) shows the effect of using our technique on the example. The garbage collection has not occurred, and the alignment of the word "HE ZHANG" is identical to the human annotation.

2 Related Work

The most related work to this paper is training phrase translation models with leave-one-out forced alignment (Wuebker et al., 2010; Wuebker et al., 2012). The differences are that their work operates at the phrase level, and their aim is to improve translation models; while our work operates at the word level, and our aim is to provide better word alignment. As word alignment is a foundation of most MT systems, our method have a wider application.

Recently, better estimation methods during the maximization step of EM have been proposed to avoid the over-fitting in WA, such as using Kneser-Ney Smoothing to back-off the expected counts (Zhang and Chiang, 2014) or integrating the smoothed l_0 prior to the estimation of probability (Vaswani et al., 2012). Our work differs from theirs by addressing the over-fitting directly in the EM algorithm by adopting a leave-one-out approach.

Bayesian methods (Gilks et al., 1996; Andrieu et al., 2003; DeNero et al., 2008; Neubig et al., 2011), also attempt to address the issue of over-fitting, however EM algorithms related to the proposed method have been shown to be more efficient (Wang et al., 2014).

[3]The probability distribution of generating target language words from w_r. The description here is only based on IBM model1 for simplicity, and the other alignment models are similar.

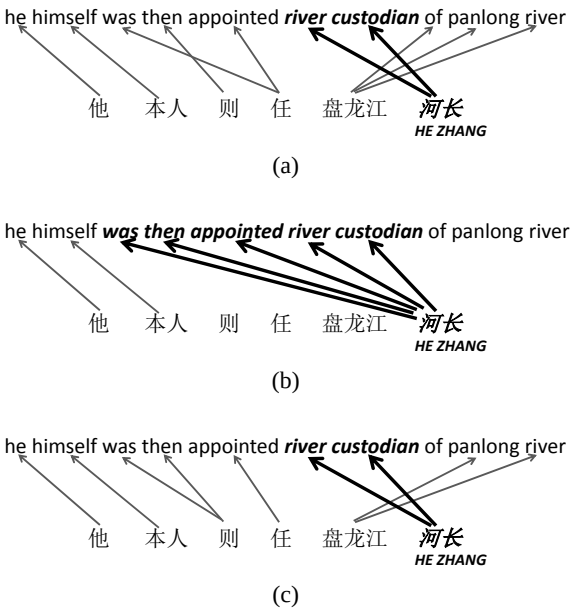


Figure 1: Examples of supervised word alignment. (a) gold alignment; (b) standard EM (GIZA++); (c) Leave-one-out alignment (proposed).

3 Methodology

This section first formulates the standard EM for WA, then presents the leave-one-out EM for WA, and finally briefly discusses handling singletons and effecient implementation. The main notation used in this section is shown in Table 1.

3.1 Standard EM for IBM Models 1, 2 and HMM Model

To perform WA through EM, the parallel corpus is taken as observed data, the alignments are taken as latent data. In order to maximize the likelihood of the alignment model θ given the data $\mathbf{S}$, the following two steps are conducted iteratively (Brown et al., 1993b; Och and Ney, 2000; Och and Ney, 2003),

Expectation Step (E step): calculating the conditional probability of alignments for each sentence pair,

$$P(\mathbf{a}|\mathbf{s},\theta) \quad = \prod_{j=1}^{J} \theta_{\text{ali}}(a_j|a_{j-1},I)\theta_{\text{lex}}(f_j|e_{a_j}) \quad (1)$$

where $\theta_{\text{ali}}(i|i',I)$ is the alignment probability and $\theta_{\text{lex}}(f|e)$ is the translation probability. Note that

$\mathbf{f}$	a foreign sentence $(f_1, \ldots, f_J)$
$\mathbf{e}$	an English sentence $(e_1, \ldots, e_I)$
$\mathbf{s}$	a sentence pair $(\mathbf{f}, \mathbf{e})$
$\mathbf{a}$	an alignment $(a_1, \ldots, a_J)$ where f_j is aligned to e_{a_j}
B_i	a list of the indexes of the foreign words which are aligned to e_i
$B_{i,k}$	the index of the k-th foreign word which is aligned to e_i
$\overline{B_i}$	is the average of all elements in B_i
ρ_i	the largest index of an English word $s.t.\ \rho_i < i$ and $\lvert B_{\rho_i}\rvert > 0$
ϕ_i	the fertility of e_i
E_i	the word class of e_i
$\theta_\cdot$	an probabilistic model
$\theta_\cdot^{\bar{\mathbf{s}}}$	a leave-one-out probabilistic model for $\mathbf{s}$
$n_x(\mathbf{s}, \mathbf{a})$	the number of times that an event x happens in $(\mathbf{s}, \mathbf{a})$
$N_x(\mathbf{s})$	the marginal number of times that an event x happens in $\mathbf{s}$

Table 1: Main Notation. Note that $N_x(\mathbf{s}) = \sum_{\mathbf{a}} n_x(\mathbf{s}, \mathbf{a}) P(\mathbf{a}|\mathbf{s})$. In practical calculation, for IBM models 1, 2 and HMM model, this summation is performed by dynamic programming; for IBM model 4, it is performed approximately using the best alignment and its neighbors.

(1) is a general form for IBM model 1, model 2 and the HMM model.

Maximization step (M step): re-estimating the probability models,

$$\theta_{\text{ali}}(i|i', I) \leftarrow \frac{\sum_{\mathbf{s}} N_{i|i',I}(\mathbf{s})}{\sum_{\mathbf{s}} N_{i',I}(\mathbf{s})} \quad (2)$$

$$\theta_{\text{lex}}(f|e) \leftarrow \frac{\sum_{\mathbf{s}} N_{f|e}(\mathbf{s})}{\sum_{\mathbf{s}} n_e(\mathbf{s})} \quad (3)$$

where $N_{i',I}(\mathbf{s})$ is the marginal number of times $e_{i'}$ is aligned to some foreign word if the length of $\mathbf{e}$ is I, or 0 otherwise; $N_{i|i',I}(\mathbf{s})$ is the marginal number of times the next alignment position after i' is i in $\mathbf{a}$ if the length of $\mathbf{e}$ is I, or 0 otherwise; $n_e(\mathbf{s})$ is the count of e in $\mathbf{e}$; $N_{f|e}(\mathbf{s}, \mathbf{a})$ is the marginal number of times e is aligned to f.

3.2 Leave-one-out EM for IBM Models 1, 2 and HMM Model

Leave-one-out EM for WA differs from standard EM in the way the alignment and translation probabilities are calculated. Each sentence pair will have its own alignment and translation probability models calculated by excluding the sentence pair itself. More formally, leave-one-out EM for WA are formulated as follows,

Leave-one-out E step: employing leave-one-out models for each $\mathbf{s}$ to calculate the conditional probability of alignments

$$P(\mathbf{a}|\mathbf{s}, \theta_{\bar{\mathbf{s}}}) = \prod_{j=1}^{J} \theta_{\text{ali}}^{\bar{\mathbf{s}}}(a_j|a_{j-1}, I) \theta_{\text{lex}}^{\bar{\mathbf{s}}}(f_j|e_{a_j}) \quad (4)$$

where $\theta_{\text{ali}}^{\bar{\mathbf{s}}}(i|i', I)$ and $\theta_{\text{lex}}^{\bar{\mathbf{s}}}(f_j|e_{a_j})$ are the leave-one-out alignment probability and translation probability, respectively.

Leave-one-out M step: re-estimating leave-one-out probability models,

$$\theta_{\text{ali}}^{\bar{\mathbf{s}}}(i|i', I) \leftarrow \frac{\sum_{\mathbf{s}' \neq \mathbf{s}} N_{i|i',I}(\mathbf{s}')}{\sum_{\mathbf{s}' \neq \mathbf{s}} N_{i',I}(\mathbf{s}')} \quad (5)$$

$$\theta_{\text{lex}}^{\bar{\mathbf{s}}}(f|e) \leftarrow \frac{\sum_{\mathbf{s}' \neq \mathbf{s}} N_{f|e}(\mathbf{s}')}{\sum_{\mathbf{s}' \neq \mathbf{s}} n_e(\mathbf{s}')}. \quad (6)$$

3.3 Standard EM for IBM Model 4

The framework of the standard EM for IBM Model 4 is similar with the one for IBM Models 1, 2 and HMM Model, but the calculation of alignment probability is more complicated.

E step: calculating the conditional probability through the reverted alignment (Och and Ney, 2003),

$$P(\mathbf{a}|\mathbf{s}, \theta) = P(B_0|B_1, \ldots, B_I) \cdot \prod_{i=1}^{I} P(B_i|B_{i-1}, e_i) \cdot \prod_{i=1}^{I} \prod_{j \in B_i} \theta_{\text{lex}}(f_j|e_i), \quad (7)$$

where B_0 means the set of foreign words aligned with the empty word; $P(B_0|B_1, \ldots, B_I)$ is assumed to be a binomial distribution for the size of B_0 (Brown et al., 1993b) or an modified distribution to relieve deficiency (Och and Ney, 2003).

The distribution $P(B_i|B_{i-1}, e_i)$ is decomposed as

$$P(B_i|B_{i-1}, e_i) = \theta_{\text{fer}}(\phi_i|e_i) \cdot \theta_{\text{hea}}(B_{i,1} - \overline{B_{\rho_i}}|E_{\rho_i}) \cdot \prod_{k=2}^{\phi_i} \theta_{\text{oth}}(B_{i,k} - B_{i,k-1}), \quad (8)$$

where θ_{fer} is a fertility model; θ_{hea} is a probability model for the head (first) aligned foreign word; θ_{oth} is a probability model for the other aligned foreign words. θ_{hea} is assumed to be conditioned

on the word class E_{ρ_i}, following the paper of (Och and Ney, 2003) and the implementation of GIZA++ and CICADA.

M step: re-estimating the probability models,

$$\theta_{\text{fer}}(\phi|e) \leftarrow \frac{\sum_{\mathbf{s}} N_{\phi|e}(\mathbf{s})}{\sum_{\mathbf{s}} \sum_{\phi'} N_{\phi'|e}(\mathbf{s})} \quad (9)$$

$$\theta_{\text{hea}}(\Delta i|E) \leftarrow \frac{\sum_{\mathbf{s}} N^{\text{hea}}_{\Delta i|E}(\mathbf{s})}{\sum_{\mathbf{s}} \sum_{\Delta i'} N^{\text{hea}}_{\Delta i'|E}(\mathbf{s})} \quad (10)$$

$$\theta_{\text{oth}}(\Delta i) \leftarrow \frac{\sum_{\mathbf{s}} N^{\text{oth}}_{\Delta i}(\mathbf{s})}{\sum_{\mathbf{s}} \sum_{\Delta i'} N^{\text{oth}}_{\Delta i'}(\mathbf{s})}, \quad (11)$$

where Δi is a difference of the indexes of two foreign words.

3.4 Leave-one-out EM for IBM Model 4

The leave-one-out treatment were applied to the three component probability models θ_{fer}, θ_{hea} and θ_{oth} of IBM model 4.

Leave-one-out E step: calculating the conditional probability through leave-one-out probability models

$$P(\mathbf{a}|\mathbf{s}, \theta^{\bar{\mathbf{s}}}) = P(B_0|B_1, \ldots, B_I) \cdot \prod_{i=1}^{I} P^{\bar{\mathbf{s}}}(B_i|B_{i-1}, e_i) \cdot \prod_{i=1}^{I} \prod_{j \in B_i} \theta^{\bar{\mathbf{s}}}_{\text{lex}}(f_j|e_i), \quad (12)$$

$$P^{\bar{\mathbf{s}}}(B_i|B_{i-1}, e_i) = \theta^{\bar{\mathbf{s}}}_{\text{fer}}(\phi_i|e_i) \cdot \theta^{\bar{\mathbf{s}}}_{\text{hea}}(B_{i,1} - \overline{B_{\rho_i}}|E_{\rho_i}) \cdot \prod_{k=2}^{\phi_i} \theta^{\bar{\mathbf{s}}}_{\text{oth}}(B_{i,k} - B_{i,k-1}). \quad (13)$$

Leave-one-out M step: re-estimating the leave-one-out probability models,

$$\theta^{\bar{\mathbf{s}}}_{\text{fer}}(\phi|e) \leftarrow \frac{\sum_{\mathbf{s}' \neq \mathbf{s}} N_{\phi|e}(\mathbf{s}')}{\sum_{\mathbf{s}' \neq \mathbf{s}} \sum_{\phi'} N_{\phi'|e}(\mathbf{s}')} \quad (14)$$

$$\theta^{\bar{\mathbf{s}}}_{\text{hea}}(\Delta i|E) \leftarrow \frac{\sum_{\mathbf{s}' \neq \mathbf{s}} N^{\text{hea}}_{\Delta i|E}(\mathbf{s}')}{\sum_{\mathbf{s}' \neq \mathbf{s}} \sum_{\Delta i'} N^{\text{hea}}_{\Delta i'|E}(\mathbf{s}')} \quad (15)$$

$$\theta^{\bar{\mathbf{s}}}_{\text{oth}}(\Delta i) \leftarrow \frac{\sum_{\mathbf{s}' \neq \mathbf{s}} N^{\text{oth}}_{\Delta i}(\mathbf{s}')}{\sum_{\mathbf{s}' \neq \mathbf{s}} \sum_{\Delta i'} N^{\text{oth}}_{\Delta i'}(\mathbf{s}')}. \quad (16)$$

3.5 Handling Singletons

Singletons are the words that occur only once in corpora. Singletons cause problems when applying leave-one-out to lexicalized models such as the translation model $\theta^{\bar{\mathbf{s}}}_{\text{lex}}$ and the fertility model $\theta^{\bar{\mathbf{s}}}_{\text{fer}}$. When calculating (6) and (14) for singletons, the denominators become zero, thus the probabilities are undefined.

For singletons, there is no prior information to guide their alignment, so we back off to uniform distributions. In that case, the alignments are primarily determined by the rest of the sentence.

In addition, singletons can be in the target side of the translation model $\theta^{\bar{\mathbf{s}}}_{\text{lex}}$. In that case, the probabilities become zero. This is handled by setting a minimum probability value of 1.0×10^{-12}, which was decided by pilot experiments.

3.6 Implementation Details

To alleviate memory requirements and increase speed, our implementation did not build or store the local alignment models explicitly for each sentence pair. The following formula was used to efficiently calculate (5), (6) and (14–16) to build temporary probability models,

$$\sum_{\mathbf{s}' \neq \mathbf{s}} N_x(\mathbf{s}') = (\sum_{\mathbf{s}'} N_x(\mathbf{s}')) - N_x(\mathbf{s}), \quad (17)$$

where x is a alignment event. Our implementation maintained global counts of all alignment events $\sum_{\mathbf{s}'} N_x(\mathbf{s}')$, and (considerably smaller) local counts $N_x(\mathbf{s})$ from each sentence pair $\mathbf{s}$.

Take the translation model $\theta^{\bar{\mathbf{s}}}_{\text{lex}}$ for example. For a sentence pair $\mathbf{s} = (f_1 \ldots f_J, e_1 \ldots e_I)$, it is caculated as,

$$\theta^{\bar{\mathbf{s}}}_{\text{lex}}(f_j|e_i) = \frac{(\sum_{\mathbf{s}'} N_{(f_j|e_i)}(\mathbf{s}')) - N_{(f_j|e_i)}(\mathbf{s})}{(\sum_{\mathbf{s}'} n_{e_i}(\mathbf{s}')) - n_{e_i}(\mathbf{s})}. \quad (18)$$

The global counts to be maintained are $\sum_{\mathbf{s}'} N_{(f_j|e_i)}(\mathbf{s}')$ and $n_{e_i}(\mathbf{s}')$, and the local counts are $\sum_{\mathbf{s}} N_{(f_j|e_i)}(\mathbf{s})$ and $n_{e_i}(\mathbf{s})$. Therefore the memory cost is,

$$|\mathcal{E}| \cdot (|\mathcal{F}| + 1) + \sum_{\mathbf{s}} I_{\mathbf{s}}(J_{\mathbf{s}} + 1), \quad (19)$$

where $|\mathcal{E}|$ is the size of English vocabulary, $|\mathcal{F}|$ is the size of foreign language vocabulary, $I_{\mathbf{s}}$ is the length of the English sentence of $\mathbf{s}$, and $J_{\mathbf{s}}$ is the length of the foreign sentence of $\mathbf{s}$.

The calculation of the leave-one-out translation model is performed for each English word and foreign word in $\mathbf{s}$. Therefore, the time cost is,

$$\sum_{\mathbf{s}} I_{\mathbf{s}}(J_{\mathbf{s}} + 1). \quad (20)$$

In addition, because the local counts $N_{(f_j|e_i)}(\mathbf{s})$ and $n_{e_i}(\mathbf{s})$ are read in order, storing them in a external memory such as a hard disk will not slow down the running speed much. This will reduce the memory cost to

$$|\mathcal{E}| \cdot (|\mathcal{F}| + 1). \quad (21)$$

This cost is independent to the number of sentence pairs[4].

The speed of the proposed method can be boosted through parallelism. These calculations on each sentence pair can be performed independently. We found empirically that when our implementation of the proposed method is run on a 16-core computer, it finishes the task earlier than GIZA++[5].

4 Experiments

The proposed WA method was tested on two language pairs: Chinese-English and Japanese-English (Table 2). Performance was measured both directly using the agreement with reference to manual WA annotations, and indirectly using the BLEU score in end-to-end machine translation tasks. GIZA++ and our own implementation of standard EM were used as baselines.

4.1 Experimental Settings

The Chinese-English experimental data consisted of the GALE WA corpus and the OpenMT corpus. They are from the same domain, both contain newswire texts and web blogs. The OpenMT evaluation 2005 was used as a development set for MERT tuning (Och, 2003), and the OpenMT evaluation 2006 was used as a test set. The Japanese-English experimental data was the Kyoto Free Translation Task (Neubig, 2011)[6]. The corpus contains a set of 1,235 sentence pairs that are manually word aligned.

The corpora were processed using a standard procedure for machine translation. The English texts were tokenized with the tokenization script released with Europarl corpus (Koehn, 2005) and converted to lowercase; the Chinese texts were segmented into words using the Stanford Word Segmenter (Xue et al., 2002)[7]; the Japanese texts were segmented into words using the Kyoto Text Analysis Toolkit (KyTea[8]). Sentences longer than 100 words or those with foreign/English word length ratios between larger than 9 were filtered out.

GIZA++ was run with the default Moses settings (Koehn et al., 2007). The IBM model 1, HMM model, IBM model 3 and IBM model 4 were run with 5, 5, 3 and 3 iterations. We implemented the proposed leave-one-out EM and standard EM in IBM model 1, HMM model and IBM model 4. In the original work (Och and Ney, 2003) this combination of models achieved comparable performance to the default Moses settings. They were run with 5, 5 and 6 iterations.

The standard EM was re-implemented as a baseline to provide a solid basis for comparison, because GIZA++ contains many undocumented details. Our implementation is based on the toolkit of CICADA (Watanabe and Sumita, 2011; Watanabe, 2012; Tamura et al., 2013)[9]. We named the implemented aligner AGRIPPA, to support our in-house decoders OCTAVIAN and AUGUSTUS.

In all experiments, WA was performed independently in two directions: from foreign languages to English, and from English to foreign languages. Then the grow-diag-final-and heuristic was used to combine the two alignments from both directions to yield the final alignments for evaluation (Och and Ney, 2000; Och and Ney, 2003).

4.2 Word Alignment Accuracy

Word alignment accuracy of the baseline and the proposed method is shown in Table 3 in terms of precision, recall and F_1 (Och and Ney, 2003). The proposed method gave rise to higher quality alignments in all our experiments. The improvement in F_1, precision and recall based on IBM Model 4 is in the range 8.3% to 9.1% compared with the GIZA++ baseline, and in the range 5.0% to 17.2% compared with our own baseline.

The most meaningful result comes from the comparison of the models trained using standard EM log-likelihood training, and the proposed EM leave-one-out log-likelihood training. These models are identical except for way in which the model likelihood is calculated. In all our experiments the proposed method gave rise to higher quality alignments. The standard EM implementation achieved

[4] We found the memory of our server is large enough, so we did not implement it

[5] We plan to make our code public available.

[6] http://www.phontron.com/kftt/

[7] http://nlp.stanford.edu/software/segmenter.shtml

[8] http://www.phontron.com/kytea/

[9] http://www2.nict.go.jp/univ-com/multi_trans/cicada/

Corpus	# Sent. pairs	# Foreign Words	# English Words
Chinese-English (GALE WA, OpenMT)			
WA	18,057	392,447	518,137
Train	442,967	12,265,072	13,444,927
Eval. 05	1,082†	29,688	138,952
Eval. 06	1,664†	37,827	189,059
Japanese-English (Kyoto Free Translation)			
WA	1,235	34,403	30,822
Train	329,882	6,085,131	5,911,486
Develop	1,166	26,856	24,309
Test	1,160	28,501	26,734

Table 2: Experimental Data. † Each consists of one foreign sentence and four English reference sentences.

Models	standard EM (GIZA++)			standard EM (ours)			Leave-one-out(prop.)		
	F_1	P	R	F_1	P	R	F_1	P	R
Chinese-English (GALE WA, OpenMT)									
Model 1	0.498	0.656	0.401	0.518	0.670	0.423	**0.553**	**0.689**	**0.461**
HMM	0.584	0.720	0.491	0.593	0.722	0.503	**0.665**	**0.774**	**0.583**
Model 4	0.624	0.698	0.565	0.593	0.688	0.522	**0.677**	**0.756**	**0.612**
Japanese-English (Kyoto Free Translation)									
Model 1	0.508	0.601	0.439	0.513	0.606	0.444	**0.535**	**0.618**	**0.471**
HMM	0.573	0.667	0.502	0.579	0.665	0.512	**0.626**	**0.687**	**0.575**
Model 4	0.577	0.594	0.561	0.570	0.617	0.530	**0.628**	**0.648**	**0.609**

Table 3: Word alignment accuracy measured by F_1, precision and recall.

alignment performance approximately comparable to GIZA++, whereas the proposed method exceeded the performance of both implementations.

4.3 End-to-end Translation Quality

BLEU scores achieved by the phrase-based and hierachical SMT systems[10] which were trained from different alignment results, are shown in Table 4. Each experiment was conducted three times to mitigate the variance in the results due to MERT. The results show that the proposed alignment method achieved the highest BLEU score in all experiments. The improvement over the baseline is in range 0.03 to 1.03 for phrase-based systems, and ranged from 0.43 to 1.30 for hierarchical systems.

Hierarchical systems benifit more from the proposed method than phrase-based systems. We think this is because that hierarchical systems are more sensitive to word alignment quality than phrase-based systems. Phrase-based systems only

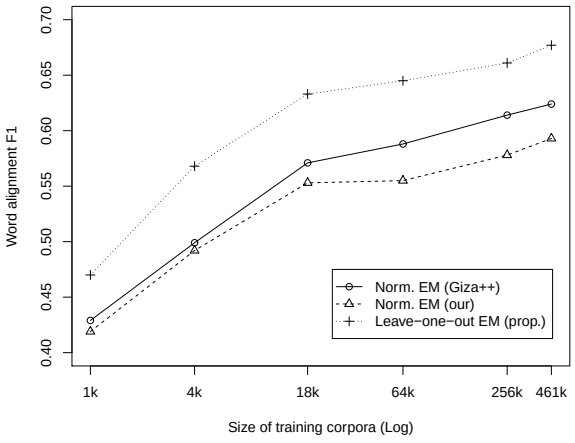


Figure 2: Curve of word alignment accuracy (F_1) under training corpora of different sizes.

[10] from the Moses toolkit

SMT Systems	standard EM (GIZA++)	standard EM (ours)	Leave-one-out (prop.)
Chinese-English (GALE WA, OpenMT)			
Phrase-based	31.85 ± 0.26	31.01 ± 0.18	$\mathbf{32.04 \pm 0.08}$
Hierarchical	32.27 ± 0.23	31.40 ± 0.26	$\mathbf{32.70 \pm 0.14}$
Japanese-English (Kyoto Free Translation)			
Phrase-based	18.35 ± 0.27	18.20 ± 0.20	$\mathbf{18.38 \pm 0.11}$
Hierarchical	19.48 ± 0.08	19.39 ± 0.02	$\mathbf{20.10 \pm 0.07}$

Table 4: End-to-end translation quality measured by BLEU

Corpus size	standard EM (GIZA++)			standard EM (ours)			Leave-one-out(prop.)		
	F_1	P	R	F_1	P	R	F_1	P	R
1K	0.429	0.466	0.397	0.419	0.463	0.382	**0.470**	**0.568**	**0.402**
4K	0.499	0.547	0.459	0.492	0.549	0.445	**0.568**	**0.668**	**0.494**
18K†	0.571	0.630	0.521	0.553	0.621	0.499	**0.633**	**0.721**	**0.565**
64K	0.588	0.659	0.531	0.555	0.638	0.492	**0.645**	**0.712**	**0.590**
256K	0.614	0.687	0.554	0.578	0.667	0.511	**0.661**	**0.718**	**0.612**
461K	0.624	0.698	0.565	0.593	0.688	0.522	**0.677**	**0.756**	**0.612**

Table 5: Effect of training corpus size on word alignment accuracy measured by F_1, precision and recall (Chinese-English). † the whole manually word aligned corpus

Corpus size	stan.(GIZA++)	stan.(ours)	LOO(prop.)	Gold
Phrase-based				
1k	7.86	7.66	9.38	**10.01**
4k	15.27	15.49	17.06	**17.57**
18K†	22.15	21.72	**24.41**	24.11
64K	28.10	27.91	**29.23**	NA
256K	31.05	30.82	**31.51**	NA
461K	31.85	31.01	**32.04**	NA
Hierarchical				
1k	7.53	7.54	9.19	**10.62**
4k	14.89	15.51	17.91	**18.31**
18K†	22.85	22.56	**24.66**	24.52
64K	28.82	28.22	**29.78**	NA
256K	31.47	30.21	**31.72**	NA
461K	32.27	31.04	**32.70**	NA

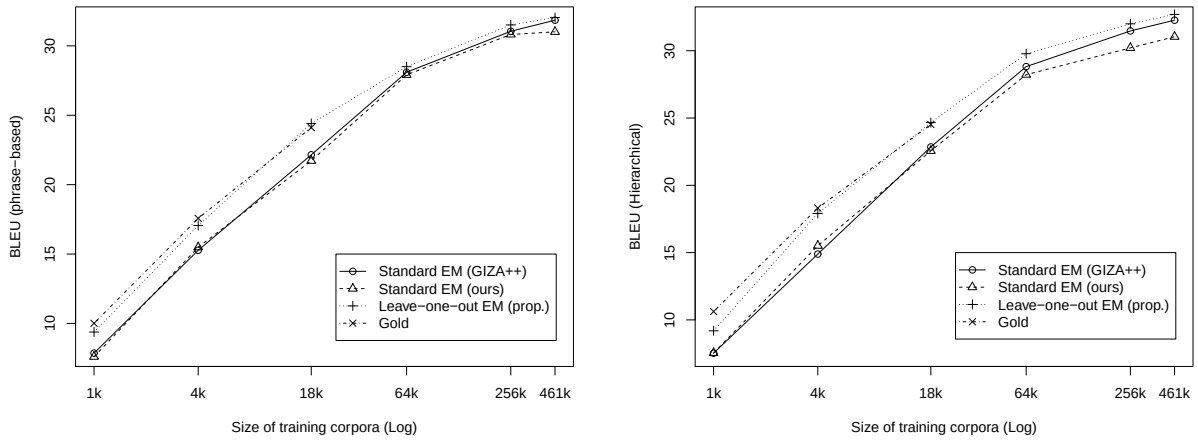
Table 6: Effect of training corpus size on end-to-end translation quality measured by BLEU (Chinese-English). † the whole manually word aligned corpus

take contiguous parallel phrase pairs as translation rules, while hierarchical systems also use patterns made by subtracting (inner) short parallel phrases from (outer) longer parallel phrases. Both the outer and inner phrases typically need to be noise-free in order to produce high quality rules. This puts a high demand on the alignment quality.

4.4 Effect of Training Corpus Size

Training corpora of different sizes were employed to perform unsupervised WA experiments and MT experiments (see Tables 5 and 6).

The training corpora were randomly sampled from the Chinese-English manual WA corpora and the parallel training corpus. The manual WA corpus has a priority for being sampled so that the gold WA annotation is available for MT experi-



(a) (b)

Figure 3: Curves of translation quality (BLEU) under training corpora of different sizes. (a) Phrase-based MT; (b) Hierarchical MT.

ments.

The settings of the unsupervised WA experiments and the MT experiments are the same with the previous experiments. In the WA experiments, GIZA++, our implemented standard EM and the proposed leave-one-out EM are applied to training corpora with the same parameter settings as the previous. In the MT experiments, the WA results of different methods and the gold WA (if available) are employed to extract translation rules; the rest settings including language models, development and test corpus, and parameters are the same as the previous.

On word alignment accuracy, the proposed method achieved improvements of F_1 from 0.041 to 0.090 under the different training corpora (Table 5. The maximum improvement compared with GIZA++ is 0.069 when the training corpus has 4,000 sentence pairs. The maximum improvement compared with our own implement is 0.090 when the training corpus has 64,000 sentence pairs.

Figure 2 shows that the extent of improvements slightly changes under different training corpora, but they are all quite stable and obvious.

On translation quality, the proposed method achieved improvements of BLEU under the different training corpora. The improvements ranged from 0.19 to 1.72 for phrase-based MT and ranged from 0.25 to 3.02 (see Table 5). The improvements are larger under smaller training corpora (see Figure 3).

In addition, the BLEUs achieved by the proposed method is close to the ones achieved by gold WA annotations. The proposed method slightly outperforms the gold WA annotations when using the full manual WA corpus of 18,057 sentence pairs.

4.5 Comparison to l_0-Normalization and Kneser-Ney Smoothing Methods

The proposed leave-one-word word alignment method was empirically compared to l_0-normalized GIZA++ (Vaswani et al., 2012)[11] and Kneser-Ney smoothed GIZA++ (Zhang and Chiang, 2014)[12]. l_0-normalization and Kneser-Ney smoothing methods are established methods to overcome the sparse problem. This enables the probability distributions on rare words to be estimated more effectively. In this way, these two GIZA++ variants are related to the proposed method.

l_0-normalized GIZA++ and Kneser-Ney smoothed GIZA++ were run with the same settings as GIZA++, which came from the default settings of MOSES. For the settings of l_0-normalized GIZA++ that are not in common with GIZA++ were the default settings. As for Kneser-Ney smoothed GIZA++, the smooth switches of IBM models 1 – 4 and HMM model

[11] http://www.isi.edu/~avaswani/giza-pp-l0.html

[12] https://github.com/hznlp/giza-kn

	GIZA++			l_0-Normalization			Kneser-Ney Smooth.			Leave-one-out(prop.)		
Word Alignment Quality												
	F_1	P	R	F_1	P	R	F_1	P	R	F_1	P	R
All Words	0.624	0.698	0.565	0.629	0.700	0.571	0.656	0.726	0.599	**0.678**	**0.755**	**0.615**
S.W.F=1	0.458	0.435	0.483	0.448	0.471	0.427	**0.515**	0.532	**0.499**	0.398	**0.693**	0.279
S.W.F$\leq$2	0.466	0.451	0.481	0.461	0.485	0.440	**0.522**	0.545	**0.501**	0.450	**0.707**	0.330
S.W.F$\leq$5	0.476	0.480	0.473	0.478	0.509	0.451	**0.534**	0.572	**0.501**	0.502	**0.722**	0.385
S.W.F$\leq$10	0.485	0.505	0.466	0.491	0.531	0.456	**0.541**	0.593	**0.498**	0.529	**0.733**	0.414
Translation Quality (BLEU)												
Phrase-based	31.85 ± 0.26			31.52 ± 0.06			31.94 ± 0.19			**32.04 ± 0.08**		
Hierarchical	32.27 ± 0.23			32.20 ± 0.04			32.47 ± 0.33			**32.70 ± 0.14**		

Table 7: Empirical Comparision with l_0-Normalized and Kneser-Ney Smoothed GIZA++'s

were turned on.

The experimental results are presented in Table 7. The experiments were run on the Chinese-English language pair. The word alignment quality was evaluated separately for all words and for various levels of rare words. The leave-one-out method outperformed related methods in terms of precision, recall and F_1 when evaluated on all words.

Rare words were categorized based on the number of occurences in the source-language text of the training data. The evaluations were carried out on the subset of alignment links that had a rare word on the source side. Table 7 presents the results for thresholds 1, 2, 5 and 10. The proposed method achieved much higher precision on rare words than the other methods, but performed poorly on recall. The Kneser-Ney Smoothed GIZA++ had higher recall. The explanation might be that the leave-one-out method punishes rare words more than the Kneser-Ney smoothing method, by totally removing the derived expected counts of current sentence pair from the alignment models. This leads to rare words being passively aligned. In other words, the leave-one-out method would align rare words unless the confidence is high. Therefore, we plan to seek a method to integrate Kneser-Ney smoothing into the proposed leave-one-out method in the future work.

The BLEU scores achieved by phrase-based SMT and hierarchical SMT for different alignment methods are presented in Table 7. The proposed method outperforms the other methods. The Kneser-Ney Smoothed GIZA++ performed the second best. We tried to further analyze the relation between word alignment and BLEU, but found the analysis was obscured by the many processing stages. These stages include parallel phrase extraction (or translation rule extraction from hierarchical SMT), log-linear model, MERT tuning and practical decoding where a lot of pruning happened.

5 Conclusion

This paper proposes a leave-one-out EM algorithm for WA to overcome the over-fitting problem that occurs when using standard EM for WA. The experimental results on Chinese-English and Japanese-English corpora show that both the WA accuracy and the end-to-end translation are improved.

In addition, we have a interesting finding about the effect of manual WA annotations on training MT systems. In a Chinese-English parallel training corpus of 18,057 sentence pairs, the manual WA annotation outperformed the unsupervised WA results produced by standard EM algorithms. However, the unsupervised WA results produced by proposed leave-one-out EM algorithm outperformed the manual WA annotation.

Our future work will focus on increasing the gains in end-to-end translation quality through the proposed leave-one-out aligner. It is a interesting question why GIZA++ achieved competitive BLEU scores though its alignment accuracy measured by F_1 was substantially lower. The answer to this question which may reveal essence of good word alignment for MT and eventually help to improve MT. In addition, we plan to improve the proposed method by integrating Kneser-Ney smoothing.

Acknowledgments

We appreciated the valuable comments from the reviewers.

References

Christophe Andrieu, Nando De Freitas, Arnaud Doucet, and Michael I. Jordan. 2003. An introduction to MCMC for machine learning. *Machine learning*, 50(1-2):5–43.

Ondrej Bojar, Christian Buck, Chris Callison-Burch, Christian Federmann, Barry Haddow, Philipp Koehn, Christof Monz, Matt Post, Radu Soricut, and Lucia Specia. 2013. Findings of the 2013 workshop on statistical machine translation. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 1–44.

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, Meredith J. Goldsmith, Jan Hajic, Robert L. Mercer, and Surya Mohanty. 1993a. But dictionaries are data too. In *Proceedings of the Workshop on Human Language Technology*, HLT '93, pages 202–205, Stroudsburg, PA, USA. Association for Computational Linguistics.

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

Mauro Cettolo, Jan Niehues, Sebastian Stüker, Luisa Bentivogli, and Marcello Federico. 2013. Report on the 10th IWSLT evaluation campaign. In *Proceedings of the International Workshop on Spoken Language Translation*, pages 29–38.

Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 1–38.

John DeNero and Dan Klein. 2007. Tailoring word alignments to syntactic machine translation. In *Proceedings of the 45th Annual Meeting on Association for Computational Linguistics*, pages 17–24.

John DeNero, Alexandre Bouchard-Côté, and Dan Klein. 2008. Sampling alignment structure under a bayesian translation model. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 314–323, Stroudsburg, PA, USA. Association for Computational Linguistics.

Chris Dyer, Victor Chahuneau, and Noah A Smith. 2013. A simple, fast, and effective reparameterization of ibm model 2. In *Proceedings of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 644–648.

Kuzman Ganchev, Joao V. Graça, and Ben Taskar. 2008. Better alignments = better translations? *Proceedings of the 46th Annual Meeting on Association for Computational Linguistics*, page 42.

Walter R. Gilks, Sylvia Richardson, and David J. Spiegelhalter. 1996. *Markov chain Monte Carlo in practice*, volume 2. CRC press.

Isao Goto, Bin Lu, Ka Po Chow, Eiichiro Sumita, and Benjamin K Tsou. 2011. Overview of the patent machine translation task at the NTCIR-9 workshop. In *Proceedings of NTCIR*, volume 9, pages 559–578.

Marcin Junczys-Dowmunt and Arkadiusz Sza. 2012. Symgiza++: Symmetrized word alignment models for machine translation. In Pascal Bouvry, Mieczyslaw A. Klopotek, Franck Leprvost, Malgorzata Marciniak, Agnieszka Mykowiecka, and Henryk Rybinski, editors, *Security and Intelligent Information Systems (SIIS)*, volume 7053 of *Lecture Notes in Computer Science*, pages 379–390, Warsaw, Poland. Springer.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics on Interactive Poster and Demonstration Sessions*, pages 177–180. Association for Computational Linguistics.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *Proceedings of MT Summit*, volume 5, pages 79–86.

Percy Liang, Ben Taskar, and Dan Klein. 2006. Alignment by agreement. In *Proceedings of the North American Chapter of the Association of Computational Linguistics: Human Language Technologies*, pages 104–111. Association for Computational Linguistics.

Robert C. Moore. 2004. Improving IBM word-alignment model 1. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 518. Association for Computational Linguistics.

Graham Neubig, Taro Watanabe, Eiichiro Sumita, Shinsuke Mori, and Tatsuya Kawahara. 2011. An unsupervised model for joint phrase alignment and extraction. In *ACL*, pages 632–641.

Graham Neubig. 2011. The Kyoto free translation task. http://www.phontron.com/kftt.

Franz Josef Och and Hermann Ney. 2000. A comparison of alignment models for statistical machine translation. In *Proceedings of the 18th conference on Computational linguistics-Volume 2*, pages 1086–1090. Association for Computational Linguistics.

Franz Josef Och and Hermann Ney. 2003. A Systematic Comparison of Various Statistical Alignment Models. *Computational Linguistics*, 29(1):19–51.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 160–167. Association for Computational Linguistics.

Nicolas Le Roux and Francis Bach. 2011. Local component analysis. Technical report.

Akihiro Tamura, Taro Watanabe, Eiichiro Sumita, Hiroya Takamura, and Manabu Okumura. 2013. Part-of-speech induction in dependency trees for statistical machine translation. In *Proceedings of the 51th Annual Meeting of the Association for Computational Linguistics*, pages 841–851.

João V Graça, Kuzman Ganchev, and Ben Taskar. 2010. Learning tractable word alignment models with complex constraints. *Computational Linguistics*, 36(3):481–504.

Ashish Vaswani, Liang Huang, and David Chiang. 2012. Smaller alignment models for better translations: unsupervised word alignment with the l_0-norm. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 311–319. Association for Computational Linguistics.

Xiaolin Wang, Masao Utiyama, Andrew Finch, and Eiichiro Sumita. 2014. Empirical study of unsupervised chinese word segmentation methods for smt on large-scale corpora. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 752–758, Baltimore, Maryland, June. Association for Computational Linguistics.

Taro Watanabe and Eiichiro Sumita. 2011. Machine translation system combination by confusion forest. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1249–1257. Association for Computational Linguistics.

Taro Watanabe. 2012. Optimized online rank learning for machine translation. In *Proceedings of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 253–262. Association for Computational Linguistics.

Joern Wuebker, Arne Mauser, and Hermann Ney. 2010. Training phrase translation models with leaving-one-out. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 475–484. Association for Computational Linguistics.

Joern Wuebker, Mei-Yuh Hwang, and Chris Quirk. 2012. Leave-one-out phrase model training for large-scale deployment. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 460–467. Association for Computational Linguistics.

Nianwen Xue, Fu-Dong Chiou, and Martha Palmer. 2002. Building a large-scale annotated chinese corpus. In *Proceedings of the 19th International Conference on Computational Linguistics*, pages 1–8. Association for Computational Linguistics.

Hui Zhang and David Chiang. 2014. Kneser-ney smoothing on expected counts. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 765–774, Baltimore, Maryland, June. Association for Computational Linguistics.

Generalized Agreement for Bidirectional Word Alignment

Chunyang Liu[†], **Yang Liu**[†*], **Huanbo Luan**[†], **Maosong Sun**[†], and **Heng Yu**[‡]
[†] State Key Laboratory of Intelligent Technology and Systems
Tsinghua National Laboratory for Information Science and Technology
Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China
[‡] Samsung R&D Institute of China, Beijing 100028, China
{liuchunyang2012,liuyang.china,luanhuanbo}@gmail.com, sms@tsinghua.edu.cn
h0517.yu@samsung.com

Abstract

While agreement-based joint training has proven to deliver state-of-the-art alignment accuracy, the produced word alignments are usually restricted to one-to-one mappings because of the hard constraint on agreement. We propose a general framework to allow for arbitrary loss functions that measure the disagreement between asymmetric alignments. The loss functions can not only be defined between asymmetric alignments but also between alignments and other latent structures such as phrase segmentations. We use a Viterbi EM algorithm to train the joint model since the inference is intractable. Experiments on Chinese-English translation show that joint training with generalized agreement achieves significant improvements over two state-of-the-art alignment methods.

1 Introduction

Word alignment is a natural language processing task that aims to specify the correspondence between words in two languages (Brown et al., 1993). It plays an important role in statistical machine translation (SMT) as word-aligned bilingual corpora serve as the input of translation rule extraction (Koehn et al., 2003; Chiang, 2007; Galley et al., 2006; Liu et al., 2006).

Although state-of-the-art generative alignment models (Brown et al., 1993; Vogel et al., 1996) have been widely used in practical SMT systems, they fail to model the symmetry of word alignment. While word alignments in real-world bilingual data usually exhibit complicated mappings (i.e., mixed with one-to-one, one-to-many, many-to-one, and many-to-many links), these models assume that each target word is aligned to exactly one source word. To alleviate this problem, heuristic methods (e.g., grow-diag-final) have been proposed to combine two asymmetric alignments (source-to-target and target-to-source) to generate symmetric bidirectional alignments (Och and Ney, 2003; Koehn and Hoang, 2007).

Instead of using heuristic symmetrization, Liang et al. (2006) introduce a principled approach that encourages the agreement between asymmetric alignments in two directions. The basic idea is to favor links on which both unidirectional models agree. They associate two models via the agreement constraint and show that agreement-based joint training improves alignment accuracy significantly.

However, enforcing agreement in joint training faces a major problem: the two models are restricted to one-to-one alignments (Liang et al., 2006). This significantly limits the translation accuracy, especially for distantly-related language pairs such as Chinese-English (see Section 5). Although posterior decoding can potentially address this problem, Liang et al. (2006) find that many-to-many alignments occur infrequently because posteriors are sharply peaked around the Viterbi alignments. We believe that this happens because their model imposes a *hard* constraint on agreement: the two models must share the same alignment when estimating the parameters by calculating the products of alignment posteriors (see Section 2).

In this work, we propose a general framework for imposing agreement constraints in joint training of unidirectional models. The central idea is to use the expectation of a loss function, which measures the disagreement between two models, to replace the original probability of agreement. This allows for many possible ways to quantify agreement. Experiments on Chinese-English translation show that our approach outperforms two state-of-the-art baselines significantly.

*Corresponding author: Yang Liu.

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1828–1836,
Lisbon, Portugal, 17-21 September 2015.

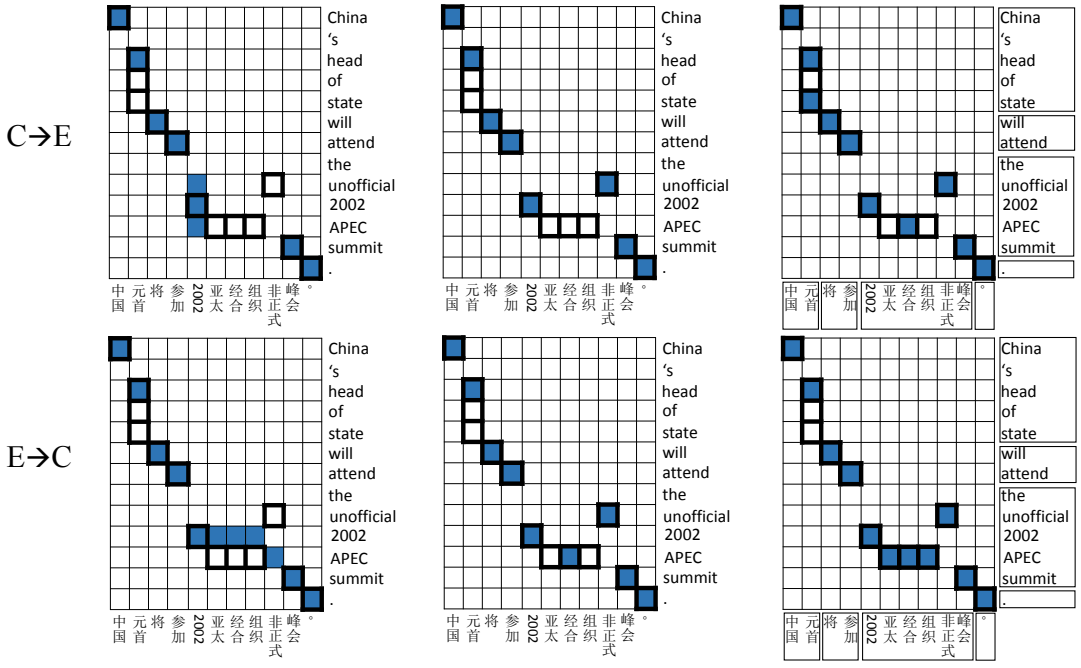


Figure 1: Comparison of (a) independent training without agreement, (b) joint training with agreement, and (c) joint training with generalized agreement. Bold squares are gold-standard links and solid squares are model predictions. The Chinese and English sentences are segmented into phrases in (c). Joint training with agreement achieves a high precision but generally only produces one-to-one alignments. We propose generalized agreement to account for not only the consensus between asymmetric alignments, but also the conformity of alignments to other latent structures such as phrase segmentations.

2 Background

2.1 Asymmetric Alignment Models

Given a source-language sentence $\mathbf{e} \equiv e_1^I = e_1, \ldots, e_I$ and a target-language sentence $\mathbf{f} \equiv f_1^J = f_1, \ldots, f_J$, a source-to-target translation model (Brown et al., 1993; Vogel et al., 1996) can be defined as

$$P(\mathbf{f}|\mathbf{e};\boldsymbol{\theta}_1) = \sum_{\mathbf{a}_1} P(\mathbf{f}, \mathbf{a}_1|\mathbf{e};\boldsymbol{\theta}_1) \quad (1)$$

where $\mathbf{a}_1$ denotes the source-to-target alignment and $\boldsymbol{\theta}_1$ is the set of source-to-target translation model parameters.

Likewise, the target-to-source translation model is given by

$$P(\mathbf{e}|\mathbf{f};\boldsymbol{\theta}_2) = \sum_{\mathbf{a}_2} P(\mathbf{e}, \mathbf{a}_2|\mathbf{f};\boldsymbol{\theta}_2) \quad (2)$$

where $\mathbf{a}_2$ denotes the target-to-source alignment and $\boldsymbol{\theta}_2$ is the set of target-to-source translation model parameters.

Given a training set $D = \{\langle \mathbf{f}^{(s)}, \mathbf{e}^{(s)} \rangle\}_{s=1}^S$, the two models are trained independently to maximize the log-likelihood of the training data for each direction, respectively:

$$\mathcal{L}(\boldsymbol{\theta}_1) = \sum_{s=1}^{S} \log P(\mathbf{f}^{(s)}|\mathbf{e}^{(s)};\boldsymbol{\theta}_1) \quad (3)$$

$$\mathcal{L}(\boldsymbol{\theta}_2) = \sum_{s=1}^{S} \log P(\mathbf{e}^{(s)}|\mathbf{f}^{(s)};\boldsymbol{\theta}_2) \quad (4)$$

One key limitation of these generative models is that they are **asymmetric**: each target word is restricted to be aligned to exactly one source word (including the empty cept) in the source-to-target direction and vice versa. This is undesirable because most real-world word alignments are **symmetric**, in which one-to-one, one-to-many, many-to-one, and many-to-many links are usually mixed. See Figure 1(a) for example. Therefore, a number of heuristic symmetrization methods such as intersection, union, and grow-diag-final have been proposed to combine asym-

metric alignments (Och and Ney, 2003; Koehn and Hoang, 2007).

2.2 Alignment by Agreement

Rather than using heuristic symmetrization methods, Liang et al. (2006) propose a principled approach to jointly training of the two models via enforcing **agreement**:

$$\begin{aligned} & J(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2) \\ = & \sum_{s=1}^{S} \log P(\mathbf{f}^{(s)}|\mathbf{e}^{(s)}; \boldsymbol{\theta}_1) + \\ & \log P(\mathbf{e}^{(s)}|\mathbf{f}^{(s)}; \boldsymbol{\theta}_2) + \\ & \log \sum_{\mathbf{a}} P(\mathbf{a}|\mathbf{f}^{(s)}, \mathbf{e}^{(s)}; \boldsymbol{\theta}_1) \times \\ & P(\mathbf{a}|\mathbf{e}^{(s)}, \mathbf{f}^{(s)}; \boldsymbol{\theta}_2) \end{aligned} \quad (5)$$

Note that the last term in Eq. (5) encourages the two models to agree on asymmetric alignments. While this strategy significantly improves alignment accuracy, the joint model is prone to generate one-to-one alignments because it imposes a *hard* constraint on agreement: the two models must share the same alignment when estimating the parameters by calculating the products of alignment posteriors. In Figure 1(b), the two one-to-one alignments are almost identical except for one link. This makes the posteriors to be sharply peaked around the Viterbi alignments (Liang et al., 2006). As a result, the lack of many-to-many alignments limits the benefits of joint training to end-to-end machine translation.

3 Generalized Agreement for Bidirectional Alignment

Our intuition is that the agreement between two alignments can be defined as a loss function, which enables us to consider various ways of quantification (Section 3.1) and even to incorporate the dependency between alignments and other latent structures such as phrase segmentations (Section 3.2).

3.1 Agreement between Word Alignments

The key idea of generalizing agreement is to leverage *loss functions* that measure the difference between two unidirectional alignments. For example, the last term in Eq. (5) can be re-written as

$$\begin{aligned} & \sum_{\mathbf{a}} P(\mathbf{a}|\mathbf{f}^{(s)}, \mathbf{e}^{(s)}; \boldsymbol{\theta}_1) P(\mathbf{a}|\mathbf{e}^{(s)}, \mathbf{f}^{(s)}; \boldsymbol{\theta}_2) \\ = & \sum_{\mathbf{a}_1} \sum_{\mathbf{a}_2} P(\mathbf{a}_1|\mathbf{f}^{(s)}, \mathbf{e}^{(s)}; \boldsymbol{\theta}_1) \times \\ & P(\mathbf{a}_2|\mathbf{e}^{(s)}, \mathbf{f}^{(s)}; \boldsymbol{\theta}_2) \times \\ & \delta(\mathbf{a}_1, \mathbf{a}_2) \end{aligned} \quad (6)$$

Note that the last term in Eq. (6) is actually the expected value of agreement:

$$\mathbb{E}_{\mathbf{a}_1|\mathbf{f}^{(s)}, \mathbf{e}^{(s)}; \boldsymbol{\theta}_1} \left[\mathbb{E}_{\mathbf{a}_2|\mathbf{e}^{(s)}, \mathbf{f}^{(s)}; \boldsymbol{\theta}_2} \left[\delta(\mathbf{a}_1, \mathbf{a}_2) \right] \right] \quad (7)$$

Our idea is to replace $\delta(\mathbf{a}_1, \mathbf{a}_2)$ in Eq. (6) with an arbitrary loss function $\Delta(\mathbf{a}_1, \mathbf{a}_2)$ that measures the difference between $\mathbf{a}_1$ and $\mathbf{a}_2$. This gives the new joint training objective with generalized agreement:

$$\begin{aligned} & J(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2) \\ = & \sum_{s=1}^{S} \log P(\mathbf{f}^{(s)}|\mathbf{e}^{(s)}; \boldsymbol{\theta}_1) + \\ & \log P(\mathbf{e}^{(s)}|\mathbf{f}^{(s)}; \boldsymbol{\theta}_2) - \\ & \log \sum_{\mathbf{a}_1} \sum_{\mathbf{a}_2} P(\mathbf{a}_1|\mathbf{f}^{(s)}, \mathbf{e}^{(s)}; \boldsymbol{\theta}_1) \times \\ & P(\mathbf{a}_2|\mathbf{e}^{(s)}, \mathbf{f}^{(s)}; \boldsymbol{\theta}_2) \times \\ & \Delta(\mathbf{a}_1, \mathbf{a}_2) \end{aligned} \quad (8)$$

Obviously, Liang et al. (2006)'s training objective is a special case of our framework. We refer to its loss function as **hard matching**:

$$\Delta_{\text{HM}}(\mathbf{a}_1, \mathbf{a}_2) = 1 - \delta(\mathbf{a}_1, \mathbf{a}_2) \quad (9)$$

We are interested in developing a soft version of the hard matching loss function because this will help to produce many-to-many symmetric alignments. For example, in Figure 1(c), the two alignments share most links but still allow for disagreed links to capture one-to-many and many-to-one links. Note that the union of the two asymmetric alignments is almost the same with the gold-standard alignment in this example.

While there are many possible ways to define a **soft matching** loss function, we choose the difference between disagreed and agreed link counts because it is easy and efficient to calculate during search:

$$\Delta_{\text{SM}}(\mathbf{a}_1, \mathbf{a}_2) = |\mathbf{a}_1 \cup \mathbf{a}_2| - 2|\mathbf{a}_1 \cap \mathbf{a}_2| \quad (10)$$

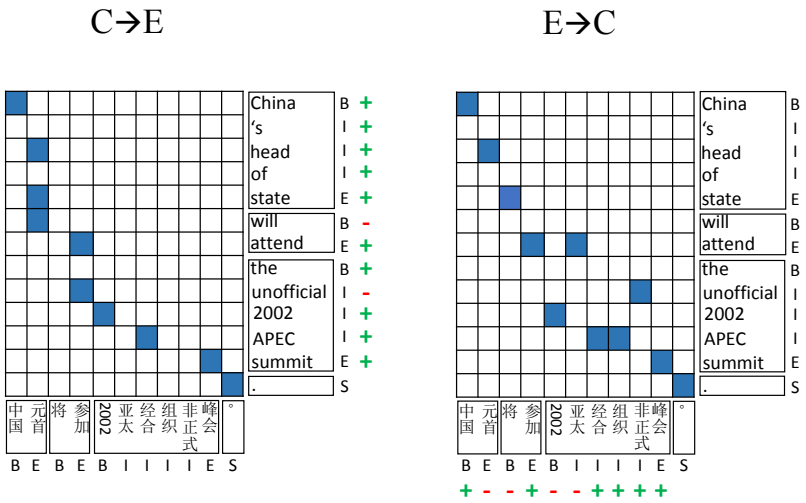


Figure 2: Generalized agreement between word alignments and phrase segmentations. The Chinese and English sentences are segmented into phrases using B (beginning), I (internal), E (ending), S (single) labels. We expect that word alignment does not violate the phrase segmentation. The word "unofficial" in the C → E alignment is labeled with "-" because "unofficial" and "2002" belong to the same English phrase but their counterparts are separated in two Chinese phrases. Words that do not violate the phrase alignment are labeled with "+". See Section 3.2 for details.

3.2 Agreement between Word Alignments and Phrase Segmentations

Our framework is very general and can be extended to include the agreement between word alignment and other latent structures such as phrase segmentations.

The words in a Chinese sentence often constitute phrases that are translated as units in English and vice versa. Inspired by the alignment consistency constraint widely used in translation rule extraction (Koehn et al., 2003), we make the following assumption to impose a structural agreement constraint between word alignment and phrase segmentation: *source words in one source phrase should be aligned to target words belonging to the same target phrase and vice versa.*

For example, consider the C → E alignment in Figure 2. We segment Chinese and English sentences into phrases, which are sequences of consecutive words. Since "2002" and "APEC" belong to the same English phrase, their counterparts on the Chinese side should also belong to one phrase.

While this assumption can potentially improve the correlation between word alignment and phrase-based translation, a question naturally arises: how to segment sentences into phrases? Instead of leveraging chunking, we treat phrase segmentation as a latent variable and train the joint alignment and segmentation model from unlabeled data in an unsupervised way.

Formally, given a target-language sentence $\mathbf{f} \equiv f_1^J = f_1, \ldots, f_J$, we introduce a latent variable $\mathbf{b} \equiv b_1^J = b_1, \ldots, b_J$ to denote a *phrase segmentation*. Each label $b_j \in \{B, I, E, S\}$, where B denotes the beginning word of a phrase, I denotes the internal word, E denotes the ending word, and S denotes the one-word phrase. Figure 2 shows the label sequences for the sentence pair.

We use a first-order HMM to model phrase segmentation of a target sentence:

$$P(\mathbf{f}; \boldsymbol{\lambda}_1) = \sum_{\mathbf{b}_1} P(\mathbf{f}, \mathbf{b}_1; \boldsymbol{\lambda}_1) \quad (11)$$

Similarly, the hidden Markov model for the phrase segmentation of the source sentence can be defined as

$$P(\mathbf{e}; \boldsymbol{\lambda}_2) = \sum_{\mathbf{b}_2} P(\mathbf{e}, \mathbf{b}_2; \boldsymbol{\lambda}_2) \quad (12)$$

Then, we can combine word alignment and phrase segmentation and define the joint training objective as

$$\begin{aligned} & J(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2) \\ = & \sum_{s=1}^{S} \log P(\mathbf{f}^{(s)} | \mathbf{e}^{(s)}; \boldsymbol{\theta}_1) + \end{aligned}$$

1: **procedure** VITERBIEM(D)
2: Initialize $\boldsymbol{\Theta}^{(0)}$
3: **for all** $k = 1, \ldots, K$ **do**
4: $\hat{\mathbf{H}}^{(k)} \leftarrow$ SEARCH($D, \boldsymbol{\Theta}^{(k-1)}$)
5: $\boldsymbol{\Theta}^{(k)} \leftarrow$ UPDATE($D, \hat{\mathbf{H}}^{(k)}$)
6: **end for**
7: **return** $\hat{\mathbf{H}}^{(K)}, \boldsymbol{\Theta}^{(K)}$
8: **end procedure**

Algorithm 1: A Viterbi EM algorithm for learning the joint word alignment and phrase segmentation model from bilingual corpus. D is a bilingual corpus, $\boldsymbol{\Theta}^{(k)}$ is the set of model parameters at the k-th iteration, $\mathbf{H}^{(k)}$ is the set of Viterbi latent variables at the k-th iteration.

$$\begin{aligned} &\log P(\mathbf{e}^{(s)}|\mathbf{f}^{(s)}; \boldsymbol{\theta}_2) + \\ &\log P(\mathbf{f}^{(s)}; \boldsymbol{\lambda}_1) + \\ &\log P(\mathbf{e}^{(s)}; \boldsymbol{\lambda}_2) - \\ &\log \mathcal{E}(\mathbf{f}^{(s)}, \mathbf{e}^{(s)}, \boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2) \end{aligned} \quad (13)$$

where the expected loss is given by

$$\begin{aligned} &\mathcal{E}(\mathbf{f}^{(s)}, \mathbf{e}^{(s)}, \boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2) \\ = &\sum_{\mathbf{a}_1} \sum_{\mathbf{a}_2} \sum_{\mathbf{b}_1} \sum_{\mathbf{b}_2} P(\mathbf{a}_1|\mathbf{f}^{(s)}, \mathbf{e}^{(s)}; \boldsymbol{\theta}_1) \times \\ &\quad P(\mathbf{a}_2|\mathbf{e}^{(s)}, \mathbf{f}^{(s)}; \boldsymbol{\theta}_2) \times \\ &\quad P(\mathbf{b}_1|\mathbf{f}^{(s)}; \boldsymbol{\lambda}_1) \times \\ &\quad P(\mathbf{b}_2|\mathbf{e}^{(s)}; \boldsymbol{\lambda}_2) \times \\ &\quad \Delta(\mathbf{a}_1, \mathbf{a}_2, \mathbf{b}_1, \mathbf{b}_2) \end{aligned} \quad (14)$$

We define a new loss function **segmentation violation** to measure the degree that an alignment violates phrase segmentations.

$$\begin{aligned} &\Delta_{\text{SV}}(\mathbf{a}_1, \mathbf{a}_2, \mathbf{b}_1, \mathbf{b}_2) \\ = &\sum_{j=1}^{J-1} \beta(\mathbf{a}_1, j, \mathbf{b}_1, \mathbf{b}_2) + \sum_{i=1}^{I-1} \beta(\mathbf{a}_2, i, \mathbf{b}_2, \mathbf{b}_1) \end{aligned} \quad (15)$$

where $\beta(\mathbf{a}_1, j, \mathbf{b}_1, \mathbf{b}_2)$ evaluates whether two links $l_1 = (j, a_j)$ and $l_2 = (j+1, a_{j+1})$ violate the phrase segmentation:

1. $\mathbf{f}_j$ and $\mathbf{f}_{j+1}$ belong to one phrase but $\mathbf{e}_{a_j}$ and $\mathbf{e}_{a_{j+1}}$ belong to two phrases, or
2. $\mathbf{f}_j$ and $\mathbf{f}_{j+1}$ belong to two phrases but $\mathbf{e}_{a_j}$ and $\mathbf{e}_{a_{j+1}}$ belong to one phrase.

The β function returns 1 if there is violation and 0 otherwise.

1: **procedure** SEARCH(D, $\boldsymbol{\Theta}$)
2: $\hat{\mathbf{H}} \leftarrow \emptyset$
3: **for all** $s \in \{1, \ldots, S\}$ **do**
4: $\hat{\mathbf{a}}_1 \leftarrow$ ALIGN($\mathbf{f}^{(s)}, \mathbf{e}^{(s)}, \boldsymbol{\theta}_1$)
5: $\hat{\mathbf{a}}_2 \leftarrow$ ALIGN($\mathbf{e}^{(s)}, \mathbf{f}^{(s)}, \boldsymbol{\theta}_2$)
6: $\hat{\mathbf{b}}_1 \leftarrow$ SEGMENT($\mathbf{f}^{(s)}, \boldsymbol{\lambda}_1$)
7: $\hat{\mathbf{b}}_2 \leftarrow$ SEGMENT($\mathbf{e}^{(s)}, \boldsymbol{\lambda}_2$)
8: $\mathbf{h}_0 \leftarrow \langle \hat{\mathbf{a}}_1, \hat{\mathbf{a}}_2, \hat{\mathbf{b}}_1, \hat{\mathbf{b}}_2 \rangle$
9: $\hat{\mathbf{h}} \leftarrow$HILLCLIMB($\mathbf{f}^{(s)}, \mathbf{e}^{(s)}, \mathbf{h}_0, \boldsymbol{\Theta}$)
10: $\hat{\mathbf{H}} \leftarrow \hat{\mathbf{H}} \cup \{\hat{\mathbf{h}}\}$
11: **end for**
12: **return** $\hat{\mathbf{H}}$
13: **end procedure**

Algorithm 2: A search algorithm for finding the Viterbi latent variables. $\hat{\mathbf{a}}_1$ and $\hat{\mathbf{a}}_2$ denote Viterbi alignments, $\hat{\mathbf{b}}_1$ and $\hat{\mathbf{b}}_2$ denote Viterbi segmentations. They form a starting point $\mathbf{h}_0$ for the hill climbing algorithm, which keeps changing alignments and segmentations until the model score does not increase. $\hat{\mathbf{h}}$ is the final set of Viterbi latent variables for one sentence.

In Figure 2, we use "+" to label words that do not violate the phrase segmentations and "-" to label violations.

In practice, we combine the two loss functions to enable word alignment and phrase segmentation to benefit each other in a joint search space:

$$\begin{aligned} &\Delta_{\text{SM+SV}}(\mathbf{a}_1, \mathbf{a}_2, \mathbf{b}_1, \mathbf{b}_2) \\ = &\Delta_{\text{SM}}(\mathbf{a}_1, \mathbf{a}_2) + \Delta_{\text{SV}}(\mathbf{a}_1, \mathbf{a}_2, \mathbf{b}_1, \mathbf{b}_2) \end{aligned} \quad (16)$$

4 Training

Liang et al. (2006) indicate that it is intractable to train the joint model. For simplicity and efficiency, they exploit a simple heuristic procedure that leverages the product of posterior marginal probabilities. The intuition behind the heuristic is that links on which two models disagree should be discounted because the products of the marginals are small (Liang et al., 2006).

Unfortunately, it is hard to develop a similar heuristic for our model that allows for arbitrary loss functions. Alternatively, we resort to a Viterbi EM algorithm, as shown in Algorithm 1. The algorithm takes the training data $D = \{\langle \mathbf{f}^{(s)}, \mathbf{e}^{(s)} \rangle\}_{s=1}^{S}$ as input (line 1). We use $\boldsymbol{\Theta}^{(k)} = \langle \boldsymbol{\theta}_1^{(k)}, \boldsymbol{\theta}_2^{(k)}, \boldsymbol{\lambda}_1^{(k)}, \boldsymbol{\lambda}_2^{(k)} \rangle$ to denote the set of model parameters at the k-th iteration. After initializing the model parameters (line 2), the algorithm alternates between searching for the Viterbi alignments

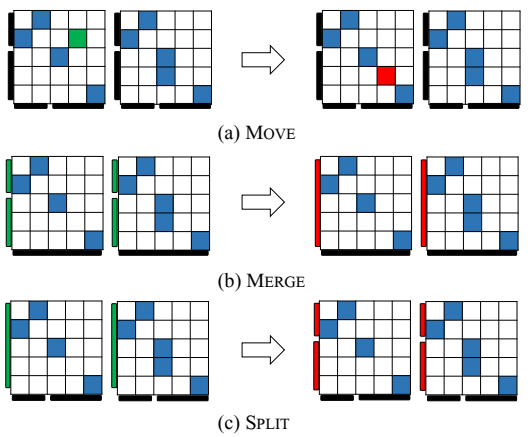


Figure 3: Operators used in the HILLCLIMB procedure.

and segmentations $\hat{\mathbf{H}}^{(k)}$ using the SEARCH procedure (line 4) and updating model parameters using the UPDATE procedure (line 5). The algorithm terminates after running for K iterations.

It is challenging to search for the Viterbi alignments and segmentations because of complicated structural dependencies. As shown in Algorithm 2, our strategy is first to find Viterbi alignments and segmentations independently using the ALIGN and SEGMENT procedures (lines 4-7), which then serve as a starting point for the HILLCLIMB procedure (lines 8-9).

Figure 3 shows three operators we use in the HILLCLIMB procedure. The MOVE operator moves a link in an alignment, the MERGE operator merges two phrases into one phrase, and the SPLIT operator splits one phrase into two smaller phrases. Note that each operator can be further divided into two variants: one for the source side and another for the target side.

5 Experiments

5.1 Setup

We evaluate our approach on Chinese-English alignment and translation tasks.

The training corpus consists of 1.2M sentence pairs with 32M Chinese words and 35.4M English words. We used the SRILM toolkit (Stolcke, 2002) to train a 4-gram language model on the Xinhua portion of the English GIGAWORD corpus, which contains 398.6M words. For alignment evaluation, we used the Tsinghua Chinese-English word alignment evaluation data set.[1] The evaluation metric is alignment error rate (AER) (Och and Ney, 2003). For translation evaluation, we used the NIST 2006 dataset as the development set and the NIST 2002, 2003, 2004, 2005, and 2008 datasets as the test sets. The evaluation metric is case-insensitive BLEU (Papineni et al., 2002).

We used both phrase-based (Koehn et al., 2003) and hierarchical phrase-based (Chiang, 2007) translation systems to evaluate whether our approach improves translation performance. For the phrase-based model, we used the open-source toolkit Moses (Koehn and Hoang, 2007). For the hierarchical phrase-based model, we used an in-house re-implementation on par with state-of-the-art open-source decoders.

We compared our approach with two state-of-the-art generative alignment models:

1. GIZA++ (Och and Ney, 2003): unsupervised training of IBM models (Brown et al., 1993) and the HMM model (Vogel et al., 1996) using EM,

2. BERKELEY (Liang et al., 2006): unsupervised training of joint HMMs using EM.

For GIZA++, we trained IBM Model 4 in two directions with the default setting and used the grow-diag-final heuristic to generate symmetric alignments. For BERKELEY, we trained joint HMMs using the default setting. The hyper-parameter of posterior decoding was optimized on the development set.

We used first-order HMMs for both word alignment and phrase segmentation. Our joint alignment and segmentation model were trained using the Viterbi EM algorithm for five iterations. Note that the Chinese-to-English and English-to-Chinese alignments are generally non-identical but share many links (see Figure 1(c)). Then, we used the grow-diag-final heuristic to generate symmetric alignments.

5.2 Comparison with GIZA++ and BERKELEY

Table 1 shows the comparison of our approach with GIZA++ and BERKELEY in terms of AER and BLEU. GIZA++ trains two asymmetric models independently and uses the grow-diag-final (i.e., GDF) for symmetrization. BERKELEY

[1] http://nlp.csai.tsinghua.edu.cn/~ly/systems/TsinghuaAligner/TsinghuaAligner.html

system	training	agreement	loss	sym.	AER	BLEU
GIZA++	indep.	N/A	N/A	GDF	21.35	24.46
BERKELEY	joint	word-word	HM	PD	20.52	24.54
this work	joint	word-word	SM	GDF	22.19	25.11
		word-word, word-phrase	SM+SV		22.01	25.78

Table 1: Comparison with GIZA++ and BERKELEY. "word-word" denotes the agreement between Chinese-to-English and English-to-Chinese word alignments. "word-phrase" denotes the agreement between word alignments and phrase segmentations. "HM" denotes the hard matching loss function, "SM" denotes soft matching, and "SV" denotes segmentation violation. "GDF" denotes grow-diag-final. "PD" denotes posterior decoding. The BLEU scores are evaluated on NIST08 test set.

alignment	loss	translation	NIST06	NIST02	NIST03	NIST04	NIST05	NIST08
GIZA++	N/A	phrase	29.57	31.82	31.67	32.20	30.48	24.46
		hier.	30.72	33.90	33.12	33.54	32.28	24.72
BERKELEY	HM	phrase	29.87	32.21	32.48	32.06	30.59	24.54
		hier.	29.52	33.59	32.70	32.95	29.52	24.29
this work	SM	phrase	30.04^{*}	32.75^{**++}	32.35^{**}	32.47^{*+}	30.86^{*+}	25.11^{**++}
		hier.	30.71^{++}	34.50^{**++}	33.89^{**++}	34.02^{*++}	32.83^{**++}	24.32
	SM+SV	phrase	30.60^{**++}	33.37^{**++}	33.24^{**++}	33.15^{**++}	31.57^{**++}	25.78^{**++}
		hier.	30.88^{++}	34.53^{**++}	34.04^{**++}	33.66^{++}	32.93^{**++}	25.17^{*++}

Table 2: Results on (hierarchical) phrase-based translation. The evaluation metric is case-insensitive BLEU. "HM" denotes the hard matching loss function, "SM" denotes soft matching, and "SV" denotes segmentation violation. "*": significantly better than GIZA++ ($p < 0.05$). "**": significantly better than GIZA++ ($p < 0.01$). "+": significantly better than BERKELEY ($p < 0.05$). "++": significantly better than BERKELEY ($p < 0.01$).

trains two models jointly with the hard-matching (i.e., HM) loss function and uses posterior decoding for symmetrization.

For our approach, we distinguish between two variants:

1. Imposing agreement between word alignments (i.e., word-word) that uses the soft matching loss function (i.e., SM) (see Section 3.1);

2. Imposing agreement between word alignments and phrase segmentations (i.e., word-word, word-phrase) that uses both the soft matching and segmentation violation loss functions (i.e., SM+SV) (see Section 3.2).

We used the grow-diag-final heuristic for symmetrization.

For the alignment evaluation, we find that our approach achieves higher AER scores than the two baseline systems. One possible reason is that links in the intersection of two symmetric alignments or two symmetric models agree usually correspond to sure links in the gold-standard annotation. Our approach loosens the hard constraint on agreement and makes the posteriors less peaked around the Viterbi alignments.

For the translation evaluation, we used the phrase-based system Moses to report BLEU scores on the NIST 2008 test set. We find that both the two variants of our approach significantly outperforms the two baselines ($p < 0.01$).

5.3 Results on (Hierarchical) Phrase-based Translation

Table 2 shows the results on phrase-based and hierarchical phrase-based translation systems. We find that our approach systematically outperforms GIZA++ and BERKELEY on all NIST datasets.

In particular, generalizing the agreement to model the discrepancy between word alignment and phrase segmentation is consistently beneficial for improving translation quality, suggesting that it is important to introduce structural constraints into word alignment to increase the correlation between alignment and translation.

While "SM+SV" improves over "SM" significantly on phrase-based translation, the margins on the hierarchical phrase-based system are relatively smaller. One possible reason is that the "SV"

system	loss	$\|\mathcal{A}_{C\to E}\|$	$\|\mathcal{A}_{E\to C}\|$	$\|\mathcal{A}_{C\to E} \cap \mathcal{A}_{E\to C}\|$	F1
GIZA++	N/A	29.39M	27.64M	17.07M	59.86
BERKELEY	HM	29.12M	28.09M	21.30M	74.46
this work	SM	29.84M	29.31M	20.24M	68.42
	SM+SV	30.04M	29.50M	20.54M	69.00

Table 3: Agreement evaluation of GIZA++, BERKELEY and our approach. The F1 score reflects how well two asymmetric alignments agree with each other.

loss function can better account for phrase-based rather than hierarchical phrase-based translation. It is possible to design new loss functions tailored to hierarchical phrase-based translation.

We also find that the BLEU scores of BERKELEY on hierarchical phrase-based translation are much lower than those on phrase-based translation. This might result from the fact that BERKELEY is prone to produce one-to-one alignments, which are not optimal for hierarchical phrase-based translation.

5.4 Agreement Evaluation

Table 3 compares how well two asymmetric models agree with each other among GIZA++, BERKELEY and our approach. We use F1 score to measure the degree of agreement:

$$\frac{2|\mathcal{A}_{C\to E} \cap \mathcal{A}_{E\to C}|}{|\mathcal{A}_{C\to E}| + |\mathcal{A}_{E\to C}|} \quad (17)$$

where $\mathcal{A}_{C\to E}$ is the set of Chinese-to-English alignments on the training data and $\mathcal{A}_{E\to C}$ is the set of English-to-Chinese alignments.

It is clear that independent training leads to low agreement and joint training results in high agreement. BERKELEY achieves the highest value of agreement because of the hard constraint.

6 Related Work

This work is inspired by two lines of research: (1) agreement-based learning and (2) joint modeling of multiple NLP tasks.

6.1 Agreement-based Learning

The key idea of agreement-based learning is to train a set of models jointly by encouraging them to agree on the hidden variables (Liang et al., 2006; Liang et al., 2008). This can also be seen as a particular form of posterior constraint or posterior regularization (Graça et al., 2007; Ganchev et al., 2010). The agreement is prior knowledge and indirect supervision, which helps to train a more reasonable model with biased guidance.

While agreement-based learning provides a principled approach to training a generative model, it constrains that the sub-models must share the same output space. Our work extends (Liang et al., 2006) to introduce arbitrary loss functions that can encode prior knowledge. As a result, Liang et al. (2006)'s model is a special case of our framework. Another difference is that our framework allows for including the agreement between word alignment and other structures such as phrase segmentations and parse trees.

6.2 Joint Modeling of Multiple NLP Tasks

It is well accepted that different NLP tasks can help each other by providing additional information for resolving ambiguities. As a result, joint modeling of multiple NLP tasks has received intensive attention in recent years, including phrase segmentation and alignment (Zhang et al., 2003), alignment and parsing (Burkett et al., 2010), tokenization and translation (Xiao et al., 2010), parsing and translation (Liu and Liu, 2010), alignment and named entity recognition (Chen et al., 2010; Wang et al., 2013).

Among them, Zhang et al. (2003)'s integrated search algorithm for phrase segmentation and alignment is most close to our work. They use Point-wise Mutual Information to identify possible phrase pairs. The major difference is we train models jointly instead of integrated decoding.

7 Conclusion

We have presented generalized agreement for bidirectional word alignment. The loss functions can be defined both between asymmetric alignments and between alignments and other latent structures such as phrase segmentations. We develop a Viterbi EM algorithm to train the joint model. Experiments on Chinese-English translation show that joint training with generalized agreement achieves

significant improvements over two baselines for (hierarchical) phrase-based MT systems. In the future, we plan to investigate more loss functions to account for syntactic constraints.

Acknowledgments

Yang Liu and Maosong Sun are supported by the 863 Program (2015AA011808), the National Natural Science Foundation of China (No. 61331013 and No. 61432013), and Samsung R&D Institute of China. Huanbo Luan is supported by the National Natural Science Foundation of China (No. 61303075). This research is also supported by the Singapore National Research Foundation under its International Research Centre@Singapore Funding Initiative and administered by the IDM Programme. We sincerely thank the reviewers for their valuable suggestions. We also thank Yue Zhang, Meng Zhang and Shiqi Shen for their insightful discussions.

References

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

David Burkett, John Blitzer, and Dan Klein. 2010. Joint parsing and alignment with weakly synchronized grammars. In *Proceedings of NAACL-HLT 2010*.

Yufeng Chen, Chengqing Zong, and Keh-Yih Su. 2010. On jointly recognizing and aligning bilingual named entities. In *Proceedings of ACL 2010*.

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.

Michel Galley, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang, and Ignacio Thayer. 2006. Scalable inference and training of context-rich syntactic translation models. In *Proceedings of COLING-ACL 2006*, pages 961–968, Sydney, Australia, July.

Kuzmann Ganchev, João Graça, Jennifer Gillenwater, and Ben Taskar. 2010. Posterior regularization for structured latent variable models. *Journal of Machine Learning Research*.

Joao V Graça, Kuzman Ganchev, and Ben Taskar. 2007. Expectation maximization and posterior constraints. In *Proceedings of NIPS 2007*.

Philipp Koehn and Hieu Hoang. 2007. Factored translation models. In *Proceedings of EMNLP-CoNLL 2007*, pages 868–876, Prague, Czech Republic, June.

Philipp Koehn, Franz J. Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of HLT-NAACL 2003*, pages 127–133, Edmonton, Canada, May.

Percy Liang, Ben Taskar, and Dan Klein. 2006. Alignment by agreement. In *Proceedings of HLT-NAACL 2006*, pages 104–111, New York City, USA, June.

Percy Liang, Dan Klein, and Michael I. Jordan. 2008. Agreement-based learning. In *Advances in Neural Information Processing Systems (NIPS)*.

Yang Liu and Qun Liu. 2010. Joint parsing and translation. In *Proceedings of COLING 2010*.

Yang Liu, Qun Liu, and Shouxun Lin. 2006. Tree-to-string alignment template for statistical machine translation. In *Proceedings of COLING/ACL 2006*.

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

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a methof for automatic evaluation of machine translation. In *Proceedings of ACL 2002*.

Andreas Stolcke. 2002. Srilm - an extensible language modeling toolkit. In *Proceedings of ICSLP 2002*.

Stephan Vogel, Hermann Ney, and Christoph Tillmann. 1996. Hmm-based word alignment in statistical translation. In *Proceedings of COLING 1996*.

Mengqiu Wang, Wanxiang Che, and Christopher D. Manning. 2013. Joint word alignment and bilingual named entity recognition using dual decomposition. In *Proceedings of ACL 2013*.

Xinyan Xiao, Yang Liu, Young-Sook Hwang, Qun Liu, and Shouxun Lin. 2010. Joint tokenization and translation. In *Proceedings of COLING 2010*.

Ying Zhang, Stephan Vogel, and Alex Waibel. 2003. Integrated phrase segmentation and alignment algorithm for statistical machine translation. In *Proceedings of Natural Language Processing and Knowledge Engineering, 2003*. IEEE.

A Transition-based Model for Joint Segmentation, POS-tagging and Normalization

Tao Qian[1,3], **Yue Zhang**[2], **Meishan Zhang**[2]*, **Yafeng Ren**[1] **and Donghong Ji**[1]
[1]Computer School, Wuhan University, Wuhan, China
[2]Singapore University of Technology and Design
[3]College of Computer Science and Technology, Hubei University of Science and Technology, XianNing, China
{taoqian, renyafeng, dhji}@whu.edu.cn
{yue_zhang, meishan_zhang}@sutd.edu.sg

Abstract

We propose a transition-based model for joint word segmentation, POS tagging and text normalization. Different from previous methods, the model can be trained on standard text corpora, overcoming the lack of annotated microblog corpora. To evaluate our model, we develop an annotated corpus based on microblogs. Experimental results show that our joint model can help improve the performance of word segmentation on microblogs, giving an error reduction in segmentation accuracy of 12.02%, compared to the traditional approach.

1 Introduction

Microblogs, such as Twitter, SMS and Weibo, has become an important research topic in NLP. Previous work has shown that off-the-shelf NLP tools can perform poorly on microblogs (Foster et al., 2011; Gimpel et al., 2011; Han and Baldwin, 2011). One of the major challenges for microblog processing is the issue of informal words. For example, "tmrw" has been frequently used in tweets for "tomorrow", causing OOV problems.

Text normalization has been introduced as a pre-processing step for microblog processing, which transforms informal words into their standard forms. Most work in the literature focuses on English microblog normalization, treating it as a noisy channel problem (Pennell and Liu, 2014; Cook and Stevenson, 2009; Yang and Eisenstein, 2013) or a translation problem (Aw et al., 2006; Contractor et al., 2010; Li and Liu, 2012; Zhang et al., 2014c), and training models based on words.

Lack of annotated corpora, text normalization is more challenging for Chinese. Unlike English, Chinese informal words are more difficult to mechanically normalize for two main reasons. First, Chinese does not have word delimiters. Second, Chinese informal words manifest diversity, such as abbreviations, neologisms, unconventional spellings and phonetic substitutions. Intuitively, there is mutual dependency between Chinese word segmentation and normalization, and therefore two tasks should be solved jointly.

Wang and Kan (2013) proposed a joint model to process word segmentation and informal word detection. However, text normalization was not included in the joint model. Kaji et al (2014) proposed a joint model for word segmentation, POS tagging and normalization for Japanese Microblogs, which was trained on a partially annotated microblog corpus. Their method requires special annotation for text normalization, which can be expensive.

In this paper, we propose a joint model for Chinese text normalization, word-segmentation and POS tagging, which can be trained using standard segmentation and POS tagging annotation, overcoming the lack of an annotated corpus on Chinese microblogs. Our model is based on Zhang and Clark (2010), with an extended set of transition actions to handle joint normalization. In our model, word segmentation and POS tagging are based on normalized text transformed from informal text. Assuming that the majority of informal words can be normalized into formal equivalents (Han et al., 2012; Li and Yarowsky, 2008), we seek standard forms of informal words from an automatically constructed normalization dictionary.

To evaluate our model, we developed an annotated corpus of microblog texts. Results show that our model achieves the best performances on three tasks compared with several baseline systems.

2 Text Normalization

Text normalization is a relatively new research topic. There are no precise definitions of a text

*corresponding author

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1837–1846, Lisbon, Portugal, 17-21 September 2015.

normalization task that are widely accepted by researchers. The task is generally divided into three categories: lexical-level, sentence-level and discourse-level normalization. In this paper we focus on lexical-level normalization, which aims to transform informal words into their standard forms.

Lexical normalization can be regarded as a spelling correction problem. However, researches on spelling correction focus on typographic and cognitive/orthographic errors (Kukich, 1992), while text normalization focuses on lexical variants, such as phonetic substitutions, abbreviation and paraphrases.

Unlike English, for which informal words are detected according to whether they are out of vocabulary, Chinese informal words manifest diversity. Wang et al. (2013) divided informal words into three types: phonetic substitutions, abbreviations and neologisms. Li and Yarowsky (2008) classified them into four types: homophone, abbreviation, transliteration and others. Due to variant characteristics, they normalise informal words by training a model per type, leading to increased system complexity.

Research reveals that most lexical variants have an unambiguous standard form (Han et al., 2012; Li and Yarowsky, 2008). The validity of this assumption is also empirically assessed on our corpus annotation in Section 6.1. Based on this assumption, we seek standard forms of informal words from a constructed normalization dictionary, avoiding diversity on informal words.

3 Joint Segmentation and Normalization

3.1 Transition-based Segmentation

We adapt the segmenter of Zhang and Clark (2007) as our baseline segmenter. Given an input sentence x, the baseline segmenter finds a segmentation by maximizing:

$$F(x) = \underset{y \epsilon Gen(x)}{\operatorname{argmax}} Score(y) \quad (1)$$

where $Gen(x)$ denotes the set of all possible segmentations for an input sentence.

Zhang and Clark (2007) proposed a graph-based scoring model, with features based on complete words and word sequences. We adapt their method slightly, under a transition-based framework (Zhang and Clark, 2011), which gives us a consistent way of defining all models in this paper.

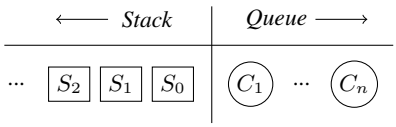


Figure 1: A state of transition-based model.

Here a transition model is defined as a quadruple $M = (C, T, W, C_t)$, where C is a state space, T is a set of transitions, each of which is a function: $C \rightarrow C$, W is an input sentence $c_1 ... \ c_n$, C_t is a set of terminal states. A model scores the output by scoring the corresponding transition sequence.

As shown in Figure 1, a state is a tuple $ST = (S, Q)$, where S contains partially segmented sequences, and $Q = (c_i, c_{i+1}, ..., c_n)$ is the sequence of input characters that have not been processed. When the character c_i is processing, the transition system would operate one of two actions that are defined as follows:

(1) APP(c_i), removing c_i from Q, and appending it to the last (partial) word in S.

(2) SEP(c_i), removing c_i from Q, making the last word in S as completed, and adding c_i as a new partial word.

Given the sentence "工作压力大啊！(How great work pressure is!)", the sequences of action "SEP(工), APP(作), SEP(压), APP(力), SEP(大), SEP(啊), SEP(!)" can be used to analyze its structure.

3.2 Joint Segmentation and Normalization

Our SN model extends the transition-based segmentation model. In addition to the actions APP and SEP, the transition system also contains a SEPS action, which substitutes an in formal word on the top of S if it exists in the normalization dictionary. Figure 2 gives a normalization transition process for the sentence "工作鸭梨大啊！(How great work pressure is!)". During processing the character "大(big)", the following actions can be applied.

(1) APP("大(big)"), appending "大(big)" to the last word "鸭梨(yālí, pear)" in the informal labeled sequence.

(2) SEP("大(big)"), making the last word "鸭梨(yālí, pear)" in the informal labeled sequence as a completed word, and adding "大(big)" as a new partial word.

(3) SEPS("大(big)", "压力(yālì, pressure)"), operating the action SEP("大(big)"), and using

Sentence: 工作鸭梨大啊! (How great work pressure is!)

State	Action	Stack	Queue	Dictionary
S_i	.....	Org: 工作 鸭梨 work pear Nor: 工作 work	大啊! big ah!	鸭梨- 压力 pear - pressure 孩纸- 孩子 child paper - child 围脖- 微博 neckerchief - microblog 盆友- 朋友 basin friend - friend .....
S_{i+1}	APP("大")	Org: 工作 鸭梨大 work pear big Nor: 工作 work	啊! (ah!)	
	SEP("大")	Org: 工作 鸭梨 大 work pear big Nor: 工作 work		
	SEPS("大", "压力")	Org: 工作 鸭梨 大 work pear big Nor: 工作 压力 work pressure		

Figure 2: Transition actions for joint segmentation and normalization.

Algorithm 1: Decoder

Input: sent, Dictionary // sent: informal sentence
Output:Best normalization sentence
1. *agenda* ←*NULL*
2. **for** *idx* ***in*** *[*0..LEN*(sent)]:*
3. **for** *state* ***in*** *agenda:*
4. *new* ←APP*(state, sent[idx])*
5. ADDITEM*(agenda, new)*
6. *new* ←SEP*(state, sent[idx])*
7. ADDITEM*(agenda, new)*
8. *norWords*←GETNWORD*(state.lastWord)*
9. **for** *word* ***in*** *norWords*
10. *new* ←SEPS*(state,sent[idx],word)*
12. ADDITEM*(agenda, new)*
13. *agenda* ←N-BEST*(agenda)*
14. **return** BEST*(agenda)*

the standard form "压力(yālì, pressure)" for the informal word " 鸭梨(yālí, pear)".

Given the sentence "工作鸭梨大啊！(How great work pressure is!)", the sequences of action "SEP(工), APP(作), SEP(鸭), APP(梨), SEPS(大, 压力), SEP(啊), SEP(!)" can be used to analyze its structure.

Lexical substitution is based on a normalization dictionary whose entries consist of <lexical variant, standard form> pairs. The output is a pair of labeled sequences, containing the informal labeled sequence and the corresponding formal labeled sequence. To rank the candidates, both labeled sequences can be scored. However, lacking annotated corpora on informal texts, we only use the score of formal labeled sequence in our model. The advantage is that we can train our model by using standard corpus only, overcoming the lack of annotated corpora on informal texts.

3.3 Training and Decoding

We apply the global training and beam-search decoding framework of Zhang and Clark (2011). An agenda is used by the decoder to keep the N-best states during the incremental process. Before decoding starts, the agenda is initialized with the initial state. When a character is processed, existing states are removed from the agenda and extended with all possible actions, and the N-best newly generated states are put back onto the agenda. After all states have been terminal, the highest-scored state from the agenda is taken as the output.

Algorithm 1 shows pseudocode for the decoder. ADDITEM adds a new item into the agenda, N-BEST returns the N highest-scored items from the agenda, and BEST returns the highest-scored item from the agenda. GETNWORD returns a possible standard form set of last word, seeking from normalization dictionary. APP appends a character to the last word in a state, SEP joins a character as the start of a new word in a state, SEPS operates SEP and replaces the last word by a possible standard form.

3.4 Features

In the experiments, we use the segmentation feature templates of Zhang and Clark (2011). These features are effective for segmentation on formal text. However, for text normalization, these features contain insufficient information. Our experiments show that by using Zhang and Clark's features, the F-Score on normalization is only 0.4207.

Prior work has shown that the language statistic information is important for text normalization (Wang et al., 2013; Li and Yarowsky, 2008; Kaji and Kitsuregawa, 2014). As a result, we extract language model features by using word-based language model learned from a large quantity of standard texts. In particular, 1-gram, 2-gram, 3-gram features are extracted. Every type of n-gram is divided into ten probability ranges. For example, if the probability of the word bigram: " 压力- 大" (high pressure) is in the 2_{nd} range, the feature is represented as "word-2-gram=2".

In our experiments, language models are trained on the Gigaword corpus[1] with SRILM tools[2]. To train a word-based language model, we segmented the corpus using our re-implementation of Zhang and Clark (2010). Results show that language model information not only improves the perfor-

[1]https://catalog.ldc.upenn.edu/LDC2003T05
[2]http://www.speech.sri.com/projects/srilm/

mance of text normalization, but also increases the performance of word-segmentation.

4 Extension for Joint Segmentation, Normalization and POS tagging

4.1 Joint Segmentation and POS Tagging

In order to reduce the error propagation of word segmentation, joint models have been applied to some NLP tasks, such as POS tagging (Zhang and Clark, 2010; Kruengkrai et al., 2009) and Parsing (Zhang et al., 2014a; Qian and Liu, 2012; Zhang et al., 2014b).

We take the joint word segmentation and POS tagging model of Zhang and Clark (2010) as the joint baseline. It extends from transition-based segmenter, adding POS arguments to the original actions. In Figure 1, when the current character c_i is processing, the transition system for ST would operate as follows :

(1) APP(c_i), removing c_i from Q, and appending it to the last (partial) word in S with the same POS tag, .

(2) SEP(c_i, pos), removing c_i from Q, making the last word in S as completed, and adding c_i as a new partial word with a POS tag "pos".

Given the sentence "工作压力大啊！(How great work pressure is!)", the sequences of action "SEP(工, NN), APP(作), SEP(压, NN), APP(力), SEP(大, VA), SEP(啊, SP), SEP(!, PU)" can be used to analyze its structure.

4.2 Joint Segmentation, Normalization and POS Tagging

Our joint model extends the model of Zhang and Clark (2010) by adding a SEPS action, which substitutes formal word for last word in S if exists in the dictionary. On the other hand, it can also be regarded as an extension of the joint segmentation and normalization model, adding POS arguments to the original actions.

Using the same example shown in Figure 2, the following three actions can be applied for the character " 大（big)":

(1) APP("大(big)"), appending "大(big)" to the last word "鸭梨(yālí, pear)" in the informal labeled sequence, which remain with the same POS tag "NN".

(2) SEP("大(big)", VA), making the last word "鸭 梨(yālí, pear)" in the informal labeled sequence as a completed word and adding "大(big)" as a new partial word with a POS tag "VA".

Text	Relation
海归也称海龟。(Overseas returnees is also referred to as turtles.)	(海归, 海龟) (overseas returnee, turtle)
一棵树，有点高，上面挂了好多人。恩，这棵树叫高数（高等数学）。(A tree, seemingly a little high, fails a lot of people. Well, this tree is called high number (advanced mathematics))	(高等数学, 高数) (advanced mathematics, high number)

Table 1: Relation patterns in microblogs.

(3) SEPS("大(big)", VA, "压 力(yālì, pressure)"), operating the action SEP("大(big)", VA), and using the standard form "压力(yālì, pressure)" for the informal word "鸭梨(yālí, pear)".

Given the sentence "工 作 鸭 梨 大 啊 ！ (How great work pressure is!)", the sequences of action "SEP(工, NN), APP(作), SEP(鸭, NN), APP(梨), SEPS(大, VA, 压 力), SEP(啊, SP), SEP(!, PU)" can be used to analyze its structure.

We use the same training and decoding framework as our joint segmentation, normalization and POS tagging model, as described in section 3.3.

5 Construction of Normalization Dictionary

Although large-scale normalization dictionaries are difficult to obtain, informal/formal relations could be extracted from large-scale web corpora (Li and Yarowsky, 2008), and informal words are mainly derived using fixed word-formation patterns. In this paper, we adopt two methods to construct a normalization dictionary.

The first method is to extract informal/formal pairs from large-scale text. In general, many informal and formal words co-occur in the same texts or similar contexts. We can find their relations with text patterns. As shown in Table 1, the first example follows the "formal也称informal" ("也称" means "is also referred to as") definition pattern, while the second example follows the pattern "informal(formal)". This gives us a reliable way to seed and bootstrap a list of informal/formal pairs.

We use a bootstrapping algorithm to extract informal/formal pairs from large-scale microblogs. First, a small set of example relations are collected manually. Second, using these relations as a seed set, we extract the text patterns, with which we identify more new relations from the data and aug-

informal也是formal的意思, formal也称informal, informal(formal), 为什么要把formal称为informal, formal其实叫informal,informal:...对formal的称谓, informal谐音自formal, “formal”缘何变“informal”, 用“informal”取代“formal”, informal就是formal的意思, informal有新的意义formal, formal的缩写是informal, 把formal说成informal, 网络...formal叫informal, 称...formal为informal, informal是formal的谐音, “formal”咋就成了“informal”, 当formal变成informal, informal可看作是formal的简称, 将formal写成informal, 网络...infromal的意思是formal.

Table 2: Examples of text patterns.

ment them into the seed set. Table 2 shows the initial text patterns extracted form the examples. The procedure iterates until it cannot identify new relations. There is much noise in the extracted informal/formal pairs. We re-rank them using a similarity-based classifier with weak supervision, with the positive pairs being inserted into dictionary.

The second method is to generate new informal/formal pairs using word-formation patterns extracted from informal/formal pairs. Although Chinese informal words manifest diversity, informal words are mainly derived using fixing word-formation methods, such as compounds, phonetic substitutions, abbreviations, acronym, reduplication. We can learn the pattern of informal word-formation from informal/formal pairs. For example, in informal/formal pair “ 妹纸(mèizhǐ, sister paper)/妹子(mèizǐ, sister)”, informal word “妹纸(mèizhǐ, sister paper)” is builded from formal word “妹子(mèizǐ, sister)” by the pattern “子→纸”. Using this pattern, we can generate many new informal/formal pairs, such as “汉纸(hànzhǐ, man paper)/汉子(hànzǐ, man)”, “男纸(nánzhǐ, man paper)/男子(nánzǐ, man)”, “孙纸(sūnzhǐ, grandson paper)/ 孙子(sūnzǐ, grandson)”, in which the formal words contain character “子”.

In the experiments, we constructed the normalization dictionary consisting of 32,787 informal/formal word pairs in total. The dictionary is used to tamper the formal training data for the joint segmentation and normalization systems with 25% of the formal words in the dictionary being replaced with their informal equivalents.

	Num	Ratio	Agree
Phonetic Substitutions	572	0.870	0.95
Abbreviation	69	0.105	0.97
Paraphrases	17	0.025	0.90
Total	658	1	0.95

Table 3: Frequency distribution and annotation agreement on various types of informal words.

6 Experiments

6.1 Microblog Corpus Annotation

To evaluate our model, we develop a microblog corpus. Our annotated corpus is collected from Sina Weibo[3], which is the largest microblogging platform in China. More than 1,000,000 Chinese posts are crawled using Sina Weibo API. Among these, 4,000 posts were randomly selected. We follow Wang et al. (2012) and apply rules to preprocess the corpus'URLs, emoticons, “@usernames” and Hashtags as pre-segmented words. As a result, we obtain 2,000 sentences as a source of the corpus.

Two human participants annotated the 2,000 sentences by using the tools we developed. The tools can simultaneously annotate word boundaries, POS and text normalization. We used the CTB scheme for word segmentation and POS tagging. We divided informal words into three types: Phonetic Substitutions, Abbreviation, Paraphrases. In total, we annotated 1,129 informal word-pairs in the 2,000 sentences, which contained 658 different informal words.

Table 3 shows the frequency distribution and annotation agreement over three types of informal words in corpus. The Cohen's kappa is 0.95 for informal words annotation, which shows that it is easy for humans to distinguish informal words, and validates our assumption that informal word generally has one formal word equivalent.

6.2 Settings and Measures

Our model is trained on the Chinese Treebank (CTB) 7[4], which is a large, word segmented, POS tagged and fully bracketed Chinese news corpus. The annotated microblog corpus is randomly divided into two parts: 1,000 sentences for development and 1,000 sentences for test.

The standard F-measure is used to measure the

[3]http://www.weibo.com/

[4]https://catalog.ldc.upenn.edu/LDC2010T07

	Development		Test	
	Seg-F	Nor-F	Seg-F	Nor-F
S;N	0.8859	0.3956	0.8885	0.4058
SN	0.8946	0.4053	0.8945	0.4207
S;N+lm	0.9101	0.5897	0.9132	0.6276
SN+lm	**0.9202**	**0.6009**	**0.9240**	**0.6392**

Table 4: Segmentation and normalization results. S;N denotes the pipeline model. SN denotes the joint model. lm denotes language model features.

	Segmentation		
models	N-R	I-R	ALL-R
S;N	0.8711	0.5100	0.8624
SN	0.8716	0.6653	0.8652
S;N+lm	0.9143	0.4229	0.9025
SN+lm	**0.9149**	**0.7752**	**0.9109**

Table 5: Formal and informal word accuracies on the development test. N-R denotes the recall rate of formal words, I-R denotes the recall rate of informal words, ALL-R denotes the recall rate of all words.

accuracies of word segmentation, POS tagging and text normalization, where the accuracy is *F = 2PR/(P+R)*. In addition, we use recall rates to evaluate the identification accuracies of formal, informal and all words. The recall rate of formal words *N-R* is defined as the percentage of gold standard output formal words that are correctly segmented, the recall rate of informal words *I-R* is defined as the percentage of gold-standard output informal words that are correctly segmented and the recall rate of all words *ALL-R* is defined as the percentage of gold standard output words that are correctly segmented.

6.3 Joint Segmentation and Normalization

Our development set is used to decide the beam size and the number of training iterations. The best performances on the development set are obtained when the beam size is set to 16 and the number of iterations is set to 32.

Comparison with pipeline We investigate the influence of the language model and analyze the result compared to the baseline. Table 4 shows the results on the development and test sets, where SN model is joint model and S;N is pipeline model. Our SN model performs better on segmentation than pipeline S;N model, demonstrating the effectiveness of normalization.

Table 5 shows the accuracies (i.e., recall rate) of formal and informal word identification on the development set. After normalization, the accuracy of informal word identification has a large improvement, and the accuracy of formal word identification also increases. This shows that formal words can be better recognized when informal words are identified correctly. It demonstrates that text normalization is effective for both informal words and formal words.

The effect of language model From Table 4, we observe that the performances increase when using language model features. Particularly, the normalization accuracy improves more significantly. It indicates that statistical language model knowledge play an important role on text normalization. Using language model features, our SN model improves more in the segmentation F-Score compared with the baseline system.

Furthermore, we also find that the language model features are helpful to identifying the formal words, as shown in Table 5. The identification accuracy of informal words increases on the SN model, while the accuracy decreases on the S;N model. Due to the relatively low frequency of informal words, they score lower on informal text by using the language model information, resulting in incorrect word segmentations. This illustrates that our joint model is more suitable for microblogs than the pipeline method.

6.4 Joint Segmentation, Normalization and POS tagging

We compare the following models on word segmentation, text normalization and POS tagging.

ST Our re-implementation of Zhang and Clark(2010). We investigate how the joint model contributes to improving accuracy of word segmentation and POS tagging in microblog domain.

S;N;T It is a pipe-line method for segmentation, normalization and POS tagging. The segmentation model does not use the features of POS. The normalization model uses segmentation information, but not features of POS. The POS tagging model does not need to segmentation.

SN;T It is another pipe-line method that first performs segmentation and normalization, then performs POS tagging. The SN model does not use the features of POS, and the POS tagging model does not need to segmentation.

SNT Our joint segmentation, normalization, and POS tagging model.

6.4.1 Results

Table 6 shows the final results on the test set. Previous work has shown that the systems trained on news data give poor accuracies of word segmentation and POS tagging in the microblog domain. As shown in Table 6,the F-Score of segmentation and POS tagging is 0.902 and 0.8163 respectively by using the Stanford segmenter and POS tagger.

Comparing ST and SNT, we find that text normalization can enhance word segmentation and POS tagging in the microblog. SNT achieved larger improvements over the baseline with language features, reducing segmentation errors by 12.02% and POS errors by 3.63%.

Another goal of the experiment is to illustrate whether the three tasks benefit from each other. Comparing SN;T to S;N;T shows that the performance increases by join segmentation and normalization. It indicates that segmentation and text normalization benefit from each other. On other hand, our SNT model yields better performance than SN;T. It indicates that POS features are effective for segmentation and text normalization, and hence three tasks benefit from each other.

The effect of the normalization dictionary The dictionary plays an important role in our model, which reduces the number of OOV words. Intuitively, the performance is higher when the coverage of dictionary is larger. In the experiments, the coverage of our dictionary on the development and tests are 45.8%,48.2% respectively.

To investigate the effect of the dictionary on our model, we manually construct ten dictionaries from our development data, with coverage between 10% and 100%. Figure 3 shows the F-score curves of test set on segmentation and POS-tagging for both SNT+lm and ST+lm model by different dictionaries. With the coverage of the dictionaries increasing from 10% to 100%, the F-score generally increases. When the coverage is greater than about 20%, the F-score for joint model is higher than for the baseline model.

6.4.2 Error Analysis

We found two major categories of errors. Abbreviation is sometimes incorrectly normalised, especially an informal word mapping to more than one formal word. For example, informal word “美偶” mapped to “美国偶像” (American idol), which consists of two words: “美国” (American) and “偶像” (idol). However, our model cannot normalise the word “美偶” in the experiment.

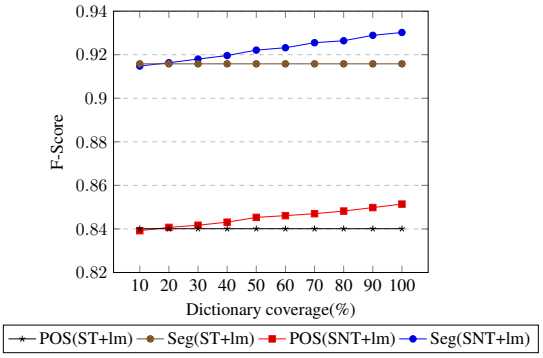


Figure 3: Results of SNT+lm and ST+lm based on different dictionaries for test set.

	Seg-F	POS-F	Nor-F
Stanford	**0.9058**	0.8163	
ST	0.8934	0.8263	
S;N;T	0.8885	0.8197	0.4058
SN;T	0.8945	0.8287	0.4207
SNT	0.8995	**0.8296**	**0.4391**
ST+lm	0.9162	0.8401	
S;N;T+lm	0.9132	0.8341	0.6276
SN;T+lm	0.9240	0.8439	0.6392
SNT+lm	**0.9261**	**0.8459**	**0.6413**

Table 6: Results on the test set. ST denotes the joint segmentation and POS tagging model. S;N;T denotes the pipeline model. SN denotes the joint segmentation and normalization model. SNT denotes the joint segmentatin. normalization and POS tagging model. lm denotes language model features. Seg-F denotes the F-Score of segmentation. POS-F denotes the F-Score of POS tagging. Nor-F denotes the F-Score of normalization.

Another type of error is phonetic substitutions of numbers, which are sometimes identified incorrectly. For example. “7456” is identified as a number in the experiments, but it means “气死我了” (I’m so angry). To settle this problem, it needs more context information.

6.5 Results of Lexical Normalization

It is interesting to explore how well the joint model can normalize informal words. We compare our results with two existing systems on text normalization based on our annotated microblog corpus.

(1) **WangDT** We re-implement Wang et al. (2013), which formalized the task as a classification problem and proposed rule-based and statistical features to model three plausible channels that explain the connection between formal and informal pairs. We use a single decision tree classifier

	P	R	F
SNT+lm	**0.9027**	0.4920	**0.6413**
WangDT	0.6214	**0.5543**	0.5859
LYTop1	0.6338	0.4920	0.5540

Table 7: Results of lexical normalization.

in the experiment.

(2) **LYTop1** Li and Yarowsky (2008) formalized the task as a ranking problem and proposed a conditional log-linear model to normalization. In the experiment, we select top 1 as the standard form of informal word.

We use the same division with 1000 sentences for training and 1000 for test. The training data is used for both the WangDT and LY. We re-segment the corpus using Stanford tools for the two baselines. WangDT uses CRF to detection informal words and LYTop1 uses the informal words detected using our joint model.

Although it is a little unfair for the two baselines compared with our joint model, which uses the external knowledge - normalization dictionary. The experiments can partly reflect some conclusions. Table 7 shows the results of normalization by different systems. The performance of our model is the best among the three systems. In particular, the precision in our SNT model improves upon the baselines significantly. The main reason is that our model is based on global features over whole sentences, while the two baselines based on local windows features.

7 Related Work

There has much work on text normalization. The task is generally treated as a noisy channel problem (Pennell and Liu, 2014; Cook and Stevenson, 2009; Yang and Eisenstein, 2013; Sonmez and Ozgur, 2014) or a translation problem (Aw et al., 2006; Contractor et al., 2010; Li and Liu, 2012; Zhang et al., 2014c). For English, most recent work (Han and Baldwin, 2011; Gouws et al., 2011; Han et al., 2012) uses two-step unsupervised approaches to first detect and then normalize informal words. They aim to produce and use informal/formal word lexicons and mappings.

In processing Chinese informal text, Wong and Xia (2008) address the problem of informal words in bulletin board system (BBS) chats by employing pattern matching. Xia et al. (2005) also use SVM-based classification to recognize Chinese informal sentences chats. Both methods have their advantages: the learning-based method does better on recall, while the pattern matching performs better on precision.

Li and Yarowsky (2008) tackle the problem of identifying informal/formal Chinese word pairs by generating candidates from Baidu search engine and ranking using a conditional log-linear model. Zhang et al. (2014c) analyze the phenomena of mixed text in Chinese microblogs, proposing a two-stage method to normalise mixed texts. However, their models employ pipelined words segmentation, resulting in reduced performance.

Wang and Kan (2013) propose a joint model to process word segmentation and informal word detection. However, text normalization is split to another task (Wang et al., 2013). Our joint model process word segmentation, POS tagging and normalization simultaneously. Kaji et al. (2014) propose a joint model for word segmentation, POS tagging and normalization for Japanese Microblogs. Their model is trained on a partially annotated microblog corpus. In contrast, our model can be trained on existing annotated corpora in standard text.

Researchers have recently developed various microblog corpora annotated with rich linguistic information. Gimpel et al. (2011) and Foster et al. (2011) annotate English microblog posts with POS tags. Han and Baldwin (2011) release a microblog corpus annotated with normalized words. Duan et al. (2012) develop a Chinese microblog corpus annotated with segmentation for SIGHAN bakeoff. Wang et al. (2013) release a Chinese microblog corpus for word segmentation and informal word detection. However, there are no microblog corpora annotated Chinese word segmentation, POS tags, and normalized sentences.

Our work is alse related to the work of word segmentation (Zhang and Clark, 2007; Zhang et al., 2013; Chen et al., 2015) and joint word segmentation and POS-tagging (Jiang et al., 2008; Zhang and Clark, 2010). A comprehensive survey is out of the scope of this paper, but interested readers can refer to Pei et al. (Pei et al., 2014) for a recent literature review of the fields.

To evaluate our model, we develop an annotated microblog corpus with word segmentation, POS tags, and normalization. Furthermore, we train our model by using a standard segmented and POS tagged corpus. We also present a comprehensive evaluation in terms of precision and recall on our

microblog test corpus. Such an evaluation has not been conducted in previous work due to the lack of annotated corpora for Chinese microblogs.

8 Conclusion

We proposed a joint model of word segmentation, POS tagging and normalization, in which the three tasks benefit from each other. The model is trained on standard corpora, hence there is no need to retrain it for new microblog corpora. The results demonstrated that the model can improve the performance of word segmentation and POS tagging with text normalization on microblogs, and our model can benefit from the language statistical information, which is not suitable to segment word and tag POS directly for microblogs because of the relatively low frequency of informal words.

In our model, lexical substitution is based on a normalization dictionary, which avoids the diversity of informal words, simplifying this problem for real world applications. The codes of the joint model and data set are published at the website: `https://github.com/qtxcm/JointModelNSP`.

Acknowledgments

We thank all reviewers for the insightful comments. This work is supported by the State Key Program of National Natural Science Foundation of China (No.61133012), the National Natural Science Foundation of China (No.61373108, 61373056, 61202193), the Key Program of Natural Science Foundation of Hubei, China(No.2012FFA088), the National Philosophy Social Science Major Bidding Project of China (No.11&ZD189) and the Singapore Ministration and Education (MOE) AcRF project T2MOE201301.

References

AiTi Aw, Min Zhang, Juan Xiao, and Jian Su. 2006. A phrase-based statistical model for sms text normalization. In *Proceedings of the COLING/ACL*, pages 33–40.

Xinchi Chen, Xipeng Qiu, Chenxi Zhu, and Xuanjing Huang. 2015. Gated recursive neural network for chinese word segmentation. In *Proceedings of the 53rd ACL*, pages 1744–1753, July.

Danish Contractor, Tanveer A Faruquie, and L Venkata Subramaniam. 2010. Unsupervised cleansing of noisy text. In *Proceedings of the 23rd COLING*, pages 189–196.

Paul Cook and Suzanne Stevenson. 2009. An unsupervised model for text message normalization. In *Proceedings of the workshop on computational approaches to linguistic creativity*, pages 71–78.

Huiming Duan, Zhifang Sui, Ye Tian, and Wenjie Li. 2012. The cips-sighan clp 2012 chinese word segmentation on microblog corpora bakeoff. In *Proceedings of the Second CIPS-SIGHAN Joint Conference on Chinese Language Processing, Tianjin, China*, pages 35–40.

Jennifer Foster, Özlem Çetinoglu, Joachim Wagner, Joseph Le Roux, Stephen Hogan, Joakim Nivre, Deirdre Hogan, and Josef Van Genabith. 2011. # hardtoparse: Pos tagging and parsing the twitterverse. In *AAAI 2011 Workshop on Analyzing Microtext*, pages 20–25.

Kevin Gimpel, Nathan Schneider, Brendan O'Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A Smith. 2011. Part-of-speech tagging for twitter: Annotation, features, and experiments. In *Proceedings of the 49th ACL*, pages 42–47.

Stephan Gouws, Donald Metzler, Congxing Cai, and Eduard Hovy. 2011. Contextual bearing on linguistic variation in social media. In *Proceedings of the Workshop on Languages in Social Media*, pages 20–29.

Bo Han and Timothy Baldwin. 2011. Lexical normalisation of short text messages: Makn sens a# twitter. In *Proceedings of the 49th ACL*, pages 368–378.

Bo Han, Paul Cook, and Timothy Baldwin. 2012. Automatically constructing a normalisation dictionary for microblogs. In *Proceedings of the 2012 EMNLP*, pages 421–432.

Wenbin Jiang, Liang Huang, Qun Liu, and Yajuan Lü. 2008. A cascaded linear model for joint chinese word segmentation and part-of-speech tagging. In *Proceedings of ACL-08: HLT*, pages 897–904.

Nobuhiro Kaji and Masaru Kitsuregawa. 2014. Accurate word segmentation and pos tagging for japanese microblogs: Corpus annotation and joint modeling with lexical normalization. In *Proceedings of the 2014 EMNLP)*, pages 99–109, October.

Canasai Kruengkrai, Kiyotaka Uchimoto, Jun'ichi Kazama, Yiou Wang, Kentaro Torisawa, and Hitoshi Isahara. 2009. An error-driven word-character hybrid model for joint chinese word segmentation and pos tagging. In *Proceedings of the Joint Conference of the 47th ACL and the 4th AFNLP*, pages 513–521.

Karen Kukich. 1992. Techniques for automatically correcting words in text. *ACM Computing Surveys (CSUR)*, 24(4):377–439.

Chen Li and Yang Liu. 2012. Normalization of text messages using character-and phone-based machine translation approaches. In *Thirteenth Annual Conference of the International Speech Communication Association*.

Zhifei Li and David Yarowsky. 2008. Mining and modeling relations between formal and informal chinese phrases from web corpora. In *Proceedings of the 2008 EMNLP*, pages 1031–1040.

Wenzhe Pei, Tao Ge, and Baobao Chang. 2014. Max-margin tensor neural network for chinese word segmentation. In *Proceedings of the 52nd ACL*, pages 293–303, June.

Deana L Pennell and Yang Liu. 2014. Normalization of informal text. *Computer Speech & Language*, 28(1):256–277.

Xian Qian and Yang Liu. 2012. Joint chinese word segmentation, pos tagging and parsing. In *Proceedings of the 2012 EMNLP*, pages 501–511, July.

Cagil Sonmez and Arzucan Ozgur. 2014. A graph-based approach for contextual text normalization. In *Proceedings of the 2014EMNLP*, pages 313–324.

Aobo Wang and Min-Yen Kan. 2013. Mining informal language from chinese microtext: Joint word recognition and segmentation. In *ACL (1)*, pages 731–741.

Aobo Wang, Tao Chen, and Min-Yen Kan. 2012. Re-tweeting from a linguistic perspective. In *Proceedings of the second workshop on language in social media*, pages 46–55.

Aobo Wang, Min-Yen Kan, Daniel Andrade, Takashi Onishi, and Kai Ishikawa. 2013. Chinese informal word normalization: an experimental study. In *Proceedings of IJCNLP*, pages 127–135.

Kam-Fai Wong and Yunqing Xia. 2008. Normalization of chinese chat language. *Language Resources and Evaluation*, 42(2):219–242.

Yunqing Xia, Kam-Fai Wong, and Wei Gao. 2005. Nil is not nothing: Recognition of chinese network informal language expressions. In *4th SIGHAN Workshop on Chinese Language Processing at IJCNLP*, volume 5.

Yi Yang and Jacob Eisenstein. 2013. A log-linear model for unsupervised text normalization. In *EMNLP*, pages 61–72.

Yue Zhang and Stephen Clark. 2007. Chinese segmentation with a word-based perceptron algorithm. In *Proceedings of the 45th ACL*, pages 840–847, June.

Yue Zhang and Stephen Clark. 2010. A fast decoder for joint word segmentation and pos-tagging using a single discriminative model. In *Proceedings of the 2010 EMNLP*, pages 843–852.

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

Longkai Zhang, Houfeng Wang, Xu Sun, and Mairgup Mansur. 2013. Exploring representations from unlabeled data with co-training for Chinese word segmentation. In *Proceedings of the 2013 EMNLP*, pages 311–321, October.

Meishan Zhang, Yue Zhang, Wanxiang Che, and Ting Liu. 2014a. Character-level chinese dependency parsing. In *Proceedings of the 52nd ACL*, pages 1326–1336, June.

Meishan Zhang, Yue Zhang, Wanxiang Che, and Ting Liu. 2014b. Type-supervised domain adaptation for joint segmentation and pos-tagging. In *Proceedings of the 14th EACL*, pages 588–597.

Qi Zhang, Huan Chen, and Xuanjing Huang. 2014c. Chinese-english mixed text normalization. In *Proceedings of the 7th ACM international conference on Web search and data mining*, pages 433–442. ACM.

Multilingual discriminative lexicalized phrase structure parsing

Benoit Crabbé
Alpage – Université Paris Diderot – Inria – IUF
Place Paul Ricoeur 75013 Paris
benoit.crabbe@univ-paris-diderot.fr

Abstract

We provide a generalization of discriminative lexicalized shift reduce parsing techniques for phrase structure grammar to a wide range of morphologically rich languages. The model is efficient and outperforms recent strong baselines on almost all languages considered. It takes advantage of a dependency based modelling of morphology and a shallow modelling of constituency boundaries.

1 Introduction

Lexicalized phrase structure parsing techniques were first introduced by Charniak (2000) and Collins (2003) as generative probabilistic models. Nowadays most statistical models used in natural language processing are discriminative: discriminative models provide more flexibility for modelling a large number of variables and conveniently expressing their interactions. This trend is particularly striking if we consider the literature in dependency parsing. Most state of the art multilingual parsers are actually weighted by discriminative models (Nivre and Scholz, 2004; McDonald et al., 2005; Fernández-González and Martins, 2015).

With respect to multilingual phrase structure parsing, the situation is quite different. Most parsers focus on fixed word order languages like English or Chinese as exemplified by Zhu et al. (2013). Despite a few exceptions (Collins et al., 1999), multilingual state of the art results are generally derived from the generative model of Petrov et al. (2006). Although more recently Hall et al. (2014) introduced a conditional random field parser that clearly improved the state of the art in the multilingual setting.

Both Petrov et al. (2006) and Hall et al. (2014) frame their parsing model to model in priority regular surfacic patterns and word order: Petrov et al. (2006) crucially infers category refinements (called category 'splits') in order to specialize the grammar on recurrent informative patterns observed on input spans. Hall et al. (2014) relies on a similar intuition : the model essentially aims to capture regularities on the spans of constituents and their immediate neighbourhood, following earlier intuitions of Klein and Manning (2004). This modelling strategy has two main motivations. First it reduces the burden of feature engineering, making it easier to generalize to multiple languages. Second it avoids modeling explicitly bilexical dependencies for which parameters are notoriously hard to estimate from small data sets such as existing treebanks.

On the other hand this strategy becomes less intuitive when it comes to modeling free word order languages where word order and constituency should in principle be less informative. As such, the good results reported by Hall et al. (2014) are surprising. It suggests that word order and constituency might be more relevant than often thought for modelling free word order languages.

Nevertheless, free word order languages also tend to be morphologically rich languages. This paper shows that a parsing model that can effectively take morphology into account is key for parsing these languages. More specifically, we show that an efficient lexicalized phrase structure parser - modelling both dependencies and morphology - already significantly improves parsing accuracy. But we also show that an additional modelling of spans and constituency provides additional robustness that contributes to yield state of the art results on almost all languages considered, while remaining quite efficient. Moreover, given the availability of existing multi-view treebanks (Bhatt et al., 2009; Seddah et al., 2013; Qiu et al., 2014), our proposed solution only requires a lightweight infrastructure to achieve multilin-

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1847–1856,
Lisbon, Portugal, 17-21 September 2015.

gual parsing without requiring costly language-dependent modifications such as feature engineering.

The paper is organized as follows. We first review the properties of multiview treebanks (Section 2). As these treebanks typically do not provide directly head annotation, an information required for lexicalized parsing, we provide an automated multilingual head annotation procedure (Section 3). We then describe in section 4 a variant of lexicalized shift reduce parsing that we use for the multilingual setting. It provides a way to integrate morphology in the model. Section 5 finally describes a set of experiments designed to test our main hypothesis and to point out the improvements over state of the art in multilingual parsing.

2 Multi-view treebanks

Multi-view treebanks are treebanks annotated both for constituents and dependencies that have the property to be token-wise aligned (Bhatt et al., 2009; Seddah et al., 2013; Qiu et al., 2014) . These double annotations are typically obtained by converting a constituency or dependency annotation into the other annotation type. This method was used for the construction of the dataset for the SPMRL 2013 shared task (Seddah et al., 2013), which contains multi-view treebanks for a number of morphologically rich languages, for which either constituency or dependency treebanks were available. The same kind of process was applied to the Penn TreeBank using the Stanford conversion system to produce dependency annotations (de Marneffe et al., 2006). In this paper, we use both of these datasets.

Although in multi-view treebanks each sentence is annotated both for constituency and dependency, they are not normalized for categories nor lexical features accross languages such as dependencies in the Google Universal Treebank (McDonald et al., 2013). What is more, the dependency and constituency structures may sometimes strongly differ. For some languages, like Hungarian, the conversion has involved some manual reannotation (Vincze et al., 2010).

3 Head annotation procedure

Lexicalized phrase structure parsers traditionally use hand-crafted heuristics for head annotation (Collins, 2003). Although these heuristics are available for some languages, for others they are non existent or non explicit and typically hidden in conversion procedures. In order to leverage the burden of managing language specific heuristics, we first automate head annotation by taking advantage of the multi-view annotation.

We begin by introducing some notation. Assuming a sentence $W = w_1 \dots w_n$, the dependency annotation of this sentence is assumed to be a dependency forest (Kuhlmann and Nivre, 2006). A dependency graph $G = \langle V, E \rangle$ where $V = \{1 \dots n\}$ is the set of word indexes or vertices and $E \subseteq V \times V$ is a set of dependency links. By convention, a dependency (i, j) means that i governs j. A dependency forest is a dependency graph such that a node has at most a single incoming edge and where there is no cycle. A node with no incoming edge is a root of the dependency forest and a dependency tree is a dependency forest with a single root. For some languages, such as German or Basque, the dependency structures found in the data set are actually dependency forests.

Lexicalized parsing relies on head annotation, in other words each node in a constituency tree is associated with the word index of its head. More formally, let A be the set of nodes in the c-tree, head annotation can be represented as a function $h : A \mapsto \{1 \dots n\}$ which maps each node $a \in A$ to the index of its head in the input sentence. h is obtained by leveraging head-related information associated with each rule in the grammar. More precisely, each rule $\tau \rightarrow \gamma$, with $\gamma = a_1 \dots a_k$, is associated with a head index i ($1 \leq i \leq k$) that states that the head $h(\tau)$ of any node labeled τ in a constituency tree that is built using this rule is the same as the head of the right-hand side symbol a_i.

A **Naive** h function is straightforwardly defined as the annotation of each local rule part of the tree in a bottom-up fashion:[1]

Base: $h(w_i) = i \qquad \forall w_i \in W$
Recurrence:

$$h(\tau) = \begin{cases} h(a_i) & \text{if } \forall_{a_j : a_j \in \gamma, i \neq j} \, (h(a_i), h(a_j)) \in E \\ \perp & \text{otherwise} \end{cases}$$

When $\exists a \in A$ such that $h(a) = \perp$ we say that the annotation has failed.

However the naive procedure fails in a large number of cases. Failures fall into the four patterns that are illustrated in Figure 1. For each of

[1] The additional case of unary rules is straightforward and left to the reader.

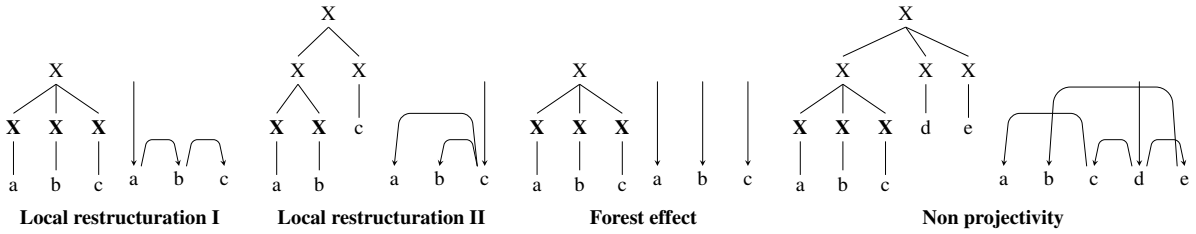


Figure 1: Patterns of the causes of problems taking place during head annotation

the patterns we have highlighted in bold the symbols for which the **Naive** procedure currently fails. *Local Restructuration I* is where the c-structure is flatter than the d-structure. Here the **Naive** procedure fails because $(a, c) \notin E$. *Local Restructuration II* is where the d-structure is flatter than the c-structure. The procedure fails because neither $(a, b) \in E$ nor $(b, a) \in E$. *Forest Effect* is where the d-structure is a dependency forest (here $E = \emptyset$). And finally *Non Projectivity* is where the d-tree is non projective.

We can easily correct the naive procedure for *Local Restructuration I* by taking advantage of $E+$, the non reflexive transitive closure of E, thus yielding the following **Corrected** procedure:

Base: $h(w_i) = i \qquad \forall w_i \in W$
Recurrence:

$$h(\tau) = \begin{cases} h(a_i) & \text{if } \forall_{a_j : a_j \in \gamma, i \neq j} (h(a_i), h(a_j)) \in E^+ \\ \perp & \text{otherwise} \end{cases}$$

The three other cases are more problematic, since their correction would somehow require altering the structure of either the c-tree or the d-tree. Refraining from altering the constituency data set we instead use a catch-all procedure that essentially creates the problematic head annotation by analogy with the rest of the data, yielding a fully **Robust** procedure that is guaranteed to succeed in any case:

Base: $h(w_i) = i \qquad \forall w_i \in W$
Recurrence:

$$h(\tau) = \begin{cases} h(a_i) & \text{if } \forall_{a_j : a_j \in \gamma, i \neq j} (h(a_i), h(a_j)) \in E^+ \\ h(a_{\text{KNN}(\tau \rightarrow \gamma)}) & \text{otherwise} \end{cases}$$

where KNN($\tau \rightarrow \gamma$) is a function returning a guess for the position of the head in γ, the right hand side of the rule, based on similarity to successfully head annotated rules.

The details are as follows. KNN($\tau \rightarrow \gamma$) supposes a dataset $D = (R_i, \mathcal{H}_i)_{i=1}^N$ of successfully head annotated rules. In this dataset, each rule $R_i = \tau \rightarrow \gamma$ is associated with $\mathcal{H}_i$ the position of the head in γ. We define the similarity between two rules $R_1 = \tau^{(1)} \rightarrow a_1^{(1)} \dots a_k^{(1)}$ and $R_2 = \tau^{(2)} \rightarrow a_1^{(2)} \dots a_{k'}^{(2)}$ to be the Levenshtein distance between $\tau^{(1)}, a_1^{(1)} \dots a_k^{(1)}$ and $\tau^{(2)}, a_1^{(2)} \dots a_{k'}^{(2)}$. In practice for a given rule R the function returns the most frequent $\mathcal{H}$ among the 5 most similar rules in the data set.

The full head annotated data set D is built by reading off the rules from the trees successfully annotated in the treebank by the **Corrected** procedure in a first pass. A second pass yields the final annotation by running the **Robust** procedure.

Analysis of the conversion We report in Table 1 an overall quantification of the conversion procedure: % Success (**Corrected**) reports the number of trees succesfully annotated by the **Corrected** procedure and Silver UAS reports an UAS score obtained by comparing the reference dependency trees to the conversion of those obtained from the Robust conversion of the head-annotated phrase structure trees back to dependency structures. The conversion works well apart from four languages (Arabic, Basque, German and Hungarian) which cause more difficulties.

Language	% Success (**Corrected**)	Silver UAS
ARABIC	61.7	92.0
BASQUE	54.2	82.7
ENGLISH	99.9	98.8
FRENCH	99.9	99.3
GERMAN	98.4	72.1
HEBREW	98.2	99.0
HUNGARIAN	85.9	80.1
KOREAN	100.0	100.0
POLISH	98.6	98.8
SWEDISH	99.6	98.8

Table 1: Quantification of the conversion

In order to better understand the problems faced by the conversion procedure, we manually inspected the errors returned by the **Corrected** pro-

cedure. For each language, we sampled 20 examples of failures encountered and we manually categorized the errors using the four patterns illustrated in Figure 1. Across languages, 49.9% of the errors come from the pattern *Local Restructuration II* and 50% from the pattern *Forest effect* and more suprisingly, we found only one example in our sample from the pattern *Non projectivity* in the Hungarian treebank. This overall average hides however an important variation across treebanks. The *Forest effect* is indeed massively found in the Basque[2] (100%) and German treebanks and more marginally in the Hungarian data set. Most of the time, these are cases of short word sequences (2 to 5 tokens) where all nodes are annoted as roots of the dependency trees. The *Local restructuration II* is mostly found in the Arabic, Hebrew and Polish treebanks and less frequently in Hungarian. Arabic and Hebrew tend to follow a binary annotation scheme partially inspired by X-Bar, hence creating additional constituent structures that are not directly inferrable from the dependency annotation. Polish uses this restructuration in patterns involving coordination. More surprisingly, non projective patterns, which we expected to be a significant feature of these languages, remain marginal in comparison to annotation related idiosyncrasic problems.

4 Parsing algorithm

This section provides an overview of the design of the constituent parsing system. There are three recent proposals for beam-based discriminative shift reduce parsing for phrase structure grammar with a structured perceptron and beam search (Zhu et al., 2013; Crabbé, 2014; Mi and Huang, 2015). All three proposals point out that for weighted phrase structure parsing, the shift reduce algorithm requires a special treatment of unary rules in order to compare derivations of the same length. They all provide different management schemes for these unaries.

The work described here draws on the LR algorithm introduced by Crabbé (2014), but provides a simpler algorithm, it precisely describes the management of unary rules and clarifies how spans and morphological information is represented (see section 5).

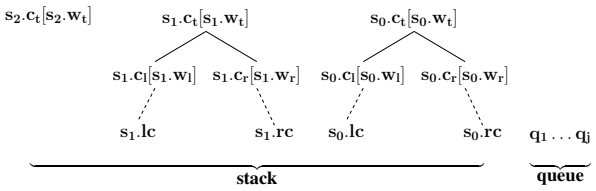
[2]The constituency conversion of the Basque treebank also contains a recurrent attachment error of the punctuations which we ignored when computing this statistic.

For each language, the grammar is induced from a treebank using the following preprocessing steps. The corpus is first head-annotated with the **Robust** head annotation procedure. Second, the treebank is head-markovized (order 0) and unary productions that do not emit tokens[3] are collapsed into unique symbols. Once this has been done we assume that tokens to be parsed are a list of couples (tag, wordform). The preprocessing steps ensure the binarized treebank implicitly encodes a binary lexicalized grammar whose rules are either in Chomsky Normal Form (CNF) like in (a) $X[h] \rightarrow A[x]\,B[h]$, $X[h] \rightarrow A[h]\,B[x]$, $X[t] \rightarrow t$ or are also of the form (b) $X[h] \rightarrow A[h]\,t$, $X[t] \rightarrow A[h]\,t$, $X[h] \rightarrow t\,B[h]$, $X[t] \rightarrow t\,B[h]$ where A, B, X are delexicalized non-terminals, h, x, t are tokens (terminals) and $A[h], A[x] \ldots X[t]$ are lexicalized non-terminals. Given a grammar in CNF, we can prove that for a sentence of length n, the number of derivation steps for a shift reduce parser is $3n - 1$. However our tagset-preserving transformation also introduces rules of the form (b), which explains why the number of derivation steps may vary from $2n - 1$ to $3n - 1$.

To ensure that a derivation is of length $3n - 1$, the parser forces each shift to be followed by either a unary reduction or an alternative dummy Ghost Reduction (GR). Given the pre-processed treebank we infer the set A of actions used by the parser. Let Σ be the set of non-terminal symbols (including temporary symbols) read off from the binary treebank. The set of actions contains one Shift (S), one Ghost Reduction (GR) a set of $|\Sigma|$ unary reductions (RU-X), one for each symbol, a set of $|\Sigma|$ binary left reductions (RL-X) and a set of $|\Sigma|$ binary right reductions (RR-X) (see also Sagae and Lavie (2006) and Figure 3 for details).

The parser itself is organized around two data structures: a stack of symbols, $\mathbf{S} = \ldots|s_2|s_1|s_0$, whose topmost element is s_0. Symbols are lexicalized non terminals or tokens of the form $A[x]$. The second structure is a queue statically filled with tokens $T = t_1 \ldots t_n$. Parsing is performed by sequentially generating configurations C of the form $\langle j, \mathbf{S}, \cdot \rangle$ where $\mathbf{S}$ is a stack and j is the index of the first element of the queue. Given an initial configuration $C_0 = \langle 1, \epsilon, \perp \rangle$, a derivation step $C_{t-1} \overset{a_{t-1}}{\Rightarrow} C_t$ generates a new configuration C_t by applying an action $a_{t-1} \in A$ as defined in Figure 3. The derivation is complete and successful

[3]In order not to alter the tagset of the treebank.



Language	gen	num	case	mood	aspect	other
ARABIC	gender	number	case	mood	aspect	-
BASQUE	-	NUM	KAS	MDN	ASP	DADUDIO,ERL,NOR(I|K)?
ENGLISH	-	-	-	-	-	-
FRENCH	g	n	-	m	-	mwe
GERMAN	gender	number	case	mood	-	-
HEBREW	gen	num	-	-	-	tense
HUNGARIAN	-	Num	Cas	Mood	-	SubPOS
KOREAN	-	-	case-type	-	-	verb-type
POLISH	gender	number	case	-	aspect	post-prepositionality
SWEDISH	gender	number	case	verbform	-	perfectform

Figure 2: Features available for scoring. $\mathbf{s_x}$ denote a position in the stack. Stack positions are local trees of depth 1, features can access its **t**op, **l**eft and **r**ight nodes. The suffixes $\mathbf{c_p}, \mathbf{w_p}, \mathbf{lc}, \mathbf{rc}$ denote respectively the delexicalized category, the head token, the left corner token, the right corner token of a stack position. For tokens elements accessible from the stack ($\mathbf{s_x.w_x}$) and from the queue ($\mathbf{q_x}$), features can access the **word** form, pos **tag** or any morphological feature **m** available for that language as described in the table at the right

$$\textsc{Init} \quad \langle 1, \epsilon, \bot \rangle : 0$$
$$\textsc{Goal} \quad \langle n+1, \tau, \bot \rangle : w$$

$$\textsc{Shift} \quad \frac{\langle j, \mathbf{S}, \bot \rangle : w}{\langle j+1, \mathbf{S} \mid t_j.tag[t_j.word]), \top \rangle : w + F(S, \langle j, \mathbf{S} \rangle)}$$

$$\text{RL(X)} \quad \frac{\langle j, \mathbf{S}_\ominus \mid c_1[t_1]\ c_0[t_0], \bot \rangle : w}{\langle j, \mathbf{S}_\ominus \mid X[t_1], \bot \rangle : w + F(RL(X), \langle j, \mathbf{S} \rangle)}$$

$$\text{RR(X)} \quad \frac{\langle j, \mathbf{S}_\ominus \mid c_1[t_1]\ c_0[t_0], \bot \rangle : w}{\langle j, \mathbf{S}_\ominus \mid X[t_0], \bot \rangle : w + F(RR(X), \langle j, \mathbf{S} \rangle)}$$

$$\text{RU(X)} \quad \frac{\langle j, \mathbf{S}_\ominus \mid c_0[t_0], \top \rangle : w}{\langle j, \mathbf{S}_\ominus \mid X[t_0], \bot \rangle : w + F(RU(X), \langle j, \mathbf{S} \rangle)}$$

$$\text{GR(X)} \quad \frac{\langle j, \mathbf{S}_\ominus \mid c_0[t_0], \top \rangle : w}{\langle j, \mathbf{S}_\ominus \mid c_0[t_0], \bot \rangle : w + F(GR, \langle j, \mathbf{S} \rangle)}$$

Figure 3: Weighted inference rules

once the action C_{3n-1} is generated. A derivation sequence $C_{0 \Rightarrow \tau}$ is a sequence of derivation steps $C_0 \overset{a_0}{\Rightarrow} \dots \overset{a_{\tau-1}}{\Rightarrow} C_\tau$

Weighted prediction The choice of the action $a \in A$ at each derivation step is naturally non-deterministic. Determinism is provided by a weighting function based on a linear model of the form:

$$W(C_{0 \Rightarrow \tau}) = \sum_{i=0}^{\tau-1} \mathbf{w} \cdot \mathbf{\Phi}(a_i, C_i) = \sum_{i=0}^{\tau-1} F(a_i, C_i)$$

where $\mathbf{w} \in \mathbb{R}^d$ is a weight vector and $\mathbf{\Phi}(a_i, C_i) \in \{0,1\}^d$ is a feature vector. The best parse is then the successful derivation with the maximum score:

$$\hat{C}_{0 \Rightarrow 3n-1} = \underset{C_{0 \Rightarrow 3n-1} \in \text{GEN}_{3n-1}}{\text{argmax}} \quad W(C_{0 \Rightarrow 3n-1})$$

In practice, we use a beam of size K at each time step and lossy feature hashing, which makes the inference approximative.

For the purpose of computing weights, we extend the representation of the stack and queue elements such that the feature functions have access to a richer context than just simple lexicalized symbols of the form $A[x]$. As described in Figure 2 (left), features can also access the immediate left and right children of s_0 and s_1 as well as their left and right corner tokens. This allows us to encode the span models described in Section 5. We also use tuple-structured tokens encoding not only the word-form and the tag but also additional custom lexical features such as those enumerated in Figure 2 (right). This allows us to express the morphological models described in Section 5.

Finally, the parameters $\mathbf{w}$ are estimated with a parallel averaged structured perceptron designed to cope with inexact inference (beam search): we specifically rely on max-violation updates of Huang et al. (2012) and on minibatches to accelerate and parallelize training (Shalev-Shwartz et al., 2007; Zhao and Huang, 2013).

5 Experiments

The experiments aim to compare the contribution of span based features approximating some intuitions of Hall et al. (2014) for shift reduce parsing and morphological features for parsing free word order languages. We start by describing the evaluation protocol and by defining the models used.

We use the standard SPMRL data set (Seddah et al., 2013). Part of speech tags are generated with Marmot (Müller et al., 2013), a CRF tagger specifically designed to provide tuple-structured tags. The training and development sets are tagged by 10-fold jackknifing. Head annotation is supplied by the **Robust** procedure described in Section 3. The parser is systematically trained for 25 epochs with a max violation update perceptron, a beam of size 8 and a minibatch size of 24.

To enable a comparison with other published results, the evaluation is performed with a version of `evalb` provided by the SPMRL organizers (Seddah et al., 2013) which takes punctuation into account.

Baseline model (B) The baseline model uses a set of templates identical to those of Zhu et al. (2013) for parsing English and Chinese except that we have no specific templates for unary reductions.

Span-based model (B+S) This model extends the **B** model by modeling spans. The span model approximates an intuition underlying Hall et al. (2014): constituent boundaries contain very informative tokens (typically function words). These tokens together with the pattern of their neighborhood provide key clues for detecting and (sub-)typing constituents. Moreover, parameter estimation for frequent functional words should suffer less from data sparseness issues than the estimation of bilexical dependencies on lexical head words. The model includes conjunctions of nonterminal symbols on the stack with their left and right corners (words or tags) and also their immediately adjacent tokens across constituents. Using the notation given in Figure 2 we specifically included the following matrix templates :

$$\mathbf{s_0.c_t\&s_0.lc.word\&s_0.rc.word}$$
$$\mathbf{s_1.c_t\&s_1.lc.word\&s_1.rc.word}$$
$$\mathbf{s_0.c_t\&s_0.lc.word\&s_1.rc.word}$$
$$\mathbf{q_1.word\&s_0.lc.word\&s_0.rc.word}$$
$$\mathbf{q_2.word\&s_0.lc.word\&s_0.rc.word}$$

from which we derived additional backoff templates where only a single corner condition is expressed and/or words are replaced by tags.

Morphological model (B+M) This model extends the **B** model by adding morphological features. This model aims to approximate the intuition that morphological features such as case are key for identifying the structure of free word order languages. As feature engineering may become in principle quite complex once it comes to morphology, we targeted fairly crude models with the goal of providing a proof of concept. Therefore the morphologically informed models use as input a rich set of morphological features specified in Figure 2 (right) predicted by the CRF tagger (Müller et al., 2013) with the same jackkniffing as before. The content of Figure 2 provides an explicit indication of the actual features defined in the original treebanks (see Seddah et al. (2013) and references therein for details), while the columns are indicative normalized names. For Basque most of the additional morphological features further encode case and verbal subcategorization. For French the *mwe* field abbreviates IOB predicted tags derived from multi-word expression annotations found in the original dataset.

Now let M be the set of values enumerated for a language in Figure 2 (right), we systematically added the following templates to model **B**:

$$\begin{array}{ll}
\mathbf{s_0.w_t.m\&s_1.w_t.m\&q_1.tag} & \forall \mathbf{m} \in M \\
\mathbf{s_0.w_t.m\&s_1.c_t\&q_1.m} & \forall \mathbf{m} \in M \\
\mathbf{s_0.c_t.m\&s_1.w_t.m\&q_1.m} & \forall \mathbf{m} \in M \\
\mathbf{s_0.w_t.m\&q_1.m\&q_2.tag} & \forall \mathbf{m} \in M \\
\mathbf{s_0.w_t.m\&q_1.tag\&q_2.m} & \forall \mathbf{m} \in M \\
\mathbf{s_0.c_t\&q_1.m\&q_2.m} & \forall \mathbf{m} \in M
\end{array}$$

Essentially the model expresses interactions between morphological features from the constituent heads on the top of the stack and the morphological features from the tokens at the beginning of the queue.

Mixed model (B+S+M) Our last model is the union of the span model (**B+S**) and the morphological model (**B+M**).

Results (development) We measured the impact of the model variations on the development set for c-parsing on the SPMRL data sets (Table 2). We immediately observe that modelling spans tends to improve the results, in particular for languages where the head annotation is more problematic: Arabic[4], Basque, German and Hungarian and also Swedish however. So the span-based model seems to improve the parser's robustness in cases when dependencies lack precision. For this model, the average behaviour is similar to that of Hall et al. (2014) although the variance is high.

On the other hand, the morphological model tends to be most important for languages where head annotation is easier: French, Korean, Polish and Swedish. It is key for very richly inflected languages such as Basque and Hungarian even though our head annotation is more approximative[5]. A

[4] Although not detailed in the paper, we also observe that for Arabic, the morphological features are generally predicted with a lower accurracy by the tagger than for other languages.

[5] As annotation schemes are not normalized across languages, it is important to stress that these observations are very unlikely to be representative of the linguistic properties of these languages.They are more likely to be a result of annotation choices. For example Korean is a strongly agglutinative language for which much of the morphology is already encoded in the tag set.

	Model	Arabic	Basque	French	German	Hebrew	Hungarian	Korean	Polish	Swedish	Avg
1	Base	79.46	74.67	79.66	82.61	90.43	84.34	81.96	91.68	75.60	82.26
2	Base+S	80.59	76.39	80.15	83.63	90.63	85.62	82.21	91.75	77.49	83.16
3	Base+M	80.17	83.69	**81.05**	83.66	90.40	87.75	82.79	92.72	77.50	84.41
4	Base+S+M	**81.25**	**84.01**	80.87	**84.08**	**90.69**	**88.27**	**83.09**	**92.78**	**77.87**	**84.77**
5	Hall-Klein 14	78.89	83.74	79.40	83.28	88.06	87.44	81.85	91.10	75.95	83.30

F1-scores provided by `evalb-spmrl` (Seddah et al., 2013). Takes punctuation into account and penalizes unparsed sentences.

Table 2: Development F-scores

comparison with Hall et al. (2014) also reveals that for Basque, Hungarian and Swedish, taking into account morphological information largely explains our improved results.

Results (test) We observe in Table 3 that our joint B+S+M model yields a state of the art c-parser on almost all languages considered[6]. It is quite clear that both our span and morphology enhanced models could be dramatically improved, but it shows that with reasonable feature engineering, these two sub-models are largely sufficient to improve the state of the art in c-parsing for these languages over strong baselines. Although in principle the Berkeley parsers (Petrov et al., 2006; Hall et al., 2014) are designed to be language-generic with an underlying design that is surprisingly accurate for free word order languages end up suffering from a lack of sensitivity to morphological information. Finally we also observe that our phrase structure parser clearly outperforms the TurboParser setup described by Fernández-González and Martins (2015) in which an elaborate output conversion procedure generates c-parses from d-parses.

Comparison with related work We conclude with a few comparisons with related work. This will enable us to show that our approach is not only accurate but also efficient. A comparison with dependency parsers will also allow us to better identify the properties of our proposal.

In order to test efficiency, we compared our parser to c-parsers trained on Penn Treebank (PTB) for which we have running times reported by Fernández-González and Martins (2015). This required first assigning heads, for which we used the Stanford tool for converting PTB to Basic Dependencies, and then used our **Robust** conversion method. We performed a simple test using the PTB standard split with the same experimental setting as before, except that we use the standard `evalb` scorer (Table 5). Although the time comparison remains indicative, it is clear that the parsing framework described in this paper is not only reasonably accurate on a fixed word order language such as English but it is also quite efficient. Parsing accuracies might be different with other head annotation schemes (See e.g. Elming et al. (2013) for illustrations). In our case, we compare the (B+S) model with automated head annotation to the Collins head annotation as implemented in the Standord CORE NLP library (Manning et al., 2014), where we can see that the Collins handcrafted head annotation yields better results than the automated one on English[7].

System (single parsers)	F1 (EVALB)	(Toks/sec)
Hall-Klein 14	88.6	12
StanfordSR	89.1	655
Charniak 00	89.5	-
This paper (B+S)	89.7	2150 ◇
This paper (B+S) [Collins]	90.0	2150 ◇
Petrov 06	90.1	169
Fernandez-Martins 15	90.2	957
Zhu et al. 13	90.4	1290

Alls scores and times except ◇ are measured by Fernández-González and Martins (2015) on an intel Xeon 2.3Ghz. ◇ denotes the use of a different architecture (2.4Ghz intel).

Table 5: Penn treebank test (WSJ 23)

The question is now to which extend c-trees encode meaningful dependencies? As lexicalized c-trees encode unlabeled dependency trees, our parser also directly outputs unlabeled d-trees by

[6]For Basque, our problem comes from a recurrent inconsistency in the SPMRL data set. As annotated in the c-trees, the punctuation induces a modification of the d-structure: c-trees encode a different governor for punctuation marks than d-trees. This not only causes problem to our head annotation procedure but also for the parser to solving these attachments. A simple correction results in a significant improvement of these parsing results. However we decided to leave the data untouched in order to preserve fair comparisons with other systems.

[7]This pattern does not seem to be systematic: on French we could also compare with head annotations described in (Arun and Keller, 2005) and we observed a slight improvement when using the automated procedure.

Parser (single)	Arabic	Basque	French	German	Hebrew	Hungarian	Korean	Polish	Swedish	Avg
Petrov 06	79.19	70.50	80.38	78.30	86.96	81.62	71.42	79.23	79.19	78.45
Petrov 06 + tags	78.66	74.74	79.76	78.28	85.42	85.22	78.56	86.75	80.64	81.17
Hall-Klein 14	78.75	83.39	79.70	78.43	87.18	88.25	80.18	90.66	82.00	83.72
Fernandez-Martins 15	-	**85.90**	78.75	78.66	88.97	88.16	79.28	91.20	82.80	84.22
This paper (B+S+M)	**81.31**	84.94	**80.84**	**79.26**	**89.65**	**90.14**	**82.65**	**92.66**	**83.24**	**85.42**
Best semi/ensemble	81.32	88.24	82.53	81.66	89.80	91.72	83.81	90.50	85.50	86.72

F-scores provided by `evalb-spmrl` (Seddah et al., 2013). It takes punctuation into account and penalizes unparsed sentences. The average ignores Arabic for comparison with TurboParser. Petrov 06 + tags is the Berkeley parser with externally predicted pos tags (Seddah et al., 2013)

Table 3: Multilingual test (F-scores, phrase structure parsing)

System	English	French	Korean	Hebrew	Polish	Swedish	Arabic	Basque	German	Hungarian
This paper (B+S+M)	91.75	**86.68**	**87.22**	**85.28**	**88.61**	**86.22**	80.64	73.68	67.20	74.46
Best d-parser (single)	**91.95**	85.80	85.84	81.05	88.12	84.54	**84.57**	**84.33**	**87.65**	**83.71**
Best semi/ensemble	-	89.19	89.10	87.41	91.75	88.48	88.32	89.96	91.64	89.81

Unlabeled Accuracy Scores. Best other is the best single parser UAS result reported either in SPMRL 13 or SPMRL 14 shared tasks.
Best ensemble is the best semi-supervised or ensemble system from either SPMRL 13 or SPMRL 14 (Björkelund et al., 2013; Björkelund et al., 2014).

Table 4: Multilingual test (UAS, dependency parsing)

simply reading them off from the lexicalized c-structure. We report in Table 4 the UAS evaluation of those dependencies and we compare them to the best results obtained by dependency parsers in both SPMRL13 and SPMRL14 shared tasks. For each language, the comparison is made with the best single dependency parsing system[8]. For English we compare against Standard TurboParser - which seems to be the most similar to our system- when parsing to Basic Stanford dependencies. The comparison with semi-supervised and ensemble parsers still provides a reasonable upperline (Björkelund et al., 2013).

As can be seen in Table 4, our results partly generalize the observation summarized by Cer et al. (2010) and Kong and Smith (2014) that phrase structure parsers tend to provide better dependencies than genuine dependency parsers for parsing to Stanford Dependencies. For English, our UAS is similar to that of TurboParser, but in a broader multilingual framework, the left side of the table shows that the unlabeled dependencies are clearly better than those of genuine dependency parsers. On the right side of the table are languages for which our dependencies are actually worse. This is not a surprise, since these are also the languages for which head annotation was more problematic in the first place. This last observation suggests that a lexicalized c-parser can also provide very accurate dependencies. A way to further generalize this observation to problematic languages would be either to design a less immediate post-processing conversion scheme or to further normalize the data set to obtain the correct heads from the outset.

[8]In practice it turns out that these are either DYALOG-SR (de La Clergerie, 2013) or sometimes MALTOPTIMIZER (Ballesteros and Nivre, 2012)

6 Conclusion

Lexicalized phrase structure parsing of morphologically rich languages used to be difficult since existing implementations targeting essentially English or Chinese do not allow a straightforward integration of morphology. Given multi-view treebanks, we achieve multilingual parsing with a language-agnostic head annotation procedure. Once this procedure has created the required data representation for lexicalized parsing, only modest and weakly language dependent feature engineering is required to achieve state-of-the-art accuracies on all languages considered: a minimal interface with morphology already contributes to improving accuracy, and this is specifically the case when heads are accurately identified. When heads are only approximatively identified, span-based configurational modelling tends to correct the approximation.

Leaving aside details concerning conversion and data normalization, we generally found that the unlabeled dependencies modelled by the lexicalized c-parser also tend to be highly accurate. For languages where c-annotations and d-annotations are less compatible, additional language renormalizations would help to get better comparisons.

As suggested in this paper, future work for parsing morphologically rich languages will require to focus both on feature selection and on the interface between syntax and morphology, which means in our case the interface between the segmenter, the tagger and the parser.

Acknowledgments

The author wishes to thank Djamé Seddah for insigthful discussions regarding the work reported in this paper as well as R. Bawden, M. Coavoux and B. Sagot for their careful proofreading.

References

Abhishek Arun and Frank Keller. 2005. Lexicalization in crosslinguistic probabilistic parsing: The case of french. In *Association for Computational Linguistics*.

Miguel Ballesteros and Joakim Nivre. 2012. Maltoptimizer: An optimization tool for maltparser. In *13th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*.

Rajesh Bhatt, Bhuvana Narasimhan, Martha Palmer, Owen Rambow, Dipti Sharma, and Fei Xia. 2009. A multi-representational and multi-layered treebank for hindi/urdu. In *Proceedings of the Third Linguistic Annotation Workshop (LAW III)*.

Anders Björkelund, Ozlem Cetinoglu, Richárd Farkas, Thomas Mueller, and Wolfgang Seeker. 2013. (re)ranking meets morphosyntax: State-of-the-art results from the SPMRL 2013 shared task. In *Proceedings of the Fourth Workshop on Statistical Parsing of Morphologically-Rich Languages*.

Anders Björkelund, Özlem Cetinoğlu, Agnieszka Faleńska, Richárd Farkas, Thomas Müller, Wolfgang Seeker, and Zsolt Szántó. 2014. The ims-wrocaw-szeged-cis entry at the spmrl 2014 shared task: Reranking and morphosyntax meet unlabeled data. In *Fifth Workshop on Statistical Parsing of Morphologically-Rich Languages*.

Daniel M. Cer, Marie-Catherine de Marneffe, Daniel Jurafsky, and Christopher D. Manning. 2010. Parsing to stanford dependencies: Trade-offs between speed and accuracy. In *Proceedings of the Language Ressources and Evaluation Conference (LREC)*.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *ANLP*, pages 132–139.

Michael Collins, Jan Hajic, Lance A. Ramshaw, and Christoph Tillmann. 1999. A statistical parser for Czech. In *27th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Michael Collins. 2003. Head-driven statistical models for natural language parsing. *Computational Linguistics*, 29(4):589–637.

Benoit Crabbé. 2014. An LR-inspired generalized lexicalized phrase structure parser. In *25th International Conference on Computational Linguistics (COLING)*.

Eric Villemonte de La Clergerie. 2013. Exploring beam-based shift-reduce dependency parsing with dyalog: Results from the spmrl 2013 shared task. In *4th Workshop on Statistical Parsing of Morphologically Rich Languages (SPMRL2013)*.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. 2006. Generating typed dependency parses from phrase structure parses. In *Language Ressources and Evaluation Conference (LREC)*.

Jakob Elming, Anders Johannsen, Sigrid Klerke, Emanuele Lapponi, Hector Martinez, and Anders Sogaard. 2013. Down-stream effects of tree-to-dependency conversions. In *Proceedings of NAACL-HLT*.

Daniel Fernández-González and André F. T. Martins. 2015. Parsing as reduction. In *Proceeding of the Association of Computational Linguistics (ACL), 2015*.

David Hall, Greg Durrett, and Dan Klein. 2014. Less grammar, more features. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014*, pages 228–237.

Liang Huang, Suphan Fayong, and Yang Guo. 2012. Structured perceptron with inexact search. In *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics (NAACL-HLT)*.

Dan Klein and Christopher D. Manning. 2004. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, pages 478–485.

Lingpeng Kong and Noah A. Smith. 2014. An empirical comparison of parsing methods for stanford dependencies. *CoRR*, abs/1404.4314.

Marco Kuhlmann and Joakim Nivre. 2006. Mildly non-projective dependency structures. In *21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (ACL/COLING)*.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*.

Ryan T. McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajic. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP)*.

Ryan McDonald, J. Nivre, Y. Quirmbach-Brundage, Y. Goldberg, D. Das, K. Ganchev, K. Hall, S. Petrov, H. Zhang, O. Tackstrom, C. Bedini, N. Bertomeu Castello, and J. Lee. 2013. Universal dependency annotation for multilingual parsing. In *Proceeding of the Association of Computational Linguistics (ACL)*.

Haitao Mi and Liang Huang. 2015. Shift-reduce constituency parsing with dynamic programming and pos tag lattice. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT)*.

Thomas Müller, Helmut Schmid, and Hinrich Schütze. 2013. Efficient higher-order crfs for morphological tagging. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Joakim Nivre and Mario Scholz. 2004. Deterministic dependency parsing of english text. In *COLING 2004, 20th International Conference on Computational Linguistics*.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference (ACL/COLING)*.

Likun Qiu, Yue Zhang, Peng Jin, and Houfeng Wang. 2014. Multi-view chinese treebanking. In *25th International Conference on Computational Linguistics (COLING)*.

Kenji Sagae and Alon Lavie. 2006. A best-first probabilistic shift-reduce parser. In *21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (ACL/COLING)*.

Djamé Seddah, Reut Tsarfaty, Sandra Kubler, Marie Candito, Jinho D. Choi, Richárd Farkas, Jennifer Foster, Iakes Goenaga, Koldo Gojenola Galletebeitia, Yoav Goldberg, Spence Green, Nizar Habash, Marco Kuhlmann, Wolfgang Maier, Yuval Marton, Joakim Nivre, Adam Przepiórkowski, Ryan Roth, Wolfgang Seeker, Yannick Versley, Veronika Vincze, Marcin Woliski, Alina Wróblewska, and Eric Villemonte de la Clergerie. 2013. Overview of the spmrl 2013 shared task: A cross-framework evaluation of parsing morphologically rich languages. In *Proceedings of the Fourth SPMRL Workshop, Seattle, USA*.

Shai Shalev-Shwartz, Yoram Singer, and Nathan Srebro. 2007. Pegasos: Primal estimated sub-gradient solver for SVM. In *Machine Learning, Proceedings of the Twenty-Fourth International Conference (ICML)*.

Veronika Vincze, Dora Szauter, Attila Almasi, Gyorgy Mora, Zoltan Alexin, and Janos Csirik. 2010. Hungarian dependency treebank. In *Proceedings of Language Ressources and Evalutation Conference (LREC)*.

Kai Zhao and Liang Huang. 2013. Minibatch and parallelization for online large margin structured learning. In *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics (NAACL-HLT)*.

Muhua Zhu, Yue Zhang, Wenliang Chen, Min Zhang, and Jingbo Zhu. 2013. Fast and accurate shift-reduce constituent parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, (ACL)*.

Hierarchical Low-Rank Tensors for Multilingual Transfer Parsing

Yuan Zhang
CSAIL, MIT
yuanzh@csail.mit.edu

Regina Barzilay
CSAIL, MIT
regina@csail.mit.edu

Abstract

Accurate multilingual transfer parsing typically relies on careful feature engineering. In this paper, we propose a hierarchical tensor-based approach for this task. This approach induces a compact feature representation by combining atomic features. However, unlike traditional tensor models, it enables us to incorporate prior knowledge about desired feature interactions, eliminating invalid feature combinations. To this end, we use a hierarchical structure that uses intermediate embeddings to capture desired feature combinations. Algebraically, this hierarchical tensor is equivalent to the sum of traditional tensors with shared components, and thus can be effectively trained with standard online algorithms. In both unsupervised and semi-supervised transfer scenarios, our hierarchical tensor consistently improves UAS and LAS over state-of-the-art multilingual transfer parsers and the traditional tensor model across 10 different languages.[1]

1 Introduction

The goal of multilingual syntactic transfer is to parse a resource lean target language utilizing annotations available in other languages. Recent approaches have demonstrated that such transfer is possible, even in the absence of parallel data. As a main source of guidance, these methods rely on the commonalities in dependency structures across languages. These commonalities manifest themselves through a broad and diverse set of indicators, ranging from standard arc features used in monolingual parsers to typological properties needed to guide cross-lingual sharing (e.g., verb-subject ordering preference). In fact, careful feature engineering has been shown to play a crucial role in state-of-the-art multilingual transfer parsers (Täckström et al., 2013).

Verb-subject: {head POS=VERB} ∧ {modifier POS=NOUN} ∧{label=subj} ∧ {direction=LEFT}∧ {82A=SV}
Noun-adjective: {head POS=NOUN} ∧ {modifier POS=ADJ}∧ {direction=LEFT} ∧ {87A=Adj-Noun}

Table 1: Example verb-subject and noun-adjective typological features. 82A and 87A denote the WALS (Dryer et al., 2005) feature codes for verb-subject and noun-adjective ordering preferences.

Tensor-based models are an appealing alternative to manual feature design. These models automatically induce a compact feature representation by factorizing a tensor constructed from atomic features (e.g., the head POS). No prior knowledge about feature interactions is assumed. As a result, the model considers all possible combinations of atomic features, and addresses the parameter explosion problem via a low-rank assumption.

In the multilingual transfer setting, however, we have some prior knowledge about legitimate feature combinations. Consider for instance a typological feature that encodes verb-subject preferences. As Table 1 shows, it is expressed as a conjunction of five atomic features. Ideally, we would like to treat this composition as a single non-decomposable feature. However, the traditional tensor model decomposes this feature into multiple dimensions, and considers various combinations of these features as well as their individual interactions with other features. Moreover, we want to avoid invalid combinations that con-

[1]The source code is available at https://github.com/yuanzh/TensorTransfer.

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1857–1867,
Lisbon, Portugal, 17-21 September 2015.

join the above feature with unrelated atomic features. For instance, there is no point to constructing features of the form {head POS=ADJ} ∧ {head POS=VERB} ∧ ··· ∧ {82A=SV} as the head *POS* takes a single value. However, the traditional tensor technique still considers these unobserved feature combinations, and assigns them non-zero weights (see Section 7). This inconsistency between prior knowledge and the low-rank assumption results in a sub-optimal parameter estimation.

To address this issue, we introduce a hierarchical tensor model that constrains parameter representation. The model encodes prior knowledge by explicitly excluding undesired feature combinations over the same atomic features. At the bottom level of the hierarchy, the model constructs combinations of atomic features, generating intermediate embeddings that represent the legitimate feature groupings. For instance, these groupings will not combine the verb-subject ordering feature and the POS head feature. At higher levels of the hierarchy, the model combines these embeddings as well as the expert-defined typological features over the same atomic features. The hierarchical tensor is thereby able to capture the interaction between features at various subsets of atomic features. Algebraically, the hierarchical tensor is equivalent to the sum of traditional tensors with shared components. Thus, we can use standard online algorithms for optimizing the low-rank hierarchical tensor.

We evaluate our model on labeled dependency transfer parsing using the newly released multilingual universal dependency treebank (McDonald et al., 2013). We compare our model against the state-of-the-art multilingual transfer dependency parser (Täckström et al., 2013) and the direct transfer model (McDonald et al., 2011). All the parsers utilize the same training resources but with different feature representations. When trained on source languages alone, our model outperforms the baselines for 7 out of 10 languages on both unlabeled attachment score (UAS) and labeled attachment score (LAS). On average, it achieves 1.1% UAS improvement over Täckström et al. (2013)'s model and 4.8% UAS over the direct transfer. We also consider a semi-supervised setting where multilingual data is augmented with 50 annotated sentences in the target language. In this case, our model achieves improvement of 1.7% UAS over Täckström et al. (2013)'s model and 4.5% UAS over the direct transfer.

2 Related Work

Multilingual Parsing The lack of annotated parsing resources for the vast majority of world languages has kindled significant interest in multi-source parsing transfer (Hwa et al., 2005; Durrett et al., 2012; Zeman and Resnik, 2008; Yu et al., 2013b; Cohen et al., 2011; Rasooli and Collins, 2015). Recent research has focused on the non-parallel setting, where transfer is driven by cross-lingual commonalities in syntactic structure (Naseem et al., 2010; Täckström et al., 2013; Berg-Kirkpatrick and Klein, 2010; Cohen and Smith, 2009; Duong et al., 2015).

Our work is closely related to the selective-sharing approaches (Naseem et al., 2012; Täckström et al., 2013). The core of these methods is the assumption that head-modifier attachment preferences are universal across different languages. However, the sharing of arc direction is selective and is based on typological features. While this selective sharing idea was first realized in the generative model (Naseem et al., 2012), higher performance was achieved in a discriminative arc-factored model (Täckström et al., 2013). These gains were obtained by a careful construction of features templates that combine standard dependency parsing features and typological features. In contrast, we propose an automated, tensor-based approach that can effectively capture the interaction between these features, yielding a richer representation for cross-lingual transfer. Moreover, our model handles labeled dependency parsing while previous work only focused on the unlabeled dependency parsing task.

Tensor-based Models Our approach also relates to prior work on tensor-based modeling. Lei et al. (2014) employ three-way tensors to obtain a low-dimensional input representation optimized for parsing performance. Srikumar and Manning (2014) learn a multi-class label embedding tailored for document classification and POS tagging in the tensor framework. Yu and Dredze (2015), Fried et al. (2015) apply low-rank tensor decompositions to learn task-specific word and phrase embeddings. Other applications of tensor framework include low-rank regularization (Primadhanty et al., 2015; Quattoni et al., 2014; Singh et al., 2015) and neural tensor networks (Socher et

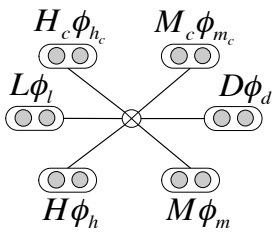


Figure 1: Visual representation for traditional multiway tensor.

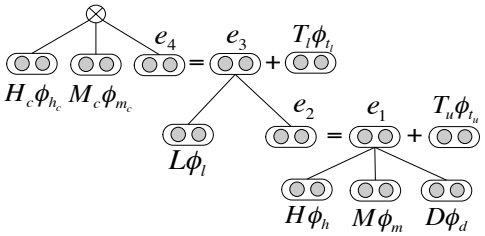


Figure 2: Visual representation for hierarchical tensor, represented as a tree structure. The tensor first captures the low-level interaction ($H\phi_h$, $M\phi_m$ and $D\phi_d$) by an element-wise product, and then combines the intermediate embedding with other components higher in the hierarchy, e.g. e_2 and $L\phi_l$. The equations show that we composite two representations by an element-wise sum.

al., 2013; Yu et al., 2013a). While these methods can automatically combine atomic features into a compact composite representation, they cannot take into account constraints on feature combination. In contrast, our method can capture features at different composition levels, and more generally can incorporate structural constraints based on prior knowledge. As our experiments show, this approach delivers higher transfer accuracy.

3 Hierarchical Low-rank Scoring for Transfer Parsing

3.1 Background

We start by briefly reviewing the traditional three-way tensor scoring function (Lei et al., 2014). The three-way tensor characterizes each arc $h \rightarrow m$ using the tensor-product over three feature vectors: the head vector ($\phi_h \in \mathbb{R}^n$), the modifier vector ($\phi_m \in \mathbb{R}^n$) and the arc vector ($\phi_{h\rightarrow m} \in \mathbb{R}^l$). ϕ_h captures atomic features associated with the head, such as its POS tag and its word form. Similarly, ϕ_m and $\phi_{h\rightarrow m}$ capture atomic features associated with the modifier and the arc respectively. The tensor-product of these three vectors is a rank-1 tensor:

$$\phi_h \otimes \phi_m \otimes \phi_{h\rightarrow m} \in \mathbb{R}^{n\times n\times l}$$

This rank-1 tensor captures all possible combinations of the atomic features in each vector, and therefore significantly expands the feature set. The tensor score is the inner product between a three-way parameter tensor $A \in \mathbb{R}^{n\times n\times l}$ and this rank-1 feature tensor:

$$vec(A) \cdot vec(\phi_h \otimes \phi_m \otimes \phi_{h\rightarrow m})$$

where $vec(\cdot)$ denotes the vector representation of a tensor. This tensor scoring method avoids the parameter explosion and overfitting problem by assuming a low-rank factorization of the parameters A. Specifically, A is decomposed into the sum of r rank-1 components:

$$A = \sum_{i=1}^{r} U(i) \otimes V(i) \otimes W(i)$$

where r is the rank of the tensor, $U, V \in \mathbb{R}^{r\times n}$ and $W \in \mathbb{R}^{r\times l}$ are parameter matrices. $U(i)$ denotes the i-th row of matrix U and similarly for $V(i)$ and $W(i)$. Figure 1 shows the representation of a more general multiway factorization. With this factorization, the model effectively alleviates the feature explosion problem by projecting sparse feature vectors into dense r-dimensional embeddings via U, V and W. Subsequently, the score is computed as follows:

$$S_{tensor}(h \rightarrow m) = \sum_{i=1}^{r} [U\phi_h]_i [V\phi_m]_i [W\phi_{h\rightarrow m}]_i$$

where $[\cdot]_i$ denotes the i-th element of the matrix.

In multilingual transfer, however, we want to incorporate typological features that do not fit in any of the components. For example, if we add the verb-subject ordering preference into $\phi_{h\rightarrow m}$, the tensor will represent the concatenation of this preference with a noun-adjective arc, even though this feature should never trigger.

3.2 Hierarchical Low-rank Tensor

To address this issue, we propose the hierarchical factorization of tensor parameters.[2] The key idea is to generate intermediate embeddings that capture the interaction of the same set of atomic

[2] In this section we focus on delexicalized transfer, and describe the lexicalization process in Section 3.3.

features as other expert-defined features. As Figure 2 shows, this design enables the model to handle expert-defined features over various subsets of the atomic features.

Now, we will illustrate this idea in the context of multilingual parsing. Table 2 summarizes the notations of the feature vectors and the corresponding parameters. Specifically, for each arc $h \rightarrow m$ with label l, we first compute the intermediate feature embedding e_1 that captures the interaction between the head ϕ_h, the modifier ϕ_m and the arc direction and length ϕ_d, by an element-wise product.

$$[e_1]_i = [H\phi_h]_i[M\phi_m]_i[D\phi_d]_i \quad (1)$$

where $[\cdot]_i$ denotes the i-th value of the feature embedding, and H, M and D are the parameter matrices as in Table 2. The embedding e_1 captures the unconstrained interaction over the *head*, the *modifier* and the *arc*. Note that ϕ_{t_u} includes expert-defined typological features that rely on the specific values of the head POS, the modifier POS and the arc direction, such as the example noun-adjective feature in Table 1. Therefore, the embedding $T_u\phi_{t_u}$ captures an expert-defined interaction over the *head*, the *modifier* and the *arc*. Thus e_1 and $T_u\phi_{t_u}$ provide two different representations of the same set of atomic features (e.g. the *head*) and our prior knowledge motivates us to exclude the interaction between them since the low-rank assumption would not apply. Thus, we combine e_1 and $T_u\phi_{t_u}$ as e_2 using an element-wise sum

$$[e_2]_i = [e_1]_i + [T_u\phi_{t_u}]_i \quad (2)$$

and thereby avoid such combinations. As Figure 2 shows, e_2 in turn is used to capture the higher level interaction with arc label features ϕ_l,

$$[e_3]_i = [L\phi_l]_i[e_2]_i \quad (3)$$

Now e_3 captures the interaction between head, modifier, arc direction, length and label. It is over the same set of atomic features as the typological features that depend on arc labels ϕ_{t_l}, such as the example verb-subject ordering feature in Table 1. Therefore, we sum over these embeddings as

$$[e_4]_i = [e_3]_i + [T_l\phi_{t_l}]_i \quad (4)$$

Finally, we capture the interaction between e_4 and context feature embeddings $H_c\phi_{h_c}$ and $M_c\phi_{m_c}$ and compute the tensor score as

Notation	Description
H, ϕ_h M, ϕ_m	Head/modifier POS tag
D, ϕ_d	Arc length and direction
L, ϕ_l	Arc label
T_u, ϕ_{t_u}	Typological features that depend on head/modifier POS but not arc label
T_l, ϕ_{t_l}	Typological features that depend on arc label
H_c, ϕ_{h_c} M_c, ϕ_{m_c}	POS tags of head/modifier neighboring words

Table 2: Notations and descriptions of parameter matrices and feature vectors in our hierarchical tensor model.

$$S_{tensor}(h \xrightarrow{l} m) = \sum_{i=1}^{r} [H_c\phi_{h_c}]_i[M_c\phi_{m_c}][e_4]_i \quad (5)$$

By combining Equation 1 to 5, we observe that our hierarchical tensor score decomposes into three multiway tensor scoring functions.

$$\begin{aligned} S_{tensor}(h \xrightarrow{l} m) &= \sum_{i=1}^{r} [H_c\phi_{h_c}]_i[M_c\phi_{m_c}]_i \\ &\quad \Big\{[T_l\phi_{t_l}]_i + [L\phi_l]_i \\ &\quad \Big([T_u\phi_{t_u}]_i + [H\phi_h]_i[M\phi_m]_i[D\phi_d]_i\Big)\Big\} \\ &= \sum_{i=1}^{r} \Big\{[H_c\phi_{h_c}]_i[M_c\phi_{m_c}]_i[T_l\phi_{t_l}]_i \\ &+ [H_c\phi_{h_c}]_i[M_c\phi_{m_c}]_i[L\phi_l]_i[T_u\phi_{t_u}]_i \\ &+ [H_c\phi_{h_c}]_i[M_c\phi_{m_c}]_i[L\phi_l]_i[H\phi_h]_i[M\phi_m]_i[D\phi_d]_i\Big\} \end{aligned} \quad (6)$$

This decomposition provides another view of our tensor model. That is, our hierarchical tensor is algebraically equivalent to the sum of three multiway tensors, where H_c, M_c and L are shared.[3] From this perspective, we can see that our tensor model effectively captures the following three sets of combinations over atomic features:

f_1: $\phi_{h_c} \otimes \phi_{m_c} \otimes \phi_{t_l}$
f_2: $\phi_{h_c} \otimes \phi_{m_c} \otimes \phi_l \otimes \phi_{t_u}$
f_3: $\phi_{h_c} \otimes \phi_{m_c} \otimes \phi_l \otimes \phi_h \otimes \phi_m \otimes \phi_d$

[3] We could also associate each multiway tensor with a different weight. In our work, we keep them weighted equally.

The last set of features f_3 captures the interaction across standard atomic features. The other two sets of features f_1 and f_2 focus on combining atomic typological features with atomic label and context features. Consequently, we explicitly assign zero weights for invalid assignments, by excluding the combination of ϕ_{t_u} with ϕ_h and ϕ_m.

3.3 Lexicalization Components

In order to encode lexical information in our tensor-based model, we add two additional components, $H_w\phi_{h_w}$ and $M_w\phi_{m_w}$, for head and modifier lexicalization respectively. We compute the final score as the interaction between the delexicalized feature embedding in Equation 5 and the lexical components. Specifically:

$$[e_5]_i = [H_c\phi_{h_c}]_i[M_c\phi_{m_c}]_i[e_4]_i$$
$$S_{tensor}(h \xrightarrow{l} m) = \sum_{i=1}^{r}[H_w\phi_{h_w}]_i[M_w\phi_{m_w}]_i[e_5]_i \quad (7)$$

where e_5 is the embedding that represents the delexicalized transfer results. We describe the features in ϕ_{h_w} and ϕ_{m_w} in Section 5.

3.4 Combined Scoring

Similar to previous work on low-rank tensor scoring models (Lei et al., 2014; Lei et al., 2015), we combine the traditional scoring and the low-rank tensor scoring. More formally, for a sentence $\mathbf{x}$ and a dependency tree $\mathbf{y}$, our final scoring function has the form

$$S(\mathbf{x},\mathbf{y}) = \gamma \sum_{h \xrightarrow{l} m \in \mathbf{y}} \mathbf{w}\cdot\phi(h \xrightarrow{l} m) + (1-\gamma)\sum_{h \xrightarrow{l} m \in \mathbf{y}} S_{tensor}(h \xrightarrow{l} m) \quad (8)$$

where $\phi(h \xrightarrow{l} m)$ is the traditional features for arc $h \rightarrow m$ with label l and $\mathbf{w}$ is the corresponding parameter vector. $\gamma \in [0,1]$ is the balancing hyper-parameter and we tune the value on the development set. The parameters in our model are $\theta = (\mathbf{w}, H, M, D, L, T_u, T_l, H_c, M_c)$, and our goal is to optimize all parameters given the training set.

4 Learning

In this section, we describe our learning method.[4] Following standard practice, we optimize the parameters $\theta = (\mathbf{w}, H, M, D, L, T_u, T_l, H_c, M_c)$ in a maximum soft-margin framework, using online passive-aggressive (PA) updates (Crammer et al., 2006).

For tensor parameter update, we employ the joint update method originally used by Lei et al. (2015) in the context of four-way tensors. While our tensor has a very high order (8 components for the delexicalized parser and 10 for the lexicalized parser) and is hierarchical, the gradient computation is nevertheless similar to that of traditional tensors. As described in Section 3.2, we can view our hierarchical tensor as the combination of three multiway tensors with parameter sharing. Therefore, we can compute the gradient of each multiway tensor and take the sum accordingly. For example, the gradient of the label component is

$$\partial L = \sum_{h \xrightarrow{l} m \in \mathbf{y}^*} \Big((H_c\phi_{h_c}) \odot (M_c\phi_{m_c}) \odot [(T_u\phi_{t_u}) + (H\phi_h) \odot (M\phi_m) \odot (D\phi_d)]\Big) \otimes \phi_l - \sum_{h \xrightarrow{l} m \in \tilde{\mathbf{y}}} \Big((H_c\phi_{h_c}) \odot (M_c\phi_{m_c}) \odot [(T_u\phi_{t_u}) + (H\phi_h) \odot (M\phi_m) \odot (D\phi_d)]\Big) \otimes \phi_l \quad (9)$$

where $\odot$ is the element-wise product and $+$ denotes the element-wise addition. $\mathbf{y}^*$ and $\tilde{\mathbf{y}}$ are the gold tree and the maximum violated tree respectively. For each sentence $\mathbf{x}$, we find $\tilde{\mathbf{y}}$ via cost-augmented decoding.

Tensor Initialization Given the high tensor order, initialization has a significant impact on the learning quality. We extend the previous power method for high-order tensor initialization (Lei et al., 2015) to the hierarchical structure using the algebraic view as in computing the gradient.

Briefly, the power method incrementally computes the most important rank-1 component for $H(i)$, $M(i)$ etc, for $i = 1 \ldots r$. In each iteration, the algorithm updates each component by taking the multiplication between the tensor T and the rest of the components. When we update the label component l, we do the multiplication for different

[4] Our description focuses on delexicalized transfer, and we can easily extend the method to the lexicalized case.

Feature	Description
82A	Order of Subject and Verb
83A	Order of Object and Verb
85A	Order of Adposition and Noun Phrase
86A	Order of Genitive and Noun
87A	Order of Adjective and Noun

Table 3: Typological features from WALS (Dryer et al., 2005) used to build the feature templates in our work, inspired by Naseem et al. (2012). Unlike previous work (Naseem et al., 2012; Täckström et al., 2013), we use 82A and 83A instead of 81A (order of subject, object and verb) because we can distinguish between subject and object relations based on dependency labels.

multiway tensors and then take the sum.

$$l = \langle T_0, h_c, m_c, -, t_u \rangle + \langle T_1, h_c, m_c, -, h, m, d \rangle$$

where the operator $\langle T_0, h_c, m_c, -, t_u \rangle$ returns a vector in which the i-th element is computed as $\sum_{uvw} T_0(i, u, v, w) h_c(u) m_c(v) t_u(w)$. The algorithm updates other components in a similar fashion until convergence.

5 Features

Linear Scoring Features Our traditional linear scoring features in $\phi(h \xrightarrow{l} m)$ are mainly drawn from previous work (Täckström et al., 2013). Table 3 lists the typological features from "The World Atlas of Language Structure (WALS)" (Dryer et al., 2005) used to build the feature templates in our work. We use 82A and 83A for verb-subject and verb-object order respectively because we can distinguish between these two relations based on dependency labels. Table 4 summarizes the typological feature templates we use. In addition, we expand features with dependency labels to enable labeled dependency parsing.

Tensor Scoring Features For our tensor model, feature vectors listed in Table 2 capture the five types of atomic features as follows:

(a) ϕ_h, ϕ_m: POS tags of the head or the modifier.
(b) ϕ_{h_c}, ϕ_{m_c}: POS tags of the left/right neighboring words.
(c) ϕ_l: dependency labels.
(d) ϕ_d: dependency length conjoined with direction.
(e) ϕ_{t_u}, ϕ_{t_l}: selectively shared typological features, as described in Table 4.

ϕ_{t_l}	$dir \cdot$82A$\cdot\delta(hp$=VERB$\wedge mp$=NOUN$\wedge$subj$\in l)$
	$dir \cdot$82A$\cdot\delta(hp$=VERB$\wedge mp$=PRON$\wedge$subj$\in l)$
	$dir \cdot$83A$\cdot\delta(hp$=VERB$\wedge mp$=NOUN$\wedge$obj$\in l)$
	$dir \cdot$83A$\cdot\delta(hp$=VERB$\wedge mp$=PRON$\wedge$obj$\in l)$
ϕ_{t_u}	$dir \cdot$85A$\cdot\delta(hp$=ADP$\wedge mp$=NOUN$)$
	$dir \cdot$85A$\cdot\delta(hp$=ADP$\wedge mp$=PRON$)$
	$dir \cdot$86A$\cdot\delta(hp$=NOUN$\wedge mp$=NOUN$)$
	$dir \cdot$87A$\cdot\delta(hp$=ADJ$\wedge mp$=NOUN$)$

Table 4: Typological feature templates used in our work. hp/mp are POS tags of the head/modifier. $dir \in \{$LEFT, RIGHT$\}$ denotes the arc direction. 82A-87A denote the WALS typological feature value. $\delta(\cdot)$ is the indicator function. subj $\in l$ denotes that the arc label l indicates a subject relation, and similarly for obj $\in l$.

We further conjoin atomic features (b) and (d) with the family and the typological class of the language, because the arc direction and the word order distribution depends on the typological property of languages (Täckström et al., 2013). We also add a bias term into each feature vector.

Partial Lexicalization We utilize multilingual word embeddings to incorporate partial lexical information in our model. We use the CCA method (Faruqui and Dyer, 2014) to generate multilingual word embeddings. Specifically, we project word vectors in each non-English language to the English embedding space. To reduce the noise from the automatic projection process, we only incorporate lexical information for the top-100 most frequent words in the following closed classes: pronoun, determiner, adposition, conjunction, particle and punctuation mark. Therefore, we call this feature extension partial lexicalization.[5]

We follow previous work (Lei et al., 2014) for adding embedding features. For the linear scoring model, we simply append the head and the modifier word embeddings after the feature vector. For the tensor-based model, we add each entry of the word embedding as a feature value into ϕ_{h_w} and ϕ_{m_w}. In addition, we add indicator features for the English translation of words because this improves performance in preliminary experiments. For example, for the German word und, we add the word and as a feature.

[5] In our preliminary experiments, we observe that our lexicalized model usually outperforms the unlexicalized counterparts by about 2%.

6 Experimental Setup

Dataset We evaluate our model on the newly released multilingual universal dependency treebank v2.0 (McDonald et al., 2013) that consists of 10 languages: English (EN), French (FR), German (DE), Indonesian (ID), Italian (IT), Japanese (JA), Korean (KO), Brazilian-Portuguese (PT), Spanish (ES) and Swedish (SV). This multilingual treebank is annotated with a universal POS tagset and a universal dependency label set. Therefore, this dataset is an excellent benchmark for cross-lingual transfer evaluation. For POS tags, the gold universal annotation used the coarse tagset (Petrov et al., 2011) that consists of 12 tags: noun, verb, adjective, adverb, pronoun, determiner, adposition, numeral, conjunction, particle, punctuation mark, and a catch-all tag X. For dependency labels, the universal annotation developed the Stanford dependencies (De Marneffe and Manning, 2008) into a rich set of 40 labels. This universal annotation enables labeled dependency parsing in cross-lingual transfer.

Evaluation Scenarios We first consider the unsupervised transfer scenario, in which we assume no target language annotations are available. Following the standard setup, for each target language evaluated, we train our model on the concatenation of the training data in all other source languages.

In addition, we consider the semi-supervised transfer scenario, in which we assume 50 sentences in the target language are available with annotation. However, we observe that random sentence selection of the supervised sample results in a big performance variance. Instead, we select sentences that contain patterns that are absent or rare in source language treebanks. To this end, each time we greedily select the sentence that minimizes the KL divergence between the trigram distribution of the target language and the trigram distribution of the training data after adding this sentence. The training data includes both the target and the source languages. The trigrams are based on universal POS tags. Note that our method does not require any dependency annotations. To incorporate the new supervision, we simply add the new sentences into the original training set, weighing their impact by a factor of 10.

Baselines We compare against different variants of our model.

- **Direct**: a direct transfer baseline (McDonald et al., 2011) that uses only delexicalized features in the MSTParser (McDonald et al., 2005).
- **NT-Select**: our model without the tensor component. This baseline corresponds to the prior feature-based transfer method (Täckström et al., 2013) with extensions to labeled parsing, lexicalization and semi-supervised parsing.[6]
- **Multiway**: tensor-based model where typological features are added as an additional component and parameters are factorized in the multiway structure similarly as in Figure 1.
- **Sup50**: our model trained only on the 50 sentences in the target language in the semi-supervised scenario.

In all the experiments we incorporate partial lexicalization for all variants of our model and we focus on labeled dependency parsing.

Supervised Upper Bound As a performance upper bound, we train the RBGParser (Lei et al., 2014), the state-of-the-art tensor-based parser, on the full target language training set. We train the first-order model[7] with default parameter settings, using the current version of the code.[8]

Evaluation Measures Following standard practices, we report unlabeled attachment score (UAS) and labeled attachment score (LAS), excluding punctuation. For all experiments, we report results on the test set and omit the development results because of space.

Experimental Details For all experiments, we use the arc-factored model and use Eisner's algorithm (Eisner, 1996) to infer the projective Viterbi parse. We train our model and the baselines for 10 epochs. We set a strong regularization $C = 0.001$ during learning because cross-lingual transfer contains noise and the models can easily overfit. Other hyper-parameters are set as $\gamma = 0.3$ and $r = 200$ (rank of the tensor). For partial lexicalization, we set the embedding dimension to 50.

7 Results

Table 5 and 7 summarize the results for the unsupervised and the semi-supervised scenarios. Averaged across languages, our model outperforms all

[6] We use this as a re-implementation of Täckström et al. (2013)'s model because their code is not publicly available.

[7] All multilingual transfer models in our work and in Täckström et al. (2013)'s work are first-order. Therefore, we train first-order RBGParser for consistency.

[8] `https://github.com/taolei87/RBGParser`

	Direct		NT-Select		Multiway		Ours	
	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS
EN	65.7	56.7	67.6	55.3	69.8	56.3	**70.5**	**59.8**
FR	77.9	67.4	**79.1**	**68.9**	78.4	68.3	78.9	68.8
DE	62.1	53.1	62.1	53.3	62.1	54.0	**62.5**	**54.1**
ID	46.8	39.3	57.4	37.1	59.5	38.9	**61.0**	**43.5**
IT	77.9	67.9	**79.4**	**69.4**	79.0	69.0	79.3	**69.4**
JA	57.8	16.8	69.2	20.8	69.9	20.4	**71.7**	**21.3**
KO	59.9	34.3	70.4	29.1	70.5	28.1	**70.7**	**30.5**
PT	77.7	71.0	78.5	72.0	78.3	71.9	**78.6**	**72.5**
ES	76.8	65.9	77.2	67.7	77.6	68.0	**78.0**	**68.3**
SV	**75.9**	**64.5**	74.5	62.2	74.8	62.9	75.0	62.5
AVG	67.8	53.7	71.5	53.6	72.0	53.8	**72.6**	**55.1**

Table 5: **Unsupervised:** Unlabeled attachment scores (UAS) and Labeled attachment scores (LAS) of different variants of our model with partial lexicalization in unsupervised scenario. "Direct" and "Multiway" indicate the direct transfer and the multiway variants of our model. "NT-Select" indicates our model without tensor component, corresponding to a re-implementation of previous transfer model (Täckström et al., 2013) with extensions to partial lexicalization and labeled parsing. The last column shows the results by our hierarchical tensor-based model. Boldface numbers indicate the best UAS or LAS.

Feature	Weight
87A$\wedge hp$=NOUN$\wedge mp$=ADJ	2.24×10^{-3}
87A$\wedge hp$=VERB$\wedge mp$=NOUN	8.88×10^{-4}
87A$\wedge hp$=VERB$\wedge mp$=PRON	1.21×10^{-4}
87A$\wedge hp$=NOUN$\wedge mp$=NOUN	9.48×10^{-4}
87A$\wedge hp$=ADP$\wedge mp$=NOUN	3.87×10^{-4}

Table 6: Examples of weights for feature combinations between the typological feature 87A=Adj-Noun and different types of arcs. The first row shows the weight for the valid feature (conjoined with noun→adjective arcs) and the rest show weights for the invalid features (conjoined with other types of arcs).

the baselines in both cases. Moreover, it achieves best UAS and LAS on 7 out of 10 languages. The difference is more pronounced in the semi-supervised case. Below, we summarize our findings when comparing the model with the baselines.

Impact of Hierarchical Tensors We first analyze the impact of using a hierarchical tensor by comparing against the Multiway baseline that implements traditional tensor model. As Table 6 shows, this model learns non-zero weights even for invalid feature combinations.

This disregard to known constraints impacts the resulting performance. In the unsupervised scenario, our hierarchical tensor achieves an average improvement of 0.5% on UAS and 1.3% on LAS. Moreover, our model obtains better UAS on all languages and better LAS on 9 out of 10 languages. This observation shows that the multilingual transfer consistently benefits more from a hierarchical tensor structure. In addition, we observe a similar gain over this baseline in the semi-supervised scenario.

Impact of Tensor Models To evaluate the effectiveness of tensor modeling in multilingual transfer, we compare our model against the NT-Select baseline. In the unsupervised scenario, our tensor model yields a 1.1% gain on UAS and a 1.5% on LAS. In the semi-supervised scenario, the improvement is more pronounced, reaching 1.7% on UAS and 1.9% on LAS. The relative error reduction almost doubles, e.g. 7.1% vs. 3.8% on UAS.

While both our model and NT-Select outperform Direct baseline by a large margin on UAS, we observe that NT-Select achieves a slightly worse LAS than Direct. By adding a tensor component, our model outperforms both baselines on LAS, demonstrating that tensor scoring function is able to capture better labeled features for transfer comparing to Direct and NT-Select baselines.

Transfer Performance in the Context of Supervised Results To assess the contribution of multilingual transfer, we compare against the Sup50 results in which we train our model only on 50 target language sentences. As Table 7 shows, our model improves UAS by 2.3% and LAS by 2.7%. We also provide a performance upper bound

	Semi-supervised Transfer										Supervised Parsing (RBGParser)			
	Direct		Sup50		NT-Select		Multiway		Ours		Partial Lex.		Full Lex.	
	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS
EN	76.8	70.3	79.6	74.2	81.0	75.0	81.5	75.9	**82.5**	**77.2**	88.7	84.5	92.3	90.3
FR	78.8	70.2	76.9	66.8	79.4	71.0	79.0	71.1	**79.6**	**71.8**	83.3	76.5	83.3	76.5
DE	68.4	59.8	71.0	62.4	71.3	62.1	72.1	63.2	**74.2**	**65.6**	82.0	72.8	84.5	78.2
ID	63.7	56.1	78.2	68.9	76.9	68.2	77.8	69.3	**79.1**	**70.4**	85.0	77.1	85.8	79.8
IT	78.9	70.3	77.1	69.3	80.2	72.2	80.8	72.6	**80.9**	**72.6**	85.5	79.8	87.9	84.7
JA	68.2	42.1	**76.6**	61.0	73.0	58.8	75.6	60.9	76.4	**61.3**	79.0	64.0	82.1	70.3
KO	65.3	45.2	70.1	**54.7**	66.5	50.2	67.8	52.8	**70.2**	54.2	74.0	59.1	90.9	86.1
PT	78.6	72.9	76.0	70.0	78.7	73.1	**79.3**	**73.9**	**79.3**	73.5	85.2	80.8	88.5	86.5
ES	77.0	68.5	75.2	66.5	77.0	69.0	77.6	69.5	**78.4**	**70.5**	82.0	75.0	85.8	81.6
SV	77.7	67.2	74.9	64.7	77.6	66.8	77.8	67.5	**78.3**	**67.9**	84.4	75.4	87.3	82.3
AVG	73.4	62.3	75.6	65.8	76.2	66.6	76.9	67.7	**77.9**	**68.5**	82.9	74.5	87.3	83.5

Table 7: **Semi-supervised and Supervised:** UAS and LAS of different variants of our model when 50 annotated sentences in the target language are available. "Sup50" columns show the results of our model when only supervised data in the target language is available. We also include in the last two columns the supervised training results with partial or full lexicalization as the performance upper bound. Other columns have the same meaning as in Table 5. Boldface numbers indicate the best UAS or LAS.

by training RBGParser on the full training set.[9] When trained with partial lexical information as in our model, RBGParser gives 82.9% on UAS and 74.5% on LAS with partial lexical information. By utilizing source language annotations, our model closes the performance gap between training on the 50 sentences and on the full training set by about 30% on both UAS and LAS. We further compare to the performance upper bound with full lexical information (87.3% UAS and 83.5% LAS). In this case, our model still closes the performance gap by 21% on UAS and 15% on LAS.

Time Efficiency of Hierarchical Tensors We observe that our hierarchical structure retains the time efficiency of tensor models. On the English test set, the decoding speed of our hierarchical tensor is close to the multiway counterpart (58.6 vs. 61.2 sentences per second), and is lower than the three-way tensor by a factor of 3.1 (184.4 sentences per second). The time complexity of tensors is linear to the number of low-rank components, and is independent of the factorization structure.

8 Conclusions

In this paper, we introduce a hierarchical tensor based-model which enables us to constrain learned representation based on desired feature interactions. We demonstrate that our model outperforms state-of-the-art multilingual transfer parsers and traditional tensors. These observations, taken together with the fact that hierarchical tensors are efficiently learnable, suggest that the approach can be useful in a broader range of parsing applications; exploring the options is an appealing line of future research.

[9] On average, each language has more than 10,000 training sentences.

Acknowledgments

This research is developed in a collaboration of MIT with the Arabic Language Technologies (ALT) group at Qatar Computing Research Institute (QCRI) within the Interactive sYstems for Answer Search (IYAS) project. The authors acknowledge the support of the U.S. Army Research Office under grant number W911NF-10-1-0533. We thank the MIT NLP group and the EMNLP reviewers for their comments. Any opinions, findings, conclusions, or recommendations expressed in this paper are those of the authors, and do not necessarily reflect the views of the funding organizations.

References

Taylor Berg-Kirkpatrick and Dan Klein. 2010. Phylogenetic grammar induction. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1288–1297. Association for Computational Linguistics.

Shay B Cohen and Noah A Smith. 2009. Shared logistic normal distributions for soft parameter tying in unsupervised grammar induction. In *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational*

Linguistics, pages 74–82. Association for Computational Linguistics.

Shay B Cohen, Dipanjan Das, and Noah A Smith. 2011. Unsupervised structure prediction with non-parallel multilingual guidance. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 50–61. Association for Computational Linguistics.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006. Online passive-aggressive algorithms. *The Journal of Machine Learning Research*, 7:551–585.

Marie-Catherine De Marneffe and Christopher D Manning. 2008. The stanford typed dependencies representation. In *Coling 2008: Proceedings of the workshop on Cross-Framework and Cross-Domain Parser Evaluation*, pages 1–8. Association for Computational Linguistics.

Matthew S Dryer, David Gil, Bernard Comrie, Hagen Jung, Claudia Schmidt, et al. 2005. The world atlas of language structures.

Long Duong, Trevor Cohn, Steven Bird, and Paul Cook. 2015. Cross-lingual transfer for unsupervised dependency parsing without parallel data. *Proceedings of the SIGNLL Conference on Computational Natural Language Learning*, page 113.

Greg Durrett, Adam Pauls, and Dan Klein. 2012. Syntactic transfer using a bilingual lexicon. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1–11. Association for Computational Linguistics.

Jason M Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of the 16th conference on Computational linguistics-Volume 1*, pages 340–345. Association for Computational Linguistics.

Manaal Faruqui and Chris Dyer. 2014. Improving vector space word representations using multilingual correlation. In *Proceedings of the Annual Conference of the European Chapter of the Association for Computational Linguistics.*, volume 2014.

Daniel Fried, Tamara Polajnar, and Stephen Clark. 2015. Low-rank tensors for verbs in compositional distributional semantics. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*.

Rebecca Hwa, Philip Resnik, Amy Weinberg, Clara Cabezas, and Okan Kolak. 2005. Bootstrapping parsers via syntactic projection across parallel texts. *Natural language engineering*, 11(03):311–325.

Tao Lei, Yu Xin, Yuan Zhang, Regina Barzilay, and Tommi Jaakkola. 2014. Low-rank tensors for scoring dependency structures. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 1381–1391.

Tao Lei, Yuan Zhang, Regina Barzilay, Lluís Màrquez, and Alessandro Moschitti. 2015. High-order low-rank tensors for semantic role labeling. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics*.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 91–98. Association for Computational Linguistics.

Ryan McDonald, Slav Petrov, and Keith Hall. 2011. Multi-source transfer of delexicalized dependency parsers. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 62–72. Association for Computational Linguistics.

Ryan T McDonald, Joakim Nivre, Yvonne Quirmbach-Brundage, Yoav Goldberg, Dipanjan Das, Kuzman Ganchev, Keith B Hall, Slav Petrov, Hao Zhang, Oscar Täckström, et al. 2013. Universal dependency annotation for multilingual parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 92–97.

Tahira Naseem, Harr Chen, Regina Barzilay, and Mark Johnson. 2010. Using universal linguistic knowledge to guide grammar induction. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1234–1244. Association for Computational Linguistics.

Tahira Naseem, Regina Barzilay, and Amir Globerson. 2012. Selective sharing for multilingual dependency parsing. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 629–637. Association for Computational Linguistics.

Slav Petrov, Dipanjan Das, and Ryan McDonald. 2011. A universal part-of-speech tagset. *arXiv preprint arXiv:1104.2086*.

Audi Primadhanty, Xavier Carreras, and Ariadna Quattoni. 2015. Low-rank regularization for sparse conjunctive feature spaces: An application to named entity classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*.

Ariadna Quattoni, Borja Balle, Xavier Carreras, and Amir Globerson. 2014. Spectral regularization for max-margin sequence tagging. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1710–1718.

Mohammad Sadegh Rasooli and Michael Collins. 2015. Density-driven cross-lingual transfer of dependency parsers. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Sameer Singh, Tim Rocktaschel, and Sebastian Riedel. 2015. Towards combined matrix and tensor factorization for universal schema relation extraction. In *NAACL Workshop on Vector Space Modeling for NLP*.

Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *Proceedings of the Advances in Neural Information Processing Systems*, pages 926–934.

Vivek Srikumar and Christopher D Manning. 2014. Learning distributed representations for structured output prediction. In *Proceedings of the Advances in Neural Information Processing Systems*, pages 3266–3274.

Oscar Täckström, Ryan McDonald, and Joakim Nivre. 2013. Target language adaptation of discriminative transfer parsers. In *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics*.

Mo Yu and Mark Dredze. 2015. Learning composition models for phrase embeddings. *Transactions of the Association for Computational Linguistics*, 3:227–242.

Dong Yu, Li Deng, and Frank Seide. 2013a. The deep tensor neural network with applications to large vocabulary speech recognition. *Audio, Speech, and Language Processing, IEEE Transactions on*, 21(2):388–396.

Mo Yu, Tiejun Zhao, Yalong Bai, Hao Tian, and Dianhai Yu. 2013b. Cross-lingual projections between languages from different families. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 312–317.

Daniel Zeman and Philip Resnik. 2008. Cross-language parser adaptation between related languages. In *Proceedings of the International Joint Conference on Natural Language Processing*, pages 35–42.

Diversity in Spectral Learning for Natural Language Parsing

Shashi Narayan and Shay B. Cohen
School of Informatics
University of Edinburgh
Edinburgh, EH8 9LE, UK
{snaraya2,scohen}@inf.ed.ac.uk

Abstract

We describe an approach to create a diverse set of predictions with spectral learning of latent-variable PCFGs (L-PCFGs). Our approach works by creating multiple spectral models where noise is added to the underlying features in the training set before the estimation of each model. We describe three ways to decode with multiple models. In addition, we describe a simple variant of the spectral algorithm for L-PCFGs that is fast and leads to compact models. Our experiments for natural language parsing, for English and German, show that we get a significant improvement over baselines comparable to state of the art. For English, we achieve the F_1 score of 90.18, and for German we achieve the F_1 score of 83.38.

1 Introduction

It has been long identified in NLP that a diverse set of solutions from a decoder can be reranked or recombined in order to improve the accuracy in various problems (Henderson and Brill, 1999). Such problems include machine translation (Macherey and Och, 2007), syntactic parsing (Charniak and Johnson, 2005; Sagae and Lavie, 2006; Fossum and Knight, 2009; Zhang et al., 2009; Petrov, 2010; Choe et al., 2015) and others (Van Halteren et al., 2001).

The main argument behind the use of such a diverse set of solutions (such as k-best list of parses for a natural language sentence) is the hope that each solution in the set is mostly correct. Therefore, recombination or reranking of solutions in that set will further optimize the choice of a solution, combining together the information from all solutions.

In this paper, we explore another angle for the use of a set of parse tree predictions, where all predictions are made for the same sentence. More specifically, we describe techniques to exploit diversity with spectral learning algorithms for natural language parsing. Spectral techniques and the method of moments have been recently used for various problems in natural language processing, including parsing, topic modeling and the derivation of word embeddings (Luque et al., 2012; Cohen et al., 2013; Stratos et al., 2014; Dhillon et al., 2015; Rastogi et al., 2015; Nguyen et al., 2015; Lu et al., 2015).

Cohen et al. (2013) showed how to estimate an L-PCFG using spectral techniques, and showed that such estimation outperforms the expectation-maximization algorithm (Matsuzaki et al., 2005). Their result still lags behind state of the art in natural language parsing, with methods such as coarse-to-fine (Petrov et al., 2006).

We further advance the accuracy of natural language parsing with spectral techniques and L-PCFGs, yielding a result that outperforms the original Berkeley parser from Petrov and Klein (2007). Instead of exploiting diversity from a k-best list from a single model, we estimate multiple models, where the underlying features are perturbed with several perturbation schemes. Each such model, during test time, yields a single parse, and all parses are then used together in several ways to select a single best parse.

The main contributions of this paper are two-fold. First, we present an algorithm for estimating L-PCFGs, akin to the spectral algorithm of Cohen et al. (2012), but simpler to understand and implement. This algorithm has value for readers who are interested in learning more about spectral algorithms – it demonstrates some of the core ideas in spectral learning in a rather intuitive way. In addition, this algorithm leads to sparse grammar estimates and compact models.

Second, we describe how a diverse set of predictors can be used with spectral learning techniques.

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1868–1878,
Lisbon, Portugal, 17-21 September 2015.

Our approach relies on adding noise to the feature functions that help the spectral algorithm compute the latent states. Our noise schemes are similar to those described by Wang et al. (2013). We add noise to the whole training data, then train a model using our algorithm (or other spectral algorithms; Cohen et al., 2013), and repeat this process multiple times. We then use the set of parses we get from all models in a recombination step.

The rest of the paper is organized as follows. In §2 we describe notation and background about L-PCFG parsing. In §3 we describe our new spectral algorithm for estimating L-PCFGs. It is based on similar intuitions as older spectral algorithms for L-PCFGs. In §4 we describe the various noise schemes we use with our spectral algorithm and the spectral algorithm of Cohen et al. (2013). In §5 we describe how to decode with multiple models, each arising from a different noise setting. In §6 we describe our experiments with natural language parsing for English and German.

2 Background and Notation

We denote by $[n]$ the set of integers $\{1, \ldots, n\}$. For a statement Γ, we denote by $[[\Gamma]]$ its indicator function, with values 0 when the assertion is false and 1 when it is true.

An L-PCFG is a 5-tuple $(\mathcal{N}, \mathcal{I}, \mathcal{P}, m, n)$ where:

- $\mathcal{N}$ is the set of nonterminal symbols in the grammar. $\mathcal{I} \subset \mathcal{N}$ is a finite set of *interminals*. $\mathcal{P} \subset \mathcal{N}$ is a finite set of *preterminals*. We assume that $\mathcal{N} = \mathcal{I} \cup \mathcal{P}$, and $\mathcal{I} \cap \mathcal{P} = \emptyset$. Hence we have partitioned the set of nonterminals into two subsets.
- $[m]$ is the set of possible hidden states.
- $[n]$ is the set of possible words.
- For all $a \in \mathcal{I}$, $b \in \mathcal{N}$, $c \in \mathcal{N}$, $h_1, h_2, h_3 \in [m]$, we have a binary context-free rule $a(h_1) \rightarrow b(h_2)\ \ c(h_3)$.
- For all $a \in \mathcal{P}$, $h \in [m]$, $x \in [n]$, we have a lexical context-free rule $a(h) \rightarrow x$.

Latent-variable PCFGs are essentially equivalent to probabilistic regular tree grammars (PRTGs; Knight and Graehl, 2005) where the righthand side trees are of depth 1. With general PRTGs, the righthand side can be of arbitrary depth, where the leaf nodes of these trees correspond to latent states in the L-PCFG formulation

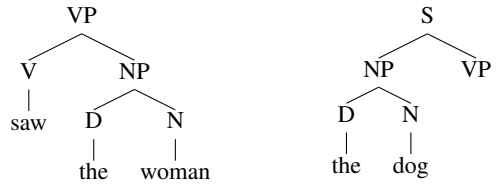


Figure 1: The inside tree (left) and outside tree (right) for the nonterminal `VP` in the parse tree `(S (NP (D the) (N dog)) (VP (V saw) (NP (D the) (N woman))))`.

above and the internal nodes of these trees correspond to interminal symbols in the L-PCFG formulation.

Two important concepts that will be used throughout of the paper are that of an "inside tree" and an "outside tree." Given a tree, the inside tree for a node contains the entire subtree below that node; the outside tree contains everything in the tree excluding the inside tree. See Figure 1 for an example. Given a grammar, we denote the space of inside trees by T and the space of outside trees by O.

3 Clustering Algorithm for Estimating L-PCFGs

We assume two feature functions, $\phi: T \rightarrow \mathbb{R}^d$ and $\psi: O \rightarrow \mathbb{R}^{d'}$, mapping inside and outside trees, respectively, to a real vector. Our training data consist of examples $(a^{(i)}, t^{(i)}, o^{(i)}, b^{(i)})$ for $i \in \{1 \ldots M\}$, where $a^{(i)} \in \mathcal{N}$; $t^{(i)}$ is an inside tree; $o^{(i)}$ is an outside tree; and $b^{(i)} = 1$ if $a^{(i)}$ is the root of tree, 0 otherwise. These are obtained by splitting all trees in the training set into inside and outside trees at each node in each tree. We then define $\Omega^a \in \mathbb{R}^{d \times d'}$:

$$\Omega^a = \frac{\sum_{i=1}^{M} [[a^{(i)} = a]] \phi(t^{(i)}) (\psi(o^{(i)}))^\top}{\sum_{i=1}^{M} [[a^{(i)} = a]]} \quad (1)$$

This matrix is an empirical estimate for the cross-covariance matrix between the inside trees and the outside trees of a given nonterminal a. An inside tree and an outside tree are conditionally independent according to the L-PCFG model, when the latent state at their connecting point is known. This means that the latent state can be identified by finding patterns that co-occur together in inside and outside trees – it is the only random variable that can explain such correlations. As such,

Inputs: An input treebank with the following additional information: training examples $(a^{(i)}, t^{(i)}, o^{(i)}, b^{(i)})$ for $i \in \{1 \dots M\}$, where $a^{(i)} \in \mathcal{N}$; $t^{(i)}$ is an inside tree; $o^{(i)}$ is an outside tree; and $b^{(i)} = 1$ if the rule is at the root of tree, 0 otherwise. A function ϕ that maps inside trees t to feature-vectors $\phi(t) \in \mathbb{R}^d$. A function ψ that maps outside trees o to feature-vectors $\psi(o) \in \mathbb{R}^{d'}$. An integer k denoting the thin-SVD rank. An integer m denoting the number of latent states.

Algorithm:
(Step 1: Singular Value Decompositions)

- Calculate SVD on Ω^a to get $\hat{U}^a \in \mathbb{R}^{(d \times k)}$ and $\hat{V}^a \in \mathbb{R}^{(d' \times k)}$ for each $a \in \mathcal{N}$.

(Step 1: Projection)

- For all $i \in [M]$, compute $y^{(i)} = (\hat{U}^{a_i})^\top \phi(t^{(i)})$ and $z^{(i)} = (\hat{V}^{a_i})^\top \psi(o^{(i)})$.
- For all $i \in [M]$, set $x^{(i)}$ to be the concatenation of $y^{(i)}$ and $z^{(i)}$.

(Step 2: Cluster Projections)

- For all $a \in \mathcal{N}$, cluster the set $\{x^{(i)} \mid a^{(i)} = a\}$ to get a clustering function $\gamma \colon \mathbb{R}^{2k} \to [m]$ that maps a projected vector $x^{(i)}$ to a cluster in $[m]$.

(Step 3: Compute Final Parameters)

- Annotate each node in the treebank with $\gamma(x^{(i)})$.
- Compute the probability of a rule $p(a[h_1] \to b[h_2]\, c[h_3] \mid a[h_1])$ as the relative frequency of its appearance in the cluster-annotated treebank.
- Similarly, compute the root probabilities $\pi(a[h])$ and preterminal rules $p(a[h] \to x \mid a[h])$.

Figure 2: The clustering estimation algorithm for L-PCFGs.

if we reduce the dimensions of Ω^a using singular value decomposition (SVD), we essentially get representations for the inside trees and the outside trees that correspond to the latent states.

This intuition leads to the algorithm that appears in Figure 2. The algorithm we describe takes as input training data, in the form of a treebank, decomposed into inside and outside trees at each node in each tree in the training set.

The algorithm first performs SVD for each of the set of inside and outside trees for all nonterminals.[1] This step is akin to CCA, which has been used in various contexts in NLP, mostly to derive representations for words (Dhillon et al., 2015; Rastogi et al., 2015). The algorithm then takes the representations induced by the SVD step, and clusters them – we use k-means to do the clustering. Finally, it maps each SVD representation to a cluster, and as a result, gets a cluster identifier for each node in each tree in the training data. These clusters are now treated as latent states that are "observed." We subsequently follow up with frequency count maximum likelihood estimate to estimate the probabilities of each parameter in the L-PCFG.

Consider for example the estimation of rules of the form $a \to x$. Following the clustering step we obtain for each nonterminal a and latent state h a set of rules of the form $a[h] \to x$. Each such instance comes from a single training example of a lexical rule. Next, we compute the probability of the rule $a[h] \to x$ by counting how many times that rule appears in the training instances, and normalize by the total count of $a[h]$ in the training instances. Similarly, we compute probabilities for binary rules of the form $a \to b\, c$.

The features that we use for ϕ and ψ are similar to those used in Cohen et al. (2013). These features look at the local neighborhood surrounding a given node. More specifically, we indicate the following information with the inside features (throughout these definitions assume that $a \to b\, c$ is at the root of the inside tree t):

- The pair of nonterminals (a, b). E.g., for the inside tree in Figure 1 this would be the pair (VP, V).
- The pair (a, c). E.g., (VP, NP).
- The rule $a \to b\, c$. E.g., VP $\to$ V NP.
- The rule $a \to b\, c$ paired with the rule at the node b. E.g., for the inside tree in Figure 1 this would correspond to the tree fragment (VP (V saw) NP).
- The rule $a \to b\, c$ paired with the rule at the node c. E.g., the tree fragment (VP V (NP D N)).
- The head part-of-speech of t paired with a. E.g., the pair (VP, V).
- The number of words dominated by t paired with a. E.g., the pair (VP, 3).

In the case of an inside tree consisting of a single rule $a \to x$ the feature vector simply indicates the identity of that rule.

For the outside features, we use:

[1] We normalize features by their variance.

- The rule above the foot node. E.g., for the outside tree in Figure 1 this would be the rule S $\rightarrow$ NP VP* (the foot nonterminal is marked with $*$).
- The two-level and three-level rule fragments above the foot node. These features are absent in the outside tree in Figure 1.
- The label of the foot node, together with the label of its parent. E.g., the pair (VP, S).
- The label of the foot node, together with the label of its parent and grandparent.
- The part-of-speech of the first head word along the path from the foot of the outside tree to the root of the tree which is different from the head node of the foot node.
- The width of the spans to the left and to the right of the foot node, paired with the label of the foot node.

Other Spectral Algorithms The SVD step on the Ω^a matrix is pivotal to many algorithms, and has been used in the past for other L-PCFG estimation algorithms. Cohen et al. (2012) used it for developing a spectral algorithm that identifies the parameters of the L-PCFG up to a linear transformation. Their algorithm generalizes the work of Hsu et al. (2009) and Bailly et al. (2010).

Cohen and Collins (2014) also developed an algorithm that makes use of an SVD step on the inside-outside. It relies on the idea of "pivot features" – features that uniquely identify latent states.

Louis and Cohen (2015) used a clustering algorithm that resembles ours but does not separate inside trees from outside trees or follows up with a singular value decomposition step. Their algorithm was applied to both L-PCFGs and linear context-free rewriting systems. Their application was the analysis of hierarchical structure of conversations in online forums.

In our preliminary experiments, we found out that the clustering algorithm by itself performs worse than the spectral algorithm of Cohen et al. (2013). We believe that the reason is two-fold: (a) k-means finds a local maximum during clustering; (b) we do hard clustering instead of soft clustering. However, we detected that the clustering algorithm gives a more diverse set of solutions, when the features are perturbed. As such, in the next sections, we explain how to perturb the models we get from the clustering algorithm (and the spectral algorithm) in order to improve the accuracy of the clustering and spectral algorithms.

4 Spectral Estimation with Noise

It has been shown that a diverse set of predictions can be used to help improve decoder accuracy for various problems in NLP (Henderson and Brill, 1999). Usually a k-best list from a single model is used to exploit model diversity. Instead, we estimate multiple models, where the underlying features are filtered with various noising schemes.

We try three different types of noise schemes for the algorithm in Figure 2:

Dropout noise: Let $\sigma \in [0, 1]$. We set each element in the feature vectors $\phi(t)$ and $\psi(o)$ to 0 with probability σ.

Gaussian (additive): Let $\sigma > 0$. For each $x^{(i)}$, we draw a vector $\varepsilon \in \mathbb{R}^{2k}$ of Gaussians with mean 0 and variance σ^2, and then set $x^{(i)} \leftarrow x^{(i)} + \varepsilon$.

Gaussian (multiplicative): Let $\sigma > 0$. For each $x^{(i)}$, we draw a vector $\varepsilon \in \mathbb{R}^{2k}$ of Gaussians with mean 0 and variance σ^2, and then set $x^{(i)} \leftarrow x^{(i)} \otimes (1 + \varepsilon)$, where $\otimes$ is coordinate-wise multiplication.

Note the distinction between the dropout noise and the Gaussian noise schemes: the first is performed on the feature vectors before the SVD step, and the second is performed after the SVD step. It is not feasible to add Gaussian noise prior to the SVD step, since the matrix Ω^a will no longer be sparse, and its SVD computation will be computationally demanding.

Our use of dropout noise here is inspired by "dropout" as is used in neural network training, where various connections between units in the neural network are dropped during training in order to avoid overfitting of these units to the data (Srivastava et al., 2014).

The three schemes we described were also used by Wang et al. (2013) to train log-linear models. Wang et al.'s goal was to prevent overfitting by introducing this noise schemes as additional regularizer terms, but without explicitly changing the training data. We do filter the data through these noise schemes, and show in §6 that all of these noise schemes do not improve the performance of our estimation on their own. However, when multiple models are created with these noise schemes,

and then combined together, we get an improved performance. As such, our approach is related to the one of Petrov (2010), who builds a committee of latent-variable PCFGs in order to improve a natural language parser.

We also use these perturbation schemes to create multiple models for the algorithm of Cohen et al. (2012). The dropout scheme stays the same, but for the Gaussian noising schemes, we follow a slightly different procedure. After noising the projections of the inside and outside feature functions we get from the SVD step, we use these projected noised features as a new set of inside and outside feature functions, and re-run the spectral algorithm of Cohen et al. (2012) on them.

We are required to add this extra SVD step because the spectral algorithm of Cohen et al. assumes the existence of linearly transformed parameter estimates, where the parameters of each nonterminal a is linearly transformed by unknown invertible matrices. These matrices cancel out when the inside-outside algorithm is run with the spectral estimate output. In order to ensure that these matrices still exactly cancel out, we have to follow with another SVD step as described above. The latter SVD step is performed on a dense $\Omega^a \in \mathbb{R}^{m \times m}$ but this is not an issue considering m (the number of latent states) is much smaller than d or d'.

5 Decoding with Multiple Models

Let $G_1, \ldots, G_p$ be a set of L-PCFG grammars. In §6, we create such models using the noising techniques described above. The question that remains is how to combine these models together to get a single best output parse tree given an input sentence.

With L-PCFGs, decoding a single sentence requires marginalizing out the latent states to find the best skeletal tree[2] for a given string. Let s be a sentence. We define $t(G_i, s)$ to be the output tree according to minimum Bayes risk decoding. This means we follow Goodman (1996), who uses dynamic programming to compute the tree that maximizes the sum of all marginals of all nonterminals in the output tree. Each marginal, for each span $\langle a, i, j \rangle$ (where a is a nonterminal and i and j are endpoints in the sentence), is computed by using the inside-outside algorithm.

[2]A skeletal tree is a derivation tree without latent states decorating the nonterminals.

In addition, let $\mu(a, i, j | G_k, s)$ be the marginal, as computed by the inside-outside algorithm, for the span $\langle a, i, j \rangle$ with grammar G_k for string s. We use the notation $\langle a, i, j \rangle \in t$ to denote that a span $\langle a, i, j \rangle$ is in a tree t.

We suggest the following three ways for decoding with multiple models $G_1, \ldots, G_p$:

Maximal tree coverage: Using dynamic programming, we return the tree that is the solution to:

$$t^* = \arg\max_t \sum_{\langle a,i,j \rangle \in t} \sum_{k=1}^{p} [[\langle a, i, j \rangle \in t(G_k, s)]].$$

This implies that we find the tree that maximizes its coverage with respect to all other trees that are decoded using $G_1, \ldots, G_p$.

Maximal marginal coverage: Using dynamic programming, we return the tree that is the solution to:

$$t^* = \arg\max_t \sum_{\langle a,i,j \rangle \in t} \sum_{k=1}^{p} \mu(a, i, j | G_k, s).$$

This is similar to maximal tree coverage, only instead of considering just the single decoded tree for each model among $G_1, \ldots, G_p$, we make our decoding "softer," and rely on the marginals that each model gives.

MaxEnt reranking: We train a MaxEnt reranker on a training set that includes outputs from multiple models, and then, during testing time, decode with each of the models, and use the trained reranker to select one of the parses. We use the reranker of Charniak and Johnson (2005).[3]

As we see later in §6, it is sometimes possible to extract more information from the training data by using a network, or a hierarchy of the above tree combination methods. For example, we get our best result for parsing by first using MaxEnt with several subsets of the models, and then combining the output of these MaxEnt models using maximal tree coverage.

[3]Implementation: `https://github.com/BLLIP/bllip-parser`. More specifically, we used the programs `extract-spfeatures`, `cvlm-lbfgs` and `best-indices`. `cvlm-lbfgs` was used with the default hyperparameters from the Makefile.

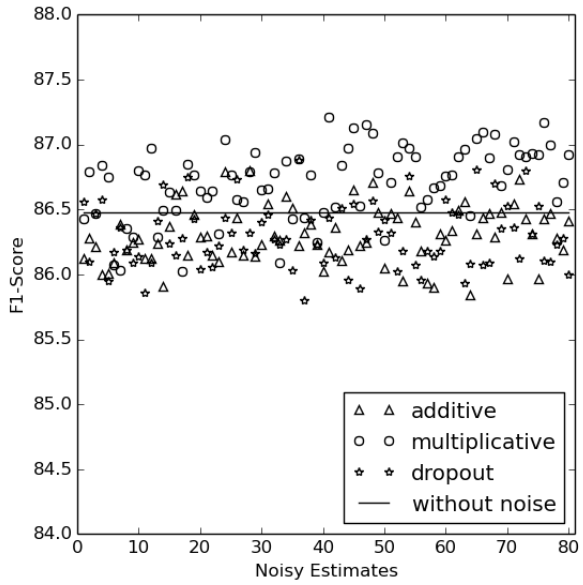


Figure 3: F_1 scores of noisy models. Each data point gives the F_1 accuracy of a single model on the development set, based on the legend. The x-axis enumerates the models (80 in total for each noise scheme).

6 Experiments

In this section, we describe parsing experiments with two languages: English and German.

6.1 Results for English

For our English parsing experiments, we use a standard setup. More specifically, we use the Penn WSJ treebank (Marcus et al., 1993) for our experiments, with sections 2–21 as the training data, and section 22 used as the development data. Section 23 is used as the final test set. We binarize the trees in training data, but transform them back before evaluating them.

For efficiency, we use a base PCFG without latent states to prune marginals which receive a value less than 0.00005 in the dynamic programming chart. The parser takes part-of-speech tagged sentences as input. We tag all datasets using Turbo Tagger (Martins et al., 2010), trained on sections 2–21. We use the F_1 measure according to the PARSEVAL metric (Black et al., 1991) for the evaluation.

Preliminary experiments We first experiment with the number of latent states for the clustering algorithm without perturbations. We use $k = 100$ for the SVD step. Whenever we need to cluster a set of points, we run the k-means algorithm 10 times with random restarts and choose the clustering result with the lowest objective value. On section 22, the clustering algorithm achieves the following results (F_1 measure): $m = 8$: 84.30%, $m = 16$: 85.98%, $m = 24$: 86.48%, $m = 32$: 85.84%, $m = 36$: 86.05%, $m = 40$: 85.43%. As we increase the number of states, performance improves, but plateaus at $m = 24$. For the rest of our experiments, both with the spectral algorithm of Cohen et al. (2012) and the clustering algorithm presented in this paper, we use $m = 24$.

Compact models One of the advantage of the clustering algorithm is that it leads to much more compact models. The number of nonzero parameters with $m = 24$ for the clustering algorithm is approximately 97K, while the spectral algorithms lead to a significantly larger number of nonzero parameters with the same number of latent states: approximately 54 million.

Oracle experiments To what extent do we get a diverse set of solutions from the different models we estimate? This question can be answered by testing the oracle accuracy in the different settings. For each type of noising scheme, we generated 80 models, 20 for each $\sigma \in \{0.05, 0.1, 0.15, 0.2\}$. Each noisy model by itself lags behind the best model (see Figure 3). However, when choosing the best tree among these models, the additively-noised models get an oracle accuracy of 95.91% on section 22; the multiplicatively-noised models get an oracle accuracy of 95.81%; and the dropout-noised models get an oracle accuracy of 96.03%. Finally all models combined get an oracle accuracy of 96.67%. We found out that these oracle scores are comparable to the one Charniak and Johnson (2005) report.

We also tested our oracle results, comparing the spectral algorithm of Cohen et al. (2013) to the clustering algorithm. We generated 20 models for each type of noising scheme, 5 for each $\sigma \in \{0.05, 0.1, 0.15, 0.2\}$) for the spectral algorithm.[4] Surprisingly, even though the spectral models were smoothed, their oracle accuracy was lower than the accuracy of the clustering algo-

[4]There are two reasons we use a smaller number of models with the spectral algorithm: (a) models are not compact (see text) and (b) as such, parsing takes comparatively longer. However, in the above comparison, we use 20 models for the clustering algorithm as well.

	Clustering			Spectral (smoothing)			Spectral (no smoothing)		
	MaxTre	MaxMrg	MaxEnt	MaxTre	MaxMrg	MaxEnt	MaxTre	MaxMrg	MaxEnt
Add	88.68	88.64	89.50	88.20	88.28	88.59	86.72	86.85	87.94
Mul	88.74	88.66	89.89	88.48	88.70	89.46	86.97	86.53	89.04
Dropout	88.68	88.56	89.80	88.64	88.71	**89.47**	88.37	88.06	89.52
All	88.84	88.75	**89.95**	88.38	88.75	89.45	87.49	87.00	**89.85**
No noise	86.48			88.53 (Cohen et al., 2013)			86.47 (Cohen et al., 2013)		

Table 1: Results on section 22 (WSJ). MaxTre denotes decoding using maximal tree coverage, MaxMrg denotes decoding using maximal marginal coverage, and MaxEnt denotes the use of a discriminative reranker. Add, Mul and Dropout denote the use of additive Gaussian noise, multiplicative Gaussian noise and dropout noise, respectively. The number of models used in the first three rows for the clustering algorithm is 80: 20 for each $\sigma \in \{0.05, 0.1, 0.15, 0.2\}$. For the spectral algorithm, it is 20, 5 for each σ (see footnotes). The number of latent states is $m = 24$. For All, we use all models combined from the first three rows. The "No noise" baseline for the spectral algorithm is taken from Cohen et al. (2013). The best figure in each algorithm block is in boldface.

rithm: 92.81% vs. 95.73%.[5] This reinforces two ideas: (i) that noising acts as a regularizer, and has a similar role to backoff smoothing, as we see below; and (ii) the noisy estimation for the clustering algorithm produces a more diverse set of parses than that produced with the spectral algorithm.

	Method	F_1
Best	Spectral (unsmoothed)	89.21
	Spectral (smoothed)	88.87
	Clustering	89.25
Hier	Spectral (unsmoothed)	89.09
	Spectral (smoothed)	89.06
	Clustering	**90.18**

Table 2: Results on section 23 (English). The first three results (Best) are taken with the best model in each corresponding block in Table 1. The last three results (Hier) use a hierarchy of the above tree combination methods in each block. It combines all MaxEnt results using the maximal tree coverage (see text).

It is also important to note that the high oracle accuracy is not just the result of k-means not finding the global maximum for the clustering objective. If we just run the clustering algorithms with 80 models as before, without perturbing the features, the oracle accuracy is 95.82%, which is lower than the oracle accuracy with the additive and dropout perturbed models. To add to this, we see below that perturbing the training set with the spectral algorithm of Cohen et al. improves the accuracy of the spectral algorithm. Since the spectral algorithm of Cohen et al. does not maximize any objective locally, it shows that the role of the perturbations we use is important.

Results Results on the development set are given in Table 1 with our three decoding methods. We present the results from three algorithms: the clustering algorithm and the spectral algorithms (smoothed and unsmoothed).[6]

It seems that dropout noise for the spectral algorithm acts as a regularizer, similarly to the backoff smoothing techniques that are used in Cohen et al. (2013). This is evident from the two spectral algorithm blocks in Table 1, where dropout noise does not substantially improve the smoothed spectral model (Cohen et al. report accuracy of 88.53% with smoothed spectral model for $m = 24$ without noise) – the accuracy is 88.64%–88.71%–89.47%, but the accuracy substantially improves for the unsmoothed spectral model, where dropout brings an accuracy of 86.47% up to 89.52%.

All three blocks in Table 1 demonstrate that decoding with the MaxEnt reranker performs the best. Also it is interesting to note that our results continue to improve when combining the output of previous combination steps further. The best result on section 22 is achieved when we combine, using maximal tree coverage, all MaxEnt outputs of the clustering algorithm (the first block in Ta-

[5] Oracle scores for the clustering algorithm: 95.73% (20 models for each noising scheme) and 96.67% (80 models for each noising scheme).

[6] Cohen et al. (2013) propose two variants of spectral estimation for L-PCFGs: smoothed and unsmoothed. The smoothed model uses a simple backedoff smoothing method which leads to significant improvements over the unsmoothed one. Here we compare our clustering algorithm against both of these models. However unless specified otherwise, the spectral algorithm of Cohen et al. (2013) refers to their best model, i.e. the smoothed model.

	Clustering			Spectral (smoothing)			Spectral (no smoothing)		
	MaxTre	MaxMrg	MaxEnt	MaxTre	MaxMrg	MaxEnt	MaxTre	MaxMrg	MaxEnt
Add	77.34	76.87	80.01	77.76	77.85	78.09	77.44	77.56	77.91
Mul	77.80	77.80	80.34	77.80	77.76	78.89	77.62	77.85	78.94
Dropout	77.37	77.17	**80.94**	77.94	78.06	79.02	77.97	78.17	79.18
All	77.71	77.51	80.86	78.04	77.89	**79.46**	77.73	77.91	**79.66**
No noise	75.04			77.71			77.07		

Table 3: Results on the development set for German. See Table 1 for interpretation of MaxTre, MaxMrg, MaxEnt and Add, Mul and Dropout. The number of models used in the first three rows for the clustering algorithm is 80: 20 for each $\sigma \in \{0.05, 0.1, 0.15, 0.2\}$. For the spectral algorithm, it is 20, 5 for each σ. The number of latent states is $m = 8$. For All, we use all models combined from the first three rows. The best figure in each algorithm block is in boldface.

ble 1). This yields a 90.68% F_1 accuracy. This is also the best result we get on the test set (section 23), 90.18%. See Table 2 for results on section 23.

Our results are comparable to state-of-the-art results for parsing. For example, Sagae and Lavie (2006), Fossum and Knight (2009) and Zhang et al. (2009) report an accuracy of 93.2%-93.3% using parsing recombination; Shindo et al. (2012) report an accuracy of 92.4 F_1 using a Bayesian tree substitution grammar; Petrov (2010) reports an accuracy of 92.0% using product of L-PCFGs; Charniak and Johnson (2005) report accuracy of 91.4 using a discriminative reranking model; Carreras et al. (2008) report 91.1 F_1 accuracy for a discriminative, perceptron-trained model; Petrov and Klein (2007) report an accuracy of 90.1 F_1. Collins (2003) reports an accuracy of 88.2 F_1.

6.2 Results for German

For the German experiments, we used the NEGRA corpus (Skut et al., 1997). We use the same setup as in Petrov (2010), and use the first 18,602 sentences as a training set, the next 1,000 sentences as a development set and the last 1,000 sentences as a test set. This corresponds to an 80%-10%-10% split of the treebank.

Our German experiments follow the same setting as in our English experiments. For the clustering algorithm we generated 80 models, 20 for each $\sigma \in \{0.05, 0.1, 0.15, 0.2\}$. For the spectral algorithm, we generate 20 models, 5 for each σ.

For the reranking experiment, we had to modify the BLLIP parser (Charniak and Johnson, 2005) to use the head features from the German treebank. We based our modifications on the documentation for the NEGRA corpus (our modifications are based mostly on mapping of nonterminals to coarse syntactic categories).

Preliminary experiments For German, we also experiment with the number of latent states. On the development set, we observe that the F_1 measure is: 75.04% for $m = 8$, 73.44% for $m = 16$ and 70.84% for $m = 24$. For the rest of our experiments, we fix the number of latent states at $m = 8$.

	Method	F_1
Best	Spectral (unsmoothed)	80.88
	Spectral (smoothed)	80.31
	Clustering	81.94
Hier	Spectral (unsmoothed)	80.64
	Spectral (smoothed)	79.96
	Clustering	**83.38**

Table 4: Results on the test set for the German data. The first three results (Best) are taken with the best model in each corresponding block in Table 3. The last three results (Hier) use a hierarchy of the above tree combination methods.

Oracle experiments The additively-noised models get an oracle accuracy of 90.58% on the development set; the multiplicatively-noised models get an oracle accuracy of 90.47%; and the dropout-noised models get an oracle accuracy of 90.69%. Finally all models combined get an oracle accuracy of 92.38%.

We compared our oracle results to those given by the spectral algorithm of Cohen et al. (2013). With 20 models for each type of noising scheme, all spectral models combined achieve an oracle accuracy of 83.45%. The clustering algorithm gets the oracle score of 90.12% when using the same number of models.

Results Results on the development set and on the test set are given in Table 3 and Table 4 respectively.

Like English, in all three blocks in Table 3, decoding with the MaxEnt reranking performs the best. Our results continue to improve when further combining the output of previous combination steps. The best result of 82.04% on the development set is achieved when we combine, using maximal tree coverage, all MaxEnt outputs of the clustering algorithm (the first block in Table 3). This also leads to the best result of 83.38% on the test set. See Table 4 for results on the test set.

Our results are comparable to state-of-the-art results for German parsing. For example, Petrov (2010) reports an accuracy of 84.5% using product of L-PCFGs; Petrov and Klein (2007) report an accuracy of 80.1 F_1; and Dubey (2005) reports an accuracy of 76.3 F_1.

7 Discussion

From a theoretical point of view, one of the great advantages of spectral learning techniques for latent-variable models is that they yield consistent parameter estimates. Our clustering algorithm for L-PCFG estimation breaks this, but there is a work-around to obtain an algorithm which would be statistically consistent.

The main reason that our algorithm is not a consistent estimator is that it relies on k-means clustering, which maximizes a non-convex objective using hard clustering steps. The k-means algorithm can be viewed as "hard EM" for a Gaussian mixture model (GMM), where each latent state is associated with one of the mixture components in the GMM. This means that instead of following up with k-means, we could have identified the parameters and the posteriors for a GMM, where the observations correspond to the vectors that we cluster. There are now algorithms, some of which are spectral, that aim to solve this estimation problem with theoretical guarantees (Vempala and Wang, 2004; Kannan et al., 2005; Moitra and Valiant, 2010).

With theoretical guarantees on the correctness of the posteriors from this step, the subsequent use of maximum likelihood estimation step could yield consistent parameter estimates. The consistency guarantees will largely depend on the amount of information that exists in the base feature functions about the latent states according to the L-PCFG model.

8 Conclusion

We presented a novel estimation algorithm for latent-variable PCFGs. This algorithm is based on clustering of continuous tree representations, and it also leads to sparse grammar estimates and compact models. We also showed how to get a diverse set of parse tree predictions with this algorithm and also older spectral algorithms. Each prediction in the set is made by training an L-PCFG model after perturbing the underlying features that estimation algorithm uses from the training data. We showed that such a diverse set of predictions can be used to improve the parsing accuracy of English and German.

Acknowledgements

The authors would like to thank David McClosky for his help with running the BLLIP parser and the three anonymous reviewers for their helpful comments. This research was supported by an EPSRC grant (EP/L02411X/1).

References

Raphaël Bailly, Amaury Habrard, and François Denis. 2010. A spectral approach for probabilistic grammatical inference on trees. In *Proceedings of ALT*.

Ezra W. Black, Steven Abney, Daniel P. Flickinger, Claudia Gdaniec, Ralph Grishman, Philip Harrison, Donald Hindle, Robert J. P. Ingria, Frederick Jelinek, Judith L. Klavans, Mark Y. Liberman, Mitchell P. Marcus, Salim Roukos, Beatrice Santorini, and Tomek Strzalkowski. 1991. A procedure for quantitatively comparing the syntactic coverage of English grammars. In *Proceedings of DARPA Workshop on Speech and Natural Language*.

Xavier Carreras, Michael Collins, and Terry Koo. 2008. TAG, Dynamic Programming, and the Perceptron for Efficient, Feature-rich Parsing. In *Proceedings of CoNLL*.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proceedings of ACL*.

Do Kook Choe, David McClosky, and Eugene Charniak. 2015. Syntactic parse fusion. In *Proceedings of EMNLP*.

Shay B. Cohen and Michael Collins. 2014. A provably correct learning algorithm for latent-variable PCFGs. In *Proceedings of ACL*.

Shay B. Cohen, Karl Stratos, Michael Collins, Dean P. Foster, and Lyle Ungar. 2012. Spectral learning of latent-variable PCFGs. In *Proceedings of ACL*.

Shay B. Cohen, Karl Stratos, Michael Collins, Dean P. Foster, and Lyle Ungar. 2013. Experiments with spectral learning of latent-variable PCFGs. In *Proceedings of NAACL*.

Michael Collins. 2003. Head-driven statistical models for natural language processing. *Computational Linguistics*, 29:589–637.

Paramveer Dhillon, Dean Foster, and Lyle Ungar. 2015. Eigenwords: Spectral word embeddings. *Journal of Machine Learning Research (to appear)*.

Amit Dubey. 2005. What to do when lexicalization fails: Parsing German with suffix analysis and smoothing. In *Proceedings of ACL*.

Victoria Fossum and Kevin Knight. 2009. Combining constituent parsers. In *Proceedings of HLT-NAACL*.

Joshua Goodman. 1996. Parsing algorithms and metrics. In *Proceedings of ACL*.

John C. Henderson and Eric Brill. 1999. Exploiting diversity in natural language processing: Combining parsers. In *Proceedings of EMNLP*.

Daniel Hsu, Sham M. Kakade, and Tong Zhang. 2009. A spectral algorithm for learning hidden Markov models. In *Proceedings of COLT*.

Ravindran Kannan, Hadi Salmasian, and Santosh Vempala. 2005. The spectral method for general mixture models. In *Learning Theory*, volume 3559 of *Lecture Notes in Computer Science*, pages 444–457. Springer.

Kevin Knight and Jonathan Graehl. 2005. An overview of probabilistic tree transducers for natural language processing. In *Computational linguistics and intelligent text processing*, volume 3406 of *Lecture Notes in Computer Science*, pages 1–24. Springer.

Annie Louis and Shay B. Cohen. 2015. Conversation trees: A grammar model for topic structure in forums. In *Proceedings of EMNLP*.

Ang Lu, Weiran Wang, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2015. Deep multilingual correlation for improved word embeddings. In *Proceedings of NAACL*.

Franco M. Luque, Ariadna Quattoni, Borja Balle, and Xavier Carreras. 2012. Spectral learning for nondeterministic dependency parsing. In *Proceedings of EACL*.

Wolfgang Macherey and Franz Josef Och. 2007. An empirical study on computing consensus translations from multiple machine translation systems. In *Proceedings of EMNLP-CoNLL*.

Mitchell P. Marcus, Beatrice Santorini, and Mary A. Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn treebank. *Computational Linguistics*, 19:313–330.

André F. T. Martins, Noah A. Smith, Eric P. Xing, Mário A. T. Figueiredo, and Pedro M. Q. Aguiar. 2010. TurboParsers: Dependency parsing by approximate variational inference. In *Proceedings of EMNLP*.

Takuya Matsuzaki, Yusuke Miyao, and Junichi Tsujii. 2005. Probabilistic CFG with latent annotations. In *Proceedings of ACL*.

Ankur Moitra and Gregory Valiant. 2010. Settling the polynomial learnability of mixtures of gaussians. In *Proceedings of IEEE Symposium on Foundations of Computer Science (FOCS)*.

Thang Nguyen, Jordan Boyd-Graber, Jeffrey Lund, Kevin Seppi, and Eric Ringger. 2015. Is your anchor going up or down? Fast and accurate supervised topic models. In *Proceedings of NAACL*.

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

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of COLING-ACL*.

Slav Petrov. 2010. Products of random latent variable grammars. In *Proceedings of HLT-NAACL*.

Pushpendre Rastogi, Benjamin Van Durme, and Raman Arora. 2015. Multiview LSA: Representation learning via generalized CCA. In *Proceedings of NAACL*.

Kenji Sagae and Alon Lavie. 2006. Parser combination by reparsing. In *Proceedings of HLT-NAACL*.

Hiroyuki Shindo, Yusuke Miyao, Akinori Fujino, and Masaaki Nagata. 2012. Bayesian symbol-refined tree substitution grammars for syntactic parsing. In *Proceedings of ACL*.

Wojciech Skut, Brigitte Krenn, Thorsten Brants, and Hans Uszkoreit. 1997. An annotation scheme for free word order languages. In *Proceedings of ANLP*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

Karl Stratos, Do-kyum Kim, Michael Collins, and Daniel Hsu. 2014. A spectral algorithm for learning class-based n-gram models of natural language. *Proceedings of UAI*.

Hans Van Halteren, Jakub Zavrel, and Walter Daelemans. 2001. Improving accuracy in word class tagging through the combination of machine learning systems. *Computational linguistics*, 27(2):199–229.

Santosh Vempala and Grant Wang. 2004. A spectral algorithm for learning mixture models. *Journal of Computer and System Sciences*, 68(4):841–860.

Sida Wang, Mengqiu Wang, Stefan Wager, Percy Liang, and Christopher D Manning. 2013. Feature noising for log-linear structured prediction. In *Proceedings of EMNLP*.

Hui Zhang, Min Zhang, Chew Lim Tan, and Haizhou Li. 2009. K-best combination of syntactic parsers. In *Proceedings of EMNLP*.

Transition-based Dependency Parsing Using Two Heterogeneous Gated Recursive Neural Networks

Xinchi Chen, Yaqian Zhou, Chenxi Zhu, Xipeng Qiu, Xuanjing Huang
Shanghai Key Laboratory of Intelligent Information Processing, Fudan University
School of Computer Science, Fudan University
825 Zhangheng Road, Shanghai, China
{xinchichen13,zhouyaqian,czhu13,xpqiu,xjhuang}@fudan.edu.cn

Abstract

Recently, neural network based dependency parsing has attracted much interest, which can effectively alleviate the problems of data sparsity and feature engineering by using the dense features. However, it is still a challenge problem to sufficiently model the complicated syntactic and semantic compositions of the dense features in neural network based methods. In this paper, we propose two heterogeneous gated recursive neural networks: tree structured gated recursive neural network (Tree-GRNN) and directed acyclic graph structured gated recursive neural network (DAG-GRNN). Then we integrate them to automatically learn the compositions of the dense features for transition-based dependency parsing. Specifically, Tree-GRNN models the feature combinations for the trees in stack, which already have partial dependency structures. DAG-GRNN models the feature combinations of the nodes whose dependency relations have not been built yet. Experiment results on two prevalent benchmark datasets (PTB3 and CTB5) show the effectiveness of our proposed model.

1 Introduction

Transition-based dependency parsing is a core task in natural language processing, which has been studied with considerable efforts in the NLP community. The traditional discriminative dependency parsing methods have achieved great success (Koo et al., 2008; He et al., 2013; Bohnet, 2010; Huang and Sagae, 2010; Zhang and Nivre, 2011; Martins et al., 2009; McDonald et al., 2005; Nivre et al., 2006; Kübler et al., 2009; Goldberg and Nivre, 2013; Choi and McCallum, 2013; Ballesteros and Bohnet, 2014). However, these methods are based on discrete features and suffer from the problems of data sparsity and feature engineering (Chen and Manning, 2014).

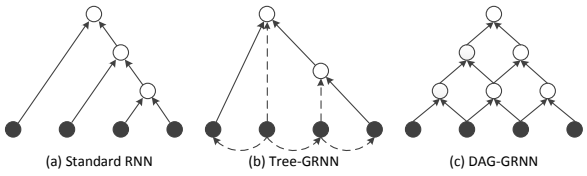


Figure 1: Sketch of three recursive neural networks (RNN). (a) is the standard RNN for constituent tree; (b) is Tree-GRNN for dependency tree, in which the dashed arcs indicate the dependency relations between the nodes; (c) is DAG-GRNN for the nodes without given topological structure.

Recently, distributed representations have been widely used in a variety of natural language processing (NLP) tasks (Collobert et al., 2011; Devlin et al., 2014; Socher et al., 2013; Turian et al., 2010; Mikolov et al., 2013b; Bengio et al., 2003). Specific to the transition-based parsing, the neural network based methods have also been increasingly focused on due to their ability to minimize the efforts in feature engineering and the boosted performance (Le and Zuidema, 2014; Stenetorp, 2013; Bansal et al., 2014; Chen and Manning, 2014; Zhu et al., 2015).

However, most of the existing neural network based methods still need some efforts in feature engineering. For example, most methods often select the first and second leftmost/rightmost children of the top nodes in stack, which could miss some valuable information hidden in the unchosen nodes. Besides, the features of the selected nodes are just simply concatenated and then fed into neural network. Since the concatenation operation is relatively simple, it is difficult to model the com-

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1879–1889,
Lisbon, Portugal, 17-21 September 2015.

plicated feature combinations which can be manually designed in the traditional discrete feature based methods.

To tackle these problems, we use two heterogeneous gated recursive neural networks, tree structured gated recursive neural network (Tree-GRNN) and directed acyclic graph gated structured recursive neural network (DAG-GRNN), to model each configuration during transition based dependency parsing. The two proposed GRNNs introduce the gate mechanism (Chung et al., 2014) to improve the standard recursive neural network (RNN) (Socher et al., 2013; Socher et al., 2014), and can model the syntactic and semantic compositions of the nodes during parsing.

Figure 1 gives a rough sketch for the standard RNN, Tree-GRNN and DAG-GRNN. Tree-GRNN is applied to the partial-constructed trees in stack, which have already been constructed according to the previous transition actions. DAG-GRNN is applied to model the feature composition of nodes in stack and buffer which have not been labeled their dependency relations yet. Intuitively, Tree-GRNN selects and merges features recursively from children nodes into their parent according to their dependency structures, while DAG-GRNN further models the complicated combinations of extracted features and explicitly exploits features in different levels of granularity.

To evaluate our approach, we experiment on two prevalent benchmark datasets: English Penn Treebank 3 (PTB3) and Chinese Penn Treebank 5 (CTB5) datasets. Experiment results show the effectiveness of our proposed method. Compared to the parser of Chen and Manning (2014), we receive 0.6% (UAS) and 0.9% (LAS) improvement on PTB3 test set, while we receive 0.8% (UAS) and 1.3% (LAS) improvement on CTB5 test set.

2 Neural Network Based Transition Dependency Parsing

2.1 Transition Dependency Parsing

In this paper, we employ the arc-standard transition systems (Nivre, 2004) and examine only greedy parsing for its efficiency. Figure 2 gives an example of arc-standard transition dependency parsing.

In transition-based dependency parsing, the consecutive configurations of parsing process can be defined as $c^{(i)} = (s^{(i)}, b^{(i)}, A^{(i)})$ which consists of a stack s, a buffer b, and a set of dependency arcs A. Then, the greedy parsing process consecutively predicts the actions based on the features extracted from the corresponding configurations. For a given sentence $w_1, \dots, w_n$, parsing process starts from a initial configuration $c^{(0)} = ([ROOT], [w_1, \dots, w_n], \emptyset)$, and terminates at some configuration $c^{(2n)} = ([ROOT], \emptyset, A^{(2n)}]$, where n is the length of the given sentence $w_{1:n}$. As a result, we derive the parse tree of the sentence $w_{1:n}$ according to the arcs set $A^{(2n)}$.

In arc-standard system, there are three types of actions: **Left-Arc**, **Right-Arc** and **Shift**. Denoting $s_j(j = 1, 2, \dots)$ as the j^{th} top element of the stack, and $b_j(j = 1, 2, \dots)$ as the j^{th} front element of the buffer, we can formalize the three actions of arc-standard system as:

- **Left-Arc**(l) adds an arc $s_2 \leftarrow s_1$ with label l and removes s_2 from the stack, resulting a new arc $l(s_1, s_2)$. Precondition: $|s| \geq 3$ (The ROOT node cannot be child node).
- **Right-Arc**(l) adds an arc $s_2 \rightarrow s_1$ with label l and removes s_1 from the stack, resulting a new arc $l(s_2, s_1)$. Precondition: $|s| \geq 2$.
- **Shift** removes b_1 from the buffer, and adds it to the stack. Precondition: $|b| \geq 1$.

The greedy parser aims to predict the correct transition action for a given configuration. There are two versions of parsing: unlabeled and labeled versions. The set of possible action candidates $\mathcal{T} = 2n_l + 1$ in the labeled version of parsing, and $\mathcal{T} = 3$ in the unlabeled version, where n_l is number of different types of arc labels.

2.2 Neural Network Based Parser

In neural network architecture, the words, POS tags and arc labels are mapped into distributed vectors (embeddings). Specifically, given the word embedding matrix $\mathbf{E}^w \in \mathbb{R}^{d_e \times n_w}$, each word w_i is mapped into its corresponding column $\mathbf{e}^w_{w_i} \in \mathbb{R}^{d_e}$ of $\mathbf{E}^w$ according to its index in the dictionary, where d_e is the dimensionality of embeddings and n_w is the dictionary size. Likewise, The POS and arc labels are also mapped into embeddings by the POS embedding matrix $\mathbf{E}^t \in \mathbb{R}^{d_e \times n_t}$ and arc label embedding matrix $\mathbf{E}^l \in \mathbb{R}^{d_e \times n_l}$ respectively, where n_t and n_l are the numbers of distinct POS tags and arc labels respectively. Correspondingly, embeddings of each POS tag t_i and each arc label l_i are $\mathbf{e}^t_{t_i} \in \mathbb{R}^{d_e}$ and $\mathbf{e}^l_{l_i} \in \mathbb{R}^{d_e}$ extracted from $\mathbf{E}^t$ and $\mathbf{E}^l$ respectively.

Configurations				Gold Actions
ID	Stack	Buffer	A	
0	[ROOT]	[He likes story books .]	$\emptyset$	
1	[ROOT He]	[likes story books .]		Shift
2	[ROOT He likes]	[story books .]		Shift
3	[ROOT likes]	[story books .]	A $\cup$ SUB(likes,He)	Left_Arc(SUB)
4	[ROOT likes story]	[books .]		Shift
5	[ROOT likes story books]	[.]		Shift
6	[ROOT likes books]	[.]	A$\cup$NMOD(books,story)	Left_Arc(NMOD)
7	[ROOT likes]	[.]	A$\cup$OBJ(likes,books)	Right_Arc(OBJ)
8	[ROOT likes .]	$\emptyset$		Shift
9	[ROOT likes]	$\emptyset$	A$\cup$P(likes,.)	Right_Arc(P)
10	[ROOT]	$\emptyset$	A$\cup$ROOT(ROOT,likes)	Right_Arc(ROOT)

Figure 2: An example of arc-standard transition dependency parsing.

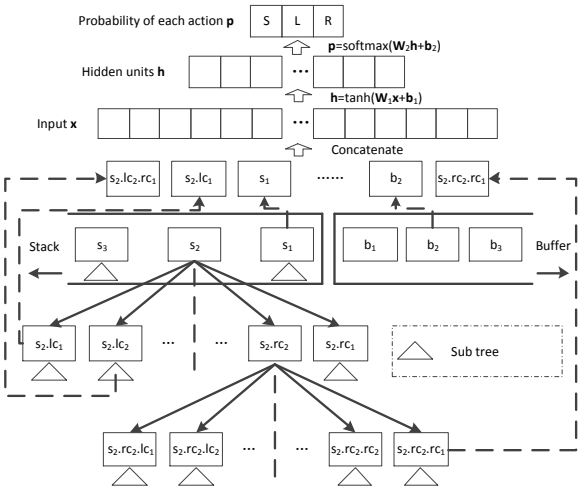


Figure 3: Architecture of neural network based transition dependency parsing.

Figure 3 gives the architecture of neural network based parser. Following Chen and Manning (2014), a set of elements S from stack and buffer (e.g. $S = \{s_2.lc_2.rc_1, s_2.lc_1, s_1, b_2, s_2.rc_2.rc_1, \dots\}$) is chosen as input. Specifically, the information (word, POS or label) of each element in the set S (e.g. $\{s_2.lc_2.rc_1.t, s_2.lc_1.l, s_1.w, s_1.t, b_2.w, \dots\}$) are extracted and mapped into their corresponding embeddings. Then these embeddings are concatenated as the input vector $\mathbf{x} \in \mathbb{R}^{\hat{d}}$. A special token NULL is used to represent a non-existent element.

We perform a standard neural network using one hidden layer with d_h hidden units followed by a softmax layer as:

$$\mathbf{h} = g(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1), \quad (1)$$

$$\mathbf{p} = \text{softmax}(\mathbf{W}_2\mathbf{h} + \mathbf{b}_2), \quad (2)$$

where $\mathbf{W}_1 \in \mathbb{R}^{d_h \times \hat{d}}$, $\mathbf{b}_1 \in \mathbb{R}^{d_h}$, $\mathbf{W}_2 \in \mathbb{R}^{|\mathcal{T}| \times d_h}$, $\mathbf{b}_2 \in \mathbb{R}^{|\mathcal{T}|}$.Here, g is a non-linear function which can be hyperbolic tangent, sigmoid, cube (Chen and Manning, 2014), etc.

3 Recursive Neural Network

Recursive neural network (RNN) is one of classical neural networks, which performs the same set of parameters recursively on a given structure (e.g. syntactic tree) in topological order (Pollack, 1990; Socher et al., 2013).

In the simplest case, children nodes are combined into their parent node using a weight matrix $\mathbf{W}$ which is shared across the whole network, followed by a non-linear function $g(\cdot)$. Specifically, given the left child node vector $\mathbf{h}_L \in \mathbb{R}^d$ and right child node vector $\mathbf{h}_R \in \mathbb{R}^d$, their parent node vector $\mathbf{h}_P \in \mathbb{R}^d$ will be formalized as:

$$\mathbf{h}_P = g\left(\mathbf{W}\begin{bmatrix}\mathbf{h}_L \\ \mathbf{h}_R\end{bmatrix}\right), \quad (3)$$

where $\mathbf{W} \in \mathbb{R}^{d \times 2d}$ and g is a non-linear function as mentioned above.

4 Architecture of Two Heterogeneous Gated Recursive Neural Networks for Transition-based Dependency Parsing

In this paper, we apply the idea of recursive neural network (RNN) to dependency parsing task. RNN needs a pre-defined topological structure. However, in each configuration during parsing, just partial dependency relations have been constructed, while the remains are still unknown. Besides, the standard RNN can just deal with the binary tree. Therefore we cannot apply the standard RNN directly.

Here, we propose two heterogeneous recursive neural networks: tree structured gated recursive neural network (Tree-GRNN) and directed acyclic graph structured gated recursive neural network

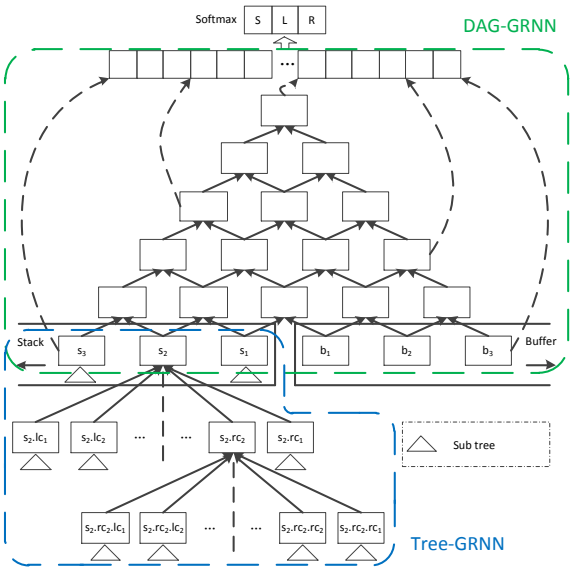


Figure 4: Architecture of our proposed dependency parser using two heterogeneous gated recursive neural networks.

(DAG-GRNN). Tree-GRNN is applied to the subtrees with partial dependency relations in stack which have already been constructed according to the previous transition actions. DAG-GRNN is employed to model the feature composition of nodes in stack and buffer which have not been labeled their dependency relations yet.

Figure 4 shows the whole architecture of our model, which integrates two different GRNNs to predict the action for each parsing configuration. The detailed descriptions of two GRNNs will be discussed in the following two subsections.

4.1 Tree Structured Gated Recursive Neural Network

It is a natural way to merge the information from children nodes into their parent node recursively according to the given tree structures in stack. Although the dependency relations have been built, it is still hard to apply the recursive neural network (as Eq. 3) directly for the uncertain number of children of each node in stack. By averaging operation on children nodes (Socher et al., 2014), the parent node cannot well capture the crucial features from the mixed information of its children nodes. Here, we propose tree structured gated recursive neural network (Tree-GRNN) incorporating the gate mechanism (Cho et al., 2014; Chung et al., 2014; Chen et al., 2015a; Chen et al., 2015b), which can selectively choose the

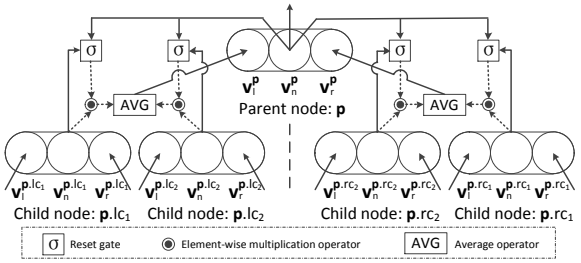


Figure 5: Minimal structure of tree structured gated recursive neural network (Tree-GRNN). The solid arrow denotes that there is a weight matrix on the link, while the dashed one denotes none.

crucial features according to the gate state. Figure 5 shows the minimal structure of Tree-GRNN model.

In Tree-GRNN, each node $\mathbf{p}$ of trees in stack is composed of three components: state vector of left children nodes $\mathbf{v}_l^{\mathbf{p}} \in \mathbb{R}^{d_c}$, state vector of current node $\mathbf{v}_n^{\mathbf{p}} \in \mathbb{R}^{d_n}$ and state vector of right children nodes $\mathbf{v}_r^{\mathbf{p}} \in \mathbb{R}^{d_c}$, where d_n and d_c indicate the corresponding vector dimensionalities. Particularly, we represent information of node $\mathbf{p}$ as a vector

$$\mathbf{v}^{\mathbf{p}} = \begin{bmatrix} \mathbf{v}_l^{\mathbf{p}} \\ \mathbf{v}_n^{\mathbf{p}} \\ \mathbf{v}_r^{\mathbf{p}} \end{bmatrix}, \tag{4}$$

where $\mathbf{v}^{\mathbf{p}} \in \mathbb{R}^q$ and $q = 2d_c + d_n$. Specifically, $\mathbf{v}_n^{\mathbf{p}}$ contains the information of current node including its word form $\mathbf{p}.w$, pos tag $\mathbf{p}.t$ and label type $\mathbf{p}.l$ as shown in Eq. 5, and $\mathbf{v}_l^{\mathbf{p}}$ and $\mathbf{v}_r^{\mathbf{p}}$ are initialized by zero vectors $\mathbf{0} \in \mathbb{R}^{d_c}$, then update as Eq. 6.

$$\mathbf{v}_n^{\mathbf{p}} = \tanh\left(\begin{bmatrix} \mathbf{e}^w_{\mathbf{p}.w} \\ \mathbf{e}^t_{\mathbf{p}.t} \\ \mathbf{e}^l_{\mathbf{p}.l} \end{bmatrix}\right), \tag{5}$$

where word embedding $\mathbf{e}^w_{\mathbf{p}.w} \in \mathbb{R}^{d_e}$, pos embedding $\mathbf{e}^t_{\mathbf{p}.t} \in \mathbb{R}^{d_e}$ and label embedding $\mathbf{e}^l_{\mathbf{p}.l} \in \mathbb{R}^{d_e}$ are extracted from embedding matrices $\mathbf{E}^w$, $\mathbf{E}^t$ and $\mathbf{E}^l$ according to the indices of the corresponding word $\mathbf{p}.w$, pos $\mathbf{p}.t$ and label $\mathbf{p}.l$ respectively. Specifically, in the case of unlabeled attachment parsing, we ignore the last term $\mathbf{e}^l_{\mathbf{p}.l}$ in Eq. 5. Thus, the dimensionality d_n of $\mathbf{v}_n^{\mathbf{p}}$ varies. In labeled attachment parsing case, we set a special token NULL to represent label $\mathbf{p}.l$ if not available (e.g. $\mathbf{p}$ is the node in stack or buffer).

By given node $\mathbf{p}$ and its left children nodes $\mathbf{p}.lc_i$ and right children nodes $\mathbf{p}.rc_i$, we update the left

children information $\mathbf{v}_l^{\mathbf{p}}$ and right children information $\mathbf{v}_r^{\mathbf{p}}$ as

$$\begin{aligned}\mathbf{v}_l^{\mathbf{p}} &= \tanh(\mathbf{W}_l \frac{1}{N_L(\mathbf{p})} \sum_i \mathbf{o}^{\mathbf{p}.lc_i} \odot \mathbf{v}^{\mathbf{p}.lc_i} + \mathbf{b}_l), \\ \mathbf{v}_r^{\mathbf{p}} &= \tanh(\mathbf{W}_r \frac{1}{N_R(\mathbf{p})} \sum_i \mathbf{o}^{\mathbf{p}.rc_i} \odot \mathbf{v}^{\mathbf{p}.rc_i} + \mathbf{b}_r),\end{aligned} \tag{6}$$

where $\mathbf{o}^{\mathbf{p}.lc_i}$ and $\mathbf{o}^{\mathbf{p}.rc_i}$ are the reset gates of the nodes $\mathbf{p}.lc_i$ and $\mathbf{p}.rc_i$ respectively as shown in Eq. 7. In addition, functions $N_L(\mathbf{p})$ and $N_R(\mathbf{p})$ result the numbers of left and right children nodes of node $\mathbf{p}$ respectively. The operator $\odot$ indicates element multiplication here. $\mathbf{W}_l \in \mathbb{R}^{d_c \times q}$ and $\mathbf{W}_r \in \mathbb{R}^{d_c \times q}$ are weight matrices. $\mathbf{b}_l \in \mathbb{R}^{d_c}$ and $\mathbf{b}_r \in \mathbb{R}^{d_c}$ are bias terms.

The reset gates $\mathbf{o}^{\mathbf{p}.lc_i}$ and $\mathbf{o}^{\mathbf{p}.rc_i}$ can be formalized as

$$\begin{aligned}\mathbf{o}^{\mathbf{p}.lc_i} &= \sigma(\mathbf{W}_o \begin{bmatrix} \mathbf{v}^{\mathbf{p}.lc_i} \\ \mathbf{v}_n^{\mathbf{p}} \end{bmatrix} + \mathbf{b}_o), \\ \mathbf{o}^{\mathbf{p}.rc_i} &= \sigma(\mathbf{W}_o \begin{bmatrix} \mathbf{v}^{\mathbf{p}.rc_i} \\ \mathbf{v}_n^{\mathbf{p}} \end{bmatrix} + \mathbf{b}_o),\end{aligned} \tag{7}$$

where σ indicates the sigmoid function, $\mathbf{W}_o \in \mathbb{R}^{q \times (q+d_n)}$ and $\mathbf{b}_o \in \mathbb{R}^q$.

By the mechanism above, we can summarize the whole information into the stack recursively from children nodes to their parent using the partial-built tree structure. Intuitively, the gate mechanism can selectively choose the crucial features of a child node according to the gate state which is derived from the current child node and its parent.

4.2 Directed Acyclic Graph Structured Gated Recursive Neural Network

Previous neural based parsing works feed the extracted features into a standard neural network with one hidden layer. Then, the hidden units are fed into a softmax layer, outputting the probability vector of available actions. Actually, it cannot well model the complicated combinations of extracted features. As for the nodes, whose dependency relations are still unknown, we propose another recursive neural network namely directed acyclic graph structured gated recursive neural network (DAG-GRNN) to better model the interactions of features.

Intuitively, the DAG structure models the combinations of features by recursively mixing the information from the bottom layer to the top layer as shown in Figure 4. The concatenation operation can be regraded as a mix of features in different levels of granularity. Each node in the directed acyclic graph can be seen as a complicated feature composition of its governed nodes.

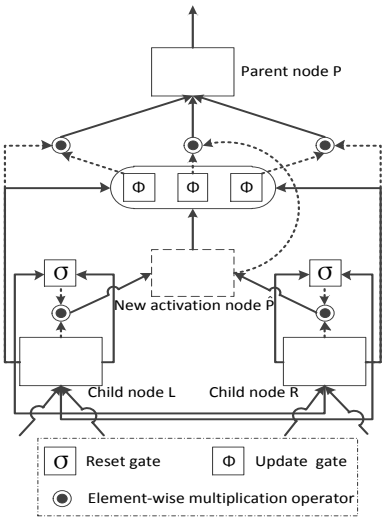


Figure 6: Minimal structure of directed acyclic graph structured gated recursive neural network (DAG-GRNN). The solid arrow denotes that there is a weight matrix on the link, while the dashed one denotes none.

Moreover, we also use the gate mechanism to better model the feature combinations by introducing two kinds of gates, namely "reset gate" and "update gate". Intuitively, each node in the network seems to preserve all the information of its governed notes without gates, and the gate mechanism similarly plays a role of filter which decides how to selectively exploit the information of its children nodes, discovering and preserving the crucial features.

DAG-GRNN structure consists of minimal structures as shown in Figure 6. Vectors $\mathbf{h}^P$, $\mathbf{h}^L$, $\mathbf{h}^R$ and $\mathbf{h}^{\hat{P}} \in \mathbb{R}^q$ denote the value of the parent node P, left child node L, right child node R and new activation node $\hat{P}$ respectively. The value of parent node $\mathbf{h}^P \in \mathbb{R}^q$ is computed as:

$$\mathbf{h}^P = \mathbf{z}_{\hat{P}} \odot \mathbf{h}^{\hat{P}} + \mathbf{z}_L \odot \mathbf{h}^L + \mathbf{z}_R \odot \mathbf{h}^R, \tag{8}$$

where $\mathbf{z}_{\hat{P}}$, $\mathbf{z}_L$ and $\mathbf{z}_R \in \mathbb{R}^q$ are update gates for new activation node $\hat{P}$, left child node L and right child node R respectively. Operator $\odot$ indicates element-wise multiplication.

The update gates $\mathbf{z}$ can be formalized as:

$$\mathbf{z} = \begin{bmatrix} \mathbf{z}_{\hat{P}} \\ \mathbf{z}_L \\ \mathbf{z}_R \end{bmatrix} \propto \exp(\mathbf{W}_z \begin{bmatrix} \mathbf{h}^{\hat{P}} \\ \mathbf{h}^L \\ \mathbf{h}^R \end{bmatrix}), \quad (9)$$

which are constrained by:

$$\begin{cases} [\mathbf{z}_{\hat{P}}]_k + [\mathbf{z}_L]_k + [\mathbf{z}_R]_k = 1, & 1 \le k \le q, \\ [\mathbf{z}_{\hat{P}}]_k \ge 0, & 1 \le k \le q, \\ [\mathbf{z}_L]_k \ge 0, & 1 \le k \le q, \\ [\mathbf{z}_R]_k \ge 0, & 1 \le k \le q, \end{cases} \quad (10)$$

where $\mathbf{W}_z \in \mathbb{R}^{3q \times 3q}$ is the coefficient of update gates.

The value of new activation node $\mathbf{h}_{\hat{P}}$ is computed as:

$$\mathbf{h}_{\hat{P}} = \tanh(\mathbf{W}_{\hat{P}} \begin{bmatrix} \mathbf{r}_L \odot \mathbf{h}_L \\ \mathbf{r}_R \odot \mathbf{h}_R \end{bmatrix}), \quad (11)$$

where $\mathbf{W}_{\hat{P}} \in \mathbb{R}^{q \times 2q}$, $\mathbf{r}_L \in \mathbb{R}^q$, $\mathbf{r}_R \in \mathbb{R}^q$. $\mathbf{r}_L$ and $\mathbf{r}_R$ are the reset gates for left child node L and right child node R respectively, which can be formalized as:

$$\mathbf{r} = \begin{bmatrix} \mathbf{r}_L \\ \mathbf{r}_R \end{bmatrix} = \sigma(\mathbf{W}_r \begin{bmatrix} \mathbf{h}_L \\ \mathbf{h}_R \end{bmatrix}), \quad (12)$$

where $\mathbf{W}_r \in \mathbb{R}^{2q \times 2q}$ is the coefficient of two reset gates and σ indicates the sigmoid function.

Intuitively, the reset gates $\mathbf{r}$ partially read the information from the left and right children, outputting a new activation node $\mathbf{h}_{\hat{P}}$, while the update gates $\mathbf{z}$ selectively choosing the information among the the new activation node $\hat{P}$, the left child node L and the right child node R. This gate mechanism is effective to model the combinations of features.

Finally, we concatenate all the nodes in the DAG-GRNN structure as input $\mathbf{x}$ of the architecture described in Section 2.2, resulting the probability vector for all available actions.

4.3 Inference

We use greedy decoding in parsing. At each step, we apply our two GRNNs on the current configuration to extract the features. After softmax operation, we choose the feasible transition with the highest possibility, and perform the chosen transition on the current configuration to get the next configuration state.

In practice, we do not need calculate the Tree-GRNN over the all trees in the stack on the current configuration. Instead, we preserve the representations of trees in the stack. When we need apply a new transition on the configuration, we update the relative representations using Tree-GRNN.

5 Training

We use the maximum likelihood (ML) criterion to train our model. By extracting training set (x_i, y_i) from gold parse trees using a shortest stack oracle which always prefers Left-Arc(l) or Right-Arc(l) action over Shift, the goal of our model is to minimize the loss function with the parameter set θ:

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m} \log \mathbf{p}(y_i|x_i;\theta) + \frac{\lambda}{2m}\|\theta\|_2^2, \quad (13)$$

where m is number of extracted training examples which is as same as the number of all configurations.

Following Socher et al. (2013), we use the diagonal variant of AdaGrad (Duchi et al., 2011) with minibatch strategy to minimize the objective. We also employ dropout strategy to avoid overfitting.

In practice, we perform DAG-GRNN with two hidden layers, which gets the best performance. We use the approximated gradient for Tree-GRNN, which only performs gradient back propagation on the first two layers.

6 Experiments

6.1 Datasets

To evaluate our proposed model, we experiment on two prevalent datasets: English Penn Treebank 3 (PTB3) and Chinese Penn Treebank 5 (CTB5) datasets.

- **English** For English Penn Treebank 3 (PTB3) dataset, we use sections 2-21 for training, section 22 and section 23 as development set and test set respectively. We adopt CoNLL Syntactic Dependencies (CD) (Johansson and Nugues, 2007) using the LTH Constituent-to-Dependency Conversion Tool.
- **Chinese** For Chinese Penn Treebank 5 (CTB5) dataset, we follow the same split as described in (Zhang and Clark, 2008). Dependencies are converted by the Penn2Malt tool with the head-finding rules of (Zhang and Clark, 2008).

Embedding size	$d_e = 50$
Dimensionality of child node vector	$d_c = 50$
Initial learning rate	$\alpha = 0.05$
Regularization	$\lambda = 10^{-8}$
Dropout rate	$p = 20\%$

Table 1: Hyper-parameter settings.

6.2 Experimental Settings

For parameter initialization, we use random initialization within (-0.01, 0.01) for all parameters except the word embedding matrix $\mathbf{E}^w$. Specifically, we adopt pre-trained English word embeddings from (Collobert et al., 2011). And we pre-train the Chinese word embeddings on a huge unlabeled data, the Chinese Wikipedia corpus, with word2vec toolkit (Mikolov et al., 2013a).

Table 1 gives the details of hyper-parameter settings of our approach. In addition, we set mini-batch size to 20. In all experiments, we only take s_1, s_2, s_3 nodes in stack and b_1, b_2, b_3 nodes in buffer into account. We also apply dropout strategy here, and only dropout at the nodes in stack and buffer with probability $p = 20\%$.

6.3 Results

The experiment results on PTB3 and CTB5 datasets are list in Table 2 and 3 respectively. On all datasets, we report unlabeled attachment scores (UAS) and labeled attachment scores (LAS). Conventionally, punctuations are excluded in all evaluation metrics.

To evaluate the effectiveness of our approach, we compare our parsers with feature-based parser and neural-based parser. For feature-based parser, we compare our models with two prevalent parsers: MaltParser (Nivre et al., 2006) and MSTParser (McDonald and Pereira, 2006). For neural-based parser, we compare our results with parser of Chen and Manning (2014). Compared with parser of Chen and Manning (2014), our parser with two heterogeneous gated recursive neural networks (Tree-GRNN+DAG-GRNN) receives 0.6% (UAS) and 0.9% (LAS) improvement on PTB3 test set, and receives 0.8% (UAS) and 1.3% (LAS) improvement on CTB5 test set.

Since that speed of algorithm is not the focus of our paper, we do not optimize the speed a lot. On CTB (UAS), it takes about 2 days to train Tree-GRNN+DAG-GRNN model with CPU only. The testing speed is about 2.7 sentences per second. All implementation is based on Python.

6.4 Effects of Gate Mechanisms

We adopt five different models: plain parser, Tree-RNN parser, Tree-GRNN parser, Tree-RNN+DAG-GRNN parser, and Tree-GRNN+DAG-GRNN parser. The experiment results show the effectiveness of our proposed two heterogeneous gated recursive neural networks.

Specifically, plain parser is as same as parser of Chen and Manning (2014). The difference between them is that plain parser only takes the nodes in stack and buffer into account, which uses a simpler feature template than parser of Chen and Manning (2014). As plain parser omits all children nodes of trees in stack, it performs poorly compared with parser of Chen and Manning (2014). In addition, we find plain parser outperforms MaltParser (standard) on PTB3 dataset making about 1% progress, while it performs poorer than MaltParser (standard) on CTB5 dataset. It shows that the children nodes of trees in stack is of great importance, especially for Chinese. Moreover, it also shows the effectiveness of neural network based model which could represent complicated features as compacted embeddings. Tree-RNN parser additionally exploits all the children nodes of trees in stack, which is a simplified version of Tree-GRNN without incorporating the gate mechanism described in Section 4.1. In anther word, Tree-RNN omits the gate terms $\mathbf{o}^{\mathbf{p}.lc_i}$ and $\mathbf{o}^{\mathbf{p}.rc_i}$ in Eq. 6. As we can see, the results are significantly boosted by utilizing the all information in stack, which again shows the importance of children nodes of trees in stack. Although the results of Tree-RNN are comparable to results of Chen and Manning (2014), it not outperforms parser of Chen and Manning (2014) in all cases (e.g. UAS on CTB5), which implies that exploiting all information without selection might lead to incorporate noise features. Moreover, Tree-GRNN parser further boosts the performance by incorporating the gate mechanism. Intuitively, Tree-RNN who exploits all the information of stack without selection cannot well capture the crucial features, while Tree-GRNN with gate mechanism could selectively choose and preserve the effective features by adapting the current gate state.

We also experiment on parsers using two heterogeneous gated recursive neural networks: Tree-RNN+DAG-GRNN parser and Tree-GRNN+DAG-GRNN parser. The similarity of

models	Dev		Test	
	UAS	LAS	UAS	LAS
Malt:standard	90.0	88.8	89.9	88.5
Malt:eager	90.1	88.9	90.1	88.7
MSTParser	92.1	90.8	92.0	90.5
Chen's Parser	92.2	91.0	92.0	90.7
Plain	91.1	90.0	91.2	89.7
Tree-RNN	92.4	91.0	92.1	90.8
Tree-GRNN	92.6	91.1	92.4	91.0
Tree-RNN+DAG-GRNN	**92.8**	**91.9**	92.4	91.5
Tree-GRNN+DAG-GRNN	92.6	**91.9**	**92.6**	**91.6**

Table 2: Performance of different models on PTB3 dataset. UAS: unlabeled attachment score. LAS: labeled attachment score.

models	Dev		Test	
	UAS	LAS	UAS	LAS
Malt:standard	82.4	80.5	82.4	80.6
Malt:eager	91.2	79.3	80.2	78.4
MSTParser	84.0	82.1	83.0	81.2
Chen's Parser	84.0	82.4	83.9	82.4
Plain	81.6	79.3	81.1	78.8
Tree-RNN	83.5	82.5	83.8	82.7
Tree-GRNN	84.2	82.5	84.3	83.1
Tree-RNN+DAG-GRNN	84.5	83.3	84.5	83.1
Tree-GRNN+DAG-GRNN	**84.6**	**83.6**	**84.7**	**83.7**

Table 3: Performance of different models on CTB5 dataset. UAS: unlabeled attachment score. LAS: labeled attachment score.

two parsers is that they all employ the DAG structured recursive neural network with gate mechanism to model the combination of features extracted from stack and buffer. The difference between them is the former one employs the Tree-RNN without gate mechanism to model the features of stack, while the later one employs the gated version (Tree-GRNN). Again, the performance of these two parsers is further boosted, which shows DAG-GRNN can well model the combinations of features which is summarized by Tree-(G)RNN structure. In addition, we find the performance does not drop a lot in almost cases by turning off the gate mechanism of Tree-GRNN, which implies that the DAG-GRNN can help selecting the information from trees in stack, even it has not been selected by gate mechanism of Tree-GRNN yet.

6.5 Convergency Speed

To further analyze the convergency speed of our approach, we compare the UAS results on development sets of two datasets for first ten epoches as shown in Figure 7 and 8. As plain parser only take the nodes in stack and buffer into account, the performance is much poorer than the rest parsers. Moreover, Tree-GRNN converges slower than Tree-RNN, which shows that it might be more difficult to learn this gate mechanism. By introducing the DAG-GRNN, both Tree-RNN and Tree-GRNN parsers become faster to converge, which shows that the DAG-GRNN is of great help in boosting the convergency speed.

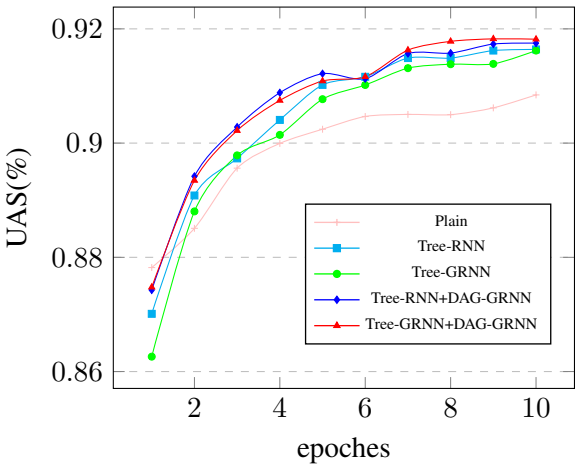


Figure 7: Performance of different models on PTB3 development set. UAS: unlabeled attachment score.

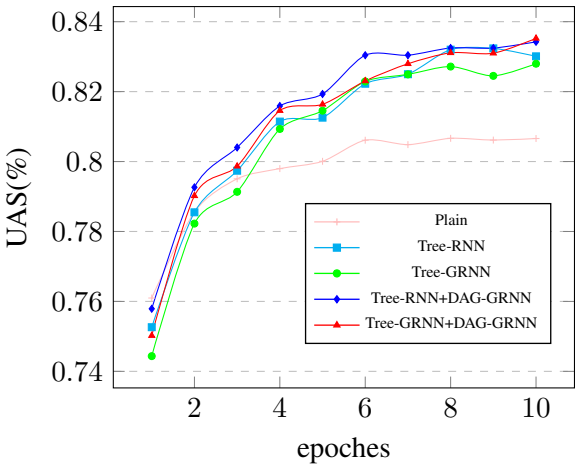


Figure 8: Performance of different models on CTB5 development set. UAS: unlabeled attachment score.

7 Related Work

Many neural network based methods have been used for transition based dependency parsing.

Chen et al. (2014) and Bansal et al. (2014) used the dense vectors (embeddings) to represent words or features and found these representations are

complementary to the traditional discrete feature representation. However, these two methods only focus on the dense representations (embeddings) of words or features.

Stenetorp (2013) first used RNN for transition based dependency parsing. He followed the standard RNN and used the binary combination to model the representation of two linked words. But his model does not achieve the performance of the traditional method.

Le and Zuidema (2014) proposed a generative re-ranking model with Inside-Outside Recursive Neural Network (IORNN), which can process trees both bottom-up and top-down. However, IORNN works in generative way and just estimates the probability of a given tree, so IORNN cannot fully utilize the incorrect trees in k-best candidate results. Besides, IORNN treats dependency tree as a sequence, which can be regarded as a generalization of simple recurrent neural network (SRNN) (Elman, 1990).

Although the two methods also used RNN, they just deal with the binary combination, which is unnatural for dependency tree.

Zhu et al. (2015) proposed a recursive convolutional neural network (RCNN) architecture to capture syntactic and compositional-semantic representations of phrases and words in a dependency tree. Different with the original recursive neural network, they introduced the convolution and pooling layers, which can model a variety of compositions by the feature maps and choose the most informative compositions by the pooling layers.

Chen and Manning (2014) improved the transition-based dependency parsing by representing all words, POS tags and arc labels as dense vectors, and modeled their interactions with neural network to make predictions of actions. Their method only relies on dense features, and is not able to automatically learn the most useful feature conjunctions to predict the transition action.

Compared with (Chen and Manning, 2014), our method can fully exploit the information of all the descendants of a node in stack with Tree-GRNN. Then DAG-GRNN automatically learns the complicated combination of all the features, while the traditional discrete feature based methods need manually design them.

Dyer et al. (2015) improved the transition-based dependency parsing using stack long short term memory neural network and received significant improvement on performance. They focused on exploiting the long distance dependencies and information, while we aims to automatically model the complicated feature combination.

8 Conclusion

In this paper, we pay attention to the syntactic and semantic composition of the dense features for transition-based dependency parsing. We propose two heterogeneous gated recursive neural networks, Tree-GRNN and DAG-GRNN. Each hidden neuron in two proposed GRNNs can be regarded as a different combination of the input features. Thus, the whole model has an ability to simulate the design of the sophisticated feature combinations in the traditional discrete feature based methods.

Although the two proposed GRNNs are only used for the greedy parsing based on arc-standard transition system in this paper, it is easy to generalize them to other transition systems and graph based parsing. In future work, we would also like to extend our GRNNs for the other NLP tasks.

Acknowledgments

We would like to thank the anonymous reviewers for their valuable comments. This work was partially funded by the National Natural Science Foundation of China (61472088, 61473092), National High Technology Research and Development Program of China (2015AA015408), Shanghai Science and Technology Development Funds (14ZR1403200).

References

Miguel Ballesteros and Bernd Bohnet. 2014. Automatic feature selection for agenda-based dependency parsing. In *Proceedings of the 25th International Conference on Computational Linguistics*.

Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2014. Tailoring continuous word representations for dependency parsing. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *The Journal of Machine Learning Research*, 3:1137–1155.

Bernd Bohnet. 2010. Very high accuracy and fast dependency parsing is not a contradiction. In *Proceedings of the 23rd International Conference on Com-*

putational Linguistics, pages 89–97. Association for Computational Linguistics.

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

Wenliang Chen, Yue Zhang, and Min Zhang. 2014. Feature embedding for dependency parsing. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 816–826, Dublin, Ireland, August.

Xinchi Chen, Xipeng Qiu, Chenxi Zhu, and Xuanjing Huang. 2015a. Gated recursive neural network for Chinese word segmentation. In *Proceedings of Annual Meeting of the Association for Computational Linguistics*.

Xinchi Chen, Xipeng Qiu, Chenxi Zhu, Shiyu Wu, and Xuanjing Huang. 2015b. Sentence modeling with gated recursive neural network. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Proceedings of EMNLP*.

Jinho D Choi and Andrew McCallum. 2013. Transition-based dependency parsing with selectional branching. In *ACL (1)*, pages 1052–1062.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

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

Jacob Devlin, Rabih Zbib, Zhongqiang Huang, Thomas Lamar, Richard Schwartz, and John Makhoul. 2014. Fast and robust neural network joint models for statistical machine translation. In *52nd Annual Meeting of the Association for Computational Linguistics, Baltimore, MD, USA, June*.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A Smith. 2015. Transition-based dependency parsing with stack long short-term memory. *arXiv preprint arXiv:1505.08075*.

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

Yoav Goldberg and Joakim Nivre. 2013. Training deterministic parsers with non-deterministic oracles. *TACL*, 1:403–414.

He He, Hal Daumé III, and Jason Eisner. 2013. Dynamic feature selection for dependency parsing. In *EMNLP*, pages 1455–1464.

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

Richard Johansson and Pierre Nugues. 2007. Extended constituent-to-dependency conversion for english. In *16th Nordic Conference of Computational Linguistics*, pages 105–112. University of Tartu.

Terry Koo, Xavier Carreras, and Michael Collins. 2008. Simple semi-supervised dependency parsing. In *ACL*.

Sandra Kübler, Ryan McDonald, and Joakim Nivre. 2009. Dependency parsing. *Synthesis Lectures on Human Language Technologies*, 1(1):1–127.

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

André FT Martins, Noah A Smith, and Eric P Xing. 2009. Concise integer linear programming formulations for dependency parsing. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pages 342–350. Association for Computational Linguistics.

Ryan T McDonald and Fernando CN Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *EACL*.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 91–98.

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

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

Joakim Nivre, Johan Hall, and Jens Nilsson. 2006. Maltparser: A data-driven parser-generator for dependency parsing. In *Proceedings of LREC*, volume 6, pages 2216–2219.

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

Jordan B Pollack. 1990. Recursive distributed representations. *Artificial Intelligence*, 46(1):77–105.

Richard Socher, John Bauer, Christopher D Manning, and Andrew Y Ng. 2013. Parsing with compositional vector grammars. In *In Proceedings of the ACL conference*. Citeseer.

Richard Socher, Andrej Karpathy, Quoc V Le, Christopher D Manning, and Andrew Y Ng. 2014. Grounded compositional semantics for finding and describing images with sentences. *Transactions of the Association for Computational Linguistics*, 2:207–218.

Pontus Stenetorp. 2013. Transition-based dependency parsing using recursive neural networks. In *NIPS Workshop on Deep Learning*.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 384–394. Association for Computational Linguistics.

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

Yue Zhang and Joakim Nivre. 2011. Transition-based dependency parsing with rich non-local features. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 188–193.

Chenxi Zhu, Xipeng Qiu, Xinchi Chen, and Xuanjing Huang. 2015. A re-ranking model for dependency parser with recursive convolutional neural network. In *Proceedings of Annual Meeting of the Association for Computational Linguistics*.

Turn-taking phenomena in incremental dialogue systems

Hatim Khouzaimi
Orange Labs
LIA-CERI, Univ. Avignon
hatim.khouzaimi@orange.com

Romain Laroche
Orange Labs
Issy-les-Moulineaux
France
romain.laroche@orange.com

Fabrice Lefèvre
LIA-CERI, Univ. Avignon
Avignon
France
fabrice.lefevre@univ-avignon.fr

Abstract

In this paper, a turn-taking phenomenon taxonomy is introduced, organised according to the level of information conveyed. It is aimed to provide a better grasp of the behaviours used by humans while talking to each other, so that they can be methodically replicated in spoken dialogue systems. Five interesting phenomena have been implemented in a simulated environment: the system barge-in with three variants (resulting from either an unclear, an incoherent or a sufficient user message), the feedback and the user barge-in. The experiments reported in the paper illustrate that how such phenomena are implemented is a delicate choice as their impact on the system's performance is variable.

1 Introduction

A spoken dialogue system is said to be incremental when it does not wait until the end of the user's utterance in order to process it (Dohsaka and Shimazu, 1997; Allen et al., 2001; Schlangen and Skantze, 2011). New audio information is captured by an incremental Automatic Speech Recognition (ASR) at a certain frequency (Breslin et al., 2013) and at each new step, the partial available information is processed immediately. Therefore, the system is able to replicate a rich set of **turn-taking phenomena** (TTP) that are performed by human beings when talking to each other (Sacks et al., 1974; Clark, 1996). Replicating these TTP in dialogue systems can help to make them more efficient (e.g. (El Asri et al., 2014)) and enhance their ability to recover from misunderstandings (Skantze and Schlangen, 2009).

Several contributions already explored different TTP like end-point detection (Raux and Eskenazi, 2008), backchannels (Meena et al., 2014; Visser et al., 2014), feedback (Skantze and Schlangen, 2009) or barge-in (Selfridge et al., 2013; Ghigi et al., 2014). However, these studies have been performed separately with no unified view and no comparison of respective merits, importance and co-influence of the different TTP. In order to have a better grasp on the concept of turn-taking in a dialogue and a guideline for the implementation, we felt the need to introduce a *taxonomy* of these TTP. Our motivation is to clarify which TTP are interesting to implement given the task at hand. As an illustration, five TTP (which we assume have the best properties to improve the dialogue efficiency) have been implemented and compared in a slot-filling simulated environment.

Section 2 introduces the TTP taxonomy and Section 3 describes the simulated environment, the experimental setup and the results. We then conclude in Section 4.

2 Turn-taking phenomena taxonomy

In linguistics and philosophy of language, a distinction is made between two different levels of a speech act analysis: *locutionary acts* and *illocutionary acts* (Austin, 1962; Searle, 1969). Loosely speaking, a *locutionary act* refers to the act of uttering sounds without taking their meaning into account. When the semantic information is the object of interest, it is an *illocutionary act*. In (Raux and Eskenazi, 2009), four basic turn-taking transitions are presented: *the turn transitions with gap*, *the turn transitions with overlap*, *the failed interruptions* and *the time outs* where only the mechanics of turn-taking are studied at a locutionary level. In (Gravano and Hirschberg, 2011), the authors propose a *turn-taking labeling scheme*, which is a modified version of the original *classification of interruptions and smooth speaker-switches* introduced in (Beattie, 1982). This classification is richer than the one in (Raux and Eskenazi, 2009) as the meaning of the turn-taker utterance

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1890–1895,
Lisbon, Portugal, 17-21 September 2015.

Table 1: *Turn-taking phenomena taxonomy. The rows/columns correspond to the levels of information added by the floor giver/taker. The phenomena in black have been implemented in the simulator.*

	T_REF_IMPL	**T_REF_RAW**	**T_REF_INTERP**	**T_MOVE**
G_NONE	FLOOR_TAKING_IMPL			INIT_DIALOGUE
G_FAIL	FAIL_IMPL	FAIL_RAW	FAIL_INTERP	
G_INCOHERENCE	INCOHERENCE_IMPL	INCOHERENCE_RAW	INCOHERENCE_INTERP	
G_INCOMPLETE	BACKCHANNEL	FEEDBACK_RAW	FEEDBACK_INTERP	
G_SUFFICIENT	REF_IMPL	REF_RAW	REF_INTERP	BARGE_IN_RESP
G_COMPLETE	REKINDLE			END_POINT

is taken into account. From a computational point of view, it is more interesting to add high-level information to classify these behaviours as semantics clearly influence turn-taking decisions (Duncan, 1972; Gravano and Hirschberg, 2011). In this paper, a more fine-grained taxonomy of TTP is introduced where utterances are considered both at locutionary and illocutionary levels.

During a floor transition, the person who starts speaking will be called T (Taker) whereas the person that was speaking just before will be called G (Giver). At the beginning of the dialogue, the person that initiates the dialogue will be called T and the other G by convention. We classify TTP given two criteria: the *quantity of information that has been injected by G* before the floor transition (rows in Table 1) and the *quantity of information that T tries to add by taking the floor* (columns in Table 1). Table 2 gives the meaning of the different criteria's labels.

Table 2: *Taxonomy labels*

G_NONE	No information given
G_FAIL	Failed trial
G_INCOHERENT	Incoherent information
G_INCOMPLETE	Incomplete information
G_INSUFFICIENT	Insufficient information
G_SUFFICIENT	Sufficient information
G_COMPLETE	Complete utterance
T_REF_IMPL	Implicit ref. to G's utterance
T_REF_RAW	Raw ref. to G's utterance
T_REF_INTERP	Reference with interpretation
T_MOVE	Dialogue move (with improvement)

At the beginning of the dialogue (G_NONE), T can implicitly announce that she wants to take the floor by using hand gestures or by clearing her throat for instance (FLOOR_TAKING_IMPL), or she can directly initiate the dialogue (INIT_DIALOGUE). If G is already speaking, her message can be not understandable by T (G_FAIL). T can warn G implicitly by frowning for example (FAIL_IMPL) or explicitly, in a raw manner by saying *Sorry?* (FAIL_RAW) or by pointing out what has not been understood (FAIL_INTERP). In addition, even if the meaning of the message has been understood, it can be incoherent with the interaction context (G_INCOHERENT, e.g. trying to book a flight from a city with no airport). Again, T can warn G implicitly (INCOHERENCE_IMPL) or explicitly, either by explaining the reason of the problem (INCOHERENCE_INTERP) or not (INCOHERENCE_RAW).

In the case G's utterance is not problematic but yet incomplete (G_INCOMPLETE), T can let her understand that she understands what has been said so far by performing a BACKCHANNEL (*Yes*, *uhum* etc.), by repeating his words exactly (FEEDBACK_RAW) or by commenting them (FEEDBACK_INTERP), for example: *Yesterday I went to this new Chinese restaurant in town... / Yeah Fing Shui / ...and it was a pretty good deal*). If G utters enough information to move the dialogue forward (G_SUFFICIENT), T can refer to an element in G's utterance implicitly (*Aha*) by reacting at the proper timing (REF_IMPL), or explicitly in a raw (REF_RAW, for example *Ok, Sunday*) or interpreted manner (REF_INTERP, for example *Yeah, Sunday is the only day when I am free*). T can also interrupt G to add some information that is relevant to the course of the dialogue (BARGE_IN_RESP). Finally, she can wait until G has finished his utterance (G_COMPLETE) and warn him that he should add more information (REKINDLE, for example: *And?*) or start a new dialogue turn (END_POINT).

In the rest of this paper, five incremental TTP that are the more used in general, and therefore studied, have been tested in a simulated environment: FAIL_RAW (Ghigi et al., 2014), INCOHERENCE_INTERP (DeVault et al., 2011), FEEDBACK_RAW (Skantze and Schlangen, 2009) and BARGE_IN_RESP from both sides, user (Selfridge et al., 2013) and system interruptions (DeVault et al., 2011), along with

INIT_DIALOGUE and END_POINT that already exist in traditional systems.

3 Simulation

3.1 Service task

A personal agenda assistant task has been implemented in the simulated dialogue system (referred to as the *service* hereafter). The user can add events to her agenda as long as they do not overlap with existing events. She can also move events in the agenda or delete them (ADD, MODIFY and DELETE actions). An event corresponds to a title, a date, a time slot, a priority, and the list of alternative dates and time slots where the event can fit, in the case the main date and slot are not available. For example: {**title:** *house cleaning*, **date:** *January* 6^{th}, **slot:** *from 18 to 20*, **priority:** *3*, **alternative 1:** *January* 7^{th}*, from 18 to 20*, **alternative 2:** *January* 9^{th}*, from 10 to 12*}.

3.2 User simulator

3.2.1 Overview

The architecture of the User Simulator (US) is built around five modules: the Natural Language Understanding (NLU) module, the Intent Manager, the Natural Language Generator (NLG), the Verbosity Manager, the ASR Output Simulator, and the Patience Manager. These modules are described in the following.

NLU module: The NLU module is very simple as the service's utterances are totally known by the US and no parsing is involved. Each one of them is associated with a specific dialogue act.

Intent Manager: The Intent Manager is somehow the brain of the US, as it determines its next intent given the general goal and the last NLU result. The general goal depends on the scenario at hand, which is in turn determined by two lists of events: the initial list (*InitList*) and the list of events to add to the agenda during the dialogue (*ToAddList*). The Intent Manager tries to add each event from the latter given the constraints imposed by the former. If the events of both lists cannot be kept, those with lower priorities are abandoned or deleted until a solution is reached.

The service asks for the different slot values in a mixed initiative way. At first, the user has the initiative in the sense that she is asked to provide all the slot values in the same utterance. If there is still missing information (because the user did not provide all the slot values or because of ASR noise), the remaining slot values are asked for one by one (system initiative).

NLG module: The NLG figures out the next sentence to utter given the current Intent Manager's output. A straightforward sentence is computed, for example, *Add the event meeting Mary on July* 6^{th} *from 18:00 until 20:00*.

Verbosity Manager: The Verbosity Manager randomly expands the NLG output with some usual prefixes (like *I would like to...*) and suffixes (like *please*, *if possible*...). Also, a few sentences are replaced with off-domain words or repeated twice as it is the case in real dialogues (Ghigi et al., 2014). For questions concerning a specific slot, neither prefixes nor suffixes are added.

Patience Manager: When the dialogue lasts too long, the US can get impatient and hang up. The US patience corresponds to a threshold on each task duration. It is randomly sampled around a mean of 180 seconds for the experiments. A speech rate of 200 words per minute is assumed for the dialogue duration estimation (Yuan et al., 2006). Moreover, a silence of one second is assumed at each regular system/user transition and a two second silence is assumed the other way round. For interruptions and accurate end-point detection, no silence is taken into account.

3.3 ASR Output Simulator

The US can run either in a *traditional mode* in the sense that it provides a complete utterance to the system then it waits for a response, or in an *incremental mode* where a growing utterance is outputted at each new word. For example: *I*, *I want*, *I want to*, *I want to add*...etc. In incremental dialogue systems, the turn increment (called the *micro-turn* in this case) could be different than the word (a small duration for example).

The ASR output simulator can be used in both modes, but as the traditional mode is a special case of the incremental one, we describe the latter only. This module computes a noisy version of each word (substitution, deletion, or insertion). It also associates a confidence score with each new partial utterance. Moreover, a word in the ASR output can change later as new words pop in (Selfridge et al., 2011; McGraw and Gruenstein, 2012). In the following, this mechanism is referred to as the *ASR instability*. At each micro-turn, the sys-

tem can keep listening to the US or decide to take the floor (see Section 3.4).

3.4 Scheduler

The same architecture as in (Khouzaimi et al., 2014) is used. A *Scheduler* module is inserted between the service and the user simulator. As the ASR output utterance grows, the partial utterances are sent, at each micro-turn, to the Scheduler. In turn, the latter transfers them to the service and waits for its responses.

The aim of this module is to make turn-taking decisions. Given the last system's response and some other features and rules determined by the designer, or learned from data, the Scheduler decides whether or not to convey that response to the client immediately or not.

3.5 Dialogue example

In the following example, the user has to delete an event, before adding another one (ASR noise is not introduced here):

SYSTEM: Hi. Welcome to your agenda management service. How can I help you?

USER: I would like to add the event birthday party on January 6^{th} from 6 pm to 11 pm if it is possible.

SYSTEM: The time slot from 6 pm to 11 pm on January 6^{th} overlaps with the event house cleaning on January 6^{th} from 7 pm to 9 pm. How can I help you?

USER: Delete the event house cleaning.

SYSTEM: Ok. So you want to delete the event house cleaning that takes place on January 6^{th} from 7 pm to 9 pm. Right?

USER: Yes.

SYSTEM: Got it. Do you want to perform some other action?

USER: Yes.

SYSTEM: Add the event birthday party on January 6^{th} from 6 pm to 11 pm.

USER: Ok. So you want to add the event birthday party on January 6^{th} from 6 pm to 11 pm. Right?

USER: Yes.

SYSTEM: Got it. Do you want to perform some other action?

SYSTEM: No.

USER: Thank you. Bye!

3.6 TTP implementation

Replicating some turn-taking phenomena like backchannels makes the system seems more realistic (Meena et al., 2014). In this work, the focus is on dialogue efficiency, therefore, the following TTP have been chosen for the implementation: FAIL_RAW, INCOHERENCE_INTERP, FEEDBACK_RAW and BARGE_IN_RESP from the user's and the system's point of view.

At each micro-turn, the system has to pick an action among three options: to wait (WAIT), to retrieve the last service's response to the client (SPEAK) or to repeat the word at position $n-2$ (if n is the current number of words, because of the ASR instability) in the current partial request (REPEAT). To replicate each selected TTP, a set of rules have been specified to make the proper decision. We review the triggering features related to each TTP accommodated to the task at hand (agenda filling).

FAIL_RAW: Depending on the last system's dialogue act, a threshold relative to the number of words without detecting a key concept in the utterance has been set. In the case of an open question (where the system waits for all the information needed in one request), if no action type has been detected after 6 words, a FAIL_RAW event is declared. The system waits for 3 words in the case of a yes/no question, for 4 words in the case of a date and for 6 words in the case of slots (some concepts need more words to be detected and the user may use additional off-domain words).

INCOHERENCE_INTERP: This event is useful to promptly react to partial requests that would eventually lead to an error, not because they were not correctly understood, but because they are in conflict with the current dialogue state. If such an inconsistency is detected, the system waits for two words (ASR instability) and if it is maintained, it takes the floor to warn the user.

FEEDBACK_RAW: If at time t, a new word is added to the partial utterance and the ratio between the last partial utterance's score and the one before last (which corresponds to the score of the last increment) is lower than 1/2, then the system waits for two words (because of the ASR instability), and if the word is still in the partial utterance, a REPEAT action is performed.

BARGE_IN_RESP (System): This TTP depends on the last system dialogue act as it determines which kind of NLU concept the system is

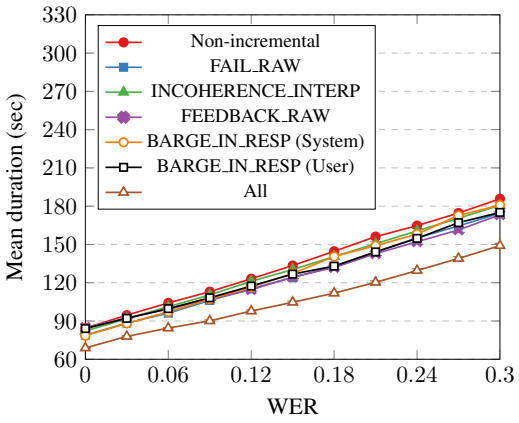


Figure 1: Simulated dialogue duration for different noise levels

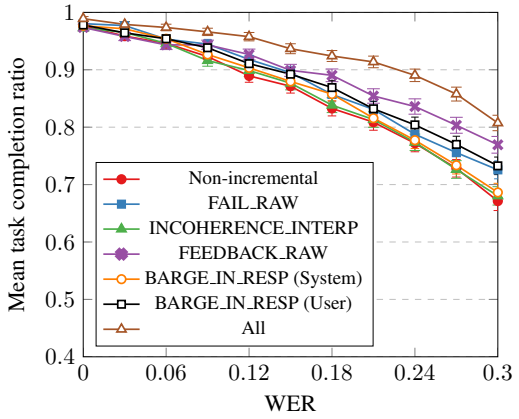


Figure 2: Simulated dialogue task completion for different noise levels

waiting for. Once it is detected, the system waits for two more words (ASR instability) and if the concept is maintained, it performs a SPEAK.

USER_BARGE_RESP (User): This event is triggered directly by the user (no system decision is involved). For each system dialogue act, the moment when a familiar user would barge-in is manually defined in the simulator.

Dialogue duration and task completion are used as evaluation criteria. The task completion rate is the ratio between the number of dialogues where the user did not hang up (because of her patience limit) and the total number of dialogues.

The five implemented TTP have been tested single-handled and in an aggregated manner (referred to as *All* strategy). They have also been been compared to a non-incremental baseline (see Figure 1 and 2). Three dialogue scenarios and different WER levels were tested. For each strategy and each WER, 1000 dialogues have been simulated for each scenario. Figure 1 (resp. Figure 2) represents the mean duration (resp. the mean task completion), with the corresponding 95% confidence intervals, for the different strategies and for WER varying between 0 and 0.3.

The FEEDBACK_RAW strategy performs best whereas INCOHERENCE_INTERP does not improve over the baseline. This is due to the fact that the system has to deal with an open slot (which set of possible values is not closed and known a priori): the event's description. The system mostly performs ADD actions, so the description slot can take any value and is never compared with existing data. This is the case of many application like message dictation for example. However, in the case of service at hand, an initial concept must be detected (the action), therefore, FAIL_RAW improves the performance. BARGE_IN_RESP from user's side is also useful here as dialogues can be long and may contain repetitive system dialogue acts. The users get familiar with the systems and may infer the end of the system's question before it ends. Obviously it is questionable that users may be patient enough (up to several minutes) to achieve such simple tasks in real life. But for the sake of the simulation it was necessary to generate dialogues long enough to have the studied TTP influence them. In a next step, increasing the service capacities (and complexity) will remedy that as a side effect. Finally, BARGE_IN_RESP from the system's side does not bring any improvement either which is due to the fact that in this task and because of input noise, in most cases, the response to the initial open question is not enough to fill all the slots. The responses to single-slot questions do not contain suffixes which explains the inefficiency of the last strategy (the US stops speaking as soon as the slot value is given).

4 Conclusion and future work

This paper introduces a new taxonomy of turn-taking phenomena in human dialogue. Then an experiment where five TTP are implemented has been run in a simulated environment. It illustrates the potentiality of the taxonomy and shows that some TTP are worth replicating in some situations but not all. In future work, we plan to perform TTP analysis in the case of real users and to optimise the hand-crafted rules introduced here to operate the floor management in the system (when to take/give the floor and according to which TTP scheme) by using reinforcement learning (Sutton and Barto, 1998; Lemon and Pietquin, 2012).

References

James Allen, George Ferguson, and Amanda Stent. 2001. An architecture for more realistic conversational systems. In *6th international conference on Intelligent user interfaces*.

J.L. Austin. 1962. *How to Do Things with Words*. Oxford.

Geoffrey Beattie. 1982. Turn-taking and interruption in political interviews: Margaret thatcher and jim callaghan compared and contrasted. *Semiotica*, 39:93–114.

Catherine Breslin, Milica Gasic, Matthew Henderson, Dongho Kim, Martin Szummer, Blaise Thomson, Pirros Tsiakoulis, and Steve Young. 2013. Continuous asr for flexible incremental dialogue. In *ICASSP*, pages 8362–8366.

Herbert H. Clark. 1996. *Using Language*. Cambridge University Press.

David DeVault, Kenji Sagae, and David Traum. 2011. Incremental interpretation and prediction of utterance meaning for interactive dialogue. *Dialogue and Discourse*, 2:143–170.

Kohji Dohsaka and Akira Shimazu. 1997. A system architecture for spoken utterance production in collaborative dialogue. In *IJCAI*.

Starkey Duncan. 1972. Some signals and rules for taking speaking turns in conversations. *Journal of Personality and Social Psychology*, 23:283–292.

Layla El Asri, Remi Lemonnier, Romain Laroche, Olivier Pietquin, and Hatim Khouzaimi. 2014. NASTIA: Negotiating Appointment Setting Interface. In *Proceedings of LREC*.

Fabrizio Ghigi, Maxine Eskenazi, M Ines Torres, and Sungjin Lee. 2014. Incremental dialog processing in a task-oriented dialog. In *Fifteenth Annual Conference of the International Speech Communication Association*.

Agustín Gravano and Julia Hirschberg. 2011. Turn-taking cues in task-oriented dialogue. *Comput. Speech Lang.*, 25(3):601–634.

Hatim Khouzaimi, Romain Laroche, and Fabrice Lefèvre. 2014. An easy method to make dialogue systems incremental. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*.

Oliver Lemon and Olivier Pietquin. 2012. *Data-Driven Methods for Adaptive Spoken Dialogue Systems*. Springer Publishing Company, Incorporated.

Ian McGraw and Alexander Gruenstein. 2012. Estimating word-stability during incremental speech recognition. In *Proceedings of the INTERSPEECH 2012 Conference*.

Raveesh Meena, Johan Boye, Gabriel Skantze, and Joakim Gustafson. 2014. Crowdsourcing street-level geographic information using a spoken dialogue system. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*.

Antoine Raux and Maxine Eskenazi. 2008. Optimizing endpointing thresholds using dialogue features in a spoken dialogue system. In *SIGDIAL*.

Antoine Raux and Maxine Eskenazi. 2009. A finite-state turn-taking model for spoken dialog systems. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, NAACL '09, pages 629–637.

Harvey Sacks, Emanuel A. Schegloff, and Gail Jefferson. 1974. A simplest systematics for the organization of turn-taking for conversation. *Language*, 50:696–735.

David Schlangen and Gabriel Skantze. 2011. A general, abstract model of incremental dialogue processing. *Dialogue and Discourse*, 2:83–111.

John Searle. 1969. *Speech Acts: An Essay in the Philosophy of Language*. Cambridge University Press, UK.

Ethan O. Selfridge, Iker Arizmendi, Peter A. Heeman, and Jason D. Williams. 2011. Stability and accuracy in incremental speech recognition. In *Proceedings of the SIGDIAL 2011 Conference*.

Ethan Selfridge, Iker Arizmendi, Peter Heeman, and Jason Williams. 2013. Continuously predicting and processing barge-in during a live spoken dialogue task. In *Proceedings of the SIGDIAL 2013 Conference*.

Gabriel Skantze and David Schlangen. 2009. Incremental dialogue processing in a micro-domain. In *ACL*.

Richard S. Sutton and Andrew G. Barto. 1998. *Reinforcement Learning, An Introduction*. The MIT Press, Cambridge, Massachusetts, London, England.

Thomas Visser, David Traum, David DeVault, and Rieks op den Akker. 2014. A model for incremental grounding in spoken dialogue systems. *Journal on Multimodal User Interfaces*.

Jiahong Yuan, Mark Liberman, and Christopher Cieri. 2006. Towards an integrated understanding of speaking rate in conversation. In *INTERSPEECH Proceedings*.

Hierarchical Latent Words Language Models for Robust Modeling to Out-Of Domain Tasks

Ryo Masumura[†‡], **Taichi Asami**[†], **Takanobu Oba**[†],
Hirokazu Masataki[†], **Sumitaka Sakauchi**[†], **Akinori Ito**[‡]
[†] NTT Media Intelligence Laboratories, NTT Corporation, Japan
[‡] Graduate School of Engineering, Tohoku University, Japan
†{masumura.ryo, asami.taichi, oba.takanobu, masataki.hirokazu,
sakauchi.sumitaka} @lab.ntt.co.jp, ‡ aito@spcom.ecei.tohoku.ac.jp

Abstract

This paper focuses on language modeling with adequate robustness to support different domain tasks. To this end, we propose a hierarchical latent word language model (h-LWLM). The proposed model can be regarded as a generalized form of the standard LWLMs. The key advance is introducing a multiple latent variable space with hierarchical structure. The structure can flexibly take account of linguistic phenomena not present in the training data. This paper details the definition as well as a training method based on layer-wise inference and a practical usage in natural language processing tasks with an approximation technique. Experiments on speech recognition show the effectiveness of h-LWLM in out-of domain tasks.

1 Introduction

Language models (LMs) are essential for automatic speech recognition or statistical machine translation (Rosenfeld, 2000). The performance of LMs strongly depends on quality and quantity of their training data. Superior performance is usually obtained by using enormous domain-matched training data sets to construct LMs (Brants et al., 2007). Unfortunately, in many cases, large amounts of domain-matched training data sets are not available. Therefore, LM technology that can robustly work for domains that differ from that of the training data is needed (Goodman, 2001).

For robust language modeling, several technologies have been proposed. Fundamental techniques are smoothing (Chen and Goodman, 1999) and clustering (Brown et al., 1992). Other solutions are Bayesian modeling (Teh, 2006) and ensemble modeling (Xu and Jelinek, 2004; Emami and Jelinek, 2005). Moreover, continuous representation of words in neural network LMs can also support robust modeling (Bengio et al., 2003; Mikolov et al., 2010). However, previous works are focused on maximizing performance in the same domain as that of the training data. In other words, it is uncertain that these technologies robustly support out-of domain tasks.

In contrast, latent words LMs (LWLMs) (Deschacht et al., 2012) are clearly effective for out-of domain tasks. We employed the LWLM to speech recognition and the resulting performance was significantly superior in out-of domain tasks while the performance was comparable in domain-matched task to conventional LMs (Masumura et al., 2013a; Masumura et al., 2013b). LWLMs are generative models that employ a latent word space. The latent space can flexibly take into account relationships between words and the modeling helps to efficiently increase the robustness to out-of domain tasks (Sec. 2).

In this paper, we focus on LWLMs and aim to make them more flexible for greater robustness to out-of domain tasks. To this end, this paper takes note of a fact that standard LWLM simply represents the latent space as n-gram model of latent words. However, function and meaning of words are essentially hierarchical and upper layers ought to be useful to increase the robustness to out-of domain tasks. The conventional LWLMs do not model the hierarchy, while the latent words are used to represent function and meaning of words. Thus, we tried to model the hierarchy in the latent space by estimating a latent word of a latent word recursively.

This paper proposes a novel LWLM with multiple latent word spaces that are hierarchically structured; we call it the hierarchical LWLM (h-LWLM). The proposed model can be regarded as a generalized form of the standard LWLMs. The hierarchical structure can take into account the abstraction process of function and meaning of words. Therefore, it can be expected that h-

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1896–1901,
Lisbon, Portugal, 17-21 September 2015.

LWLMs flexibly calculate generative probability for unseen words unlike non-hierarchical LWLMs. To create the hierarchical latent word structure from training data sets, we also propose a layer-wise inference. The inference is inspired by a deep Boltzmann machine (Salakhutdinov and Hinton, 2009) that stacks up restricted Boltzmann machines (Hinton et al., 2006). In addition, we detail an n-gram approximation technique to apply the proposed model to practical natural language processing tasks (see Sec. 3).

In experiments, we construct LMs from spontaneous lecture task data and apply them to a contact center dialogue task and a voice mail task as out-of domain tasks. The effectiveness of the proposed method is shown by perplexity and speech recognition evaluation (Sec. 4).

2 Latent Words Language Models

LWLMs are generative models with single latent word space (Deschacht et al., 2012). The latent word is represented as a specific word that is selected from the entire vocabulary. Thus, the number of latent words equals the number of observed words.

Bayesian modeling of LWLM produces the generative probability of observed word sequence $\boldsymbol{w} = w_1, \cdots, w_K$ as:

$$P(\boldsymbol{w}) = \int_{\boldsymbol{\theta}} \prod_{k=1}^{K} \sum_{h_k} P(w_k|h_k, \boldsymbol{\theta}) P(h_k|\boldsymbol{l}_k, \boldsymbol{\theta}) P(\boldsymbol{\theta}) d\boldsymbol{\theta}, \quad (1)$$

where $\boldsymbol{\theta}$ indicates a model parameter of the LWLM, $\boldsymbol{h} = h_1, \cdots, h_K$ denotes a latent word sequence and $\boldsymbol{l}_k$ denotes context latent words $h_{k-n+1}, \cdots, h_{k-1}$. $P(h_k|\boldsymbol{l}_k, \boldsymbol{\theta})$ represents the transition probability which can be expressed by an n-gram model for latent words, and $P(w_k|h_k, \boldsymbol{\theta})$ represents the emission probability that models the dependency between the observed word and the latent word. More details are shown in previous works (Deschacht et al., 2012; Masumura et al., 2013a; Masumura et al., 2013b).

3 Hierarchical LWLMs

3.1 Definition

This paper introduces h-LWLM. The proposed model has multiple latent word spaces in a hierarchical structure. Thus, it assumes that there is a latent word behind a latent word. The proposed model can be regarded as a generalized form of the standard LWLM. Thus, standard LWLMs correspond to h-LWLMs with just one layer. The latent words in all layers are represented as a specific word that is selected from the entire vocabulary.

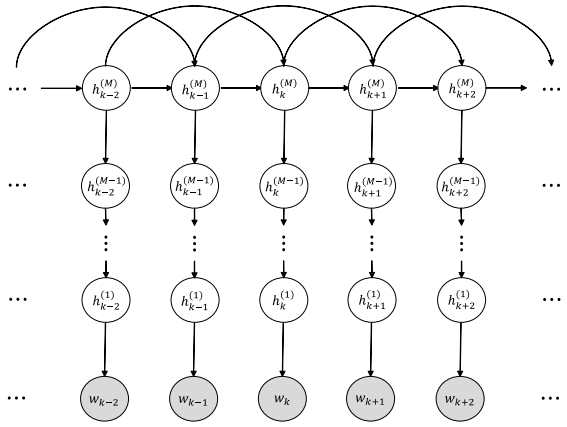


Figure 1: Graphical representation of h-LWLM.

A graphic rendering of h-LWLM is shown in Figure 1. In a generative process of the h-LWLM, a latent word in the highest layer is first generated depending on its context latent words. Next, a latent word in a lower layer is recursively generated depending on the latent word in the upper layer. Finally, an observed word is generated depending on the latent word in the lowest layer.

Bayesian modeling of h-LWLM produces the following generative probability:

$$P(\boldsymbol{w}) = \int_{\boldsymbol{\Theta}} \prod_{k=1}^{K} \sum_{h_k^{(1)}} \cdots \sum_{h_k^{(M)}} P(w_k|h_k^{(1)}, \boldsymbol{\Theta}) \cdots P(h_k^{(M-1)}|h_k^{(M)}, \boldsymbol{\Theta}) P(h_k^{(M)}|\boldsymbol{l}_k^{(M)}, \boldsymbol{\Theta}) P(\boldsymbol{\Theta}) d\boldsymbol{\Theta}, \quad (2)$$

where M is the number of layers and $\boldsymbol{\Theta}$ indicates a model parameter of h-LWLM. $\boldsymbol{h}^{(m)} = h_1^{(m)}, \cdots, h_K^{(m)}$ denotes a latent word sequence in the m-th layer. $P(h_k^{(M)}|\boldsymbol{l}_k^{(M)}, \boldsymbol{\Theta})$ represents the transition probability which is expressed by n-gram model for latent words in the highest layer. $P(h_k^{(m)}|h_k^{(m+1)}, \boldsymbol{\Theta})$ and $P(w_k|h_k^{(1)}, \boldsymbol{\Theta})$ represent the emission probabilities that respectively model the dependency between latent words in two layers and the dependency between the observed word and the latent word in the lowest layer.

As the integral with respect to $\boldsymbol{\Theta}$ is analytically

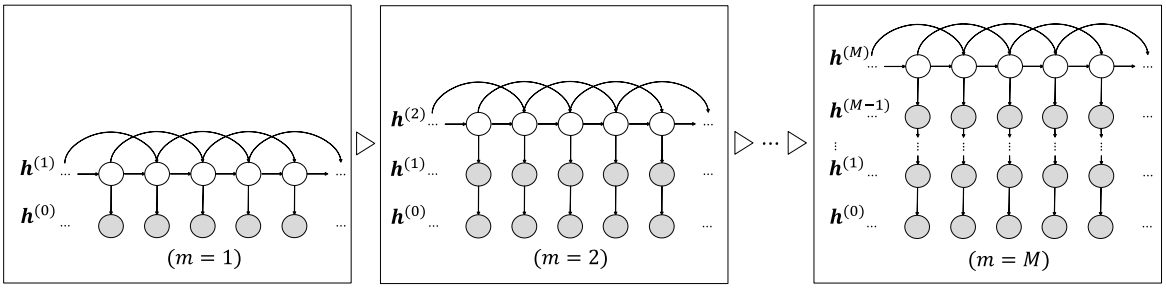


Figure 2: Layer-wise inference procedure.

Algorithm 1 :
Inference procedure for h-LWLM.

Input: Training data $\boldsymbol{w}$, number of instances T, number of layers M
Output: Model parameters $\boldsymbol{\Theta}_1, \cdots, \boldsymbol{\Theta}_T$

1: **for** $t = 1$ to T **do**
2: $\quad \boldsymbol{h}^{(0)} = \boldsymbol{w}$
3: $\quad$ **for** $m = 1$ to M **do**
4: $\quad\quad \boldsymbol{\theta}^{(m)}, \boldsymbol{h}^{(m)} \sim P(\boldsymbol{h}^{(m)} | \boldsymbol{h}^{(m-1)}, \boldsymbol{\theta}^{(m)})$
5: $\quad$ **end for**
6: $\quad \boldsymbol{\Theta}_t = \boldsymbol{\theta}^{(1)}, \cdots, \boldsymbol{\theta}^{(M)}$
7: **end for**
8: **return** $\boldsymbol{\Theta}_1, \cdots, \boldsymbol{\Theta}_T$

intractable, the equation can be approximated as:

$$P(\boldsymbol{w}) = \frac{1}{T} \prod_{k=1}^{K} \sum_{t=1}^{T} \sum_{h_k^{(1)}} \cdots \sum_{h_k^{(M)}} P(w_k | h_k^{(1)}, \boldsymbol{\Theta}_t) \cdots P(h_k^{(M-1)} | h_k^{(M)}, \boldsymbol{\Theta}_t) P(h_k^{(M)} | \boldsymbol{l}_k^{(M)}, \boldsymbol{\Theta}_t). \quad (3)$$

The probability distribution can be approximated using T instances of point estimated parameter; $\boldsymbol{\Theta}_t$ indicates the t-th point estimated parameter.

3.2 Parameter Inference

This paper proposes a layer-wise inference procedure for constructing h-LWLMs from training data. The detailed procedure is shown in **Algorithm 1**, and Figure 2 shows an image representation of the procedure as increased with the number of layers. In the procedure, LWLM structure is recursively accumulated by estimating a latent word sequence in an upper layer from a latent word sequence in the lower layer.

Line 4 in **Algorithm 1** denotes the key procedure of estimating a latent word sequence in an upper layer from a latent word sequence in the lower layer. $\boldsymbol{\theta}^{(m)}$ denotes model parameter of LWLM structure in m-th layer; it can be defined from both $\boldsymbol{h}^{(m)}$ and $\boldsymbol{h}^{(m-1)}$. For the inference of $\boldsymbol{h}^{(m)}$ from $\boldsymbol{h}^{(m-1)}$, Gibbs sampling is suitable (Casella and George, 1992; Robert et al., 1993; Scott, 2002). Gibbs sampling picks a new value for $h_k^{(m)}$ according to its probability distribution which is estimated from both $\boldsymbol{h}_{-k}^{(m)}$ and $\boldsymbol{h}^{(m-1)}$. $\boldsymbol{h}_{-k}^{(m)}$ represents all latent words in the m-th layer except for $h_k^{(m)}$. The probability distribution is given by:

$$P(h_k^{(m)} | \boldsymbol{h}_{-k}^{(m)}, \boldsymbol{h}^{(m-1)}, \boldsymbol{\theta}^{(m)}) \propto P(h_k^{(m-1)} | h_k^{(m)}, \boldsymbol{\theta}^{(m)}) \prod_{j=k}^{k+n-1} P(h_j^{(m)} | \boldsymbol{l}_j^{(m)}, \boldsymbol{\theta}^{(m)}). \quad (4)$$

For the inference, the prior distribution is necessary for each probability distribution. Usually, a hierarchical Pitman-Yor prior (Teh, 2006) is used for each transition probability and a Dirichlet prior (MacKay and Peto, 1994) is used for each emission probability.

As shown in line 6, t-th point estimated parameter $\boldsymbol{\Theta}_t$ indicates parameters of each LWLM for all layers in t-th iteration. The transition probabilities except for M-th layer are only used in the layer-wise inference procedure.

3.3 Usage

It is impractical to directly apply the h-LWLM to natural language processing tasks since the proposed model has multiple latent word spaces and we have to consider all possible latent word assignment for calculating generative probabilities. Therefore, this paper introduces an n-gram approximation technique as well as that for standard LWLM (Masumura et al., 2013a).

Algorithm 2 :
Random sampling for trained h-LWLM.

Input: Model parameters $\boldsymbol{\Theta}_1, \cdots, \boldsymbol{\Theta}_T$, number of sampled words K
Output: Sampled data $\boldsymbol{w}$

1: **for** $k = 1$ to K **do**
2: $\quad \boldsymbol{\Theta}_t \sim P(\boldsymbol{\Theta}_t) = \frac{1}{T}$
3: $\quad h_k^{(M)} \sim P(h_k^{(M)} | \boldsymbol{l}_k^{(M)}, \boldsymbol{\Theta}_t)$
4: $\quad$ **for** $m = M - 1$ to 1 **do**
5: $\quad\quad h_k^{(m)} \sim P(h_k^{(m)} | h_k^{(m+1)}, \boldsymbol{\Theta}_t)$
6: $\quad$ **end for**
7: $\quad w_k \sim P(w_k | h_k^{(1)}, \boldsymbol{\Theta}_t)$
8: **end for**
9: **return** $\boldsymbol{w} = w_1, \cdots, w_K$

The n-gram approximation is conducted as following steps. First, a lot of text data that permit h-LWLMs to be approximated by n-gram structure is generated by random sampling using trained h-LWLM. Next, an n-gram model is constructed from the generated data. The random sampling is based on **Algorithm 2**. The sampled data $\boldsymbol{w}$ in line 9 is only used for n-gram model estimation.

4 Experiments

4.1 Experimental Conditions

Our basic assumption is domain-matched training data is not available. Thus, for LM training, we used the Corpus of Spontaneous Japanese (CSJ) whose domain is a spontaneous lecture task (Maekawa et al., 2000). We divided CSJ into a training set and a small validation set (Valid). The validation set was used for optimizing several hyper parameters of LMs. For evaluation, a contact center dialogue task (Test 1) and a voice mail task (Test 2) were prepared. In contact center dialogue task, two speakers, an operator and a customer, talked to each other as in call center dialogues. 24 phone calls (24 operator channels and 24 customer channels) were used in the evaluation. In the voice mail task, a person spoke small voice messages using a smart phone. 237 messages are used in the evaluation. The training data had about 7M words, the vocabulary size was about 80K. The validation data size and test data size (both tasks) were about 20K words.

For speech recognition evaluation, we prepared an acoustic model based on hidden Markov models with deep neural networks (DNN-HMM) (Hinton et al., 2012). The DNN-HMM had 8 hidden layers with 2048 nodes. The speech recognizer was a weighted finite state transducer (WFST) decoder (Mohri et al., 2001; Hori et al., 2007).

As a baseline, 3-gram LM with interpolated Kneser-Ney smoothing (`MKN`) (Kneser and Ney, 1995) and 3-gram hierarchical Pitman-Yor LM (`HPY`) (Huang and Yor, 2007) were constructed from the training data. We also trained a class-based recurrent neural network LM with 500 hidden nodes and 500 classes (`RNN`) for comparison to state-of-the art language modeling (Mikolov et al., 2011). In addition, we constructed 3-gram standard LWLM and 3-gram h-LWLMs (`LW`). `LW` with 1 layer represents standard LWLM, and `LW` with 2-5 layers represent proposed h-LWLMs. The number of instances was set to 10 for each `LW`. For their n-gram approximation, we generated one billion words and approximated each as a 3-gram HPYLM. Moreover, we constructed interpolated model with `LW` and `HPY` (`LW+HPY`).

4.2 Results

Figure 3 shows the relation between number of layers in h-LWLM and perplexity (PPL) reduction for each condition. In addition, Table 1 shows speech recognition results in terms of word error rate (WER) for each condition. `RNN` was only tested in PPL evaluation as `RNN` cannot be converted into WFST format.

For the validation set (same domain as that of training set), PPL was not improved by the hierarchical structure in `LW`. `LW` is comparable to `MKN` and `HPY`, and inferior to `RNN` in terms of PPL. On the other hand, in test sets (out-of domain tasks), PPL improved with the increase in the number of layers in `LW`. `LW` with 5 layers was superior to 1 layer in terms of PPL and WER. The best results were obtained by `LW+HPY` with 5 layers. In fact, when we generated one billion words using a trained LWLM or trained h-LWLM, the number of observed trigrams in h-LWLM with 5 layers was 101M while the number of observed trigrams in non-hierarchical LWLM was 94M. Thus, h-LWLM can generate unseen words unlike non-hierarchical LWLM. Moreover, trigram coverage in each test data slightly increased with number of layers. These results show that h-LWLM with multiple layers offers robust performance not possible with other models while its performance in the same domain as that of training data was not improved. As a result, `LW+HPY` with 5 layers

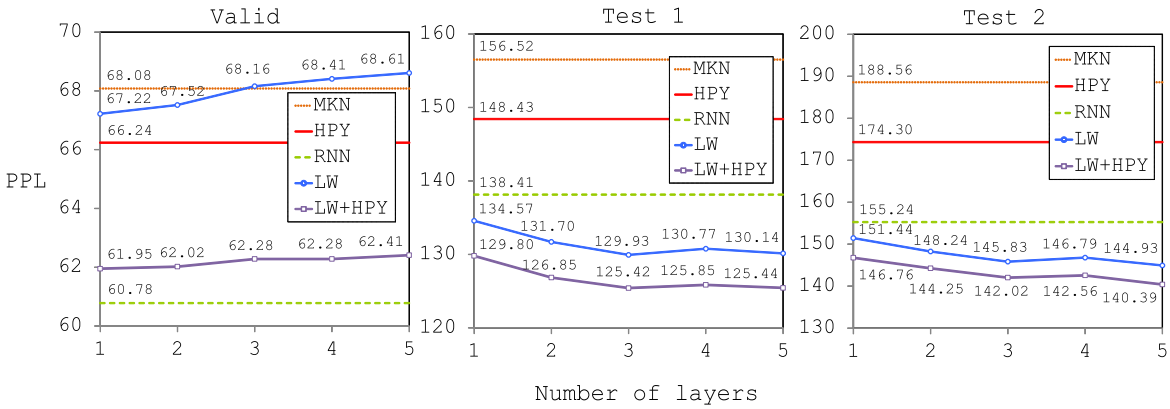


Figure 3: Perplexity (PPL) results.

Setup	Layer	Valid	Test 1	Test 2
MKN	-	24.79	38.67	32.31
HPY	-	24.67	38.29	32.00
LW	1	24.54	36.93	30.26
LW	5	24.60	36.49	29.57
LW+HPY	1	**23.62**	36.49	29.76
LW+HPY	5	23.68	**36.03**	**29.21**

Table 1: Word error rate (WER) results (%).

performed significantly better than MKN, HPY and RNN in the out-of domain tasks.

5 Conclusions

This paper proposed h-LWLM for robust modeling and detailed its definition, inference procedure, and approximation method. The proposed model has a hierarchical latent word space and it can flexibly handle linguistic phenomena not present in the training data. Our experiments showed that h-LWLM offers improved robustness to out-of domain tasks; h-LWLM is also superior to standard LWLM in terms of PPL and WER. Furthermore, our approach is significantly superior to the conventional n-gram models or the recurrent neural network LM in out-of domain tasks.

References

Yoshua Bengio, Rejean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155.

Thorsten Brants, AShok C. Popat, Peng Xu, Ftanz J. Och, and Jeffrey Dean. 2007. Large language models in machine translation. *In Proc. Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 858–867.

Peter F. Brown, Peter V. deSouza, Robert L. Mercer, Vincent J. Della Pietra, and Jenifer C. Lai. 1992. Class-based n-gram models of natural language. *Computational Linguistics*, 18:467–479.

George Casella and Edward I George. 1992. Explaining the Gibbs sampler. *The American Statistician*, 46:167–174.

Stanley F. Chen and Joshua Goodman. 1999. An empirical study of smoothing techniques for language modeling. *Computer Speech & Language*, 13:359–383.

Koen Deschacht, Jan De Belder, and Marie-Francine Moens. 2012. The latent words language model. *Computer Speech & Language*, 26:384–409.

Ahmad Emami and Frederick Jelinek. 2005. Random clusterings for language modeling. *In Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 1:581–584.

Joshua T. Goodman. 2001. A bit of progress in language modeling. *Computer Speech & Language*, 15:403–434.

Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh. 2006. A fast learning algorithm for deep bilief nets. *Neural Computation*, 18:1527–1554.

Geoffrey Hinton, Li Deng, Dong Yu, George Dahl, Abdel rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara Sainath, and Brian Kingsbury. 2012. Deep neural networks for acoustic modeling in speech recognition. *Signal Processing Magazine*, pages 1–27.

Takaaki Hori, Chiori Hori, Yasuhiro Minami, and Atsushi Nakamura. 2007. Efficient WFST-based one-pass decoding with on-the-fly hypothesis rescoring in extremely large vocabulary continuous speech recognition. *IEEE Transactions on Audio, Speech and Language Processing*, 15(4):1352–1365.

Songfang Huang and Marc Yor. 2007. Hierarchical Pitman-Yor language models for ASR in meetings. *In Proc IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 124–129.

Reinhard Kneser and Hermann Ney. 1995. Improved backing-off for m-gram language modeling. *In Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 1:181–184.

David J. C. MacKay and Linda C. Peto. 1994. A hierarchical Dirichlet language model. *Natural language engineering*, 1:289–308.

Kikuo Maekawa, Hanae Koiso, Sadaoki Furui, and Hitoshi Isahara. 2000. Spontaneous speech corpus of Japanese. *In Proc. International Conference on Language Resources and Evaluation (LREC)*, pages 947–952.

Ryo Masumura, Hirokazu Masataki, Takanobu Oba, Osamu Yoshioka, and Satoshi Takahashi. 2013a. Use of latent words language models in ASR: a sampling-based implementation. *In Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8445–8449.

Ryo Masumura, Takanobu Oba, Hirokazu Masataki, Osamu Yoshioka, and Satoshi Takahashi. 2013b. Viterbi decoding for latent words language models using Gibbs sampling. *In Proc. Annual Conference of the International Speech Communication Association (INTERSPEECH)*, pages 3429–3433.

Tomas Mikolov, Martin Karafiat, Lukas Burget, Jan Cernocky, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. *In Proc. Annual Conference of the International Speech Communication Association (INTERSPEECH)*, pages 1045–1048.

Tomas Mikolov, Stefan Kombrink Stefan, Lukas Burget, Jan Cernocky, and Sanjeev Khudanpur. 2011. Extensions of recurrent neural network language model. *In Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5528–5531.

Mehryar Mohri, Fernando Pereira, and Michael Riley. 2001. Weighted finite-state transducers in speech recognition. *Computer Speech & Language*, 16:69–88.

Christian P. Robert, Gilles Celeux, and Jean Diebolt. 1993. Bayesian estimation of hidden Markov chains: A stochastic implementation. *Statistics & Probability Letters*, 16:77–83.

Ronald Rosenfeld. 2000. Two decades of statistical language modeling: Where do we go from here? *In Proc. IEEE*, 88:1270–1278.

Ruslan Salakhutdinov and Geoffrey Hinton. 2009. Deep Boltzmann machines. *In Proc. the International Conference on Artificial Intelligence and Statistics*, 5:448–455.

Steven L. Scott. 2002. Bayesian methods for hidden Markov models: Recursive computing in the 21st century. *Journal of the American Statistical Association*, 97:337–351.

Yee Whye Teh. 2006. A hierarchical bayesian language model based on Pitman-Yor processes. *In Proc. Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 985–992.

Peng Xu and Frederick Jelinek. 2004. Random forests in language modeling. *In Proc. Empirical Methods on Natural Language Processing (EMNLP)*, pages 325–332.

A Coarse-Grained Model for Optimal Coupling of ASR and SMT Systems for Speech Translation

Gaurav Kumar [1], **Graeme Blackwood** [2], **Jan Trmal** [1], **Daniel Povey** [1], **Sanjeev Khudanpur** [1]
[1] CLSP & HLTCOE, Johns Hopkins University, Baltimore, MD, USA
[2]IBM T. J. Watson Research Center, Yorktown Heights, NY, USA
{gkumar6, dpovey1, khudanpur}@jhu.edu, blackwood@us.ibm.com

Abstract

Speech translation is conventionally carried out by cascading an automatic speech recognition (ASR) and a statistical machine translation (SMT) system. The hypotheses chosen for translation are based on the ASR system's acoustic and language model scores, and typically optimized for word error rate, ignoring the intended downstream use: automatic translation. In this paper, we present a *coarse-to-fine* model that uses features from the ASR and SMT systems to optimize this coupling. We demonstrate that several standard features utilized by ASR and SMT systems can be used in such a model at the speech-translation interface, and we provide empirical results on the Fisher Spanish-English speech translation corpus.

1 Introduction

Speech translation is the process of translating speech in the source language to text or speech in the target language. This process is typically structured as a three step pipeline. Step one involves training an Automatic Speech Recognition (ASR) system to transcribe speech to text in the source language. Step two involves extracting an appropriate form of the ASR output to translate. We will refer to this step as the Speech-Translation interface. In the simplest scenario, the ASR 1-best output can be used as the source text to translate. It may be useful to consider alternative ASR hypotheses and these take the form of an N-best list or a word-lattice. An N-best list can be included easily into the tuning and the decoding process of a statistical machine translation (SMT) system (Zhang et al., 2004). Several researchers have proposed solutions to incorporating lattices and confusion networks in this process (Saleem et al., 2004; Matusov et al., 2005; Bangalore and Riccardi, 2000; Dyer et al., 2008a; Bertoldi and Federico, 2005; Quan et al., 2005; Mathias and Byrne, 2006; Bertoldi et al., 2007). Word lattice input to SMT for tuning and decoding increases the complexity of the decoding process because of the exponential number of alternatives that are present. Finally, step three involves training and tuning a Statistical Machine Translation (SMT) system and decoding the output extracted through the speech translation interface.

This paper presents a featurized model which performs the job of hypothesis selection from the outputs of the ASR system for the input to the SMT system. Our motivation is as follows:

1. **Using downstream information** : Hypothesis selection for the input to the SMT system should be done jointly by the ASR and the SMT systems. That is, there may exist hypotheses that a trained SMT system may find easier to translate and produce better translations for than the ones that are deemed best based on the ASR acoustic and language model scores. Incorporation of knowledge from the downstream process (translation) is vital to selecting translation options, and subsequently producing better translations.

2. **Coarse-to-fine grained decoding** : An intermediate model which acts as an interface and is a weak (coarse) version of the downstream process may be able to select better hypotheses. In effect, a weak translation decoder can be used as the interface to estimate the expected translation quality of an ASR hypothesis. This method of hypothesis selection should be able to incorporate features from the ASR and the SMT system.

3. **Phrase units vs. word units** : When a phrase based SMT system is used for translation,

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1902–1907,
Lisbon, Portugal, 17-21 September 2015.

optimization for hypothesis selection at the Speech-Translation interface should be conducted using phrases as the basic unit instead of words.

2 Coarse-to-Fine Speech Translation

In this section, we describe the featurized model (coarse-grain MT decoder) for hypothesis selection that uses information from the ASR and SMT systems (impedance matching). We assume the presence of ASR and SMT systems that have been trained separately. In addition to creating almost no disruption in the traditional pipeline approach, this allows us to incorporate local gains from each system. To elaborate, our methods avoid joint optimization of the ASR and the SMT system with respect to a translation metric (Vidal, 1997; Ney, 1999), which is not feasible for larger datasets. Also, considering the dearth of speech translation training datasets, this method allows independent training of the ASR and SMT systems on data created only for ASR training and parallel data for SMT. We start by introducing the formal machinery that will be used and by presenting a simple example to motivate the model. The complete featurized model follows this exposition.

Let Σ and Γ be alphabets of words and phrases respectively in the source language. Using these, we can define the following finite state machines:

1. **Word Lattice** (L) : A finite state acceptor that accepts word-sequences in the source language ($L : \Sigma^* \rightarrow \Sigma^*$). This represents the unpruned ASR word lattice output in our model (Figure 1a).

2. **Phrase segmentation Transducer** (S) : A cyclic finite state transducer that transduces a sequence of words to phrases in the source language ($S : \Sigma^* \rightarrow \Gamma^*$). This is built from the source side of the phrase table. Each path represents one source side phrase in the phrase table. Traversing a path is equivalent to consuming the words in a phrase and producing the phrase as a token (Figure 1b).

3. **Weighted word lattice** ($\tilde{L}_{ASR}$) : A weighted version of L ($\tilde{L}_{ASR} : \Sigma^* \rightarrow \Sigma^*/\mathbb{R}^+$). We use the subscript to denote the nature/source of the weights.

4. **Phrase acceptor** ($\tilde{W}_{MT}$) : A finite state acceptor that accepts source phrases in the SMT system's phrase table ($\tilde{W}_{MT} : \Gamma \rightarrow \Gamma/\mathbb{R}$). It is weighted by features derived from the SMT system.

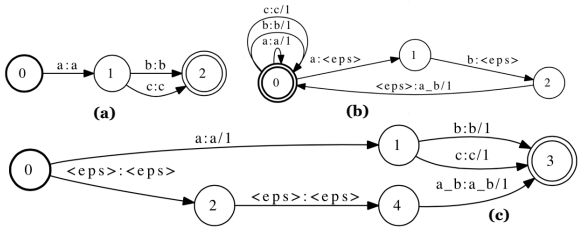


Figure 1: *A toy example for producing a phrase length weighted phrase lattice. (a) An unweighted word lattice. (b) A phrase segmentation transducer which transduces words to phrases and has a weight of one per path. Each path is a source phrase in the phrase table. (c) A phrase lattice produced by composing the word lattice and phrase segmentation transducer.*

5. **Phrase lattice** (P) : The result of the composition of a word lattice (acceptor) with the phrase segmentation transducer ($P : \Sigma^* \rightarrow \Gamma^*$). This represents all possible phrase segmentations of all the ASR hypotheses in the word lattice.

$$P = \det(\min(L \circ S))$$

We will represent weighted versions of P as $\tilde{P}_{ASR/MT}$ with subscripts to denote the origin of the weights (Figure 1c).

2.1 A simple model : Maximum Spanning Phrases

We motivate our model with this fairly simple scenario. Suppose that we believe that if our SMT input could be covered by longer source side phrases[1], we would produce better translations. This may be viewed as a tiling problem where the tiles are the source phrases in the phrase table and the goal is to select the ASR hypothesis that requires the least number of phrases to cover[2]. To achieve this using our existing machinery, we create $\tilde{S}$, a weighted version of S (Figure 1 (b)), such

[1] In phrase based translation, target translations are produced for each possible span of the input sentence allowed by the phrase table. Translation of a longer source side phrase produces fewer translation options and may be more reliable given sufficient occurrences in the training data.

[2] It may be useful to incorporate a brevity penalty here, since this approach has a strong bias towards selecting shorter hypotheses. We will use other features to counter this bias in the following sections.

that

$$w(\delta^{(\tilde{S})}) = \begin{cases} 0 & : \pi_1(\delta^{(S)}) \in \Sigma \text{ and } \pi_2(\delta^{(S)}) = \epsilon \\ 1 & : \pi_2(\delta^{(S)}) \in \Gamma \text{ and } \pi_1(\delta^{(S)}) = \epsilon \end{cases}$$

where $\delta^{(\tilde{S})}$ is an edge in $\tilde{S}$ and π_1 and π_2 are the input and output projections respectively. Using this segmentation transducer and an unweighted word lattice, L (Figure : 1 (a)), we produce a phrase lattice $\tilde{P}_{MT}$. Assuming the weights are in the log-semiring, the weight of a path $\delta^{(\tilde{P})*}$ in $\tilde{P}_{MT}$ is

$$w(\delta^{(\tilde{P})*}) = \sum_{\delta^{(\tilde{P})} \in \delta^{(\tilde{P})*}} w(\delta^{(\tilde{P})})$$

Figure 1(c) shows an example of this phrase lattice. Weights in the phrase lattice follow the same definition as the weights in the segmentation transducer. Hence, the weight of a path in the phrase lattice is simply the number of phrases used to cover this path. The shortest path [3] in the phrase lattice $\tilde{P}_{MT}$, corresponds to the hypothesis we were looking for. This simple example, demonstrates how we may be able to use SMT features (source phrase length in this case) to select hypotheses from the phrase lattice.

2.2 A general featurized model for hypothesis selection

We now present a general framework in which hypothesis selection can be carried out using knowledge (features) from the ASR and the SMT system. As described earlier, this form of 'impedance' matching allows us to select hypotheses from an unpruned ASR word lattice for which the SMT system is more likely to find good translations. Incorporating ASR weights also ensures that we take into account what the ASR system considers to be good hypotheses. We start with the previously discussed idea of a phrase lattice, using weights from the ASR system only. That is,

$$\tilde{P}_{ASR} = \det(\min(\tilde{L}_{ASR} \circ S))$$

Now, we use the weighted phrase acceptor $\tilde{W}_{MT}$ to bring in the SMT features [4]. Composing this with the weighted phrase lattice, we get

$$\tilde{P}_{ASR,MT} = \det(\min(\tilde{P}_{ASR} \circ (\tilde{W}_{MT})^*)$$

where $(\tilde{W}_{MT})^*$ is the Kleene closure of $(\tilde{W}_{MT})$. We assume that the edge weights are in the log-semiring. Hence, after these two compositions, the edge weights in $\tilde{P}_{ASR,MT}$ can be represented as

$$\begin{aligned} w(\delta^{(\tilde{P}_{ASR,MT})}) &= \sum_j \beta_j f_{j,ASR} + \sum_k \gamma_k f_{k,MT} \\ &= \sum_i \lambda_i f_i \end{aligned}$$

where $\delta^{(\tilde{P}_{ASR,MT})}$ is an edge in $\tilde{P}_{ASR,MT}$, β, γ are feature weights, f_{ASR} and f_{MT} are features from the ASR and SMT system respectively. This form represents a log-linear model (our features are already assumed to be in log-space). where f_i is any feature and λ_i is the corresponding feature weight. We may now extract the one-best, N-best or lattice input for the SMT system from $\tilde{P}_{ASR,MT}$.

2.2.1 A discussion about related techniques

1. Decoding (Translation) : Our model closely resembles a featurized finite-state transducer based translation model. If we replace the output alphabet of the acceptor $(\tilde{W}_{MT})^*$ with the target side phrases, we will actually get output in the target language. Even though this model does not explicity include reordering, the coarse-grained decoder has access to information that can enable better decisions about which hypotheses are better for the downstream process (translation).

2. Lattice Decoding : (Dyer et al., 2008b) suggests passing the entire word lattice to the SMT system. However, even if these lattices are not pruned, a beam based decoder might not consider hypotheses that our model may produce through coarse-grained decoding.

3. Language model re-scoring : One may use a bigger source language model to re-score the ASR lattice (or an N-best list). This however, does not consider any SMT features in re-scoring. With our model, we can simply use this as an additional feature.

2.2.2 Training

Training the hypothesis selection model can be carried out using standard methods for log linear models on a held-out set. This also requires decoding (translation) of a deep N-best list derived from the held-out set. The objective of training then simply becomes maximization of the translation

[3] To compute the shortest path, we switch from the log to the tropical semiring (A semiring with ordinary addition as the multiplication operator and $\max$ as the addition operator).

[4] Alternatively, we may have introduced the weights in the segmentation transducer itself. This separate machine is introduced for efficient training of this model.

quality given any metric that provides sentence level scores. Each time our model produces a hypothesis, its score can be looked up from the pre-translated N-best list. Also, whenever the weights are updated, the only structures that need to be rebuilt are $\tilde{W}^*_{MT}$ and $\tilde{P}_{ASR,MT}$ [5].

2.2.3 Features

We use the following features in our implementation of this model. However, any relevant ASR and SMT feature may be readily added to this model.

1. **ASR scores** : We incorporate the ASR acoustic (AM) and language (LM) model scores as one combined feature.

   $$f_{ASR} = LM + \alpha * AM$$

   Here, LM, AM are negative log-probabilities and α is the acoustic scaling parameter chosen to minimize ASR word error rate.

2. **Source phrase count** : As described in section 2.1, this feature may be used to capture the intuition that using a fewer number of phrases to cover the input sentence may produce better translations.

3. **Length normalized phrase unigram probability** : We may use a phrase LM feature by incorporating phrase n-gram probabilities (normalized) by length.

   $$f_{\text{uni}}(f_j) = \left[\frac{\text{freq}(f_j)}{\sum_k \text{freq}(f_k)} \right]^{\text{len}(f_j)}$$

   where f_j is a source side phrase in the phrase table.

4. **Phrase translation entropy** : For each source side phrase p_j, we may have multiple translations (e_i) in the phrase table with different translation probabilities ($p(e_i|f_j)$). A simple entropy measure can be used as a feature to estimate the confidence that the SMT system has in translating f_j.

   $$\begin{aligned} f_{\text{tr}}(p_j) &= H_{\text{tr}}(E|p_j) \\ &= -\sum_i p_{\text{tr}}(e_i|f_j) \log p_{\text{tr}}(e_i|f_j) \end{aligned}$$

5. **Lexical translation entropy** : Similarly, we can use an entropy measure based on the lexical translation probability as a feature.

   $$\begin{aligned} f_{\text{lex}}(p_j) &= H_{\text{lex}}(E|p_j) \\ &= -\sum_i p_{\text{lex}}(e_i|f_j) \log p_{\text{lex}}(e_i|f_j) \end{aligned}$$

3 Results

We use the Fisher and Callhome Spanish-English Speech Translation Corpus (Post et al., 2013) for our experiments. This Fisher dataset consists of 819 transcribed and translated telephone conversations. The corpus is split into a training, dev and two test sets (dev-2 and test). We use the dev set for training the feature weights of the proposed model.

We use the Kaldi speech recognition tools (Povey et al., 2011) to build our Spanish ASR systems. Our state-of-the-art ASR system is the p-norm DNN system of (Zhang et al., 2014). The word-error-rates on the dev and test sets of the Fisher dataset (dev, dev-2, test) are 29.80%, 29.79% and 25.30% respectively.

For the SMT system, we use the phrase based translation system of Moses (Koehn et al., 2007) with sparse features. The system is trained and tuned on the train and dev partitions of the Fisher dataset respectively. The BLEU scores of the MT output for the the dev-2 and the test partitions are 65.38% and 62.91% respectively. While decoding the ASR output, we tune on the 1-best ASR output for the dev partition. With this modified system, the BLEU scores for the ASR 1-best output of the dev2 and the test partitions are 40.06% and 40.4% respectively. We use this system as the baseline for our experiments (Table 1).

We note that if we were to use the lattice oracle[6] from our ASR system as input to the SMT system, we get a BLEU score of 46.59% for the dev2 partition of the Fisher dataset. This indicates that the best gain (+BLEU) that an oracle lattice reranker could get is only 6.53%.

To tune the weights of the coarse decoder, we decode 500-best ASR outputs for the tuning set with the SMT system. This maps each ASR hypothesis to a target language translation. An OOV feature was added to handle words that were not seen by the SMT system. The tuning process was then carried out so as to maximize the BLEU with

[5] This requires the use of one ASR feature, addressed in the "Features" section

[6] Path in the lattice that has the least word error rate.

Experiment	BLEU (dev2)	BLEU (test)
Transcripts	65.4%	62.9%
Lattice Oracle	46.59%	46.17%
ASR 1-best	40.06%	40.4%
Coarse decoder	**40.26%**	**40.46%**

Table 1: Performance when using the coarse decoder interface compared to the the decoding the human transcripts, the ASR 1-best or the lattice oracle (the path in the ASR lattice with the least WER : not available during test time.)

respect to the reference translation of the ASR hypothesis selected by the coarse grained decoder. We used ZMERT (Zaidan, 2009) for tuning which was configured to expect a 300-best list from the decoder at every iteration using the Fisher dev set. 15 iterations of tuning were carried out for each experiment. We then use the tuned weight vector to decode the Fisher-dev2 and the Fisher-test set using our coarse grained decoder. We extract the one-best output and use it as input to the pretrained SMT system (description in the preceding section). Table 1 reports the results achieved the featurized coarse grained decoder.

4 Conclusions

We present a *coarse-to-fine* featurized model which acts as the interface between ASR and SMT systems. By utilizing information from the upstream (ASR) and the downstream (SMT) systems, this model makes more informed decisions about which hypotheses from the ASR word lattice may result in better translation results. Moreover, the model takes the form of a coarse finite state transducer based translation decoder which imitates the downstream system. This enables it to estimate translation quality even before the complete SMT system is used for decoding. Finally, the proposed model is featurized and may accept any weight from the ASR and SMT system that are deemed useful for optimizing translation quality.

The Spanish Fisher corpus is one of a few conversational speech translation datasets available, and we start with a strong baseline system. We therefore persevere with the experimental setup described above, even though the maximum (oracle) improvement by any rescoring method is only 6.5% BLEU, as noted above. This partially explains the small gains reported here, and suggests that this method should be evaluated further on another corpus, e.g. the Egyptian Arabic translation dataset, with greater headroom for improvement.

Acknowledgments

This work was partially supported by NSF award No IIS 0963898 and DARPA contracts No HR0011-12-C-0015 and HR0011-51-6285. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of NSF, DARPA or the U.S. Government.

References

Srinivas Bangalore and Giuseppe Riccardi. 2000. Finite-state models for lexical reordering in spoken language translation. In *Proceedings of the Sixth International Conference on Spoken Language Processing*, pages 422–425.

N. Bertoldi and Marcello Federico. 2005. A new decoder for spoken language translation based on confusion networks. In *Proceedings of the IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 86–91.

N. Bertoldi, R. Zens, and Marcello Federico. 2007. Speech translation by confusion network decoding. In *Proceedings of the IEEE International Conference in Acoustics, Speech and Signal Processing*, volume IV, pages 1297–1300.

Christopher Dyer, Smaranda Muresan, and Philip Resnik. 2008a. Generalizing word lattice translation. In *Proceedings of ACL-08: HLT*, pages 1012–1020, Columbus, Ohio, June. Association for Computational Linguistics.

Christopher Dyer, Smaranda Muresan, and Philip Resnik. 2008b. Generalizing word lattice translation. 2008/02//.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, ACL '07, pages 177–180, Stroudsburg, PA, USA. Association for Computational Linguistics.

L. Mathias and W. Byrne. 2006. Statistical phrase-based speech translation. In *Proceedings of the IEEE International Conference in Acoustics, Speech and Signal Processing*, volume I, pages 561–564.

Evgeny Matusov, Stephan Kanthak, and Hermann Ney. 2005. On the integration of speech recognition and statistical machine translation. In *Proceedings of the 9th European Conference on Speech Communication and Technology*, pages 3177–3180.

Hermann Ney. 1999. Speech translation: coupling of recognition and translation. In *Proceedings of the IEEE International Conference in Acoustics, Speech and Signal Processing*, volume 1, pages 517–520, March.

Matt Post, Gaurav Kumar, Adam Lopez, Damianos Karakos, Chris Callison-Burch, and Sanjeev Khudanpur. 2013. General lattice decoding for improved speech-to-text translation with the Fisher and Callhome Spanish-English speech translation corpus. *Proceedings of the International Workshop on Spoken Language Translation*.

Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, Jan Silovsky, Georg Stemmer, and Karel Vesely. 2011. The Kaldi speech recognition toolkit. In *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*, December.

Vu H Quan, Marcello Federico, and Mauro Cettolo. 2005. Integrated n-best re-ranking for spoken language translation. In *Proceedings of the 9th European Conference on Speech Communication and Technology*, pages 3181–3184.

Shirin Saleem, Szu-Chen Jou, Stephan Vogel, and Tanja Schultz. 2004. Using word lattice information for a tighter coupling in speech translation systems. In *Proceedings of the International Conference on Spoken Language Processing*, pages 41–44.

Enrique Vidal. 1997. Finite-state speech-to-speech translation. In *Proceedings of the IEEE International Conference in Acoustics, Speech and Signal Processing*, volume 1, pages 111–114, April.

Omar F. Zaidan. 2009. Z-MERT: A fully configurable open source tool for minimum error rate training of machine translation systems. *The Prague Bulletin of Mathematical Linguistics*, 91:79–88.

Ruiqiang Zhang, Genichiro Kikui, Hirofumi Yamamoto, Taro Watanabe, Frank Soong, and Wai Kit Lo. 2004. A unified approach in speech-to-speech translation: integrating features of speech recognition and machine translation. In *Proceedings of the 20th international conference on Computational Linguistics*.

Xiaohui Zhang, Jan Trmal, Daniel Povey, and Sanjeev Khudanpur. 2014. Improving deep neural network acoustic models using generalized maxout networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2014, Florence, Italy, May 4-9, 2014*, pages 215–219.

Abstractive Multi-document Summarization with Semantic Information Extraction

Wei Li
Key Lab of Intelligent Info. Processing,
Institute of Computing Technology, CAS
Beijing, 100190, China
weili@kg.ict.ac.cn

Abstract

This paper proposes a novel approach to generate abstractive summary for multiple documents by extracting semantic information from texts. The concept of *Basic Semantic Unit* (*BSU*) is defined to describe the semantics of an event or action. A semantic link network on *BSUs* is constructed to capture the semantic information of texts. Summary structure is planned with sentences generated based on the semantic link network. Experiments demonstrate that the approach is effective in generating informative, coherent and compact summary.

1 Introduction

Most automatic summarization approaches are extractive which leverage only literal or syntactic information in documents. Sentences are extracted from the original documents directly by ranking or scoring and only little post-editing is made (Yih et al., 2007; Wan et al., 2007; Wang et al., 2008; Wan and Xiao, 2009). Pure extraction has intrinsic limits compared to abstraction (Carenini and Cheung, 2008).

Abstractive summarization requires semantic analysis and abstract representation of texts, which need knowledge on and beyond the texts (Zhuge, 2015a). There are some abstractive approaches in recent years: sentence compression (Knight and Marcu, 2000; Knight and Marcu, 2002; Cohn and Lapata, 2009), sentence fusion (Barzilay and McKeown, 2005; Filippova and Strube, 2008), and sentence revision (Tanaka et al., 2009). However, these approaches are sentence rewriting techniques based on syntactical analysis without semantic analysis and abstract representation.

Fully abstractive summarization approach requires a separate process for the analysis of texts that serves as an intermediate step before the generation of sentences (Genest and Lapalme, 2011). Statistics of words or phrases and syntactical analysis that have been widely used in existing summarization approaches are all shallow processing of text. It is necessary to explore summarization methods based on deeper semantic analysis.

We define the concept of *Basic Semantic Unit* (*BSU*) to express the semantics of texts. A *BSU* is an action indicator with its obligatory arguments which contain actor and receiver of the action. *BSU* is the most basic element of coherent information in texts, which can describe the semantics of an event or action. The semantic information of texts is represented by extracting *BSUs* and constructing *BSU* semantic link network (Zhuge, 2009). Semantic Link Network consists of semantic nodes, semantic links and reasoning rules (Zhuge, 2010; 2011; 2012; 2015b). The semantic nodes can be any resources. In this work, the semantic nodes are *BSUs* extracted from texts. We use semantic relatedness between *BSUs* as semantic links. Then summary can be generated based on the semantic link network through summary structure planning.

The characteristics of our approaches are as follows:

- Each *BSU* describes the semantics of an event or action. The semantic relatedness between *BSU*s can capture the context semantic relations of texts.
- The *BSU* semantic link network is an abstract representation of texts. Reduction on the network can obtain important information of texts with no redundancy.

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1908–1913,
Lisbon, Portugal, 17-21 September 2015.

- Summary is built from sentence to sentence to a coherent body of information based on the *BSU* semantic link network by summary structure planning.

2 Related Work

There are some abstractive summarization approaches in recent years. An approach *TTG* attempts to generate abstractive summary by using text-to-text generation to generate sentence for each subject-verb-object triple (Genest and Lapalme, 2011). A system that attempts to generate abstractive summaries for spoken meetings was proposed (Wang and Cardie, 2013). It identifies relation instances that are represented by a lexical indicator with an argument constituent from texts. Then the relation instances are filled into templates which are extracted by applying multiple sequence alignment. Both of these systems need to select a subset of the large volumes of generated sentences. However, our system generates summary directly by summary structure planning. It can generate well-organized and coherent summary more effectively.

A recent work aims to generate abstractive summary based on Abstract Meaning Representation (AMR) (Liu et al., 2015). It first parses the source text into AMR graphs, and then transforms them into a summary graph and plans to generate text from it. This work only focuses on the graph-to-graph transformation. The module of text generation from AMR has not been developed. The nodes and edges of AMR graph are entities and relations between entities respectively, which are sufficiently different from the *BSUs* semantic link network. Moreover, texts can be generated efficiently from the *BSUs* network. Another recent abstractive summarization method generates new sentences by selecting and merging phrases from the input documents (Bing et al., 2015). It first extracts noun phrases and verb-object phrases from the input documents, and then calculates saliency scores for them. An ILP optimization framework is used to simultaneously select and merge informative phrases to maximize the salience of phrases and meanwhile satisfy the sentence construction constraints. As the results show that the method is difficult to generate new informative sentences really different from the original sentences and may generate some none factual sentences since phrases from different sentences are merged.

Open information extraction has been proposed by (Banko et al., 2007; Etzioni et al., 2011). They extract binary relations from the web, which is different from our approach that extracts events or actions expressed in texts.

3 The Summarization Framework

Our system produces an abstractive summary for a set of topic related documents. It consists of two major components: Information extraction and summary generation.

3.1 Information Extraction

The semantic information of texts is obtained by extracting *BSU*s and constructing *BSU* semantic link network. A *BSU* is represented as an **actor-action-receiver** triple, which can both detects the crucial content and incorporates enough syntactic information to facilitate the downstream sentence generation. Some actions may not have the receiver argument. For example, "*Flight MH370 – disappear*" and "*Flight MH370 - leave - Kuala Lumpur*" are two *BSU*s.

***BSU* Extraction.** *BSU*s are extracted from the sentences of the documents. The texts are preprocessed by name entity recognition (Finkel et al., 2005) and co-reference resolution (Lee et al., 2011). Constituent and dependency parses are obtained by Stanford parser (Klein and Manning, 2003). The eligible action indicator is restricted to be a predicate verb; the eligible actor and receiver arguments are noun phrase. Both the actor and receiver arguments take the form of constituents in the parse tree. A valid *BSU* should have one action indicator and at least one actor argument, and satisfy the following constraints:

- The actor argument is the nominal subject or external subject or the complement of a passive verb which is introduced by the preposition "by" and does the action.
- The receiver argument is the direct object or the passive nominal subject or the object of a preposition following the action verb.

We create some manual rules and syntactic constraints to identify all *BSU*s based on the syntactic structure of sentences in the input texts.

Constructing *BSU* Semantic Link Network. The semantic relatedness between *BSU*s contains three parts: *Arguments Semantic Relatedness* (ASR), *Action-Verbs Semantic Relatedness* (VSR) and *Co-occurrence in the Same Sentence* (CSS). Arguments of *BSU*s include actors and receivers, which both are noun phrases and indicate concepts or entities in the text. When computing ASR, the semantic relatedness between concepts must be measured. We use the explicit

semantic analysis based on Wikipedia to compute semantic relatedness between concepts (Gabrilovich and Markovitch, 2007). When computing VSR, WordNet-based measure is used to calculate the semantic relatedness between action verbs (Mihalcea et al., 2006). CSS is measured whether two different *BSUs* co-occur in the same sentence. Semantic relations between *BSU*s are computed by linearly combining these three parts. Then *BSU*s that are extracted from the texts form a semantic link network.

Semantic Link Network Reduction. A discriminative ranker based on Support Vector Regression (SVR) (Smola and Scholkopf, 2004) is utilized to assign each *BSU* a summary-worthy score. Training data was constructed from the DUC 2005 datasets which contain both the source documents and human generated reference summaries. *BSUs* are extracted from these datasets. For each *BSU* in the source documents, if it has occurred in the corresponding human generated summaries or the semantic relatedness between the *BSU* and one *BSU* in the corresponding human generated summaries is above a threshold δ, then it is considered to be a positive sample and be assigned 1 to its summary-worthy score. Otherwise, the *BSU* is considered to be a negative sample and be assigned 0 to its summary-worthy score. Table 1 displays the features of *BSU* used in the SVR model. Then the saliency score of each *BSU* in the semantic link network is calculated by the following equation:

$$Sal(BSU_i) = SW_i * \sum_j R_{ij} \quad (1)$$

Where SW_i is the summary-worthy score of BSU_i; R_{ij} is the semantic relatedness between BSU_i and BSU_j.

BSUs in the semantic link network are clustered by hierarchical complete-link clustering methods. *BSUs* in each cluster are semantically similar. For example, *Malaysia Airlines plane - vanish* and *Flight MH370 – disappear*. Only the most important one with the largest saliency score is reserved in the network. These less important *BSUs* are eliminated. The remaining *BSU* semantic link network represents the important information of the texts with no redundancy.

3.2 Summary Generation

The summary for the documents is generated directly based on the *BSU* semantic link network. The summary should be well-structured and well-organized. It should not just be a heap of related information, but should build from sentence to sentence to a coherent body of information about a topic.

Basic Features
Number of words in actor/receiver
Number of nouns in actor/receiver
Number of new nouns in actor/receiver
Actor/receiver has capitalized word?
Actor/receiver has stopword?
Action is a phrasal verb?
Content Features
Actor/receiver has name entity?
TF/IDF/TF-IDF of action
TF/IDF/TF-IDF min max average of actor/receiver
Syntax Features
Constituent tag of actor/action/receiver
Dependency relation of action with actor
Dependency relation of action with receiver

Table 1. Features for *BSU* summary-worthy scoring. We use SVM-light with RBF kernel by default parameters (Joachims, 1999).

The summary structure is planned based on the *BSU* semantic link network. An optimal path which covers all the nodes in the network is found. The following two factors are considered when finding the optimal path: (1) **Context Semantic Coherent.** To make the summary semantic coherent, all adjacent sentences should be semantically related. We need to find an optimal path, in which every two adjacent nodes are strong semantically related. The optimal path is denoted as $P=[p_{r_1}, p_{r_2}, ..., p_{r_n}]$ and maximize $1/n\sum_{i=1}^{n-1} R_{r_i r_{i+1}}$. (2) **Clear-cut Theme.** To make the theme of generated summary clear-cut, the important content should be put in prior position. The order of the *i*th node in the path is denoted as u_i and its weight is denoted as $w_i = 1/Sal(BSU_i)$ and maximize $\sum_{i=1}^{n} u_i w_i$.

To combine the above two factors, we need to find an optimal path which covers each node only once and has the longest distance. The biased-sum weight of all nodes in the path should be maximized. The problem can be proved to be NP-hard by reduction to TSP problem. It can be formalized as an integer linear programming (ILP) as follow. x_{ij} is defined to indicate whether the optimal path goes from node i to node j.

$$x_{ij} = \begin{cases} 1 & if\ the\ path\ goes\ from\ node\ i\ to\ node\ j \\ 0 & otherwise \end{cases} \quad (2)$$

Since each node can be traversed only once, the following constraints must be satisfied.

$$\begin{aligned} &\sum_{i=1, j\neq i}^{n} x_{ij} = 1 && j = 1,...n \\ &\sum_{j=1, j\neq i}^{n} x_{ij} = 1 && i = 1,...n \end{aligned} \quad (3)$$

The nodes in the path are sequentially ordered. If the edge between two nodes is in the path, then

System	ROUGE-1	ROUGE-2	ROUGE-SU4
OurSystem	**0.42145**	**0.11016**	**0.15632**
MultiMR	0.41967	0.10302	0.15385
RankBSU	0.39123	0.08742	0.14381
TTG	0.39268	0.09645	0.14553
AveDUC	0.39684	0.09495	0.14671
NIST Baseline	0.33126	0.06425	0.11114

Table 2. Comparison results (F-measure) on DUC 2007 under ROUGE evaluation.

System	OurSystem	MultiMR	RankBSU	TTG
Pyr (Th:0.6)	**0.858**	0.845	0.832	0.834
Pyr (Th:0.65)	**0.743**	0.731	0.718	0.721

Table 3. Comparison results on DUC 2007 under the automated pyramid evaluation with two threshold value 0.6 and 0.65.

the order of the two nodes is sequentially close to each other, which can be formulated as follow:

$$\begin{aligned} & u_i - u_j + nx_{ij} \le n-1 \quad 1 \le i \ne j \le n \\ & 1 \le u_i \le n \qquad\qquad i = 1,...n \\ & u_i \in \mathbb{Z} \qquad\qquad i = 1,...n \end{aligned} \tag{4}$$

At last, we can formulate the objective function as follow:

$$\max \ 1/n \sum_{i=1}^{n} \sum_{j=1, j \ne i}^{n} R_{ij} x_{ij} + \lambda \sum_{i=1}^{n} w_i u_i \tag{5}$$

where parameter λ tunes the effect of the two parts and n is the quantity of *BSU*s in the final *BSU* semantic link network (after reduction).

Sentence Generation. After the summary structure has been planned, sentences are generated for each node in the *BSU* semantic link network. As the *BSU* contains enough semantic and syntactic information, sentence can be generated efficiently according to the following rules:

- Generate a Noun Phrase (NP) based on the actor argument to represent the subject, a NP based on the receiver argument to represent the object if present.
- Generate a Verb Phrase (VP) based on the action verb to link the components above. The tense of the verb is set to the same as in the original sentence, and most modifiers like auxiliaries and negation are conserved.
- Generate complements for the VP when the *BSU* has no receiver. The verb modifiers following the action verb such as prepositional phrases and infinitive phrases can be used as the complement, in case that the verb would have no interesting meaning without a complement.

The process of sentence generation for each node is based on the syntactic structure of the source sentence where the *BSU* is extracted from. The time and location preposition phrases which are important information of new events are kept. The generated sentences are organized according to the summary structure. If some adjacent sentences in the summary have the same subject, the subject of the latter can be substituted by a pronoun (such as it or they) to avoid repetition of noun phrases. One sample summary generated by our system for "Malaysia MH370 Disappear" news is shown in Figure 1.

4 Evaluation Results

4.1 Dataset and Experimental Settings

In order to evaluate the performance of our system, we use two datasets that have been used in recent multi-document summarization shared tasks: DUC2005 and DUC2007. Each task has a gold standard dataset consisting of document clusters and reference summaries. In our experiments, DUC2005 was used for training and parameter tuning, and DUC2007 was used for testing. Based on the tuning set, the parameter λ is set as 10 and δ is set as 0.7 after tuning.

Our system is compared with one state-of-the-art graph-based extractive approach *MultiMR* (Wan and Xiao, 2009) and one abstractive approach *TTG* (Genest and Lapalme, 2011). In addition, we have implemented another baseline *RankBSU* which uses the graph-based ranking methods on the *BSU*s network to rank *BSU*s and select the top ranked *BSU*s to generate sentences.

4.2 Results

ROUGE-1.5.5 toolkit was used to evaluate the quality of summary on DUC 2007 dataset (Lin and Hovy, 2003). The ROUGE scores of the NIST Baseline system (i.e. *NIST Baseline*) and average ROUGE scores of all the participating systems (i.e. *AveDUC*) for DUC 2007 main task were also listed. According to the results in Table 2, our system much outperforms the *NIST Baseline* and *AveDUC*, and achieves higher ROUGE scores than the abstractive approach *TTG*. So the abstract representation of texts and the information extraction process in our system are effective for multi-document summarization. Our system also achieves better performance than the baseline *RankBSU*, which demonstrates that the network reduction method is more efficient than the popular graph-based ranking methods. As compared with the state-of-art graph-based extractive method *MultiMR*, our system also achieves better performance. Furthermore, our system is abstractive with abstract representation and sentence generation. Incorrect

parser and co-reference resolution will lead to wrong extraction of *BSU*. If with more accurate parser and co-reference resolution, our system will be expected to achieve better performance.

Since ROUGE metric evaluates summaries only from word overlapping perspective, we also use the pyramid evaluation metric (Nenkova and Passonneau, 2004) which can measure the summary quality beyond simply string matching. The pyramid evaluation metric involves semantic matching of summary content units (SCUs) so as to recognize alternate realizations of the same meaning, which is a better metric for the abstractive summary evaluation. Since the manual pyramid evaluation is time-consuming and the evaluation results can't be reproducible with different groups of assessors, we use the automated version of pyramid proposed in (Passonneau et al., 2013) and adopt the same setting as in (Bing et al., 2015). Table 3 shows the evaluation results of our system and the three baseline systems on DUC 2007. The results show that the performance of our system is significantly better than the three baseline systems, which demonstrates that the summaries of our system contain more SCUs than summaries of other systems. So our system can generate more informative summary.

In addition, large volumes of news texts for popular news events are crawled from the news websites. Figure 1 and 2 show the summaries for the "Malaysia MH370 Disappear" news event generated by our system and *MultiMR* respectively. The summary by *MultiMR* contains some repetition of facts obviously. And it is just a heap of information about MH370. The summary by our system doesn't contain much repetition of facts, so it can contain more useful information. And it is built from sentence to sentence to a coherent body. Obviously, the summary by our system is more coherent and compact.

Flight MH370 disappeared after leaving Kuala Lumpur. It had been expected to land in Beijing at 06:30. It took off at 00:41 MYT from runway 32R. It ended in the southern Indian Ocean. The aircraft, a Boeing 777-200ER made a sharp turn westwards. It passed into Vietnamese airspace. The captain of another aircraft attempted to reach the crew of Malaysia Airlines flight MH370. Malaysia Airlines flight 386 was requested to attempt to contact Malaysia Airlines flight MH370 on the Lumpur Radar frequency. Malaysia Airlines flight MH17, another Boeing 777-200ER, was surpassed Malaysia Airlines flight MH370. Malaysia Airlines assumes beyond reasonable doubt there are no survivors. They reported the Malaysia Airlines flight MH370 missing. They releases passenger manifest of flight MH370. They will give US$ 5000 to the relatives of each passenger. Malaysia released preliminary report. It set up a Joint Investigation Team. Southeast Asian states have joined forces to search waters between Malaysia and Vietnam. Chinese government criticizes Malaysia for inadequate answers regarding Malaysia Airlines flight MH370. Malaysia will be deploying more ships and equipment to assist in the search. It ends hunt in South China Sea. Continued refinement of analysis of flight MH370's satellite communications identified a wide area search. Australia and Malaysia are working on a Memorandum of Understanding to cover financial and co-operation arrangements for search and recovery activities.

Figure 1. The summary of "Malaysia MH370 Disappear" news event generated by our system.

Malaysia Airlines said in a statement that **flight MH370 had disappeared** at 02:40 local time on Saturday after leaving Kuala Lumpur. Southeast Asian states have joined forces to search waters between Malaysia and Vietnam after **a Malaysia Airlines plane vanished** on a **flight to Beijing**, with 239 people on board. Flight MH370 had been expected to **land in Beijing** at 06:30. If Malaysia Airlines flight MH370 had impacted the ocean hard, resulting underwater sounds could have been detected by hydrophones, given favorable circumstances. Scientists from the CTBTO analyzed their recordings soon after **flight MH370 disappeared**, finding nothing of interest. The CMST researchers believe that the most likely explanation of the hydroacoustic data is that they come from the same event, but unrelated to Malaysia Airlines flight MH370. The lead researcher of the CMST team, Dr.Alec Duncan, believes there's a slim chance that the acoustic event is related to Malaysia Airlines flight MH370. Several IMOS recorders deployed in the Indian Ocean off northwestern Australia by CMST may have recorded data related to Malaysia Airlines flight MH370. Malaysia Airlines released the names and nationalities of the 227 passengers and 12 crew members, based on the flight manifest, later modified to include two Iranian passengers travelling on stolen passports. If the data relates to the same event, related to flight MH370, but the arc derived from analysis of the aircraft's satellite transmission is incorrect, then the most likely place to look for the aircraft would be along a line from HA01.

Figure 2. The summary of "Malaysia MH370 Disappear" news event generated by *MultiMR*.

5 Conclusions and Future Works

The proposed summarization approach is effective in information extraction and achieves good performance on DUC datasets. Through the sample summary, we can find that the approach is very effective for summarizing texts that mainly describe facts and actions of news event. Summaries generated by our system are informative, coherent and compact.

But for texts expressing opinions, the approach can't settle it appropriately. For example, when the verbs of *BSUs* are not meaningful actions, like "be", the semantic relations between them can't be appropriately computed by the methods described in the paper. More efficient methods to computer semantic relations between *BSUs* should be developed in the following work.

The sentence generation process described in the paper is just a preliminary scheme. It should be developed to generate sentence relying less on the original sentence structure and aggregating information from several different *BSUs*.

Reference

Barzilay, R., and McKeown, K. R. 2005. Sentence fusion for multidocument news summarization. *Computational Linguistics*, 31(3): 297-328.

Banko, M., Cafarella, M. J., Soderland, S., et al. 2007. Open information extraction for the web. In *IJCAI* 2007, 7: 2670-2676.

Bing, L., Li, P., Liao, Y., Lam, W., et al., 2015. Abstractive Multi-Document Summarization via Phrase Selection and Merging. In *ACL* 2015, 1587-1597

Carenini, G., and Cheung, J. C. K. 2008. Extractive vs. NLG-based abstractive summarization of evaluative text: The effect of corpus controversiality. Proceedings of the Fifth International Natural Language Generation Conference. Association for Computational Linguistics, 33-41.

Cohn, T., and Lapata, M. 2009. Sentence compression as tree transduction. *Journal of Artificial Intelligence Research*, 637-674.

Etzioni, O., Fader, A., Christensen, J., et al. 2011. Open Information Extraction: The Second Generation. In *IJCAI* 2011, 11: 3-10.

Finkel, J. R., Grenager, T., and Manning, C. 2005. Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling. In *ACL* 2005, 363-370.

Filippova, K., and Strube, M. Sentence fusion via dependency graph compression. In *EMNLP* 2008, 177-185.

Gabrilovich, E., and Markovitch, S. 2007. Computing Semantic Relatedness Using Wikipedia-based Explicit Semantic Analysis. In IJCAI 2007, 7: 1606-1611.

Genest, P. E., and Lapalme, G. 2011. Framework for abstractive summarization using text-to-text generation. In *Proceedings of the Workshop on Monolingual Text-To-Text Generation*, 64-73.

Joachims, T. 1999. Svmlight: Support vector machine. SVM-Light Support Vector Machine http://svmlight. joachims. org/, *University of Dortmund*, 19(4).

Knight, K., and Marcu, D. 2000. Statistics-based summarization-step one: Sentence compression. In *AAAI/IAAI* 2000, 703-710.

Knight, K., and Marcu, D. 2002. Summarization beyond sentence extraction: A probabilistic approach to sentence compression. In *Artificial Intelligence*. 139(1): 91-107

Klein, D., and Manning, C. D. 2003. Accurate unlexicalized parsing. In *ACL* 2003, 423-430.

Liu, F., Flanigan, J., et al. 2015. Toward Abstractive Summarization Using Semantic Representations. In *HLT-NAACL* 2015.

Lin, C. Y., and Hovy, E. 2003. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *HLT-NAACL* 2003, 71-78.

Lee, H., Peirsman, Y., Chang, A., et al. 2011. Stanford's multi-pass sieve coreference resolution system at the CoNLL-2011 shared task. In *ACL* 2011, 28-34.

Mihalcea, R., Corley, C., and Strapparava, C. 2006. Corpus-based and knowledge-based measures of text semantic similarity. In AAAI 2006, 6: 775-780.

Nenkova, A., and Passonneau, R. 2004. Evaluating content selection in summarization: The pyramid method. In *HLT-NAACL*, pages 145-152.

Passonneau, R. J., Chen, E., Guo, W., and Perin, D. 2013. Automated Pyramid Scoring of Summaries using Distributional Semantics. In *ACL*(2), pages: 143-147.

Smola, A. J., and Schölkopf, B. 2004. A tutorial on support vector regression. Statistics and computing, 14(3): 199-222.

Tanaka, H., Kinoshita, A., Kobayakawa, T., Kumano, T., and Kato, N. 2009. Syntax-driven sentence revision for broadcast news summarization. In *Proceedings of the 2009 Workshop on Language Generation and Summarisation*, 39-47.

Wan, X., Yang, J., and Xiao, J. 2007. Manifold-Ranking Based Topic-Focused Multi-Document Summarization. In *IJCAI* 2007, 7:2903-2908.

Wang, D., Li, T., Zhu, S., and Chris, D. 2008. Multi-document summarization via sentence-level semantic analysis and symmetric matrix factorization. In *SIGIR 2008*, 307-314.

Wan, X., and Xiao, J. 2009. Graph-Based Multi-Modality Learning for Topic-Focused Multi-Document Summarization. In *IJCAI* 2009, 1586-1591.

Wang, L., and Cardie, C. 2013. Domain-Independent Abstract Generation for Focused Meeting Summarization. In *ACL* 2013, 1395-1405.

Zhuge, H. 2009. Communities and Emerging Semantics in Semantic Link Network: Discovery and Learning, IEEE Transactions on Knowledge and Data Engineering, vol.21, no.6, 2009, pp. 785-799.

Zhuge, H. 2010. Interactive Semantics, Artificial Intelligence, 174(2010)190-204.

Zhuge, H. 2011. Semantic linking through spaces for cyber-physical-socio intelligence: A methodology, Artificial Intelligence, 175(2011)988-1019.

Zhuge, H. 2012. Chapter 2 in The Knowledge Grid: Toward Cyber-Physical Society. *World Scientific*.

Zhuge, H. 2015a. Dimensionality on Summarization, arXiv:1507.00209 [cs.CL], 2 July 2015.

Zhuge, H. 2015b. Mapping Big Data into Knowledge Space with Cognitive Cyber-Infrastructure, arXiv:1507.06500, 24 July 2015.

Concept-based Summarization using Integer Linear Programming: From Concept Pruning to Multiple Optimal Solutions

Florian Boudin[1] **Hugo Mougard**[1] **Benoit Favre**[2]

[1] LINA - UMR CNRS 6241, Université de Nantes, France
{florian.boudin,hugo.mougard}@univ-nantes.fr

[2] LIF - UMR CNRS 7279, Université Aix-Marseille, France
benoit.favre@lif.univ-mrs.fr

Abstract

In concept-based summarization, sentence selection is modelled as a budgeted maximum coverage problem. As this problem is NP-hard, pruning low-weight concepts is required for the solver to find optimal solutions efficiently. This work shows that reducing the number of concepts in the model leads to lower ROUGE scores, and more importantly to the presence of multiple optimal solutions. We address these issues by extending the model to provide a single optimal solution, and eliminate the need for concept pruning using an approximation algorithm that achieves comparable performance to exact inference.

1 Introduction

Recent years have witnessed increased interest in global inference methods for extractive summarization. These methods formulate summarization as a combinatorial optimization problem, i.e. selecting a subset of sentences that maximizes an objective function under a length constraint, and use Integer Linear Programming (ILP) to solve it exactly (McDonald, 2007).

In this work, we focus on the concept-based ILP model for summarization introduced by (Gillick and Favre, 2009). In their model, a summary is generated by assembling the subset of sentences that maximizes a function of the unique concepts it covers. Selecting the optimal subset of sentences is then cast as an instance of the budgeted maximum coverage problem[1].

As this problem is NP-hard, pruning low-weight concepts is required for the ILP solver to find optimal solutions efficiently (Gillick and Favre, 2009; Qian and Liu, 2013; Li et al., 2013). However, reducing the number of concepts in the model has two undesirable consequences. First, it forces the model to only use a limited number of concepts to rank summaries, resulting in lower ROUGE scores. Second, by reducing the number of items from which sentence scores are derived, it allows different sentences to have the same score, and ultimately leads to multiple optimal summaries.

To our knowledge, no previous work has mentioned these problems, and only results corresponding to the first optimal solution found by the solver are reported. However, as we will show through experiments, these multiple optimal solutions cause a substantial amount of variation in ROUGE scores, which, if not accounted for, could lead to incorrect conclusions. More specifically, the contributions of this work are as follows:

- We evaluate (Gillick and Favre, 2009)'s summarization model at various concept pruning levels. In doing so, we quantify the impact of pruning on running time, ROUGE scores and the number of optimal solutions.
- We extend the model to address the problem of multiple optimal solutions, and we sidestep the need for concept pruning by developing a fast approximation algorithm that achieves near-optimal performance.

2 Concept-based ILP Summarization

2.1 Model definition

Gillick and Favre (2009) introduce a concept-based ILP model for summarization that casts sentence selection as a maximum coverage problem. The key assumption of their model is that the value of a summary is defined as the sum of the weights of the unique concepts it contains. That way, redundancy within the summary is addressed implicitly at a sub-sentence level: a summary only benefits from including each concept once.

[1] Given a collection S of sets with associated costs and a budget L, find a subset $S' \subseteq S$ such that the total cost of sets in S' does not exceed L, and the total weight of elements covered by S' is maximized (Khuller et al., 1999).

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1914–1918,
Lisbon, Portugal, 17-21 September 2015.

Formally, let w_i be the weight of concept i, c_i and s_j two binary variables indicating the presence of concept i and sentence j in the summary, Occ_{ij} an indicator of the occurrence of concept i in sentence j, l_j the length of sentence j and L the length limit for the summary, the concept-based ILP model is described as:

$$\max \quad \sum_i w_i c_i \tag{1}$$

$$s.t. \quad \sum_j l_j s_j \leq L \tag{2}$$

$$s_j Occ_{ij} \leq c_i, \quad \forall i, j \tag{3}$$

$$\sum_j s_j Occ_{ij} \geq c_i, \quad \forall i \tag{4}$$

$$c_i \in \{0, 1\} \, \forall i$$

$$s_j \in \{0, 1\} \, \forall j$$

The constraints formalized in equations 3 and 4 ensure the consistency of the solution: selecting a sentence leads to the selection of all the concepts it contains, and selecting a concept is only possible if it is present in at least one selected sentence.

Choosing a suitable definition for concepts and a method to estimate their weights are the two key factors that affect the performance of this model. Bigrams of words are usually used as a proxy for concepts (Gillick and Favre, 2009; Berg-Kirkpatrick et al., 2011). Concept weights are either estimated by heuristic counting, e.g. document frequency in (Gillick and Favre, 2009), or obtained by supervised learning (Li et al., 2013).

2.2 Pruning to reduce complexity

The concept-level formulation of (Gillick and Favre, 2009) is an instance of the budgeted maximum coverage problem, and solving such a problem is NP-hard (Khuller et al., 1999). Keeping the number of variables and constraints small is then critical to reduce the model complexity.

In previous work, efficient summarization was achieved by pruning concepts. One way to reduce the number of concepts in the model is to remove those concepts that have a weight below a given threshold (Gillick and Favre, 2009). Another way is to consider only the top-n highest weighted concepts (Li et al., 2013). Once low-weight concepts are pruned, sentences that do not contain any remaining concepts are removed, further reducing the number of variables and constraints in the model. As such, this can be regarded as a way to approximate the problem.

Pruning concepts to reduce complexity also cuts down the number of items from which summary scores are derived. As we will see in Section 3.2, this results in a lower ROUGE scores and leads to the production of multiple optimal summaries.

The concept weighting function also plays an important role in the presence of multiple optimal solutions. Limited-range functions, such as frequency-based ones, yield many ties and increase the likelihood that different sentences have the same score. Redundancy within the set of input sentences exacerbate this problem, since highly similar sentences are likely to contain the same concepts.

2.3 Summarization parameters

For comparison purposes, we use the same system pipeline as in (Gillick et al., 2009), which is described below.

Step 1: clean input documents; a set of rules is used to remove bylines and format markup.

Step 2: split the text into sentences; we use splitta[2] (Gillick, 2009) and re-attach multi-sentence quotations.

Step 3: compute parameters needed by the model; we extract and weight the concepts.

Step 4: prune sentences shorter than 10 words, duplicate sentences and those that begin and end with a quotation mark.

Step 5: map to ILP format and solve; we use an off-the-shelf ILP solver[3].

Step 6: order selected sentences for inclusion in the summary, first by source and then by position.

Similar to previous work, we use bigrams of words as concepts. Although bigrams are rough approximations of concepts, they are simple to extract and match, and have been shown to perform well at this task. Bigrams of words consisting of two stop words[4] or containing a punctuation mark are discarded. Stemming[5] is then applied to allow more robust matching.

Concepts are weighted using document frequency, i.e. the number of source documents

[2]We use splitta v1.03, `https://code.google.com/p/splitta/`

[3]We use glpk v4.52, `https://www.gnu.org/software/glpk/`

[4]We use the stoplist in nltk, `http://www.nltk.org/`

[5]We use the Porter stemmer in nltk.

	DUC'04				TAC'08			
DF	1	2	3	4	1	2	3	4
# solutions	1.3	1.3	1.5	1.5	1.2	1.3	1.8	4.8
# concepts	2 955	676	247	107	2 909	393	127	56
# sentences	184	175	159	139	174	167	149	129
Avg. time (sec)	22.3	1.7	0.5	0.3	21.5	0.8	0.3	0.2

Table 1: Average number of optimal solutions, concepts and sentences for different minimum document frequencies. The average time in seconds for finding the first optimal solution is also reported.

where the concept was seen. Document frequency is a simple, yet effective approach to concept weighting (Gillick and Favre, 2009; Woodsend and Lapata, 2012; Qian and Liu, 2013). Reducing the number of concepts in the ILP model is then performed by pruning those concepts that occur in fewer than a given number of documents.

ILP solvers usually provide only one solution. To generate alternate optimal solutions, we iteratively add new constraints to the problem that eliminate already found optimal solutions and rerun the solver. We stop the iterations when the value of the objective function returned by the solver changes.

3 Experiments

3.1 Datasets and evaluation measures

Experiments are conducted on the DUC'04 and TAC'08 datasets. For DUC'04, we use the 50 topics from the generic multi-document summarization task (Task 2). For TAC'08, we focus only on the 48 topics from the non-update summarization task. Each topic contains 10 newswire articles for which the task is to generate a summary no longer than 100 words (whitespace-delimited tokens).

Summaries are evaluated against reference summaries using the ROUGE automatic evaluation measures (Lin, 2004). We set the ROUGE parameters to those[6] that lead to highest agreement with manual evaluation (Owczarzak et al., 2012), that is, with stemming and stopwords not removed.

3.2 Results

Table 1 presents the average number of optimal solutions at different levels of concept pruning. Overall, the average number of optimal solutions increases along with the minimum document frequency, reaching 4.8 for TAC'08 at DF $= 4$. Pruning concepts also greatly reduces the number of variables in the ILP formulation, and consequently improves the run-time for solving the problem.

Interestingly, we note that, even without any pruning, the model produces multiple optimal solutions. The choice of document frequency for weighting concepts is responsible for this as it generates many ties. Finer-grained concept weighting functions such as frequency estimation (Li et al., 2013) should therefore be preferred to limit the number of multiple optimal solutions.

The mean ROUGE recall scores of the multiple optimal solutions for different minimal document frequencies are presented in Table 2. Here, the higher the concept pruning threshold, the higher the variability of the generated summaries as indicated by the standard deviation. Best ROUGE scores are achieved without concept pruning while the best compromise between effectiveness and run-time is given when DF ≥ 3, confirming the findings of (Gillick and Favre, 2009).

To show in a realistic scenario how multiple optimal solutions could lead to different conclusions, we compare in Table 3 the ROUGE-1 scores of the summaries generated from the first optimal solution found by three off-the-shelf ILP solvers against that of the systems[7] that participated at TAC'08. We set the minimum document frequency to 3, which is often used in previous work (Gillick and Favre, 2009; Li et al., 2013), and use a two-sided Wilcoxon signed-rank to compute the number of systems that obtain significantly lower and higher ROUGE-1 recall scores[8].

Despite being comparable (p-value > 0.4), the solutions found by the three solvers support different conclusions. The solution found using GLPK

[6] We use ROUGE-1.5.5 with the parameters: `n 4 -m -a -l 100 -x -c 95 -r 1000 -f A -p 0.5 -t 0`

[7] 71 systems participated at TAC'08 but we removed ICSI1 and ICSI2 systems which are based on the concept-based ILP model.

[8] ROUGE-1 recall is most accurate metric to identify the better summary in a pair (Owczarzak et al., 2012).

	DUC'04			TAC'08		
DF	ROUGE-1	ROUGE-2	ROUGE-4	ROUGE-1	ROUGE-2	ROUGE-4
1	37.74 ±0.07	**9.48** ±0.05	1.45 ±0.02	**37.65** ±0.10	**10.63** ±0.08	**2.23** ±0.04
2	37.25 ±0.08	9.14 ±0.02	1.37 ±0.01	37.16 ±0.11	9.96 ±0.07	2.05 ±0.03
3	37.37 ±0.11	9.16 ±0.06	1.41 ±0.02	37.39 ±0.15	10.62 ±0.07	2.13 ±0.03
4	**37.96** ±0.10	9.38 ±0.05	**1.57** ±0.02	36.73 ±0.12	10.10 ±0.08	1.78 ±0.07

Table 2: Mean ROUGE recall and standard deviation for different minimum document frequencies.

Solver	ROUGE-1	↓ / ↑
GLPK	37.33	54 / 0
Gurobi	37.20	52 / 1
CPLEX	37.17	51 / 1

Table 3: ROUGE-1 recall scores for the first optimal solution found by different solvers along with the number of systems that obtain significantly lower (↓) or higher (↑) scores (p-value < 0.05).

indicates that the concept-based model achieves state-of-the-art performance whereas the solutions provided by Gurobi and CPLEX do not do so. The reason for these differences is the use of different solving strategies, involving heuristics for finding feasible solutions more quickly. This example demonstrates that multiple optimal solutions should be considered during evaluation.

3.3 Solving the multiple solution problem

Multiple optimal solutions occur when concepts alone are not sufficient to distinguish between two competing summary candidates. Extending the model so that it provides a single solution can therefore not be done without introducing a second term in the objective function. Following the observation that the frequency of a non-stop word in a document set is a good predictor of a word appearing in a human summary (Nenkova and Vanderwende, 2005), we extend equation 1 as follows:

$$\max \quad \sum_i w_i c_i + \mu \sum_k f_k t_k \qquad (5)$$

where f_k is the frequency of non-stop word k in the document set, and t_k is a binary variable indicating the presence of k in the summary. Here, we want to induce a single solution among the multiple optimal solutions given by concept weighting, and thus set μ to a small value (10^{-6}). We add further constraints, similar to equations 3 and 4, to ensure the consistency of the solution.

This extended model succeeds in giving a single solution that is at least comparable to the mean score of the multiple optimal solutions. However, it requires about twice as much time to solve which makes it impractical for large documents.

3.4 Fast approximation

Instead of pruning concepts to reduce complexity, one may consider using an approximation if results are found satisfactory. Here, similarly to (Takamura and Okumura, 2009; Lin and Bilmes, 2010) we implement the greedy heuristic proposed in (Khuller et al., 1999) that solve the budgeted maximum coverage problem with a performance guarantee $1/2 \cdot (1 - 1/e)$. Table 4 compares the performance of the model that achieves the best trade off between effectiveness and runtime, that is when $DF \geq 3$, with that of the greedy approximation without pruning.

Overall, the approximate solution is over 96% as good as the average optimal solution. Although the ILP solution marks an upper bound on performance, its solving time is exponential in the number of input sentences. The approximate method is then relevant as it marks an upper bound on speed (less than 0.01 seconds to compute) while having performance comparable to the ILP model with concept pruning (p-value > 0.3).

Dataset	ROUGE-1	ROUGE-2
DUC'04	37.14 (−0.7%)	9.37 (+2.3%)
TAC'08	36.90 (−1.3%)	10.27 (−3.3%)

Table 4: ROUGE recall scores of the approximation. The relative difference from the mean score of the multiple optimal solutions is also reported.

4 Conclusion

Multiple optimal solutions are not an issue as long as alternate solutions are equivalent. Unfortunately, summaries generated from different sets of

sentences are likely to differ. We showed through experiments that concept pruning leads to the presence of multiple optimal solutions, and that the latter cause a substantial amount of variation in ROUGE scores. We proposed an extension of the ILP that obtains unique solutions. If speed is a concern, we showed that a near-optimal approximation can be computed without pruning. The implementation of the concept-based summarization model that we use in this study is available at `https://github.com/boudinfl/sume`.

In future work, we intend to extend our study to compressive summarization. We expect that the number of optimal solutions will increase as multiple compression candidates, which are likely to be similar in content, are added to the set of input sentences.

Acknowledgments

This work was partially supported by the GOLEM project (grant of CNRS PEPS FaSciDo 2015, `http://boudinfl.github.io/GOLEM/`). We thank the anonymous reviewers and Rémi Bois for their insightful comments.

References

Taylor Berg-Kirkpatrick, Dan Gillick, and Dan Klein. 2011. Jointly learning to extract and compress. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 481–490, Portland, Oregon, USA, June. Association for Computational Linguistics.

Dan Gillick and Benoit Favre. 2009. A scalable global model for summarization. In *Proceedings of the Workshop on Integer Linear Programming for Natural Language Processing*, pages 10–18, Boulder, Colorado, June. Association for Computational Linguistics.

Dan Gillick, Benoit Favre, Dilek Hakkani-Tur, Berndt Bohnet, Yang Liu, and Shasha Xie. 2009. The icsi/utd summarization system at tac 2009. In *Proceedings of the Second Text Analysis Conference*, Gaithersburg, Maryland, USA. National Institute of Standards and Technology.

Dan Gillick. 2009. Sentence boundary detection and the problem with the u.s. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, pages 241–244, Boulder, Colorado, June. Association for Computational Linguistics.

Samir Khuller, Anna Moss, and Joseph (Seffi) Naor. 1999. The budgeted maximum coverage problem. *Information Processing Letters*, 70(1):39 – 45.

Chen Li, Xian Qian, and Yang Liu. 2013. Using supervised bigram-based ilp for extractive summarization. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1004–1013, Sofia, Bulgaria, August. Association for Computational Linguistics.

Hui Lin and Jeff Bilmes. 2010. Multi-document summarization via budgeted maximization of submodular functions. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 912–920, Los Angeles, California, June. Association for Computational Linguistics.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In Stan Szpakowicz Marie-Francine Moens, editor, *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, pages 74–81, Barcelona, Spain, July. Association for Computational Linguistics.

Ryan McDonald. 2007. A study of global inference algorithms in multi-document summarization. In *Proceedings of the 29th European Conference on IR Research*, ECIR'07, pages 557–564, Berlin, Heidelberg. Springer-Verlag.

Ani Nenkova and Lucy Vanderwende. 2005. The impact of frequency on summarization. *Microsoft Research, Redmond, Washington, Tech. Rep. MSR-TR-2005-101*.

Karolina Owczarzak, John M. Conroy, Hoa Trang Dang, and Ani Nenkova. 2012. An assessment of the accuracy of automatic evaluation in summarization. In *Proceedings of Workshop on Evaluation Metrics and System Comparison for Automatic Summarization*, pages 1–9, Montréal, Canada, June. Association for Computational Linguistics.

Xian Qian and Yang Liu. 2013. Fast joint compression and summarization via graph cuts. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1492–1502, Seattle, Washington, USA, October. Association for Computational Linguistics.

Hiroya Takamura and Manabu Okumura. 2009. Text summarization model based on maximum coverage problem and its variant. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, pages 781–789, Athens, Greece, March. Association for Computational Linguistics.

Kristian Woodsend and Mirella Lapata. 2012. Multiple aspect summarization using integer linear programming. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 233–243, Jeju Island, Korea, July. Association for Computational Linguistics.

GhostWriter: Using an LSTM for Automatic Rap Lyric Generation

Peter Potash, Alexey Romanov, Anna Rumshisky
Dept. of Computer Science
University of Massachusetts Lowell
Lowell, MA 01854
{ppotash,aromanov,arum}@cs.uml.edu

Abstract

This paper demonstrates the effectiveness of a Long Short-Term Memory language model in our initial efforts to generate unconstrained rap lyrics. The goal of this model is to generate lyrics that are similar in style to that of a given rapper, but not identical to existing lyrics: this is the task of ghostwriting. Unlike previous work, which defines explicit templates for lyric generation, our model defines its own rhyme scheme, line length, and verse length. Our experiments show that a Long Short-Term Memory language model produces better "ghostwritten" lyrics than a baseline model.

1 Introduction

Ghostwriting defines a distinction between the performer/presenter of text, lyrics, etc, and the creator of text/lyrics. The goal is to present something in a style that is believable enough to be credited to the performer. In the domain of rap specifically, rappers sometimes function as ghostwriters early on before embarking on their own public careers, and there are even businesses that provide written lyrics as a service [1]. The goal of GhostWriter is to produce a system that can take a given artist's lyrics and generate *similar* yet *unique* lyrics. To accomplish this, we must create a language model to produce text, while also understanding what 'style' means in a quantitative sense.

The contribution of this paper is three-fold: (1) we present the ghostwriting problem of producing similar yet different lyrics; (2) we present computational, quantitative evaluation methods for these two aspects; (3) we evaluate the performance of a Long Short-Term Memory (LSTM) vs n-gram model for this problem.

2 Related Work

Recent work (Sutskever et al., 2011; Graves, 2013) has shown the effectiveness of Recurrent Neural Networks (RNNs) for text generation. In their works, the authors use an RNN to create a language model at the character level. The results are inspiring, as the models learn various grammatical and punctuation rules, such as opening and closing parentheses, plus learning a large vocabulary of English words at the character level. Graves (2013) uses a variation of an RNN called LSTM architecture which creates a better language model than a regular RNN.

Text generation for artistic purposes, such as poetry and lyrics, has also been explored, often using templates and constraints (Oliveira et al., 2014; Barbieri et al., 2012). In regards to rap lyrics, Wu et al. (2013) present a system for rap lyric generation that produces a single line of lyrics that are meant to be a response to a single line of input. However, the work that is most similar to ours is that of Malmi et al. (2015). The authors create fixed 16-line verses, generating the verse line-by-line using full lines from existing rap songs. The system predicts the best next line based on the previous lines, using a system that records an 81.9% accuracy predicting next lines in already existing verses. The feature that provides the greatest accuracy gain is a neural embedding of the lines, created from the character level.

Hirjee and Brown (2010b) have developed a rhyme detection tool based on a probabilistic model (Hirjee and Brown, 2010a) that analyzes phoneme patterns in words. The model is trained on a set of lyrics that were manually annotated for rhyming words. The statistics generated by the rhyme detection tool will be an important part of

[1] http://www.rap-rebirth.com/, http://www.precisionwrittens.com/rap-ghostwriters-for-hire/

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1919–1924,
Lisbon, Portugal, 17-21 September 2015.

our evaluation (see Section 5).

3 Generating Lyrics

In a departure from previous work on poetry/lyric generation, our goal is to build a model that does not require templates/constraints to generate lyrics, while also being able to produce full verses, as opposed to single lines. The system must be able to model general human language in order to produce fluent lyrics, but it must also be able to model the style of a target artist, by understanding the artist's vocabulary and rhythmic style, in order to fully execute the ghostwriting task of producing similar yet new lyrics.

3.1 LSTM

Here we will give a very brief overview of RNNs and LSTMs. For a more detailed explanation please refer to (Graves, 2013). The foundation of an RNN (of which an LSTM is specific architecture) is a word embedding E that provides a vector representation for each of the words in our corpus. Given a history of words $w_k, ..., w_0$ we want to determine $P(w_{k+1}|w_k, ..., w_0; E, \Phi)$, where Φ is a set of parameters used by our model. In the context of an RNN we define this probability by:

$$P(w_{k+1}|w_k, ..., w_0; E, \Phi) = f(x, s) \quad (1)$$

At each time-step the RNN computes f given an observation x and a previous state s. The input goes through a transformation where it passes through one or several hidden layers.

The LSTM model uses a specific architecture for the hidden transformation, defined by the LSTM memory cell. The key feature to the LSTM memory cell is the presence of an input gate, output gate, forget gate, and cell/cell memory, which manifest themselves in the model as activation vectors. Each of these gates/cells has its own bias vector, and the hidden layer at each time-step is now a complex nonlinear combination of gate, cell, and hidden vectors.

3.2 Using LSTM for Lyrics Generation

Since previous work has shown the power of RNNs to model language, we hope that it can capture the rhythmic style of an artist by learning rhyme and meter patterns. As noted in Section 2, LSTMs have performed well at sequence forecasting, for example at learning punctuation, such as opening and closing parentheses. We see the task of rhyme detection as something similar in nature. Kaparthy et al. (2015) have also shown that LSTMs could successfully learn where to place the brackets and indentation in C++ code. In their model, certain LSTM cells activated specifically when encountering end of the line. We believe learning rhymes at the end of the line is conceptually similar to such tasks.

3.3 Verse Structure and Rhyme Inference

The goal of our model is to not just generate lyrics, but generate the structure for the lyrics as well. To do this, we have added "<endLine>" and "<endVerse>" tokens to the lyrics. From this, the system will generate its own line breaks, while also defining when a generated verse ends. This allows us to analyze non-rhyming features from (Hirjee and Brown, 2010a), such as number of syllables per line and number of lines per verse. We also desire that, by using the "<endLine>" token, the system has a better chance of understanding rhyme schemes used by an artist. For example, the LSTM can capture the pattern of "came <endLine>" followed shortly by "name <endLine>" to understand that "came" and "name" are a rhyming pair. To do this effectively, the system would need sufficient training data where rhyming pairs occur frequently enough to actually dictate a pattern, similar to (Reddy and Knight, 2011; Addanki and Wu, 2013).

4 Experimental Design

4.1 Dataset

We collected songs from 14 artists from the site *The Original Hip-Hop (Rap) Lyrics Archive - OHHLA.com - Hip-Hop Since 1992*[2]. In the present lyrics generation experiments, we used the lyrics from the rapper Fabolous. For training, we used 219 verses with at least 175 words in each verse. We selected Fabolous because his lyrics produced the highest accuracy in the artist recognition experiments in (Hirjee and Brown, 2010a). We conjecture that because of this, he had the most consistent style, making him a good choice for initial experiments.

[2] http://www.ohhla.com/

4.2 Baseline

To compare with the results of the LSTM model, we followed the work of (Barbieri et al., 2012) and created a Markov model for lyric generation. Since the goal of our work is to make an unsupervised system, we do not use any constraints or templates to produce the lyrics. Thus, our baseline simplifies to a basic n-gram model. Given a history of w_{k+n-1},...,w_k, the system generates a new token t as follows:

$$P(w_{k+n} = t|w_{k+n-1}, ..., w_k) = \frac{|w_k,...,w_{k+n-1},t|}{|w_k,...,w_{k+n-1},\bullet|} \quad (2)$$

where $|w_k...w_{k+n-1}t|$ is the amount of times the the context w_{k+n-1},...,w_1 is followed by t in the training data, and $|w_k...w_{k+n-1}\bullet|$ is the amount of times the context appears followed by any token. There is the possibility that the context has never been encountered in the training data. When this occurs, we back off to a smaller n-gram model:

$$P(w_{k+n} = t|w_{k+n-2}, ..., w_k) = \frac{|w_k,...,w_{k+n-2},\bullet,t|}{|w_k,...,w_{k+n-2},\bullet,\bullet|} \quad (3)$$

The model may have to back-off multiple times before it encounters context it has seen in the training data. Once we back-off to the point where we compute $P(w_{n+k} = t|w_k)$, we are guaranteed to have at least one non-zero probability, because w_k must have appeared in the vocabulary for it to have been generated previously.

Note that rather than backing off to a lower-order n-gram model, we use a skip-gram model which drops the words immediately preceding the predicted word. The main motivation for this is that it allows us to capture long-range dependencies, which makes it into a better baseline comparison for an LSTM.

4.3 Model Initialization

When producing lyrics with either the LSTM or baseline model, we initialize with the "<startVerse>" token. Once the model produces a token, it becomes part of the context for the next step of token generation. Our models are closed in the sense that they only produce tokens that appear in the training vocabulary.

4.4 LSTM Implementation

We used a Python implementation of an LSTM from Jonathan Raiman[3]. The LSTM is built on top of Theano (Bastien et al., 2012; Bergstra et al., 2010). Following (Graves, 2013), we set the amount of LSTM inputs/outputs to be equal to the vocabulary size. Also, to avoid the vanishing gradient problem when training RNNs, we clip the gradients in the range [-1,1]. We train our LSTM model using a Tesla K40 GPU on a single workstation.

5 Evaluation Methods

In this section, we present automated methods for evaluating the quality of generated lyrics. Ideally, judging system output in terms of, e.g. fluency, should be conducted using manual evaluation. However, conducting formal human evaluation is somewhat problematic. For a full *qualitative* evaluation of a given artist that would assess both similarity of style and novelty, the evaluator would need to know that particular artist's body of work very well. Even finding annotators who are well-versed in the general art of rap lyrics can be challenging (Addanki and Wu, 2014). While this may be possible for the present experiments that focus on a single artist, it is hardly feasible for larger-scale studies that will use our full data set that contains the lyrics of 14 different artists. We therefore propose an automated evaluation method which we believe is able to capture two critical aspects of ghostwriting, which are in fact quite tricky to capture together: being similar, yet different.

5.1 Similarity to existing lyrics

In order to evaluate the novelty of generated lyrics, we compare the similarity of the generated lyrics to the lyrics in our training set. We used an algorithm proposed by (Mahedero et al., 2005) for calculating the similarity between produced lyrics and all verses from the same artist. This algorithm is based on the well-known Inverse Document Frequency, using cosine on document vectors to calculate distance. First, we build the Term-Document Matrix with weights for each term in each song:

$$w_{ij} = f_{ij} log(\frac{N}{n_j}) \quad (4)$$

[3] https://github.com/JonathanRaiman/theano_lstm

where N is the total number of documents (verses, in our case), n_j is the number of verses that contains term j and f_{ij} is the frequency of term j in the ith verse. Using this matrix, we can calculate the cosine distance between verses and use it as a measure of similarity. When discussing similarity, we refer to the max similarity: of all verses it is most similar to, exactly *how* similar is it? The lower the max similarity score, the more novel the lyrics.

5.2 Numerical features of the lyrics

We also produced the features from (Hirjee and Brown, 2010a) for our generated verses. The statistics of these features are meant to represent how effective we are in modeling an artist's style. The point of the system is not to produce arbitrary rhymes; it is to produce rhyme types and rhyme frequency that are similar to the target artist. Following (Malmi et al., 2015), the rhyme feature we examine in this work is rhyme density. Rhyme density is defined as the total number of rhymed syllables divided by the total number of syllables.

6 Results

For the lyrics generation experiments reported here, we used the rapper Fabolous as the artist whose style we tried to emulate.

6.1 Example of Generated Lyrics (Warning: Explicit Content)

Below is a sample of lyrics generated by the LSTM model:

> Line 1: i m old enough to know better young enough to not give a fuck
> Line 2: rather hold my head high and die then live and duck
> Line 3: i got ta fuck be up and little niggaz go in
> Line 4: when i m in the feds and scoped by uncle sam
> Line 5: dope and hunn ed grams rope and hunn ed grams
> Line 6: at the same time she jerking and wiggling
> Line 7: smirking and giggling

While the pairs of rhyming end-lines in the generated lyrics are taken from the training data (the max similarity is 0.41), no more than two lines appear from a single verse. Though the generated lyrics aren't novel in a strict sense, the LSTM model is more effective than the n-gram model at using lines from multiple verses (see next Section 6.3). The rhyme density of this verse is 0.35, which is almost equal to Fabolous's average rhyme density (0.34).

6.2 Quantitative Analysis

As previously mentioned, the key to effective ghostwriting is to mirror an artist's style, but also providing original content. While vocabulary and lyrical content are key components for an artist's style, this is inherently satisfied by using words only from the training data. Thus, rhyme style – specifically rhyme density – will be the key performance indicator for imitating an artist's style. In terms of rap lyrics in general, a higher rhyme density is often better. Therefore for our system we would like a high rhyme density, but with a low max similarity score (a higher novelty). Figures 2 and 1 show the graph for rhyme density and max similarity for the LSTM and n-grams models, respectively. For the LSTM model the values are graphed compared to training iteration number – as the model becomes more fit to the data. For the n-gram model they are graphed dependent on n-gram value. For each n-gram value, we generate 10 verses and compute the average value of the two metrics. One expects that a perfectly fit LSTM model without regularization would exactly reproduce lyrics from the training data, and a high n-gram value would would also produce duplicate lyrics. This is evident in the graphs.

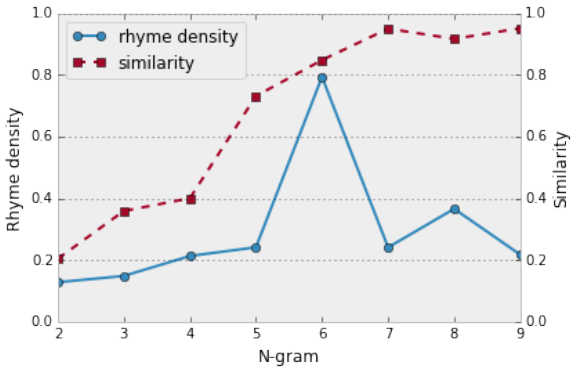


Figure 1: Values of rhyme density and max similarity versus n-gram value for the n-gram model.

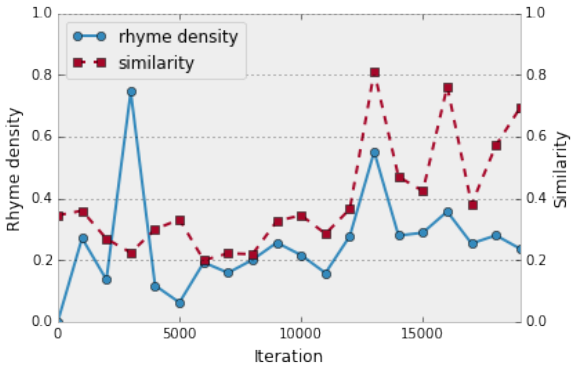


Figure 2: Values of rhyme density and max similarity versus iteration number when training the LSTM model.

6.3 Correlation of Rhyme Density and Max Similarity

Since exact replication would assuredly give a higher rhyme density than randomly produced lyrics, we desire a *low* correlation between rhyme density and max similarity. The correlation between rhyme density and max similarity for the LSTM model is 0.32, and for the n-gram model it is 0.47. When examining Figures 1 and 2 one may notice the anomalous points of high rhyme density (at $n = 6$ on the n-gram graph and 3,000 iterations for the LSTM model). After further inspection of the lyrics at these points, we see the lyrics contain repetitions of the exact same phrase. Since words are repeated frequently, the rhyme density of the lyrics is high (repeated words create rhymed phonemes, according to the rhyme detection tool). These points cause the similarity-density correlations to be artificially lower. After removing these data points, the LSTM model still has a lower correlation than the n-gram model, but the gap is much smaller: 0.71 compared to 0.75. Ultimately however, this shows that the LSTM model is better at generating original, rhyming lyrics.

6.4 Style Matching

Unfortunately, the correlation numbers do not dictate specifically the effectiveness of the LSTM model in the ghostwriting task. Instead, we can look at that max similarity values of both systems when they generate lyrics that produce a rhyme density similar to the average rhyme density of the target rapper. Looking at 100 randomly selected verses, Fabolous has an average rhyme density of 0.34. To do our analysis, first we create four regression lines, one for each metric (max similarity and rhyme density) in each model (we do not include the points of high rhyme density). Next we use the two rhyme density lines to determine at which iteration/n value the systems generate a rhyme density of 0.34. After that we plug these numbers into the two similarity lines to determine what similarity is needed to achieve the target rhyme density. The n-gram model line has a similarity of 1.28 at this point (above the max value of 1 for the metric), while the LSTM model has a value of 0.59. Based on these numbers, the LSTM model clearly outperforms the n-gram model when it comes to making original lyrics that are similar in style to our target rapper.

7 Conclusion

In this work, we have shown the effectiveness of an LSTM model for generating novel lyrics that are similar in style to a target artist. We compare the performance of the LSTM model to a much simpler system: an n-gram model. The results of our experiments show that, as an unsupervised, non-template model, the LSTM model is better able to produce novel lyrics that also reflect the rhyming style of the target artist. In future work, we plan to use more data to train our model, making it easier for our system to actually identify rhyming pairs and use them in new contexts. We also plan to encode phoneme features of words to improve rhyme discovery. Furthermore, we plan to generate lyrics from artists with a varying vocabulary size to see if it is easier to generate lyrics for an artist with a smaller vocabulary. In terms of evaluation, we hope to incorporate some method to evaluate the fluency of generated lyrics (Addanki and Wu, 2014). Lastly, to further avoid over-fitting to the training data and reproducing lyrics with a high similarity, we plan to use weight noise (Jim et al., 1996) to regularize our model.

Acknowledgments

We would like to thank the anonymous reviewers for their feedback.

References

Karteek Addanki and Dekai Wu. 2013. Unsupervised rhyme scheme identification in hip hop lyrics using hidden markov models. In *Statistical Language and Speech Processing*, pages 39–50. Springer.

Karteek Addanki and Dekai Wu. 2014. Evaluating improvised hip hop lyrics–challenges and observa-

tions. In *Proceedings of The Ninth International Conference on Language Resources and Evaluation (LREC)*.

Gabriele Barbieri, François Pachet, Pierre Roy, and Mirko Degli Esposti. 2012. Markov constraints for generating lyrics with style. In *ECAI*, pages 115–120.

Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, James Bergstra, Ian Goodfellow, Arnaud Bergeron, Nicolas Bouchard, David Warde-Farley, and Yoshua Bengio. 2012. Theano: new features and speed improvements. *arXiv preprint arXiv:1211.5590*.

James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. 2010. Theano: a cpu and gpu math expression compiler. In *Proceedings of the Python for scientific computing conference (SciPy)*, volume 4, page 3. Austin, TX.

Alex Graves. 2013. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.

Hussein Hirjee and Daniel Brown. 2010a. Using automated rhyme detection to characterize rhyming style in rap music.

Hussein Hirjee and Daniel G Brown. 2010b. Rhyme analyzer: An analysis tool for rap lyrics. In *Proceedings of the 11th International Society for Music Information Retrieval Conference*. Citeseer.

Kam-Chuen Jim, Clyde Lee Giles, and Bill G Horne. 1996. An analysis of noise in recurrent neural networks: convergence and generalization. *Neural Networks, IEEE Transactions on*, 7(6):1424–1438.

Andrej Karpathy, Justin Johnson, and Fei-Fei Li. 2015. Visualizing and understanding recurrent networks. *CoRR*, abs/1506.02078.

Jose PG Mahedero, Álvaro MartÍnez, Pedro Cano, Markus Koppenberger, and Fabien Gouyon. 2005. Natural language processing of lyrics. In *Proceedings of the 13th annual ACM international conference on Multimedia*, pages 475–478. ACM.

Eric Malmi, Pyry Takala, Hannu Toivonen, Tapani Raiko, and Aristides Gionis. 2015. Dopelearning: A computational approach to rap lyrics generation. *arXiv preprint arXiv:1505.04771*.

Hugo Gonçalo Oliveira, Raquel Hervás, Alberto Díaz, and Pablo Gervás. 2014. Adapting a generic platform for poetry generation to produce spanish poems. In *5th International Conference on Computational Creativity, ICCC*.

Sravana Reddy and Kevin Knight. 2011. Unsupervised discovery of rhyme schemes. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 77–82. Association for Computational Linguistics.

Ilya Sutskever, James Martens, and Geoffrey E Hinton. 2011. Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 1017–1024.

Dekai Wu, Karteek Addanki, Markus Saers, and Meriem Beloucif. 2013. Learning to freestyle: Hip hop challenge-response induction via transduction rule segmentation. In *2013 Conference on Empirical Methods in Natural Language Processing (EMNLP 2013), Seattle, Washington, USA*.

Better Summarization Evaluation with Word Embeddings for ROUGE

Jun-Ping Ng
Bloomberg L.P.
New York, USA
jng324@bloomberg.net

Viktoria Abrecht
Bloomberg L.P.
New York, USA
vkanchakousk@bloomberg.net

Abstract

ROUGE is a widely adopted, automatic evaluation measure for text summarization. While it has been shown to correlate well with human judgements, it is biased towards surface lexical similarities. This makes it unsuitable for the evaluation of abstractive summarization, or summaries with substantial paraphrasing. We study the effectiveness of word embeddings to overcome this disadvantage of ROUGE. Specifically, instead of measuring lexical overlaps, word embeddings are used to compute the semantic similarity of the words used in summaries instead. Our experimental results show that our proposal is able to achieve better correlations with human judgements when measured with the Spearman and Kendall rank coefficients.

1 Introduction

Automatic text summarization is a rich field of research. For example, shared task evaluation workshops for summarization were held for more than a decade in the Document Understanding Conference (DUC), and subsequently the Text Analysis Conference (TAC). An important element of these shared tasks is the evaluation of participating systems. Initially, manual evaluation was carried out, where human judges were tasked to assess the quality of automatically generated summaries. However in an effort to make evaluation more scaleable, the automatic ROUGE[1] measure (Lin, 2004b) was introduced in DUC-2004. ROUGE determines the quality of an automatic summary through comparing overlapping units such as n-grams, word sequences, and word pairs with human written summaries.

[1]Recall-Oriented Understudy of Gisting Evaluation

ROUGE is not perfect however. Two problems with ROUGE are that 1) it favors lexical similarities between generated summaries and model summaries, which makes it unsuitable to evaluate abstractive summarization, or summaries with a significant amount of paraphrasing, and 2) it does not make any provision to cater for the readability or fluency of the generated summaries.

There has been on-going efforts to improve on automatic summarization evaluation measures, such as the Automatically Evaluating Summaries of Peers (AESOP) task in TAC (Dang and Owczarzak, 2009; Owczarzak, 2010; Owczarzak and Dang, 2011). However, ROUGE remains as one of the most popular metric of choice, as it has repeatedly been shown to correlate very well with human judgements (Lin, 2004a; Over and Yen, 2004; Owczarzak and Dang, 2011).

In this work, we describe our efforts to tackle the first problem of ROUGE that we have identified above — its bias towards lexical similarities. We propose to do this by making use of word embeddings (Bengio et al., 2003). Word embeddings refer to the mapping of words into a multi-dimensional vector space. We can construct the mapping, such that the distance between two word projections in the vector space corresponds to the semantic similarity between the two words. By incorporating these word embeddings into ROUGE, we can overcome its bias towards lexical similarities and instead make comparisons based on the semantics of words sequences. We believe that this will result in better correlations with human assessments, and avoid situations where two word sequences share similar meanings, but get unfairly penalized by ROUGE due to differences in lexicographic representations.

As an example, consider these two phrases: 1) *It is raining heavily*, and 2) *It is pouring*. If we are performing a lexical string match, as ROUGE does, there is nothing in common between the

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1925–1930,
Lisbon, Portugal, 17-21 September 2015.

terms "raining", "heavily", and "pouring". However, these two phrases mean the same thing. If one of the phrases was part of a human written summary, while the other was output by an automatic summarization system, we want to be able to reward the automatic system accordingly.

In our experiments, we show that word embeddings indeed give us better correlations with human judgements when measured with the Spearman and Kendall rank coefficient. This is a significant and exciting result. Beyond just improving the evaluation prowess of ROUGE, it has the potential to expand the applicability of ROUGE to abstractive summmarization as well.

2 Related Work

While ROUGE is widely-used, as we have noted earlier, there is a significant body of work studying the evaluation of automatic text summarization systems. A good survey of many of these measures has been written by Steinberger and Ježek (2012). We will thus not attempt to go through every measure here, but rather highlight the more significant efforts in this area.

Besides ROUGE, Basic Elements (BE) (Hovy et al., 2005) has also been used in the DUC/TAC shared task evaluations. It is an automatic method which evaluates the content completeness of a generated summary by breaking up sentences into smaller, more granular units of information (referred to as "Basic Elements").

The pyramid method originally proposed by Passonneau et al. (2005) is another staple in DUC/TAC. However it is a semi-automated method, where significant human intervention is required to identify units of information, called *Summary Content Units* (SCUs), and then to map content within generated summaries to these SCUs. Recently however, an automated variant of this method has been proposed (Passonneau et al., 2013). In this variant, word embeddings are used, as we are proposing in this paper, to map text content within generated summaries to SCUs. However the SCUs still need to be manually identified, limiting this variant's scalability and applicability.

Many systems have also been proposed in the AESOP task in TAC from 2009 to 2011. For example, the top system reported in Owczarzak and Dang (2011), AutoSummENG (Giannakopoulos and Karkaletsis, 2009), is a graph-based system which scores summaries based on the similarity between the graph structures of the generated summaries and model summaries.

3 Methodology

Let us now describe our proposal to integrate word embeddings into ROUGE in greater detail.

To start off, we will first describe the word embeddings that we intend to adopt. A word embedding is really a function W, where $W : w \rightarrow \mathbb{R}^n$, and w is a word or word sequence. For our purpose, we want W to map two words w_1 and w_2 such that their respective projections are closer to each other if the words are semantically similar, and further apart if they are not. Mikolov et al. (2013b) describe one such variant, called `word2vec`, which gives us this desired property[2]. We will thus be making use of `word2vec`.

We will now explain how word embeddings can be incorporated into ROUGE. There are several variants of ROUGE, of which ROUGE-1, ROUGE-2, and ROUGE-SU4 have often been used. This is because they have been found to correlate well with human judgements (Lin, 2004a; Over and Yen, 2004; Owczarzak and Dang, 2011). ROUGE-1 measures the amount of unigram overlap between model summaries and automatic summaries, and ROUGE-2 measures the amount of bigram overlap. ROUGE-SU4 measures the amount of overlap of skip-bigrams, which are pairs of words in the same order as they appear in a sentence. In each of these variants, overlap is computed by matching the lexical form of the words within the target pieces of text. Formally, we can define this as a similarity function f_R such that:

$$f_R(w_1, w_2) = \begin{cases} 1, & \text{if } w_1 = w_2 \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

where w_1 and w_2 are the words (could be unigrams or n-grams) being compared.

In our proposal[3], which we will refer to as ROUGE-WE, we define a new similarity function f_{WE} such that:

$$f_{WE}(w_1, w_2) = \begin{cases} 0, \text{ if } v_1 \text{or } v_2 \text{ are OOV} \\ v_1 \cdot v_2, \text{ otherwise} \end{cases} \quad (2)$$

where w_1 and w_2 are the words being compared, and $v_x = W(w_x)$. *OOV* here means a situation

[2]The effectiveness of the learnt mapping is such that we can now compute analogies such as *king* $-$ *man* $+$ *woman* $=$ *queen*.

[3]`https://github.com/ng-j-p/rouge-we`

where we encounter a word w that our word embedding function W returns no vector for. For the purpose of this work, we make use of a set of 3 million pre-trained vector mappings[4] trained from part of Google's news dataset (Mikolov et al., 2013a) for W.

Reducing OOV terms for n-grams. With our formulation for f_{WE}, we are able to compute variants of ROUGE-WE that correspond to those of ROUGE, including ROUGE-WE-1, ROUGE-WE-2, and ROUGE-WE-SU4. However, despite the large number of vector mappings that we have, there will still be a large number of OOV terms in the case of ROUGE-WE-2 and ROUGE-WE-SU4, where the basic units of comparison are bigrams.

To solve this problem, we can compose individual word embeddings together. We follow the simple multiplicative approach described by Mitchell and Lapata (2008), where individual vectors of constituent tokens are multiplied together to produce the vector for a n-gram, *i.e.*,

$$W(w) = W(w_1) \times \ldots \times W(w_n) \quad (3)$$

where w is a n-gram composed of individual word tokens, *i.e.*, $w = w_1 w_2 \ldots w_n$. Multiplication between two vectors $W(w_i) = \{v_{i1}, \ldots, v_{ik}\}$ and $W(w_j) = \{v_{j1}, \ldots, v_{jk}\}$ in this case is defined as:

$$\{v_{i1} \times v_{j1}, \ldots, v_{ik} \times v_{jk}\} \quad (4)$$

4 Experiments

4.1 Dataset and Metrics

For our experiments, we make use of the dataset used in AESOP (Owczarzak and Dang, 2011), and the corresponding correlation measures.

For clarity, let us first describe the dataset used in the main TAC summarization task. The main summarization dataset consists of 44 topics, each of which is associated with a set of 10 documents. There are also four human-curated model summaries for each of these topics. Each of the 51 participating systems generated a summary for each of these topics. These automatically generated summaries, together with the human-curated model summaries, then form the basis of the dataset for AESOP.

To assess how effective an automatic evaluation system is, the system is first tasked to assign a score for each of the summaries generated by all of the 51 participating systems. Each of these summaries would also have been assessed by human judges using these three key metrics:

Pyramid. As reviewed in Section 2, this is a semi-automated measure described in Passonneau et al. (2005).

Responsiveness. Human judges are tasked to evaluate how well a summary adheres to the information requested, as well as the linguistic quality of the generated summary.

Readability. Human judges give their judgement on how fluent and readable a summary is.

The evaluation system's scores are then tested to see how well they correlate with the human assessments. The correlation is evaluated with a set of three metrics, including 1) Pearson correlation (P), 2) Spearman rank coefficient (S), and 3) Kendall rank coefficient (K).

4.2 Results

We evaluate three different variants of our proposal, ROUGE-WE-1, ROUGE-WE-2, and ROUGE-WE-SU4, against their corresponding variants of ROUGE (*i.e.*, ROUGE-1, ROUGE-2, ROUGE-SU4). It is worth noting here that in AESOP in 2011, ROUGE-SU4 was shown to correlate very well with human judgements, especially for pyramid and responsiveness, and out-performs most of the participating systems.

Tables 1, 2, and 3 show the correlation of the scores produced by each variant of ROUGE-WE with human assessed scores for pyramid, responsiveness, and readability respectively. The tables also show the correlations achieved by ROUGE-1, ROUGE-2, and ROUGE-SU4. The best result for each column has been bolded for readability.

Measure	P	S	K
ROUGE-WE-1	0.9492	**0.9138**	**0.7534**
ROUGE-WE-2	0.9765	0.8984	0.7439
ROUGE-WE-SU4	0.9783	0.8808	0.7198
ROUGE-1	0.9661	0.9085	0.7466
ROUGE-2	0.9606	0.8943	0.7450
ROUGE-SU4	**0.9806**	0.8935	0.7371

Table 1: Correlation with pyramid scores, measured with Pearson r (P), Spearman ρ (S), and Kendall τ (K) coefficients.

ROUGE-WE-1 is observed to correlate very well with the pyramid, responsiveness, and read-

[4] https://drive.google.com/file/d/0B7XkCwpI5KDYNlNUTTlSS21pQmM/edit?usp=sharing

Measure	P	S	K
ROUGE-WE-1	0.9155	**0.8192**	0.6308
ROUGE-WE-2	0.9534	0.7974	0.6149
ROUGE-WE-SU4	0.9538	0.7872	0.5969
ROUGE-1	0.9349	0.8182	**0.6334**
ROUGE-2	0.9416	0.7897	0.6096
ROUGE-SU4	**0.9545**	0.7902	0.6017

Table 2: Correlation with responsiveness scores, measured with Pearson r (P), Spearman ρ (S), and Kendall τ (K) coefficients.

Measure	P	S	K
ROUGE-WE-1	0.7846	**0.4312**	**0.3216**
ROUGE-WE-2	0.7819	0.4141	0.3042
ROUGE-WE-SU4	**0.7931**	0.4068	0.3020
ROUGE-1	0.7900	0.3914	0.2846
ROUGE-2	0.7524	0.3975	0.2925
ROUGE-SU4	0.7840	0.3953	0.2925

Table 3: Correlation with readability scores, measured with Pearson r (P), Spearman ρ (S), and Kendall τ (K) coefficients.

ability scores when measured with the Spearman and Kendall rank correlation. However, ROUGE-SU4 correlates better with human assessments for the Pearson correlation. The key difference between the Pearson correlation and Spearman/Kendall rank correlation, is that the former assumes that the variables being tested are normally distributed. It also further assumes that the variables are linearly related to each other. The latter two measures are however non-parametric and make no assumptions about the distribution of the variables being tested. We argue that the assumptions made by the Pearson correlation may be too constraining, given that any two independent evaluation systems may not exhibit linearity.

Looking at the two bigram based variants, ROUGE-WE-2 and ROUGE-WE-SU4, we observe that ROUGE-WE-2 improves on ROUGE-2 most of the time, regardless of the correlation metric used. This lends further support to our proposal to use word embeddings with ROUGE.

However ROUGE-WE-SU4 is only better than ROUGE-SU4 when evaluating readability. It does consistently worse than ROUGE-SU4 for pyramid and responsiveness. The reason for this is likely due to how we have chosen to compose unigram word vectors into bigram equivalents. The multiplicative approach that we have taken worked better for ROUGE-WE-2 which looks at contiguous bigrams. These are easier to interpret semantically than skip-bigrams (the target of ROUGE-WE-SU4). The latter, by nature of their construction, loses some of the semantic meaning attached to each word, and thus may not be as amenable to the linear composition of word vectors.

Owczarzak and Dang (2011) reports only the results of the top systems in AESOP in terms of Pearson's correlation. To get a more complete picture of the usefulness of our proposal, it will be instructive to also compare it against the other top systems in AESOP, when measured with the Spearman/Kendall correlations. We show in Table 4 the top three systems which correlate best with the pyramid score when measured with the Spearman rank coefficient. `C_S_IIITH3` (Kumar et al., 2011) is a graph-based system which assess summaries based on differences in word co-locations between generated summaries and model summaries. `BE-HM` (baseline by the organizers of the AESOP task) is the BE system (Hovy et al., 2005), where basic elements are identified using a head-modifier criterion on parse results from Minipar. Lastly, `catolicasc1` (de Oliveira, 2011) is also a graph-based system which frames the summary evaluation problem as a maximum bipartite graph matching problem.

Measure	S	K
ROUGE-WE-1	**0.9138**	0.7534
C_S_IIITH3	0.9033	**0.7582**
BE-HM	0.9030	0.7456
catolicasc1	0.9017	0.7351

Table 4: Correlation with pyramid scores of top systems in AESOP 2011, measured with the Spearman ρ (S), and Kendall τ (K) coefficients.

We see that ROUGE-WE-1 displays better correlations with pyramid scores than the top system in AESOP 2011 (*i.e.*, `C_S_IIITH3`) when measured with the Spearman coefficient. The latter does slightly better however for the Kendall coefficient. This observation further validates that our proposal is an effective enhancement to ROUGE.

5 Conclusion

We proposed an enhancement to the popular ROUGE metric in this work, ROUGE-WE.

ROUGE is biased towards identifying lexical similarity when assessing the quality of a generated summary. We improve on this by incorporating the use of word embeddings. This enhancement allows us to go beyond surface lexicographic matches, and capture instead the semantic similarities between words used in a generated summary and a human-written model summary. Experimenting on the TAC AESOP dataset, we show that this proposal exhibits very good correlations with human assessments, measured with the Spearman and Kendall rank coefficients. In particular, ROUGE-WE-1 outperforms leading state-of-the-art systems consistently.

Looking ahead, we want to continue building on this work. One area to improve on is the use of a more inclusive evaluation dataset. The AESOP summaries that we have used in our experiments are drawn from systems participating in the TAC summarization task, where there is a strong exhibited bias towards extractive summarizers. It will be helpful to enlarge this set of summaries to include output from summarizers which carry out substantial paraphrasing (Li et al., 2013; Ng et al., 2014; Liu et al., 2015).

Another immediate goal is to study the use of better compositional embedding models. The generalization of unigram word embeddings into bigrams (or phrases), is still an open problem (Yin and Schütze, 2014; Yu et al., 2014). A better compositional embedding model than the one that we adopted in this work should help us improve the results achieved by bigram variants of ROUGE-WE, especially ROUGE-WE-SU4. This is important because earlier works have demonstrated the value of using skip-bigrams for summarization evaluation.

An effective and accurate automatic evaluation measure will be a big boon to our quest for better text summarization systems. Word embeddings add a promising dimension to summarization evaluation, and we hope to expand on the work we have shared to further realize its potential.

References

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A Neural Probabilistic Language Model. *The Journal of Machine Learning Research*, 3:1137–1155.

Hoa Trang Dang and Karolina Owczarzak. 2009. Overview of the TAC 2009 Summarization Track. In *Proceedings of the Text Analysis Conference (TAC)*.

Paulo C. F. de Oliveira. 2011. CatolicaSC at TAC 2011. In *Proceedings of the Text Analysis Conference (TAC)*.

George Giannakopoulos and Vangelis Karkaletsis. 2009. AutoSummENG and MeMoG in Evaluating Guided Summaries. In *Proceedings of the Text Analysis Conference (TAC)*.

Eduard Hovy, Chin-Yew Lin, and Liang Zhou. 2005. Evaluating DUC 2005 using Basic Elements. In *Proceedings of the Document Understanding Conference (DUC)*.

Niraj Kumar, Kannan Srinathan, and Vasudeva Varma. 2011. Using Unsupervised System with Least Linguistic Features for TAC-AESOP Task. In *Proceedings of the Text Analysis Conference (TAC)*.

Chen Li, Fei Liu, Fuliang Weng, and Yang Liu. 2013. Document Summarization via Guided Sentence Compression. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 490–500.

Chin-Yew Lin. 2004a. Looking for a Few Good Metrics: ROUGE and its Evaluation. In *Working Notes of the 4th NTCIR Workshop Meeting*.

Chin-Yew Lin. 2004b. ROUGE: A Package for Automatic Evaluation of Summaries. In *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*.

Fei Liu, Jeffrey Flanigan, Sam Thomson, Norman Sadeh, and Noah A. Smith. 2015. Toward Abstractive Summarization Using Semantic Representations. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 1077–1086.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013a. Distributed Representations of Words and Phrases and their Compositionality. In *Proceedings of the 27th Annual Conference on Neural Information Processing Systems (NIPS)*, pages 3111–3119.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013b. Linguistic Regularities in Continuous Space Word Representations. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 746–751.

Jeff Mitchell and Mirella Lapata. 2008. Vector-based Models of Semantic Composition. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL)*, pages 236–244.

Jun-Ping Ng, Yan Chen, Min-Yen Kan, and Zhoujun Li. 2014. Exploiting Timelines to Enhance Multi-document Summarization. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 923–933.

Paul Over and James Yen. 2004. An Introduction to DUC 2004 Intrinsic Evaluation of Generic New Text Summarization Systems. In *Proceedings of the Document Understanding Conference (DUC)*.

Karolina Owczarzak and Hoa Trang Dang. 2011. Overview of the TAC 2011 Summarization Track: Guided Task and AESOP Task. In *Proceedings of the Text Analysis Conference (TAC)*.

Karolina Owczarzak. 2010. Overview of the TAC 2010 Summarization Track. In *Proceedings of the Text Analysis Conference (TAC)*.

Rebecca J Passonneau, Ani Nenkova, Kathleen McKeown, and Sergey Sigelman. 2005. Applying the Pyramid Method in DUC 2005. In *Proceedings of the Document Understanding Conference (DUC)*.

Rebecca J Passonneau, Emily Chen, Weiwei Guo, and Dolores Perin. 2013. Automated Pyramid Scoring of Summaries using Distributional Semantics. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 143–147.

Josef Steinberger and Karel Ježek. 2012. Evaluation Measures for Text Summarization. *Computing and Informatics*, 28(2):251–275.

Wenpeng Yin and Hinrich Schütze. 2014. An Exploration of Embeddings for Generalized Phrases. In *Proceedings of the ACL 2014 Student Research Workshop*, pages 41–47.

Mo Yu, Matthew Gormley, and Mark Dredze. 2014. Factor-based Compositional Embedding Models. In *Proceedings of the NIPS 2014 Deep Learning and Representation Learning Workshop*.

Krimping texts for better summarization

Marina Litvak
Sami Shamoon College of Engineering,
Beer Sheva, Israel
marinal@sce.ac.il

Natalia Vanetik
Sami Shamoon College of Engineering,
Beer Sheva, Israel
natalyav@sce.ac.il

Mark Last
Ben-Gurion University of the Negev,
Beer Sheva, Israel
mlast@bgu.ac.il

Abstract

Automated text summarization is aimed at extracting essential information from original text and presenting it in a minimal, often predefined, number of words. In this paper, we introduce a new approach for unsupervised extractive summarization, based on the Minimum Description Length (MDL) principle, using the Krimp dataset compression algorithm (Vreeken et al., 2011). Our approach represents a text as a transactional dataset, with sentences as transactions, and then describes it by itemsets that stand for frequent sequences of words. The summary is then compiled from sentences that compress (and as such, best describe) the document. The problem of summarization is reduced to the maximal coverage, following the assumption that a summary that best describes the original text, should cover most of the word sequences describing the document. We solve it by a greedy algorithm and present the evaluation results.

1 Introduction

Many unsupervised approaches for extractive text summarization follow the maximal coverage principle (Takamura and Okumura, 2009; Gillick and Favre, 2009), where the extract that maximally covers the information contained in the source text, is selected. Since the exhaustive solution demands an exponential number of tests, approximation techniques, such as a greedy approach or a global optimization of a target function, are utilized. It is quite common to measure text informativeness by the frequency of its components– words, phrases, concepts, and so on.

A different approach that received much less attention is based on the Minimum Description Length (MDL) principle, defining the best summary as the one that leads to the *best compression* of the text by providing its *shortest and most concise description*. The MDL principle is widely useful in compression techniques of non-textual data, such as summarization of query results for OLAP applications. (Lakshmanan et al., 2002; Bu et al., 2005) However, only a few works on text summarization using MDL can be found in the literature. Authors of (Nomoto and Matsumoto, 2001) used K-means clustering extended with MDL principle for finding diverse topics in the summarized text. Nomoto in (2004) also extended the C4.5 classifier with MDL for learning rhetorical relations. In (Nguyen et al., 2015) the problem of micro-review summarization is formulated within the MDL framework, where the authors view the tips as being encoded by snippets, and seek to find a collection of snippets that produce the encoding with the minimum number of bits.

This paper introduces a new MDL-based approach for extracting relevant sentences into a summary. The approach represents documents as a sequential transactional dataset and then compresses it by replacing frequent sequences of words by codes. The summary is then compiled from sentences that best compress (or describe) the document content. The intuition behind this approach says that a summary that best describes the original text should cover its most frequent word sequences. As such, the problem of summarization is very naturally reduced to the maximal coverage problem. We solve it by the greedy method which ranks sentences by their coverage of best compressing frequent word sequences and selects the top-ranked sentences to a summary. There are a few works that applied the common data mining techniques for calculating frequent itemsets from transactional data to the text summarization task (Baralis et al., 2012; Agarwal et al., 2011; Dalal and Zaveri, 2013), but none of them followed the MDL principle. The comparative results on three different corpora

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1931–1935,
Lisbon, Portugal, 17-21 September 2015.

show that our approach outperforms other unsupervised state-of-the-art summarizers.

2 Methodology

The proposed summarization methodology, denoted by Gamp[1], is based on the MDL principle that is defined formally as follows (Mitchell, 1997). Given a set of models $\mathcal{M}$, a model $M \in \mathcal{M}$ is considered the *best* if it minimizes $L(M) + L(D|M)$ where $L(M)$ is the bit length of the description of M and $L(D|M)$ is the bit length of the dataset D encoded with M.

In our approach, we first represent an input text as a transactional dataset. Then, using the Krimp dataset compression algorithm (Vreeken et al., 2011), we build the MDL for this dataset using its frequent sequential itemsets (word sequences). The sentences that cover most frequent word sequences are chosen to a summary. The following subsections describe our methodology in detail.

2.1 Problem statement

We are given a single text or a collection of texts about the same topic, composed of a set of sentences $S_1, \ldots, S_n$ over terms $t_1, \ldots, t_m$. The word limit W is defined for the final summary.

We represent a text as a *sequential transactional dataset*. Such a dataset consists of *transactions* (sentences), denoted by $T_1, \ldots, T_n$, and unique items (terms[2]) $I_1, \ldots, I_m$. Items are unique across the entire dataset. The number n of transactions is called the *size* of a dataset. Transaction T_i is a sequence of items from $I_1, \ldots, I_m$, denoted by $T_i = (I_{i_1}, \ldots, I_{i_k})$; the same item may appear in different places within the same transaction. *Support* of an item sequence s in the dataset is the ratio of transactions containing it as a subsequence to the dataset size n, i.e., $supp(s) = \frac{|\{T \in D | s \subseteq T\}|}{n}$. Given a support bound $Supp \in [0, 1]$, a sequence s is called *frequent* if $supp(s) \geq Supp$.

According to the MDL principle, we are interested in the minimal size of a compressed dataset $D|CT$ after frequent sequences in D are encoded with the compressing set–codes from the *Coding Table* CT, where shorter codes are assigned to more frequent sequences. The description length of non-encoded terms is assumed proportional to their length (number of characters). We rank sentences by their coverage of the best compressing set, which is the number of CT members in the sentence. The sentences with the highest coverage score are added to the summary as long as its length does not exceed W.

2.2 Krimping text

The purpose of the Krimp algorithm (Vreeken et al., 2011) is to use frequent sets (or sequences) to compress a transactional database in order to achieve MDL for that database. Let $FreqSeq$ be the set of all frequent sequences in the database. A collection CT of sequences from $FreqSeq$ (called the *Coding Table*) is called *best* when it minimizes $L(CT) + L(D|CT)$. We are interested in both *exact* and *inexact* sequences, allowing a sequence to have gaps inside it as long as the ratio of sequence length to sequence length with gaps does not exceed a pre-set parameter $Gap \in (0, 1]$. Sequences with gaps make sense in text data, as phrasing of the same fact or entity in different sentences may differ. In order to encode the database, every member $s \in CT$ is associated with its binary *prefix code* c (such as Huffman codes for 4 members: 0, 10, 110, 111), so that the most frequent code has the shortest length. We use an upper bound C on the number of encoded sequences in the coding table CT, in order to limit document compression. Krimp-based extractive summarization (see Algorithm 1) is given a document D with n sentences and m unique terms. The algorithm parameters are described in Table 1:

#	*note*	*description*	*affects*
1	**W**	summary words limit	summary length
2	**Supp**	minimal support bound – minimal fraction of sentences containing a frequent sequence	number of frequent word sequences $\|FreqSeq\|$, compression rate
3	**C**	maximal number of codes	as in 2
4	**Gap**	maximal allowed sequence gap ratio	as in 2

Table 1: Parameters of Algorithm 1.

The algorithm consists of the following steps.

1. We find all frequent term sequences in the document using Apriori-TID algorithm (R and R, 1994) for the given $Supp$ and Gap and store them in set $FreqSeq$, which is kept in Standard Candidate Order[3]. The coding table CT is initialized to contain all single normalized terms and their frequencies. CT is always kept in Standard Cover Order[4] (Steps

[1] abbreviation of two words: GAp and kriMP

[2] normalized words following tokenization, stemming, and stop-word removal

[3] first, sorted by increasing support, then by decreasing sequence length, then lexicographically

[4] first, sorted by decreasing sequence length, then by decreasing support, and finally, in lexicographical order

1 and 2 in Algorithm 1).

2. We repeatedly choose frequent sequences from the set $FreqSeq$ so that the size of the encoded dataset is minimal, with every selected sequence replaced by its code. Selection is done by computing the decrease in the size of the encoding when each one of the sequences is considered to be a candidate to be added to CT (Step 3 in Algorithm 1).

3. The summary is constructed by incrementally adding sentences with the highest coverage of encoded term sequences (Step 4 in Algorithm 1) that are not covered by previously selected sentences. The sentences are selected in the greedy manner as long as the word limit W is not exceeded.

Algorithm 1: Gamp: Krimp-based extractive summarization with gaps

Input: (1) A document, containing sentences $S_1, \ldots, S_n$ after tokenization, stemming and stop-word removal;
(2) normalized terms $T_1, \ldots, T_m$;
(3) summary size W
(3) limit C on the number of codes to use;
(4) minimal support bound $Supp$;
(5) maximal gap ratio Gap.
Output: Extractive summary $Summary$
/ STEP 1: Frequent sequence mining */*
$FreqSeq \leftarrow$ inexact frequent sequences of terms from $\{T_1, \ldots, T_m\}$ appearing in at least $Supp$ fraction of sentences and having a gap ratio of at least Gap;
Sort $FreqSeq$ according to Standard Candidate Order;
/ STEP 2: Initialize the coding table */*
Add all terms $T_1, \ldots, T_m$ and their support to CT;
Keep CT always sorted according to Standard Cover Order;
Initialize prefix codes according to the order of sequences in CT; */* STEP 3: Find the best encoding */*
$EncodedData \leftarrow$ *PrefixEncoding*$(\{S_1, \ldots, S_n\}, CT)$;
$CodeCount \leftarrow 0$;
while $CodeCount < B$ *and* $FreqSeq \neq \emptyset$ **do**
  $BestCode \leftarrow c \in FreqSeq$ such that $L(CT \cup \{c\}) + L($*PrefixEncoding*$(\{S_1, \ldots, S_n\}, CT \cup \{c\}))$ is minimal;
  $CT \leftarrow CT \cup \{BestCode\}$;
  $FreqSeq \leftarrow FreqSeq \setminus \{BestCode\}$;
  Remove supersets of $BestCode$ from $FreqSeq$;
  $CodeCount$++;
end
$Summary \leftarrow \emptyset$; */* STEP 4: Build the summary */*
while $\#words(Summary) < W$ **do**
  Find the sentence S_i that covers the largest set $\mathcal{T}$ of terms in CT and add it to $Summary$;
  Remove terms of $\mathcal{T}$ from CT;
end
return $Summary$

Example 2.1 Let dataset D contain following three sentences (taken from the "House of cards" TV show):

S_1 = *A hunter must stalk his prey until the hunter becomes the hunted.*
S_2 = *Then the prey becomes the predator.*
S_3 = *Then the predator and the hunter fight.*

After stemming, tokenization and stop-word removal we obtain unique (stemmed) terms:

t_1 = *hunter*, t_2 = *must*, t_3 = *stalk*, t_4 = *prei*,
t_5 = *becom*, t_6 = *hunt*, t_7 = *predat*, t_8 = *fight*.
$supp(t_1) = supp(t_4) = supp(t_5) = supp(t_7) = \frac{2}{3}$.
$supp(t_2) = supp(t_3) = supp(t_8) = \frac{1}{3}$.

Now we can view sentences as the following sequences of normalized terms.

$S_1 = (t_1, t_2, t_3, t_4, t_1, t_5, t_4)$
$S_2 = (t_4, t_5, t_7)$
$S_3 = (t_7, t_1, t_8)$

Initial coding table CT will contain all frequent single terms in Standard Cover Order: $t_5, t_1, t_7, t_4, t_8, t_2, t_3$. Let the minimal support bound be $\frac{2}{3}$, i.e., to be frequent a sequence must appear in at least 2 sentences, and let the gap ratio be $\frac{1}{2}$. Also, let the limit C be 4, meaning that only the first four entries of the coding table will be used for encoding. There exists a frequent sequence (t_4, t_5) that appears twice in the text, once in S_1 with a gap and once in S_2 without a gap. We add it to the coding table according to the Standard Cover Order, generate prefix codes for the first four entries, and get

$CT =$

seq	*supp*	*code*	*code len*
(**prei**, **becom**)	2/3	0	1
becom	2/3	10	2
hunter	2/3	110	3
predat	2/3	111	3
prei	2/3	–	–
...	...	–	–

Now S_1 covers 3 out of 4 entries in CT, while S_2 and S_3 cover 2 terms each. If our summary is to contain just one sentence, we select S_1.

3 Experimental settings and results

We performed experiments on three corpora from the Document Understanding Conference (DUC): 2002, 2004, and 2007 (duc, 2002 2007), summarized in Table 2. DUC 2002 and 2007 each contains a set of single documents, and DUC 2004 contains 50 sets of 10 related documents. We generated summaries per single documents in the DUC 2002 and 2007 corpora, and per each set of related documents (by considering each set of documents as one meta-document) in DUC 2004, following the corresponding length restrictions. The summarization quality was measured

corpus	# documents	# summaries	summ len	doc size (KB)	type
DUC'02	533	2-3	100	1-20	SD
DUC'04	50	4	100	20-89	MD (50x10)
DUC'07	23	4	250	28-131	SD

Table 2: Corpora statistics.

by the ROUGE-1 and ROUGE-2 (Lin, 2004) recall scores, with the word limit indicated in Table 2, without stemming and stopword removal.

The results of the introduced algorithm (Gamp) may be affected by several input parameters: minimum support ($Supp$), codes limit (C), and the maximal gap allowed between frequent words (Gap). In order to find the best algorithm settings for a general case, we performed experiments that explored the impact of these parameters on the summarization results. First, we experimented with different values of support count in the range of $[2, 10]$. The results show that we get the best summaries using the sequences that occur in at least four document sentences.

A limit on the number of codes is an additional parameter.We explored the impact of this parameter on the quality of generated summaries. As we could conclude from our experiments, the best summarization results are obtained if this parameter is set to the maximal number of words in the summary, W. Consequently, we used 100 codes for summarizing DUC 2002 and DUC 2004 documents and 250 codes for DUC 2007 documents.

The maximal gap ratio defines a pattern for generating the frequent sequences and has a direct effect on their structure and number. Our experiments showed that allowing a small gap between words of a sequence helps to improve slightly the ranking of sentences, but the improvement is not significant. Thus we used $Gap = 0.8$ in comparative experiments for all corpora. The resulting settings for each corpus are shown in Table 3.

Corpus	Supp	Max. codes	Gap
DUC 2002 and 2004	4	100	0.8
DUC 2007	4	250	0.8

Table 3: Best settings.

We compared the Gamp algorithm with the two known unsupervised state-of-the-art summarizers denoted by Gillick (Gillick and Favre, 2009) and McDonald (McDonald, 2007). As a baseline, we used a very simple approach that takes first sentences to a summary (denoted by TopK). Table 4 contains the results of comparative evaluations. The best scores are shown in bold. Gamp outperformed the other methods on all datasets (using ROUGE-1 score). The difference between the scores of Gamp and Gillick (second best system) on DUC 2007 is highly significant according to the Wilcoxon matched pairs test. Based on the same test, the difference of scores obtained on the DUC 2004 is not statistically significant. On the DUC 2002, Gamp is ranked first, with insignificant difference from the second best (McDonald's) scores. Based on this result, we can conclude that MDL-based summarization using frequent sequences works better on long documents or multi-document domain. Intuitively, it is a very logical conclusion, because single short documents do not contain a sufficient number of frequent sequences. It is noteworthy that, in addition to the greedy approach, we also evaluated the global optimization with maximizing coverage and minimizing redundancy using Linear Programming (LP). However, experimental results did not provide any improvement over the greedy approach. Therefore, we report only the results of the greedy solution.

	ROUGE-1 Recall			ROUGE-2 Recall		
Algorithm	DUC'02	DUC'04	DUC'07	DUC'02	DUC'04	DUC'07
Gamp	**0.4421**	**0.3440**	**0.3959**	0.1941	**0.0829**	**0.0942**
Gillick	0.4207	0.3314	0.3518	0.1773	0.0753	0.0650
McDonald	0.4391	0.2955	0.3500	**0.1981**	0.0556	0.0672
TopK	0.4322	0.2973	0.3525	0.1867	0.0606	0.0706

Table 4: Comparative results.

4 Conclusions

In this paper, we introduce a new approach for summarizing text documents based on their Minimal Description Length. We describe documents using frequent sequences of their words. The sentences with the highest coverage of the best compressing set are selected to a summary. The experimental results show that this approach outperforms other unsupervised state-of-the-art methods when summarizing long documents or sets of related documents. We would not recommend using our approach for summarizing single short documents which do not contain enough content for providing a high-quality description. In the future, we intend to apply the MDL method to keyword extraction, headline generation, and other related tasks.

Acknowledgments

This work was partially supported by the U.S. Department of the Navy, Office of Naval Research.

References

Nitin Agarwal, Kiran Gvr, Ravi Shankar Reddy, and Carolyn Penstein Ros. 2011. Scisumm: A multi-document summarization system for scientific articles. In *Proceedings of the ACL-HLT 2011 System Demonstrations*, pages 115–120.

Elena Baralis, Luca Cagliero, Saima Jabeen, and Alessandro Fiori. 2012. Multi-document summarization exploiting frequent itemsets. In *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, SAC '12, pages 782–786.

Shaofeng Bu, Laks V. S. Lakshmanan, and Raymond T. Ng. 2005. Mdl summarization with holes. In *Proceedings of the 31st International Conference on Very Large Data Bases*, VLDB '05, pages 433–444.

Mita K. Dalal and Mukesh A. Zaveri. 2013. Semisupervised learning based opinion summarization and classification for online product reviews. *Applied Computational Intelligence and Soft Computing*, 2013.

2002–2007. Document Understanding Conference. `http://duc.nist.gov`.

Dan Gillick and Benoit Favre. 2009. A Scalable Global Model for Summarization. In *Proceedings of the NAACL HLT Workshop on Integer Linear Programming for Natural Language Processing*, pages 10–18.

Laks V. S. Lakshmanan, Raymond T. Ng, Christine Xing Wang, Xiaodong Zhou, and Theodore J. Johnson. 2002. The generalized mdl approach for summarization. In *Proceedings of the 28th International Conference on Very Large Data Bases*, VLDB '02, pages 766–777.

Chin-Yew Lin. 2004. ROUGE: A Package for Automatic Evaluation of summaries. In *Proceedings of the Workshop on Text Summarization Branches Out (WAS 2004)*, pages 25–26.

R. McDonald. 2007. A study of global inference algorithms in multi-document summarization. In *Advances in Information Retrieval*, pages 557–564.

Thomas M. Mitchell. 1997. *Machine Learning*. McGraw-Hill, Inc., New York, NY, USA, 1 edition.

Thanh-Son Nguyen, Hady W. Lauw, and Panayiotis Tsaparas. 2015. Review synthesis for micro-review summarization. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*, WSDM '15, pages 169–178.

Tadashi Nomoto and Yuji Matsumoto. 2001. A new approach to unsupervised text summarization. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '01, pages 26–34.

Tadashi Nomoto. 2004. *Machine Learning Approaches to Rhetorical Parsing and Open-Domain Text Summarization*. Ph.D. thesis, Nara Institute of Science and Technology.

Agrawal R and Srikant R. 1994. Fast algorithms for mining association rules. In *20th International Conference on Very Large Databases*, pages 487–499.

Hiroya Takamura and Manabu Okumura. 2009. Text summarization model based on maximum coverage problem and its variant. In *EACL '09: Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 781–789.

Jilles Vreeken, Matthijs Leeuwen, and Arno Siebes. 2011. Krimp: Mining itemsets that compress. *Data Min. Knowl. Discov.*, 23(1):169–214, July.

From the Virtual to the Real World: Referring to Objects in Real-World Spatial Scenes

Dimitra Gkatzia, Verena Rieser
Department of Computer Science
Heriot-Watt University
Edinburgh EH14 4AS , UK
{d.gkatzia,v.t.rieser}@hw.ac.uk

Phil Bartie
School of Natural Sciences
University of Stirling
Stirling FK9 4LA, UK
phil.bartie@stir.ac.uk

William Mackaness
School of GeoSciences
University of Edinburgh
Edinburgh EH8 9XP, UK
william.mackaness@ed.ac.uk

Abstract

Predicting the success of referring expressions (RE) is vital for real-world applications such as navigation systems. Traditionally, research has focused on studying Referring Expression Generation (REG) in virtual, controlled environments. In this paper, we describe a novel study of spatial references from real scenes rather than virtual. First, we investigate how humans describe objects in *open, uncontrolled* scenarios and compare our findings to those reported in virtual environments. We show that REs in real-world scenarios differ significantly to those in virtual worlds. Second, we propose a novel approach to quantifying image complexity when complete annotations are not present (e.g. due to poor object recognition capabitlities), and third, we present a model for success prediction of REs for objects in real scenes. Finally, we discuss implications for Natural Language Generation (NLG) systems and future directions.

1 Introduction

REG has attracted considerable interest in the NLG community over the past 20 years (Krahmer and van Deemter, 2011; Gatt et al., 2014). While initially, the standard evaluation metric for REG was human-likeness, as compared to human corpora similarity as in TUNA (Gatt et al., 2009), the field has moved on to evaluating REG effectiveness by measuring task success in virtual interactive environments (Byron et al., 2009; Gargett et al., 2010; Janarthanam et al., 2012). Virtual environments however eliminate real-world uncertainty, such object recognition errors or cluttered scenes. In this paper, we investigate whether the lessons learnt in virtual environments can be transferred to real-world scenes. We consider the case where we are uncertain about the scene itself, i.e. we assume that the complexity of the scene is hidden and we are interested in identifying a specific object, and thus our work differs from approaches that generate descriptions for images such as (Mitchell et al., 2012; Feng and Lapata, 2013; Yang et al., 2011; Yatskar et al., 2014).

Related work has focused on computer generated objects (van Deemter et al., 2006; Viethen and Dale, 2008), crafts (Mitchell et al., 2010), or small objects in a simple background (Mitchell et al., 2013a; FitzGerald et al., 2013). One notable exception is the recent work by Kazemzadeh et al. (2014), who investigate referring expressions of objects in "complex photographs of real-world cluttered scenes". They report that REs are heavily influenced by the object type. Here, we are interested in studying REs for visual objects in urban scenes. As the success of a RE is heavily dependent on the complexity of the scene as well as its linguistic features, we are interested in modelling and thus predicting the success of a RE.

Initially, this paper presents and analyses a novel, real-world corpus REAL (to be released) – "Referring Expression Anchored Language" (Section 2), and compares the findings to those reported in virtual worlds (Gargett et al., 2010). We then provide a detailed analysis of how syntactic and semantic features contribute to the success of REs (Sections 4.1, 4.2, 4.3), accounting for unobservable latent variables, such as the complexity of the visual scene (as described in Section 3). Finally, we summarise our work and discuss the implications of our work for NLG systems (Section 5). The dataset and models will be released.

2 The REAL Corpus

The REAL corpus contains a collection of images of real-world urban scenes (Fig. 1) together with verbal descriptions of target objects (see Fig. 2)

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1936–1942,
Lisbon, Portugal, 17-21 September 2015.

Figure 1: Original picture.

Figure 2: Target object in yellow box.

Figure 3: The identified object by the validators.

generated by humans, paired with data on how successful other people were able to identify the same object based on these descriptions (Fig. 3). The data was collected through a web-based interface. The images were taken in Edinburgh (Scotland, UK), very early one summer morning. This was necessary to reduce the occlusion of city objects from buses and crowds, and to minimise lighting and weather variations between images.

2.1 Experimental Setup

There were 190 participants recruited (age between 16 to 71). Each participant was presented with an urban image (Fig. 1), where the target object was outlined by a yellow box (Fig. 2), and was asked to describe the target using free text. After completing a (self-specified) number of tasks, participants were then asked to validate descriptions provided by other participants by clicking on the object using previously unseen images (Fig. 3).

# participants	190
# images/ stimuli	32
# descriptions	868
# verifications	2618
– ambiguous	201
– not found	75
– correct	1994
– incorrect	251
– NA	7

Table 1: The REAL corpus

Overall, 868 descriptions across 32 images were collected, averaging around 27 descriptions per image. The balance of generation and validations was adjusted to ensure that all descriptions were identified by at least 3 other participants, generating 2618 image tag verifications. All cases were manually checked to determine if the 'correct' (green) or 'incorrect' (red) target had been identified Fig. 3. Overall, 76.2% of human descriptions provided were successfully identified. For the experiments reported in following sections, we summarised answers categorised as 'incorrect', 'ambitious' and 'not found' as *unsuccessful*.

2.2 Comparison to GIVE-2 Corpus

We now compare this data with human data generated for the GIVE-2 challenge (Gargett et al., 2010). In GIVE-2, the target objects have distinct attributes, such as colour and position. For instance, an effective RE in GIVE-2 could be "*the third button from the second row*". In real-world situations though, object properties are less well defined, making a finite set of pre-defined qualities unfeasible. Consider, for instance, the building highlighted in Figure 2, for which the following descriptions were collected:

1. *The Austrian looking white house with the dark wooden beams at the water side.*
2. *The white building with the x-shape balconies. It seems it's new.*
3. *The white building with the balconies by the river.*
4. *Apartments with balconies.*
5. *The nearest house on right side. It's black and white.*
6. *The white and black building on the far right, it has lots of triangles in its design.*
7. *The rightmost house with white walls and wood finishings.*

It is evident that the REAL users refer to a variety of object qualities. We observe that all participants refer to the colour of the building (*white, black and white, greyish-whitish*) and some mention location (*by the river, at the water side*).

Experimental Factors influencing Task Performance: In REAL, task success is defined as the ability to correctly identify an object, whereas in GIVE-2, task success refers to the successful completion of the navigation task. In contrast to GIVE-2, not all REAL participants were able to correctly identify the referred objects (76.2% task

	GIVE-2		REAL
	German	English	
Overall task success	100%	100%	76.2%
Task success (female)	100%	100%	78.8%
Task success (male)	100%	100%	69.6%
Length of descriptions (no. words)	5.2	4.7	16.01
Length of descriptions (female)	NA	NA	97.36
Length of descriptions (male)	NA	NA	91.38

Table 2: Descriptive statistics for GIVE-2 and REAL

success). We assume that this is because GIVE-2 was an interactive setup, where the participants were able to engage in a clarification dialogue. **Gender:** In REAL, gender was not a significant factor with respect to task success (Mann-Whitney U test, $p = 0.2$). **Length of REs (no. words):** In REAL, females tend to provide lengthier REs than males, however the difference is not statistically significant (Mann-Whitney U test, $p = 0.58$). In GIVE-2, only German females produced significantly longer descriptions than their male counterparts. **Relation between length (no. words) and task success:** The REAL data shows a positive relationship between length and success rate, i.e. for a one word increase in length, the odds of *correct* object identification is significantly increased ($p < 0.05$, Logit), i.e. longer and more complex sentences lead to more successful REs.

3 Quantifying the Image Complexity

We assume that the complexity of the urban scene represented in the image is hidden due to the lack of semantic annotations. Our dataset does not include any quantifiable image descriptions, such as computer vision output as in (Mitchell et al., 2012) or manual annotations as in (Yatskar et al., 2014). In addition, the same RE might not always result in successful identification of an object due to scene complexity. In order to marginalise the effect of the scene complexity, we exploit the multiple available data points per image. This allows us to estimate the average success rate of each referring expression $\overline{SR}_{RE}$ (the proportion of successful validations) and the average success rate of each image $\overline{SR}_i$ (the proportion of the correctly identified objects in the image). We use $\overline{SR}_i$ to marginalise over the (hidden) image complexity, where we assume that some pictures are inherently more complex than others and thus achieve lower success rates. Similar normalisation methods are used for user ratings to account for the fact that some users are more "tolerant" and in general give higher ratings (Jin and Si, 2004). We employ *Gaussian normalisation* (Resnick et al., 1994) to normalise image success rates by considering the following factors:

1. *Shift of average success rate per image:* some images are inherently easier than others and gain higher success rates, independently of the REs used. This factor can be accounted by subtracting average success rates of all images from the average rating of a specific image x.
2. *Different ratings:* there are 27 REs per image on average, some of which are harder to understand than others, thus they gain lower success rates. To account for this, the success rates of each image are divided by the overall SR variance.

The normalised image success rate (NSR_i) per image x is defined by the following equation:

$$NSR_i(x) = \frac{SR_i(x) - \overline{SR_i}}{\sqrt{\sum_n (SR_i(x) - \overline{SR_i})^2}} \quad (1)$$

Using the (NSR_i), we now investigate the REs in terms of their linguistic properties, including automatically annotated syntactic features and manually annotated semantic features.

4 Modelling REG Success

Unlike previous work, we use both successful and unsuccessful REs in order to build a model that is able to predict the success or the failure of a RE.

4.1 Syntactic Analysis of REG Success

We use the Stanford CoreNLP tool (Manning et al., 2014) to syntactically annotate the REs and we investigate which linguistic features contribute to the RE success in relation to the image complexity. Note that these analyses are based on normalised values, as discussed in Section 3).

Predicting RE Success Rate ($\overline{SR}_{RE}$): Initially, we compare successful and unsuccessful REs by taking the upper and lower quartiles and extracting their syntactic features., i.e. the top and bottom 25% of REs with respect to their average success rate, and group them into two groups. We then extract syntactic features of these two groups and compare their frequencies (occurrence per RE), means, and standard deviations (Table 3), and compare them using a t-test ($p < 0.05$). The difference between successful and unsuccessful ex-

	Successful REs			Successful REs		
	Mean	SD	Freq.	Mean	SD	Freq.
NP*	7.35	3.958	100	6.7	3.8	100
VP	1.45	1.673	41.8	1.46	1.923	58
PRN	.02	.181	2.1	.03	.193	2.4
NNP*	.57	1.131	27	.38	.918	19.3
NN*	4.2	2.284	98.8	3.79	2.441	98.1
DT	2.59	1.791	86.7	2.63	1.813	85.4
JJ*	1.92	1.61	80.9	1.66	1.288	81.1
CC	.4	.645	32.8	.31	.588	25
PP	2.52	1.754	92.3	2.54	1.778	85.8
ADJP	.2	.536	16.2	.041	.597	19.3
ADVP	.27	.538	22.8	.25	.539	19.8
RB	.34	.639	26.6	.34	.859	21.2
VBN*	.22	.465	20.3	.31	.445	10.8
NNS	.61	.902	40.7	.72	.782	53.3
CD	.27	.552	22	.25	.478	23.6

Table 3: Statistics regarding the linguistic features in successful vs unsuccessful referring expressions. (* denotes significant difference at $p < 0.05$).

pressions lies in the use of NP (Noun phrases), NNP (Proper noun, singular), NN (Noun, singular or mass), JJ (Adjective) and VBN (Verb, past participle) (Table 3). Successful REs include more NPs, including NNPs and NNs, which indicates that more than one reference is used to describe and distinguish a target object. This could mean that distractors are explicitly mentioned and eliminated or that the object of interest has a complex appearance, as opposed to simply structured objects, such as buttons, in GIVE-2. For example, the following description refers to a complex object:

> *The **large American-style wooden building** with **balcony painted cream and red/brown**. **Ground floor** is a **cafe with tables and parasols** outside.*

In addition, successful REs contain significantly more adjectives and verbs in past participle[1], which indicates that the object was further described and distinguished using its attributes, as for instance the following description:

> *Large modern **glass fronted** building, **butted up** against traditional Victorian terrace, slightly **set back** from road and with facing **bowed** frontage.*

The main difference between successful and unsuccessful REs is the amount of detail provided to describe and distinguish the target object. This is also in-line with our previous results that success is positively correlated to the number of words used (Section 2.2) and it might explain why humans overspecify.

[1]A participle is a form of a verb that is used in a sentence as modifier, and thus plays a role similar to that of an adjective or adverb, such as *built* or *worn*.

Models	R^2
Syntactic: NP+PP+ADVP+CD+length	.15
Semantic model: taxonomic + absolute	.338
Joint model: PP + taxonomic + absolute	.407

Table 4: Models and their fit.

To further verify this hypothesis, we build a predictive model of average success rate, using *multiple step-wise linear regression* with syntactic features as predictors. We find a significant ($p < 0.05$) positive relationship between success rate and NP, PP (Prepositional phrase), ADVP (Adverbial phrase), CD (Cardinal number), and length (Table 4). NPs are used to distinguish and describe the target object. ADVPs and PPs serve a similar function to adjectives in this case, i.e. to describe further attributes, especially spatial ones, like "the one near the river", "next to the yellow building". Cardinal numbers are used to refer to complex structured features of the target object, e.g. *two-story building* or *two large double doors*.

Predicting Image Success Rate ($\overline{NSR}_{image}$): We repeat a similar analysis for estimating how syntactic features relate to image success rate, i.e. how the image complexity, as estimated from the success rate of an image, influences how humans describe the target object, i.e. how human generated descriptions change with respect to the image complexity as estimated from the (normalised) success rate of an image. We find that humans use significantly more PPs and number of words ($p < 0.05$) when describing complex images.

In sum, syntactical features, which further describe and distinguish the target object (such as NPs, ADJ, and ADVPs and PPs) indicate successful REs. However, they cannot fully answer the question of "what makes a RE successful", therefore we enrich our feature set using manually annotated semantic features.

4.2 Semantic Analysis of REG

We extract semantic features by annotating spatial frames of reference as described in (Gargett et al., 2010). We annotate a sample of the corpus (100 instances), which allows us to perform a direct comparison between the two corpora.

Comparison to GIVE-2 Corpus: We observe that in the REAL corpus, the *taxonomic* property, the *relative property* and the *macro-level landmark*

Spatial Frame	REAL	GIVE-2	
		German	English
Taxonomic Property	**92***	53.66	58.51
Absolute Property	**57***	85.37	92.53
Relative Property	**15***	6.83	4.56
Viewer-centred	15	15.61	12.45
Micro-level landmark intrinsic	**9***	13.17	17.84
Distractor Intrinsic	**5***	10.73	14.11
Macro-level landmark intrinsic	**43***	6.83	4.15
Deduction by elimination	1	0.98	3.32

Table 5: Frequency of semantic frames in REAL vs. GIVE-2 (* denotes significant differences at $p < 0.05$, χ^2 test).

intrinsic property of the object in question are used significantly more often than in the GIVE-2 corpus (Table 5)[2].

In contrast, in GIVE-2 the *absolute property* of the object, such as the colour, and references to *distractors* are used significantly more often than in REAL. These results reflect the fact that scenes in REAL were more complex, and as such, relative properties to other objects and landmarks were used more often. In GIVE-2, target objects were mostly buttons, therefore, absolute descriptions (*"the blue button"*) or referring to an intrinsic distractor (*"the red button next to the green"*) are more frequent. In addition, real-world environments are dynamic. Humans choose to refer to immovable objects (*macro-level landmarks*) more often than in closed-world environments. In GIVE-2, immovable objects are limited to walls, ceilings or floors, whereas in REAL there is a wide range of immovable objects /landmarks that a user can refer to, e.g. another building, rivers, parks, shops, etc. Landmark descriptions will play an important role in future navigation systems (Kandangath and Tu, 2015).

Predicting RE Success Rate ($\overline{SR}_{RE}$): Next, we analyse which spatial frames significantly contribute to task success, using *multiple step-wise linear regression*.We find that *taxonomic* and *absolute* properties significantly ($p < 0.05$) contribute to the success of a referring expression (Table 4). Semantic features explain more of the variance observed in $\overline{SR}_{RE}$, than syntactic features.

[2]Note that for GIVE-2 we consider both, the German and the English data.

4.3 Joint Model of REG Success

Both syntactic and semantic features contribute to the success of a RE. Therefore, we construct a joint model for predicting $\overline{SR}_{RE}$ using *step-wise linear regression* over the joint feature space. We find that both syntactic and semantic features significantly ($p < 0.05$) contribute to the success of a RE, see Table 4. This model explains almost half of the variation observed in $\overline{SR}_{RE}$ ($R^2 = .407$). Clarke et al. (2013) reports an influence of visual salience on REG, therefore, in future, we will investigate the influence of visual features.

5 Discussion and Conclusions

From the results presented, the following conclusions can be drawn for real-world NLG systems. Firstly, semantic features have a bigger impact on the success rate of REs than syntactic features, i.e. content selection is more important than surface realisation for REG. Secondly, semantic features such as *taxonomic* and *absolute* properties can significantly contribute to RE success. Taxonomic properties refer to the type of target object, and in general depend on the local knowledge of the information giver. Similarly, the success of the RE will depend on the expertise of the information follower. As such, modelling the user's level of knowledge (Janarthenam et al., 2011) and stylistic differences (Di Fabbrizio et al., 2008) is crucial. Absolute properties refer to object attributes, such as colour. Attribute selection for REG has attracted a considerable amount of attention, therefore it would be interesting to investigate how these automatic attribute selection algorithms perform in real-world, interactive environments. Finally, the more complex scenes seem to justify longer and more complex descriptions. As such, there is an underlying trade-off which needs to be optimised, e.g. following the generation framework described in (Rieser et al., 2014).

In future, we will compare existing REG algorithms on our dataset, in a similar experiment to Mitchell et al. (2013b). Then, we will extend existing algorithms to take into account other properties such as material (e.g. "wooden"), components of the referred object (e.g. "balconies") etc. Finally, we will incorparate such an algorithm in interactive settings to investigate the influence of user dialogue behaviour and the influence of visual features, such as salience (Clarke et al., 2013), in order to improve the fit of our predictive model.

Acknowledgments

This research received funding from the EPSRC GUI project Generation for Uncertain Information (EP/L026775/1). The data has been collected through the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 216594 (SPACEBOOK project).

References

Donna Byron, Alexander Koller, Kristina Striegnitz, Justine Cassell, Robert Dale, Johanna Moore, and Jon Oberlander. 2009. Report on the First NLG Challenge on Generating Instructions in Virtual Environment (GIVE). In *12th European Workshop in Natural Language Generation (ENLG)*.

Alasdair D.F. Clarke, Micha Elsner, and Hannah Rohde. 2013. Where's wally: The influence of visual salience on referring expression generation. *Frontiers in Psychology*, 4(329).

Giuseppe Di Fabbrizio, Amanda Stent, and Srinivas Bangalore. 2008. Referring expression generation using speaker-based attribute selection and trainable realization. In *5th International Natural Language Generation Conference (INLG)*.

Yansong Feng and Mirella Lapata. 2013. Automatic caption generation for news images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(4):797–812.

Nicholas FitzGerald, Yoan Artzi, and Luke Zettlemoyer. 2013. Learning distributions over logical forms for referring expression generation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Andrew Gargett, Konstantina Garoufi, Alexander Koller, and Kristina Striegnitz. 2010. The GIVE-2 Corpus of Giving Instructions in Virtual Environments. In *7th International Conference on Language Resources and Evaluation (LREC)*.

Albert Gatt, Anja Belz, and Eric Kow. 2009. The TUNA-REG Challenge 2009: Overview and Evaluation Results. In *12th European Workshop in Natural Language Generation (ENLG)*.

Albert Gatt, Emiel Krahmer, and Kees van Deemter. 2014. Models and empirical data for the production of referring expressions. *Language, Cognition and Neuroscience*, 29(8):899 – 911.

Srinivasan Janarthanam, Xingkun Liu, and Oliver Lemon. 2012. A web-based evaluation framework for spatial instruction-giving systems. In *Proc. of Annual Meeting of the Association for Computational Linguistics (ACL)*.

Srinivasan Janarthenam, Helen Hastie, Oliver Lemon, and Xingkun Liu. 2011. "the day after the day after tomorrow?" a machine learning approach to adaptive temporal expression generation: training and evaluation with real users. In *12th Annual Meeting of the Special Interest Group on Discourse and Dialogue*.

Rong Jin and Luo Si. 2004. A study of methods for normalizing user ratings in collaborative filtering. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '04, pages 568–569, New York, NY, USA. ACM.

Anil Kandangath and Xiaoyuan Tu. 2015. Humanized navigation instructions for mapping applications. US Patent, 04.

Sahar Kazemzadeh, Vicente Ordonez, Mark Matten, and Tamara Berg. 2014. ReferItGame: Referring to Objects in Photographs of Natural Scenes. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Emiel Krahmer and Kees van Deemter. 2011. Computational generation of referring expressions: A survey. *Computational Linguistics*, 38(1):173–218, 2015/02/20.

Christopher Manning, Mihai Surdeanu, John Finkel, Jenny Bethard, and David McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In *52nd Annual Meeting of the Association for Computational Linguistics (ACL): System Demonstrations*.

Margaret Mitchell, Kees van Deemter, and Ehud Reiter. 2010. Natural reference to objects in a visual domain. In *6th International Natural Language Generation Conference (INLG)*.

Margaret Mitchell, Xufeng Han, Jesse Dodge, Alyssa Mensch, Amit Goyal, Alex Berg, Kota Yamaguchi, Tamara Berg, Karl Stratos, and Hal Daumé, III. 2012. Midge: Generating image descriptions from computer vision detections. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, EACL '12, pages 747–756.

Margaret Mitchell, Ehud Reiter, and Kees van Deemter. 2013a. Typicality and object reference. In *Cognitive Science (CogSci)*.

Margaret Mitchell, Kees van Deemter, and Ehud Reiter. 2013b. Generating expressions that refer to visible objects. In *North American Chapter of the Association for Computational Linguistics (NAACL)*.

Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstorm, and John Riedl. 1994. GroupLens: an open architecture for collaborative filtering of netnews. In *ACM Conference on Computer Supported Cooperative Work (CSCW)*.

Verena Rieser, Oliver Lemon, and Simon Keizer. 2014. Natural language generation as incremental planning under uncertainty: Adaptive information presentation for statistical dialogue systems. *Audio, Speech, and Language Processing, IEEE/ACM Transactions on*, 22(5):979–994, May.

Kees van Deemter, Ielka van der Sluis, and Albert Gatt. 2006. Building a semantically transparent corpus for the generation of referring expressions. In *4th International Natutral Language Generation Conference*.

Jette Viethen and Robert Dale. 2008. The use of spatial relations in referring expression generation. In *5th International Natural Language Generation Conference (INLG)*.

Yezhou Yang, Ching Lik Teo, Hal Daumé, III, and Yannis Aloimonos. 2011. Corpus-guided sentence generation of natural images. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Mark Yatskar, Michel Galley, Lucy Vanderwende, and Luke Zettlemoyer. 2014. See no evil, say no evil: Description generation from densely labeled images. In *Proceedings of the Third Joint Conference on Lexical and Computational Semantics (*SEM 2014)*.

An Unsupervised Bayesian Modelling Approach to Storyline Detection from News Articles

Deyu Zhou[†‡] **Haiyang Xu**[†] **Yulan He**[§]
[†] School of Computer Science and Engineering, Southeast University, China
[‡] State Key Laboratory for Novel Software Technology, Nanjing University, China
[§] School of Engineering and Applied Science, Aston University, UK
d.zhou@seu.edu.cn, h.xu@seu.edu.cn, y.he@cantab.net

Abstract

Storyline detection from news articles aims at summarizing events described under a certain news topic and revealing how those events evolve over time. It is a difficult task because it requires first the detection of events from news articles published in different time periods and then the construction of storylines by linking events into coherent news stories. Moreover, each storyline has different hierarchical structures which are dependent across epochs. Existing approaches often ignore the dependency of hierarchical structures in storyline generation. In this paper, we propose an unsupervised Bayesian model, called dynamic storyline detection model, to extract structured representations and evolution patterns of storylines. The proposed model is evaluated on a large scale news corpus. Experimental results show that our proposed model outperforms several baseline approaches.

1 Introduction

The rapid development of online news media sites is accompanied by the generation of tremendous news reports. Facing such massive amount of news articles, it is crucial to develop an automated tool which can provide a temporal summary of events and their evolutions related to a topic from news reports. Therefore, storyline detection, aiming at summarising the development of certain related events, has been studied in order to help readers quickly understand the major events reported in news articles. It has attracted great attention recently. Kawamae (2011) proposed a trend analysis model which used the difference between temporal words and other words in each document to detect topic evolution over time. Ahmed et al. (2011) proposed a unified framework to group temporally and topically related news articles into same storylines in order to reveal the temporal evolution of events. Tang and Yang (2012) developed a topic-user-trend model, which incorporates user interests into the generative process of web contents. Radinsky and Horvitz (2013) built storylines based on text clustering and entity entropy to predict future events. Huang and Huang (2013) developed a mixture-event-aspect model to model sub-events into local and global aspects and utilize an optimization method to generate storylines. Wang et al. (2013) proposed an evolutionary multi-branch tree clustering method for streaming text data in which the tree construction is casted as an online posterior estimation problem by considering both the current tree and the previous tree simultaneously.

With the fast development of social media platforms, newsworthy events are widely scattered not only on traditional news media but also on social media (Zhou et al., 2015). For example, Twitter, one of the most widely adopted social media platforms, appears to cover nearly all newswire events (Petrovic et al., 2013). Therefore, approaches have also been proposed for storyline summarization on social media. Given a user input query of an ongoing event, Lin et al. (2012) extracted the storyline of an event by first obtaining relevant tweets and then generating storylines via graph optimization. In (Li and Li, 2013), an evolutionary hierarchical Dirichlet process was proposed to capture the topic evolution pattern in storyline summarization.

However, most of the aforementioned approaches do not represent events in the form of structured representation. More importantly, they ignore the dependency of the hierarchical structures of events at different epochs in a storyline. In this paper, we propose a dynamic storyline detection model to overcome the above limitations.

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1943–1948,
Lisbon, Portugal, 17-21 September 2015.

We assume that each document could belong to one storyline s, which is modelled as a joint distribution over some named entities e and a set of topics z. Furthermore, to link events at different epochs and detect different types of storylines, the weighted sum of storyline distribution of previous epochs is employed as the prior of the current storyline distribution. The proposed model is evaluated on a large scale news corpus. Experimental results show that our proposed model outperforms several baseline approaches.

2 Methodology

To model the generation of a storyline in consecutive time periods for a stream of documents, we propose an unsupervised latent variable model, called dynamic storyline detection model (DSDM), The graphical model of DSDM is shown in Figure 1.

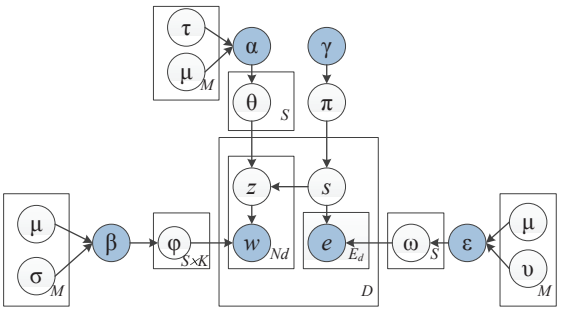


Figure 1: The Dynamic Storyline Detection model.

In this model, we assume that the storyline-topic-word, storyline-topic and storyline-entity probabilities at time t are dependent on the previous storyline-topic-word, storyline-topic and storyline-entity distributions in the last M epochs. For a certain period of time, we assume that each document could belong to one storyline s, which is modelled as a joint distribution over some named entities e and a set of topics z. This assumption essentially encourages documents published around similar time that involve the same named entities and discuss similar topics to be grouped into the same storyline. As the storyline distribution is shared across documents with the same named entities and similar topics, it essentially preserves the ambiguity that for example, documents comprising the same person and location may or may not belong to the same storyline.

The generative process of DSDM is shown below:

For each time period t from 1 to T:

- Draw a distribution over storylines $\boldsymbol{\pi}_s^t \sim$ Dirichlet(γ_s^t).
- For each storyline $s \in \{1...S\}$:
  - Draw a distribution over topics $\boldsymbol{\theta}_s^t \sim$ Dirichlet(α_s^t).
  - Draw a distribution over named entities $\boldsymbol{\omega}_s^t \sim$ Dirichlet(ϵ_s^t).
  - For each topic $k \in \{1...K\}$, draw a word distribution $\boldsymbol{\varphi}_{s,k}^t \sim$ Dirichlet(β_s^t).
- For each document $d \in \{1...D\}$:
  - Choose a storyline indicator $\boldsymbol{s}_d^t \sim$ Multinomial(π_s^t).
  - For each named entity $e \in \{1...E_d\}$:
    - Choose a named entity $\boldsymbol{e} \sim$ Multinomial(ω_s^t).
  - For other word positions $n \in \{1...N_d\}$:
    - Choose a topic $\boldsymbol{z}_n \sim$ Multinomial(θ_s^t).
    - Choose a word $\boldsymbol{w}_n \sim$ Multinomial($\varphi_{s,z}^t$).

We define an evolutionary matrix of storyline indicator s and topic z, $\sigma_{s,z,m}^t$, where each column $\sigma_{s,z,m}^t$ denotes storyline-topic-word distribution of storyline indicator s and topic z at epoch m, an evolutionary topic matrix of storyline indicator s, τ_s^t, where each column $\tau_{s,m}^t$ denotes storyline-topic distribution of storyline indicator at epoch m, an evolutionary entity matrix of storyline indicator s, υ_s^t, where each column $\upsilon_{s,m}^t$ denotes storyline-entity distribution of storyline indicator s.

We attach a vector of $M+1$ weights $\mu_{s,z}^t = \{\mu_{s,z,m}^t\}_{m=0}^M (\mu_{s,z,m}^t > 0, \sum_{m=0}^M \mu_{s,z,m}^t = 1)$, with its components representing the weights that each $\sigma_{s,z,m}^t$ contributes to calculating the priors of $\varphi_{s,z}^t$. We do it similarly for θ_s^t and ω_s^t. The Dirichlet prior for the storyline-topic-word distribution, the storyline-topic distribution and the storyline-entity distribution, respectively, at epoch t are:

$$\beta_{s,z}^t = \sum_{m=0}^{M} \mu_{s,z,m}^t \times \sigma_{s,z,m}^t \quad (1)$$

$$\alpha_s^t = \sum_{m=0}^{M} \mu_{s,m}^t \times \tau_{s,m}^t \quad (2)$$

$$\epsilon_s^t = \sum_{m=0}^{M} \mu_{s,m}^t \times \upsilon_{s,m}^t \quad (3)$$

In our experiments, the weight parameters are set to be the same regardless of storylines or topics. They are only dependent on the time window using an exponential decay function, $\mu_m =$

$\exp(-0.5 \times m)$ where m stands for the mth epoch counting backwards in the past M epochs. That is, more recent documents would have a relatively stronger influence on the model parameters in the current epoch compared to earlier documents. It is also possible to estimate the weights directly from data. We leave it as our future work.

The storyline-topic-word distribution $\varphi^t_{s,z}$, the storyline-topic distribution θ^t_s and the storyline-entity distribution ω^t_s at the current epoch t are generated from the Dirichlet distribution parameterized by $\beta^t_{s,z}, \alpha^t_s, \epsilon^t_s, \varphi^t_{s,z} \sim Dir(\beta^t_{s,z}), \varphi^t_{s,k} \sim Dir(\alpha^t_s), \omega^t_s \sim Dir(\epsilon^t_s)$. With this formulation, we can ensure that the mean of the Dirichlet parameter for the current epoch becomes proportional to the weighted sum of the word, topic distribution, and entity distribution at previous epochs.

3 Inference and Parameter Estimation

We use collapsed Gibbs sampling (Griffiths and Steyvers, 2004) to infer the parameters of the model, given observed data D. Gibbs sampling is a Markov chain Monte Carlo method which allows us repeatedly sample from a Markov chain whose stationary distribution is the posterior of interest, s^t_d and $z^t_{d,n}$ here, from the distribution over that variable given the current values of all other variables and the data. Such samples can be used to empirically estimate the target distribution. Letting the subscript $-d$ denote the quantity that excludes counts in document d, the conditional posterior for s_d is:

$$
\begin{aligned}
P(s^t_d = j | \boldsymbol{s}^t_{-d}, \boldsymbol{z}, \boldsymbol{w}, \Lambda) \propto \frac{\{N_j\}_{-d} + \gamma}{D_{-d} + S\gamma} \\
\times \prod_{e=1}^{E} \frac{\prod_{b=1}^{n^{(d)}_{j,e}} N_{j,e} - b + \epsilon^t_{j,e}}{\prod_{b=1}^{n^{E(d)}_{j}} n^E_j - b + \sum_{e=1}^{E} \epsilon^t_{j,e}} \\
\times \prod_{k=1}^{K} \frac{\prod_{b=1}^{n^{(d)}_{j,k}} n_{j,k} - b + \alpha^t_{j,k}}{\prod_{b=1}^{n^{(d)}_{j}} n_j - b + \sum_{k=1}^{K} \alpha^t_{j,k}} \\
\times \prod_{v=1}^{V} \frac{\prod_{b=1}^{n^{(d)}_{j,k,v}} n_{j,k,v} - b + \beta^t_{j,k,v}}{\prod_{b=1}^{n^{(d)}_{j,k}} n_{j,k} - b + \sum_{v=1}^{V} \beta^t_{j,k,v}},
\end{aligned}
$$

where N_j denotes the number of documents assigned to storyline indicator j in the whole corpus, D is the total number of documents, $n_{j,e}$ is the number of times named entity e is assigned with storyline indicator j, n^E_j denotes the total number of named entities with storyline indicator j in the document collection, $n_{j,k}$ is the number of times words with topic label k with storyline indicator j, n_j is the total number of words (excluding named entities) in the corpus with storyline indicator j, $n_{j,k,v}$ is the number of words v with storyline indicator j and topic label k in the document collection, counts with (d) notation denote the counts relating to document d only.

Letting the index $x = (d, n)$ denote nth word in document d and the subscript $-x$ denote a quantity that excludes data from the nth word position in document d. We only sample a topic z_x if the nth word is not a named entity based on the following conditional posterior:

$$
\begin{aligned}
&P(z^t_x = k | s_d = j, \boldsymbol{z}_{-x}, \boldsymbol{w}, \Lambda) \\
&\propto \frac{\{n^t_{j,k}\}_{-x} + \alpha^t_{j,k}}{\{n_j\}_{-x} + \sum_{k=1}^{K} \alpha^t_{j,k}} \times \frac{\{n^t_{j,k,w_n}\}_{-x} + \beta^t_{j,k,v}}{\{n^t_{j,k}\}_{-x} + \sum_{v=1}^{V} \beta^t_{j,k,v}}
\end{aligned}
$$

Once the latent variables s and z are known, we can easily estimate the model parameters $\pi, \Theta, \varphi, \psi, \omega$. We set the hyperparameters $\alpha = \gamma = 0.1, \beta = \epsilon = 0.01$ for the current epoch (i.e., $m = 0$), and gather statistics in the previous 7 epochs (i.e., $M = 7$) to set the Dirichlet priors for the storyline-topic-word distribution $\varphi^t_{s,z}$, the storyline-topic distribution θ^t_s and the storyline-entity distribution ω^t_s in the current epoch t, and run Gibbs sampler for 1000 iterations and stop the iteration once the log-likelihood of the training data converges under the learned model.

4 Experiments

4.1 Dataset

We crawled and parsed the GDELT Event Database[1] containing news articles published in May 2014. We manually annotated one-week data containing 101,654 documents and identified 77 storylines for evaluation. We also report the results of our model on the one-month data containing 526,587 documents. But we only report the precision and not recall of the storylines extracted since it is time consuming to identify all the true storylines in such a large dataset. In our experiments, we used the Stanford Named Entity Recognizer for identifying the named entities. In addition, we removed common stopwords and only kept tokens

[1] http://data.gdeltproject.org/events/index.html

which are verbs, nouns, or adjectives in these news articles.

4.2 Baselines

We chose the following three methods as the baseline approaches.

1. K-Means + Cosine Similarity (KMCS): the method first applies K-Means to cluster news documents for each day, then link storylines detected in different days based on the cosine similarity measurement.

2. LDA + Cosine Similarity (LDCS): the method first splits news documents on a daily basis, then applies the Latent Dirichlet Allocation (LDA) model to detect the latent storylines for the documents in each day, in which each storyline is modelled as a joint distribution over named entities and words, and finally links storylines detected in different days using the cosine similarity measurement.

3. Dynamic LDA (DLDA)[2]: this is the dynamic LDA (Blei and Lafferty, 2006) where the topic-word distributions are linked across epochs based on the Markovian assumption. That is, the topic-word distribution at the current epoch is only influenced by the topic-word distribution in the previous epoch.

4.3 Evaluation Metric

To evaluate the performance of the proposed approach, we use precision, recall and F-score which are commonly used in evaluating information extraction systems. The precision is calculated based on the following criteria: 1) The entities and keywords extracted refer to the same storyline. 2) The duration of the storyline is correct. We assume that the start date (or end date) of a storyline is the publication date of the first (or last) news article about it.

4.4 Experimental Results

The proposed model is compared against the baseline approaches on the annotated one-week data which consist of 77 storylines. The number of storylines, S, and the number of topics, K, are both set to 100. The number of historical epochs, M, which is taken into account for setting the Dirichlet priors for the storyline-topic-word, the storyline-topic and the storyline-entity distributions, is set to 7. The evaluation results of our proposed approach in comparison to the three baselines are presented in Table 1.

Method	Precision	Recall	F-score
KMCS	22.73	32.47	26.74
LDCS	34.29	31.17	32.66
DLDA	62.67	61.03	61.84
DSDM	**70.67**	**68.80**	**69.27**

Table 1: Performance comparison of the storyline extraction results in terms of Precision (%), Recall (%) and F-score (%).

It can be observed from Table 1 that simply using K-means to cluster news articles in each day and linking similar stories across different days in hoping of identifying storylines gives the worst results. Using LDA to detect stories in each day improves the precision dramatically. The dynamic LDA model assumes topics (or stories) in the current epoch evolves from the previous epoch and further improves the storyline detection results significantly. Our proposed model aims to capture the long distance dependencies in which the statistics gathered in the past 7 days are taken into account to set the Dirichlet priors of the storyline-topic-word, storyline-topic and storyline-entity distributions in the current epoch. It gives the best performance and outperforms dynamic LDA by nearly 7% in F-measure.

To study the impact of the number of topics on the performance of the proposed model, we conducted experiments on the one-month data with different number of topics varying between 100 and 200. In all these experiments, the number of storylines, S, is set to 200, based on the speculation that about 40 storylines in the annotated one-week data last for one month and about 40 new storylines occur each week. Table 2 shows the precision of the proposed method under different number of topics. It can be observed that the performance of the proposed approach is quite stable across different number of topics.

K	100	150	200
Precision	69.6%	70.2%	69.9%

Table 2: The precision of our method with various number (K) of topics.

[2]Topic number is set to 100 for both DLDA and LDCS. Cluster number is also set to 100 for KMCS.

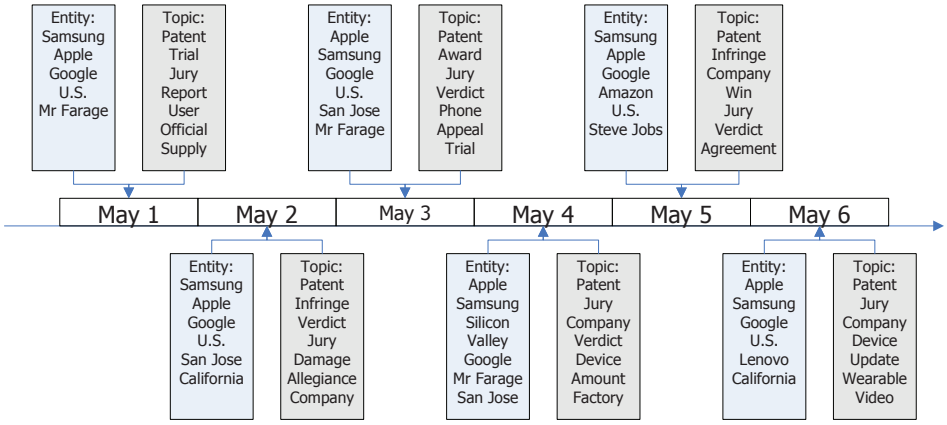


Figure 2: Storyline about the patent infringement case between Apple and Samsung was extracted by the proposed Model.

4.5 Structured Browsing

We illustrate the evolution of storylines by using structured browsing, from which the structured information (entity, topic, keywords) about storylines and the duration of storylines can be easily observed. Figure 2 shows the storyline about "The patent infringement case between Apple and Samsung". It can be observed that in the first two days, the hierarchical structure consists of entities (Apple, Samsung) and keywords (trial, patent, infringe). The case has gained significant attention in the next three days when US jury orders Samsung to pay Apple $119.6 million. It can be observed that the stories in the next three days also consist of entities (Apple, Samsung), but with different keywords (award, patent, win). The last day's story gives an overall summary and consists of entities (Apple, Samsung) and keywords (jury, patent, company).

To further investigate the storylines detected by the proposed model, we randomly selected three detected storylines. The first one is about "the patent infringement case between Apple and Samsung". It is a short-term storyline lasting for 6 day as shown in Figure 3. The second one is about "India election", which is a long-term storyline lasting for one month. The third one is about "Pistorius shoot Steenkamp", which is an intermittent storyline, lasting for a total of 22 days but with no relevant news reports in certain days as shown in Figure 3. It can be observed that the proposed model can detect not only continuous but also intermittent storylines, which further demonstrates the advantage of the proposed model.

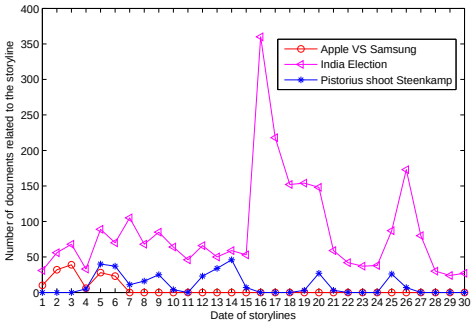


Figure 3: The number of documents on each day relating to the three storylines.

5 Conclusions and Future Work

In this paper, we have proposed an unsupervised Bayesian model to extract storylines from news corpus. Experimental results show that our proposed model is able to extract both continuous and intermittent storylines and outperforms a number of baselines. In future work, we will consider modelling background topics explicitly and investigating more principled ways in setting the weight parameters of the statistics gathered in the historical epochs. Moreover, we will also explore the impact of different scale of the dependencies from historical epochs on the distributions of the current epoch.

Acknowledgments

This work was funded by the National Natural Science Foundation of China (61528302), State Education Ministry, the Fundamental Research Funds for the Central Universities, and the Innovate UK under the grant number 101779.

References

Amr Ahmed, Qirong Ho, Jacob Eisenstein, Eric Xing, Alexander J Smola, and Choon Hui Teo. 2011. Unified analysis of streaming news. In *Proceedings of the 20th international conference on World wide web*, pages 267–276. ACM.

David M Blei and John D Lafferty. 2006. Dynamic topic models. In *Proceedings of the 23rd international conference on Machine learning*, pages 113–120. ACM.

Thomas L. Griffiths and Mark Steyvers. 2004. Finding scientific topics. In *Proceedings of the National Academy of Sciences 101 (Suppl. 1)*, pages 5228–5235.

Lifu Huang and Lian'en Huang. 2013. Optimized event storyline generation based on mixture-event-aspect model. In *Proceedings of the 2013 Conference on Empirical Methods on Natural Language Processing*, pages 726–735. ACL.

Noriaki Kawamae. 2011. Trend analysis model: trend consists of temporal words, topics, and timestamps. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 317–326. ACM.

Jiwei Li and Sujian Li. 2013. Evolutionary hierarchical dirichlet process for timeline summarization. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 556–560. ACL.

Chen Lin, Chun Lin, Jingxuan Li, Dingding Wang, Yang Chen, and Tao Li. 2012. Generating event storylines from microblogs. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 175–184. ACM.

Saša Petrovic, Miles Osborne, Richard McCreadie, Craig Macdonald, Iadh Ounis, and Luke Shrimpton. 2013. Can twitter replace newswire for breaking news? In *Proceedings of the 7th International AAAI Conference on Weblogs and Social Media*.

Kira Radinsky and Eric Horvitz. 2013. Mining the web to predict future events. In *Proceedings of the sixth ACM international conference on Web search and data mining*, pages 255–264. ACM.

Xuning Tang and Christopher C Yang. 2012. TUT: a statistical model for detecting trends, topics and user interests in social media. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 972–981. ACM.

Xiting Wang, Shixia Liu, Yangqiu Song, and Baining Guo. 2013. Mining evolutionary multi-branch trees from text streams. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 722–730. ACM.

Deyu Zhou, Liangyu Chen, and Yulan He. 2015. An unsupervised framework of exploring events on twitter: Filtering, extraction and categorization. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI 2015) , Austin, Texas, USA, January 25 C 30, 2015*, pages 2468–2474.

Topical Coherence for Graph-based Extractive Summarization

Daraksha Parveen[†] **Hans-Martin Ramsl**[‡] **Michael Strube**[†]

†NLP Group and Research Training Group AIPHES
Heidelberg Institute for Theoretical Studies gGmbH
Heidelberg, Germany
{daraksha.parveen|michael.strube}@h-its.org

‡SAP SE
Walldorf, Germany
hans-martin.ramsl@sap.com

Abstract

We present an approach for extractive single-document summarization. Our approach is based on a weighted graphical representation of documents obtained by topic modeling. We optimize importance, coherence and non-redundancy simultaneously using ILP. We compare ROUGE scores of our system with state-of-the-art results on scientific articles from *PLOS Medicine* and on DUC 2002 data. Human judges evaluate the coherence of summaries generated by our system in comparision to two baselines. Our approach obtains competitive performance.

1 Introduction

Summarization systems take a long document as input and generate a concise document as output. Several summarization variants exist such as generic, query-based, multi-document and single document, but the basic requirements for summarization remain the same. Summaries should contain salient information so that the reader will not miss anything from the original document. Also, the reader is not interested in repetitive information, so summaries should not include redundant information. Finally, summaries should be coherent and of high readability.

We introduce a completely unsupervised graph-based summarization using latent drichlet allocation (LDA, Blei and Lafferty (2009)). LDA is a simple model for topic modeling where topic probabilities are assigned words in documents. The probabilities can be used to measure the semantic relatedness between words and hence the topical coherence of a document. We use topical coherence as a means to ensure the coherence of extractive single-document summaries. Remus and Biemann (2013) apply LDA to compute lexical chains while Gorinski and Lapata (2015) also develop a graph-based summarization system which takes coherence into account.

Our work is based on the bipartite entity graph introduced by Guinaudeau and Strube (2013). However, in their graph one set of nodes corresponds to entities whereas in our graph it corresponds to topics. The entity graph has already been used by Parveen and Strube (2015) for summarization. Their graph is unweighted and sparse, whereas our topical graph is weighted and dense.

We apply our topical graph on the dataset introduced by Parveen and Strube (2015). This dataset contains scientific articles from the journal *PLOS Medicine*[1]. Every *PLOS Medicine* article is accompanied by an editor's summary and an authors' abstract. We use both as gold summaries for evaluation. Results obtained on the *PLOS Medicine* dataset using the topical graph are as good as using the entity graph and significantly better than several baselines and the graph-based system *TextRank* (Mihalcea and Tarau, 2004). We use the DUC 2002 dataset to compare our results with state-of-the-art techniques. In contrast to the *PLOS Medicine* data the DUC 2002 dataset contains very small articles. Still, our technique gives comparable results to the state-of-the-art. This shows that our technique is flexible and scalable despite being unsupervised.

2 Our Method

2.1 Document Representation

A graph-based representation has been used by well known summarization systems such as LexRank (Erkan and Radev, 2004) and TextRank (Mihalcea and Tarau, 2004). The graph used by both is of one mode type where sentences are nodes which are connected by weighted edges.

[1] http://journals.plos.org/plosmedicine/

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1949–1954,
Lisbon, Portugal, 17-21 September 2015.

Weights express sentence similarity.

We use a bipartite graph representation of documents (Figure 1). The bipartite graph, $G = (V_s, V_t, E_{t,s})$, has two sets of nodes where V_s represents sentences and V_t topics. The two sets of nodes are connected with edge $E_{t,s}$, if a word in a sentence s is present in a topic t. If multiple words are present in topic t of sentence s, then the edge weight is the logarithmic sum of probabilities of words in topic t. We normalize the edge weight by dividing them by the length of the sentence. Hence long sentences will not benefit from their lengths. We call the resulting graph *topical graph*.

2.2 Sentence Ranking

The final summary should contain only important sentences. Therefore, we give a score to every sentence in a document to obtain important sentences. Following Parveen and Strube (2015) we apply the HITS (Hyperlink Induced Topic Search) (Kleinberg, 1999) algorithm for ranking sentences by importance, since our graph is a bipartite graph. It puts nodes of a graph in two sets: *hub nodes* and *authority nodes*.

For the HITS algorithm the rank of nodes needs to be initialized. We initialize the topic rank $Rank_{t_i} = 1$ and the sentence rank $Rank_{s_i} = 1 + sim(s_i, title)$. The $title$ in the sentence rank is the title of the article. $sim(s_i, title)$ is the cosine similarity between the sentence s_i and the title of the article. After initialization of all nodes in the weighted topical graph, the HITS algorithm is applied to obtain ranks of sentences.

2.3 Coherence Measure

Guinaudeau and Strube (2013) represent a document by the entity graph, a bipartite graph consisting of sentence and entity nodes. They perform a one-mode projection on sentence nodes, compute the coherence of a document on the basis of the one-mode projections and use the coherence measure for summary coherence rating. Building upon this work, Parveen and Strube (2015) integrate this coherence measure to directly generate coherent summaries. Instead of the entity graph we here use the topical graph to incorporate the coherence measure. Parveen and Strube (2015) use an unweighted projection graph whereas we use a weighted projection graph of a topical graph to compute the coherence. The weighted one mode projection of the topical graph is shown in Figure 1, bottom right.

$$weighted\_coh(s_i, P) = weighted\_Outdegree(s_i, P) \tag{1}$$

$$norm\_weighted\_coh(s_i, P) = \frac{weighted\_coh(s_i, P)}{\sum weighted\_coh(s_i, P)} \tag{2}$$

Equation 1 calculates the outdegree of every sentence from the weighted projection graph. However $weighted\_coh(s_i, P)$ in Equation 1 is not a normalized value. The normalized coherence value is in Equation 2. Afterwards, we use this coherence value in the optimization phase for the selection of sentences.

2.4 Optimization

McDonald (2007) introduces summarization as an optimization task which takes care of importance, redundancy and coherence simultaneously. In this paper, we also propose a model for single document summarization which is based on integer linear programming (ILP). We consider ranks obtained by the HITS algorithm as sentence importance. The weighted coherence measure is calculated using Equation 1 and Equation 2. *PLOS Medicine* articles are very long and contain repetitive information, so we have to deal with redundancy even in single-document summarization. Therefore we model non-redundancy as topic coverage in the final summary: the more topics in a summary, the less redundant the summary will be. The ILP objective function is shown in Equation 3. $f_i(X)$ is the function which maximizes importance, $f_c(X)$ maximizes coherence, and $f_t(Y)$ maximizes topic coverage.

$$Objective\,function : \max_{X,Y}(f_i(X) + f_c(X) + f_t(Y)) \tag{3}$$

X is a variable for sentences which contains boolean variables x_i, where $0 < i < n$ is the number of sentences. Y is a variable for topics which contains boolean variables y_j, where $0 < j < m$ is the number of topics.

$$f_t(Y) = \sum_{j=1}^{m} y_j \tag{4}$$

Constraints ensure that the system satisfies additional requirements such as summary length:

$$\sum_{i=1}^{n} x_i \leq Len(summary) \tag{5}$$

$$\sum_{j \in T_i} y_j \geq |Topics_{x_i}| \cdot x_i, \quad \text{for } i = 1, \ldots, n \tag{6}$$

S_1 WHO recommends prompt diagnosis and quinine plus clindamycin for treatment of uncomplicated malaria in the first trimester and artemisinin-based combination therapies in subsequent trimesters.

S_2 We undertook a systematic review of women's access to and healthcare provider adherence to WHO case management policy for malaria in pregnant women.

S_3 Data were appraised for quality and content.

S_4 Determinants of women's access and providers' case management practices were extracted and compared across studies.

	Topic1 (t1)	Topic2 (t2)	Topic3 (t3)	Topic4 (t4)	Topic5 (t5)	Topic6 (t6)	Topic7 (t7)	Topic8 (t8)	Topic9 (t9)	Topic10 (t10)	Topic11 (t11)	.....	Topic500 (t500)
S_1	-	0.3	0.46	0.50	0.23	0.55	0.89	0.56	0.09	-	-	.....	0.73
S_2	0.3	0.29	0.3	-	0.66	0.35	0.36	0.63	-	-	0.55	.....	-
S_3	-	0.28	-	0.43	-	0.39	-	0.42	0.65	-	0.12	.....	0.58
S_4	0.22	0.43	0.29	0.6	-	-	0.14	0.26	-	-	-	.....	0.41

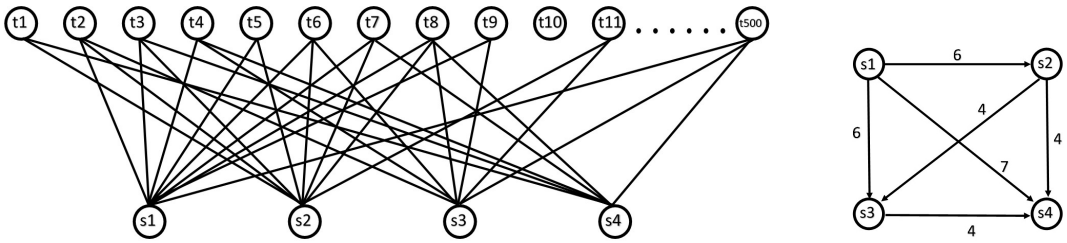


Figure 1: Abstract from *PLOS Medicine*, topical grid, bipartite topical graph, one-mode projection

$$\sum_{i \in S_j} x_i \geq y_j, \quad \text{for } j = 1, \ldots, m \tag{7}$$

The final summary should be shorter than the original text and it should also have a length limit (Equation 5). The results on *PLOS medicine* data (Section 3) are limited to 5 sentences. We have also experimented with multiple lengths. Increasing the summary length increases ROUGE scores. DUC 2002 summaries are limited to 100 words.

Equation 6 shows that topics present in sentence x_i are selected, when sentence x_i is selected. Therefore, $x_i = 1$ and $T_i = Topics_{x_i}$. The constraint holds, because $\sum_{j \in T_i} y_j = |Topics_{x_i}|$. Furthermore, if sentence $x_i = 0$, i.e., it is not selected, then there must be topics which are already present in selected sentences. Hence, the constraint holds, $\sum_{j \in T_i} y_j \geq 0$.

Equation 7 constrains the selection of topics. If topic $y_j = 1$, then at least one sentence containing this topic has been selected. Therefore $\sum_{i \in S_j} x_i \geq 1$, and the constraint holds. If topic $y_j = 0$, then sentences containing this topic are not selected, so $\sum_{i \in S_j} x_i = 0$, and the constraint holds.

3 Experiments

Following Parveen and Strube (2015), we evaluate on the science genre, i.e. *PLOS Medicine* articles, and on the news genre, i.e. DUC 2002 data.

3.1 Datasets

PLOS Medicine articles are considerably longer than DUC 2002 documents. The average number of sentences per document is 154 in *PLOS Medicine* and 25 in DUC 2002. Benefits of using *PLOS Medicine* articles for experiments are:

- They are accompanied by an authors' abstract.
- They have a summary written by an editor.
- They are formatted in XML.
- They contain explicit full forms of abbreviations.

Editor's summaries have a different perspective, writing style and length than authors' abstracts. We use both as gold summaries for evaluation. Following Parveen and Strube (2015) we report the results using editor's summaries and author's abstracts independently. To compare with the state-of-the-art in single-document summarization, we also evaluate on DUC 2002 data.

3.2 Experimental Setup

We use the XML version of *PLOS Medicine* articles. We extract the contents excluding figures, tables and references. Editor's summary and authors' abstract are separated from the content for evaluation. The *PLOS Medicine* XML provides explicit full forms when abbreviations are introduced. We replace abbreviations with their full form in the summary. We then remove non-alphabetical characters. After this we parse articles using the Stanford parser (Klein and Manning, 2003). We perform pronoun resolution using the coreference resolution system by Martschat (2013)[2]. We use *gensim* to generate the topics. For generating topics we use a dataset containing scientific articles from biology, which contains 221,385 documents and about 50 million sentences[3]. We also use Wikipedia to compare with topics from a general domain.

The HITS algorithm is applied on the bipartite graph for computing sentence importance. We calculate the coherence values of sentences on weighted one-mode projection graphs. The importance and coherence of a sentence is used in the optimization phase[4] which returns a binary value for each sentence.

3.3 Results

Results on *PLOS Medicine* are shown in Tables 1 and 2. We evaluate using ROUGE-SU4 and ROUGE-2 (Lin, 2004). We limit the length of the summaries to five sentences and the number of topics to 2000 in the topical graph. We also experimented with varying numbers of topics, i.e. 500, 1000 and 2000, and varying summary length limits. The results changed only marginally. The general trends remained the same.

We compare our system with four different baselines and two versions of the entity graph. *Lead* selects the top five sentences, *Random* five sentences randomly. *MMR* is an implementation of maximal marginal relevance (Carbonell and Goldstein, 1998). *TextRank* is the graph-based system by Mihalcea and Tarau (2004)[5]. *Egraph* is the entity graph based system by Parveen and Strube (2015). *Egraph + Coh.* is their system which includes a coherence measure, which is calculated by using the unweighted projection graph. *Egraph + Coh. + Pos.* combines the coherence measure and positional information.

Systems	R-SU4	R-2
Lead	0.067	0.055
Random	0.048	0.031
MMR	0.069	0.048
TextRank	0.068	0.048
Egraph	0.121	0.090
Egraph + Coh.	0.130	0.096
Egraph + Coh. + Pos.	0.131	0.098
Tgraph (n=2000)	0.123	0.091
Tgraph (n=2000) + Coh.	0.129	0.095
Tgraph (n=2000) + Coh. + Pos.	0.125	0.092

Table 1: *PLOS Medicine*, editor's summaries

Systems	R-SU4	R-2
Lead	0.105	0.077
Random	0.093	0.589
MMR	0.118	0.098
TextRank	0.134	0.101
Egraph	0.200	0.170
Egraph + Coh.	0.219	0.175
Egraph + Coh. + Pos.	0.224	0.189
Tgraph (n=2000)	0.217	0.173
Tgraph (n=2000) + Coh.	0.221	0.179
Tgraph (n=2000) + Coh. + Pos.	0.215	0.174

Table 2: *PLOS Medicine*, authors' abstracts

Our system outperforms all baselines substantially, as shown in Tables 1 (editor's summaries) and 2 (authors' abstracts). We observe improvements in the results when including coherence in the topical graph. We obtain best results with Tgraph + Coh., where the number of topics is 2000. In Tgraph, penalizing coherence measures with positional information lowers ROUGE scores. While including positional information into the entity graph obtains the best results on the *PLOS Medicine* dataset, positional information does not appear to be beneficial for the topical graph. Absolute ROUGE scores are higher when using abstracts as gold summaries, because the abstracts are shorter than editor's summaries.

We compare results using biology journals (Table 3) and Wikipedia (Table 4) to generate topics. The topical graph is denser when using biology journals compared to the graph generated from Wikipedia. Results using the in-domain biology journals as data to generate topics are better than using general domain Wikipedia data. The scores are highest with 2000 topics. For Bio topic the differences are negligible, however.

We also compare results on DUC 2002 to

[2] http://www.smartschat.de/software/

[3] http://www.datawrangling.com/some-datasets-available-on-the-web/

[4] We use Gurobi, http://www.gurobi.com

[5] https://kenai.com/projects/textsummarizer

Topics	R-1	R-2	R-SU4
Tgraph (n=500) + Coh.	0.279	0.090	0.125
Tgraph (n=1000) + Coh.	0.289	0.093	0.128
Tgraph (n=2000) + Coh.	0.291	0.095	0.129

Table 3: *PLOS Medicine*, editor's summ., Bio topic

Topics	R-1	R-2	R-SU4
Tgraph (n=500) + Coh.	0.208	0.060	0.098
Tgraph (n=1000) + Coh.	0.258	0.073	0.106
Tgraph (n=2000) + Coh.	0.283	0.086	0.121

Table 4: *PLOS Medicine*, editor's summ., Wiki topic

check against the state-of-the-art on a well-known dataset. *Lead* performs well on DUC 2002 as shown in Table 5, because important information appears in the initial lines of news articles. *DUC 2002 Best* is the result reported by the top performing system at DUC 2002. This system actually obtains better results than *TextRank* (Mihalcea and Tarau, 2004) and the more recent system *UniformLink* (Wan and Xiao, 2010). Our system *Tgraph + Coh.* performs better than the well known best systems on DUC 2002 and slightly better than *Egraph + Coh.* However the difference between the results of Tgraph and Egraph are not significant. In contrast to the entity graph based system, the coherence measure in our system is calculated by using a topic-based weighted projection graph, which is denser and hence more informative.

3.4 Human Coherence Judgements

In addition to ROUGE scores, we use human judgements for evaluating the coherence of our summaries. We asked four PhD students in natural language processing to evaluate our summaries on the basis of coherence. We randomly selected ten summaries of scientific articles from three different systems, *TextRank*, *Lead* and *Tgraph + Coh.* We asked the human judges to rank the summaries according to their coherence. So, the summary which is best in coherence gets rank 1, second best gets rank 2, and worst gets rank 3. We calculated the Kendall concordance coefficient (W) (Siegel and Castellan, 1988) to measure the judges' agreement. We obtain $W = 0.61$, which indicates a relatively high agreement.

Systems	R-1	R-2	R-SU4
Lead	0.459	0.180	0.201
DUC 2002 Best	0.480	0.228	
TextRank	0.470	0.195	0.217
UniformLink (k = 10)	0.471	0.201	
Egraph + Coh.	0.479	0.238	0.230
Tgraph (n=2000) + Coh.	0.481	0.243	0.242

Table 5: DUC 2002, single-document summarization

To compare the three systems, we take the average over the ranks. The overall rank of *TextRank* is 2.625, *Lead* is 1.675 and *Tgraph + Coh.* is 1.8. *Lead* performs best, because it selects the top five consecutive sentences, which are coherent as the original authors intended them to be. However, the overall ranks of *Lead* and *Tgraph + Coh.* are not significantly different, whereas *TextRank*'s overall rank is significantly worse than both. Hence, *Tgraph + Coh.* performs very well in our human judgement coherence experiment.

4 Discussion

In this paper we introduced the topical graph for single document summarization. We experimented with multiple numbers of topics on the scientific article dataset. Our system performs well when including the weighted coherence measure in the optimization phase. The results are comparable with the entity graph. However, the entity graph is less informative and very sparse as compared to the topical graph. Our system does not need annotated training data and, except for the number of topics, no optimization of parameters. Hence, we consider it unsupervised.

Acknowledgments

This work has been funded by the Klaus Tschira Foundation, Heidelberg, Germany. The first author has been supported by a Heidelberg Institute for Theoretical Studies Ph.D. scholarship. This work has been supported by the German Research Foundation as part of the Research Training Group "Adaptive Preparation of Information from Heterogeneous Sources" (AIPHES) under grant No. GRK 1994/1. We would like to thank our colleagues Sebastian Martschat, Nafise Moosavi, Alex Judea and Mohsen Mesgar who served as human subjects and commented on earlier drafts.

References

David M. Blei and John D. Lafferty. 2009. Topic models. In A. Srivastava and M. Sahami, editors, *Text Mining: Classification, Clustering, and Applications*. Chapman & Hall, Boca Raton, Flo.

Jaime G. Carbonell and Jade Goldstein. 1998. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval,* Melbourne, Australia, 24–28 August 1998, pages 335–336.

Güneş Erkan and Dragomir R. Radev. 2004. LexRank: Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research*, 22:457–479.

Philip John Gorinski and Mirella Lapata. 2015. Movie script summarization as graph-based scene extraction. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies,* Denver, Col., 31 May – 5 June 2015, pages 1066–1076.

Camille Guinaudeau and Michael Strube. 2013. Graph-based local coherence modeling. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers),* Sofia, Bulgaria, 4–9 August 2013, pages 93–103.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics,* Sapporo, Japan, 7–12 July 2003, pages 423–430.

Jon M. Kleinberg. 1999. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Proceedings of the Text Summarization Branches Out Workshop at ACL '04,* Barcelona, Spain, 25–26 July 2004, pages 74–81.

Sebastian Martschat. 2013. Multigraph clustering for unsupervised coreference resolution. In *51st Annual Meeting of the Association for Computational Linguistics: Proceedings of the Student Research Workshop,* Sofia, Bulgaria, 5–7 August 2013, pages 81–88.

Ryan McDonald. 2007. A study of global inference algorithms in multi-document summarization. In *Proceedings of the European Conference on Information Retrieval,* Rome, Italy, 2-5 April 2007.

Rada Mihalcea and Paul Tarau. 2004. TextRank: Bringing order into texts. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing,* Barcelona, Spain, 25–26 July 2004, pages 404–411.

Daraksha Parveen and Michael Strube. 2015. Integrating importance, non-redundancy and coherence in graph-based extractive summarization. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence,* Buenos Aires, Argentina, 25–31 July 2015, pages 1298–1304.

Steffen Remus and Chris Biemann. 2013. Three knowledge-free methods for automatic lexical chain extraction. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies,* Atlanta, Georgia, 9–14 June 2013, pages 989–999.

Sidney Siegel and N. John Castellan. 1988. *Nonparametric Statistics for the Behavioral Sciences*. McGraw-Hill, New York, 2nd edition.

Xiaojun Wan and Jianguo Xiao. 2010. Exploiting neighborhood knowledge for single document summarization and keyphrase extraction. *ACM Transactions on Information Systems*, 28(2):8 pages.

Summarizing Student Responses to Reflection Prompts

Wencan Luo
Computer Science Department
University of Pittsburgh
Pittsburgh, PA 15260, USA
wencan@cs.pitt.edu

Diane Litman
Computer Science Department and LRDC
University of Pittsburgh
Pittsburgh, PA 15260, USA
litman@cs.pitt.edu

Abstract

We propose to automatically summarize student responses to reflection prompts and introduce a novel summarization algorithm that differs from traditional methods in several ways. First, since the linguistic units of student inputs range from single words to multiple sentences, our summaries are created from extracted phrases rather than from sentences. Second, the phrase summarization algorithm ranks the phrases by the number of students who semantically mention a phrase in a summary. Experimental results show that the proposed phrase summarization approach achieves significantly better summarization performance on an engineering course corpus in terms of ROUGE scores when compared to other summarization methods, including MEAD, LexRank and MMR.

1 Introduction

Educational research has demonstrated the effectiveness of *reflection prompts* (Boud et al., 2013) to enhance interaction between instructors and students (Van den Boom et al., 2004; Menekse et al., 2011). However, summarizing student responses to these prompts for large courses (e.g., introductory STEM, MOOCs) is an onerous task for humans and poses challenges for existing summarization methods. First, the linguistic units of student inputs range from single words to multiple sentences. Second, we assume that the concepts (represented as phrases) mentioned by more students should get more attention from the instructor. Based on this assumption, we introduce the notion of *student coverage*, defined as the number of students who semantically mention a particular phrase. The more student coverage a phrase has, the more important it is. To illustrate the new task, an example is shown in Table. 1.

Reflection prompt:
Describe what was confusing or needed more detail.

Student Responses:
S1: Graphs of attraction/repulsive & interatomic separation
S2: Property related to bond strength
S3: The activity was difficult to comprehend as the text fuzzing and difficult to read.
S4: Equations with bond strength and Hooke's law
S5: I didn't fully understand the concept of thermal expansion
S6: The activity (Part III)
S7: Energy vs. distance between atoms graph and what it tells us
S8: The graphs of attraction and repulsion were confusing to me...(*rest omitted, 53 student responses in total.*)

Human Reference Summary:
1) Graphs of attraction/ repulsive & atomic separation [10]
2) Properties and equations with bond strength [7]
3) Coefficient of thermal expansion [6]
4) Activity part III [4]

Table 1: An example reflection prompt, 53 student responses and a gold-standard summary. The numbers in the square brackets indicate the number of students who semantically mention each phrase (i.e., student coverage).

In this work, we propose a phrase summarization method that addresses the above challenges. First, our summaries are created from extracted phrases rather than from sentences. Phrases are easy to read and browse like keywords, and fit better on small devices when compared to sentences. For example, including phrases such as "*I didn't fully understand*" (S5) and "*were confusing to me*" (S8) in the summary is a waste of space. Second, we adopt a metric clustering paradigm with a semantic distance to estimate the student coverage of each phrase in the summary; a semantic metric allows similar phrases to be grouped together even if they are in different textual forms. Experimental

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1955–1960,
Lisbon, Portugal, 17-21 September 2015.

results demonstrate the utility of our approach.

Although not the focus of this paper, we have also built a mobile application called CourseMIR-ROR[1] that utilizes the proposed summarization algorithm (Luo et al., 2015). Fan et al. (2015) report a preliminary study about the usage of the application.

2 Related Work

While summarization systems that extract sentences are dominant, others have published in "summarization" at other levels besides the sentence. For example, Ueda et al. (2000) developed an "at-a-glance" summarization method with handcrafted rules. Recently, keyphrase extraction (Hasan and Ng, 2014; Liu et al., 2009; Medelyan et al., 2009; Wu et al., 2005) has received considerable attention, aiming to select important phrases from input documents, which is similar to phrase summarization. In this paper, we propose a general framework to adapt sentence summarization to phrase summarization.

Clustering has been used to score sentences and has shown good improvement in text summarization (Yang et al., 2012; Li and Li, 2014; Gung and Kalita, 2012). In this work, we are using a metric clustering with semantic similarity to estimate the student coverage at a phrase level. Similarly, both diversity-based summarization (Carbonell and Goldstein, 1998; Zhang et al., 2005; Zhu et al., 2007) and our proposed method aim to estimate and maximize student coverage by minimizing redundancy in the output phrases. Differently, our method performs the redundancy reduction at a cluster level (a group of phrases) rather than penalize redundancy with a greedy iterative procedure sentence by sentence, and not only the information content is considered, but also the information source.

3 Data

Our data consists of student responses collected from 53 undergraduates enrolled in an introduction to materials science and engineering class. The students were asked to complete a survey at the end of each of 25 lectures during a semester, consisting of three carefully designed reflection prompts: 1) "Describe what you found most interesting in today's class." 2) "Describe what was confusing or needed more detail." 3) "Describe what you learned about how you learn."

	min	max	mean	std
Student-WC	1	91	9.2	7.3
TA-PWC	1	26	7.1	4.9
TA-WC	6	103	29.4	23.2
TA-PC	2	12	4.2	2.2

Table 2: **W**ord **C**ount (WC) in student responses (Student-WC), WC per phrase in TA's summary (TA-PWC), WC in TA's summary (TA-WC) and phrase count in TA's summary (TA-PC)

In total, more than 900 responses were collected for each prompt. Currently, gold-standard summaries of 12 out of 25 lectures are created by the teaching assistant for that course for each reflection prompt. The summaries include not only the important phrases, but also the number of students who mentioned them (i.e., student coverage). 4 lectures are randomly selected as a development set and the remaining data used as a test set, yielding 12 sets of development data and 24 sets of testing data, each with a prompt, the students' responses and the gold-standard summary. [2]

The statistics of the student responses and the TA's summary are shown in Table 2. The phrases summarized by the TA are significantly shorter than the student responses (7.1 vs. 9.2, $p<0.001$).

4 Proposed Method

We formulate our task as a standard extractive summarization problem. Unlike standard sentence-level extraction where the input and output are sentences, the input of our task ranges from words or phrases to full sentences. The output is a list of important phrases and the summary length (either # of phrases or words) is no more than L.

The proposed algorithm involves three stages: *candidate phrase extraction*, *phrase clustering*, and *phrase ranking*.

4.1 Candidate phrase extraction

We extract noun phrases (NPs) from the input using a syntax parser from the Senna toolkit (Collobert, 2011), preserving the most important con-

[1]Homepage: http://www.coursemirror.com; free download link in Google Play Store: https://play.google.com/store/apps/details?id=edu.pitt.cs.mips.coursemirror

[2]This data is publicly available at the CourseMIR-ROR website: http://www.coursemirror.com/download/dataset.

tent from the original responses without losing too much context information compared to keywords. For example, "the concept of thermal expansion" (S5) is extracted as a candidate phrase. Only NPs are considered because all reflection prompts used in the task are asking about "what", and knowledge concepts are usually represented as NPs.[3]

Due to the noisy data, malformed phrases are excluded, including single stop words (e.g. "it", "I", "there", "nothing") and phrases starting with a punctuation mark (e.g. "'t", "+ indexing").

4.2 Phrase clustering

Although phrases are more meaningful and less ambiguous compared to keywords, they suffer from the sparsity problem, especially in our data set when 89.9% of the phrases appeared only once. The challenge is the fact that students use different words for the same meaning (e.g., "bicycle parts" and "bike elements").

We use a clustering paradigm with a semantic distance metric to address this issue. Among different clustering algorithms, K-Medoids (Kaufman and Rousseeuw, 1987) fits well for our problem. First, it works with an arbitrary distance matrix between datapoints. It allows to use pairwise semantic similarity-based distance between phrases, yielding metric clustering. Second, it is robust to noise and outliers because it minimizes a sum of pairwise dissimilarities instead of squared Euclidean distances. It shows better performance than an LDA-based approach to group students' short answers for the purpose of semi-automated grading (Basu et al., 2013). Since K-Medoids picks a random set of seeds to initialize as the cluster centers (called medoids), the clustering algorithm runs 100 times and the cluster with the minimal within-cluster sum of the distances is retained to reduce random effects.

Distance metric. The semantic similarity is implemented using SEMILAR (Rus et al., 2013), using the *latent semantic analysis* trained on the Touchstone Applied Science Associates corpus (Ştefănescu et al., 2014). The distance matrix D is constructed from the similarity matrix S by applying the following transformation: $D = e^{-S}$, which is similar to the common heat kernel but without normalization[4].

[3]In our data, no advantage is observed by including other constituents like verb and prepositional phrases.

[4]This is not normalized to the range between 0 and 1 since we only care about the relative distance.

Number of clusters. For setting the number of clusters without tuning, we adopted a method from Wan and Yang (2008), by letting $K = \sqrt{V}$. where K is the number of clusters and V is the number of candidate phrases instead of the number of sentences.

4.3 Phrase ranking

In order to estimate the *student coverage*, phrases are clustered with the algorithm introduced above. We assume the phrases in a cluster are semantically similar to each other and any phrase in a cluster can represent it as a whole. Therefore the coverage of a phrase is assumed to be the same as the coverage of a cluster, which is a union of the students covered by each phrase in the cluster.

To select the most representative phrase in a cluster, LexRank (Erkan and Radev, 2004), a graph-based algorithm for computing relative importance of textual units (working for both sentences and phrases), is used to score the extracted candidate phrases. The top ranked phrase in the cluster is added to the output summary. This process starts from the cluster that has the most estimated student coverage and repeats for the next cluster until the length limit is reached.

Note that when the student coverage is the same between two clusters, the score of the top-ranked phrases in the clusters according to LexRank is used to break the tie: the higher, the better.

5 Experiments

We use the ROUGE evaluation metric (Lin, 2004) and report R-1 (unigrams), R-2 (bigrams), and R-SU4 (bigrams with skip distance up to 4 words), including the recall (R), precision (P) and F-Measure (F). These scores measure the overlap between human-generated summaries and a machine-generated summary.

We design and compare a number of other summarization methods to evaluate the proposed phrase summarization approach.

Keyphrase extraction. Maui (Medelyan et al., 2009) is selected as the baseline, which is one of the state-of-the-art keyphrase extraction methods.

Sentence to phrase summarization. Existing sentence summarization techniques can be used for phrase summarization by extracting candidate phrases and treating them as sentences. Within this framework, we adapt MEAD (Radev et al., 2004) and LexRank (Erkan and Radev, 2004) to

	R-1			R-2			R-SU4		
	R	P	F	R	P	F	R	P	F
Keyphrase	.171	.364	.211	.057	.134	.071	.039	.168	.049
OriMEAD	**.397**	.185	.219	.117	.069	.073	**.157**	.051	.045
MEAD	.341	.269	.265	.122	.102	.099	.126	.094	.072
MEAD+MMR	.360	.279	.277	**.130**	.106	.104	.142	.099	.078
LexRank	.325	.355	.307	.107	.110	.102	.120	.145	.098
LexRank+MMR	.328	.367	.312	.111	.126	.110	.117	.154	.098
Clustering+Medoid	.279	**.473**	.327	.078	.129	.091	.068	**.216**	.087
Proposed	.319	$.448^{\dagger *}$	$\mathbf{.340}^{\dagger}$	.122	$\mathbf{.176}^{\dagger *}$	**.134**	.112	$.205^{\dagger *}$	$\mathbf{.109}^{\dagger}$

Table 3: Summarization performance. The last row is our proposed approach. The highest score for each column is shown in bold. † indicates that the improvement over the MEAD+MMR baseline is statistically significant. * indicates that the improvement over LexRank+MMR is statistically significant.

our task. We also include the original MEAD[5] for comparison (named as OriMEAD).

Diversity-based summarization. We applied the MMR (Carbonell and Goldstein, 1998), a popular diversity-based summarization method as a post-processing step to the MEAD (**MEAD+MMR**) and LexRank (**LexRank+MMR**) baselines.[6]

Clustering+Medoid. To show the performance using the clustering alone, this baseline selects the medoid phrase instead of using LexRank to rank the phrases in a cluster to form the summary.

Results. The performance on the test set is shown in Table 3 with the length limit L as 4 phrases (the average phrase number in the TA's summary). Similar results can be observed when the length limit is based on the number of words, but cannot be reported here due to page limit.

First, our proposed method (last row), which clusters the extracted phrases and uses LexRank to score them, can outperform all the baselines over all three ROUGE scores in terms of F-measure. In addition, the proposed model performs better than the clustering and LexRank alone. Through a paired *t*-test, our model outperforms LexRank statistically in terms of precision for all three ROUGE scores and significantly improves Clustering+Medoid on all R-2 scores (except the precision with 0.06 p-value). We believe that the semantic similarity based clustering complements LexRank in two ways: 1) LexRank depends on the cosine similarity of TF-IDF vectors to build the graph while the clustering takes semantic similarity into account. 2) The clustering performed a global selection to form a summary by grouping similar phrases and ranking them by the number of covered students (similar to what the human did). Compared to LexRank, our approach captures the student coverage explicitly. While modifying LexRank by using semantic similarity is possible, estimating the student coverage is not straightforward.

Second, OriMEAD tends to select long sentences, resulting in a high recall but a low precision. The phrase version (MEAD) improves both the P and F scores by removing unnecessary parts in the original sentences.

Lastly, the proposed method outperforms the MMR based baselines on the precision and F-measure of all three ROUGE scores. We observed that the MMR baselines suffer from the issue of diverse expressions used the students (e.g., "graphs" and "charts").

6 Conclusion

In this paper, we presented a novel application to summarize student feedback to reflection prompts by a combination of phrase extraction, phrase clustering and phrase ranking. It makes use of metric clustering to rank the phrases by their student coverage, taking the information source into account. Experimental results demonstrate the good effectiveness of the model. While the proposed method improved the performance against MMR, other summarization methods without an additional MMR component do exist, in-

[5]The default Length parameter in MEAD is changed to 1 from its default value 9 and the position feature is removed, yielding better performance.

[6]For each MMR based baseline, the parameter is optimized with a grid search on the development data set.

cluding SumBasic (Vanderwende et al., 2007), KLSUM and TopicSUM (Haghighi and Vanderwende, 2009). An initial experiment shows they do not yield better performance with default parameters. However, we will revisit it since these methods are meant for full sentences and are not optimized within the phrase framework.

In the future, we plan to have additional annotation to evaluate the relative importance using the student coverage numbers. We also deployed CourseMIRROR in a statistics class in Spring 2015 and have created gold-standard summaries, which will allow us to both replicate the intrinsic evaluation of this paper with a new and larger dataset as well conduct an extrinsic evaluation beyond ROUGE scores. Finally, we are interested in applying our summarization approach to other types of user-generated content from mobile-applications (e.g., review comments).

Acknowledgments

This research is partially supported by an internal grant from the Learning Research and Development Center at the University of Pittsburgh. We thank Muhsin Menekse for providing the data set. We thank Jingtao Wang and Xiangmin Fan for developing the CourseMIRROR application and for valuable suggestions about the proposed summarization algorithm. We also thank anonymous reviewers for insightful comments and suggestions.

References

Sumit Basu, Chuck Jacobs, and Lucy Vanderwende. 2013. Powergrading: a clustering approach to amplify human effort for short answer grading. *Transactions of the Association for Computational Linguistics*, 1:391–402.

David Boud, Rosemary Keogh, David Walker, et al. 2013. *Reflection: Turning experience into learning*. Routledge.

Jaime Carbonell and Jade Goldstein. 1998. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '98, pages 335–336. ACM.

Ronan Collobert. 2011. Deep learning for efficient discriminative parsing. In *International Conference on Artificial Intelligence and Statistics*, number EPFL-CONF-192374.

Dan Ştefănescu, Rajendra Banjade, and Vasile Rus. 2014. Latent semantic analysis models on wikipedia and tasa. In *The 9th Language Resources and Evaluation Conference (LREC 2014)*, pages 26–31.

Günes Erkan and Dragomir R. Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *J. Artif. Int. Res.*, 22(1):457–479, December.

Xiangmin Fan, Wencan Luo, Muhsin Menekse, Diane Litman, and Jingtao Wang. 2015. CourseMIRROR: Enhancing large classroom instructor-student interactions via mobile interfaces and natural language processing. In *Works-In-Progress of ACM Conference on Human Factors in Computing Systems*. ACM.

James Gung and Jugal Kalita. 2012. Summarization of historical articles using temporal event clustering. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL HLT '12, pages 631–635.

Aria Haghighi and Lucy Vanderwende. 2009. Exploring content models for multi-document summarization. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 362–370, Boulder, Colorado, June.

Kazi Saidul Hasan and Vincent Ng. 2014. Automatic keyphrase extraction: A survey of the state of the art. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 1262–1273, June.

Leonard Kaufman and Peter Rousseeuw. 1987. Clustering by means of medoids. *Statistical Data Analysis Based on the L1-Norm and Related Method*, pages 405–416.

Yanran Li and Sujian Li. 2014. Query-focused multi-document summarization: Combining a topic model with graph-based semi-supervised learning. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 1197–1207, August.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, pages 74–81.

Zhiyuan Liu, Peng Li, Yabin Zheng, and Maosong Sun. 2009. Clustering to find exemplar terms for keyphrase extraction. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, EMNLP '09, pages 257–266.

Wencan Luo, Xiangmin Fan, Muhsin Menekse, Jingtao Wang, and Diane Litman. 2015. Enhancing instructor-student and student-student interactions with mobile interfaces and summarization. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 16–20, June.

Olena Medelyan, Eibe Frank, and Ian H. Witten. 2009. Human-competitive tagging using automatic keyphrase extraction. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 3*, EMNLP '09, pages 1318–1327.

Muhsin Menekse, Glenda Stump, Stephen J. Krause, and Michelene T.H. Chi. 2011. The effectiveness of students daily reflections on learning in engineering context. In *Proceedings of the American Society for Engineering Education (ASEE) Annual Conference*, Vancouver, Canada.

Dragomir R. Radev, Hongyan Jing, Małgorzata Styś, and Daniel Tam. 2004. Centroid-based summarization of multiple documents. *Inf. Process. Manage.*, 40(6):919–938, November.

Vasile Rus, Mihai C Lintean, Rajendra Banjade, Nobal B Niraula, and Dan Stefanescu. 2013. Semilar: The semantic similarity toolkit. In *ACL (Conference System Demonstrations)*, pages 163–168.

Yoshihiro Ueda, Mamiko Oka, Takahiro Koyama, and Tadanobu Miyauchi. 2000. Toward the "at-a-glance" summary: Phrase-representation summarization method. In *Proceedings of the 18th Conference on Computational Linguistics - Volume 2*, COLING '00, pages 878–884.

Gerard Van den Boom, Fred Paas, Jeroen JG Van Merrienboer, and Tamara Van Gog. 2004. Reflection prompts and tutor feedback in a web-based learning environment: effects on students' self-regulated learning competence. *Computers in Human Behavior*, 20(4):551 – 567.

Lucy Vanderwende, Hisami Suzuki, Chris Brockett, and Ani Nenkova. 2007. Beyond sumbasic: Task-focused summarization with sentence simplification and lexical expansion. *Inf. Process. Manage.*, 43(6):1606–1618, November.

Xiaojun Wan and Jianwu Yang. 2008. Multi-document summarization using cluster-based link analysis. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '08, pages 299–306.

Yi-fang Brook Wu, Quanzhi Li, Razvan Stefan Bot, and Xin Chen. 2005. Domain-specific keyphrase extraction. In *Proceedings of the 14th ACM International Conference on Information and Knowledge Management*, CIKM '05, pages 283–284.

Rui Yang, Zhan Bu, and Zhengyou Xia. 2012. Automatic summarization for chinese text using affinity propagation clustering and latent semantic analysis. In *Proceedings of the 2012 International Conference on Web Information Systems and Mining*, WISM'12, pages 543–550.

Benyu Zhang, Hua Li, Yi Liu, Lei Ji, Wensi Xi, Weiguo Fan, Zheng Chen, and Wei-Ying Ma. 2005. Improving web search results using affinity graph. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '05, pages 504–511. ACM.

Xiaojin Zhu, Andrew Goldberg, Jurgen Van Gael, and David Andrzejewski. 2007. Improving diversity in ranking using absorbing random walks. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics*, pages 97–104, April.

Extractive Summarization by Maximizing Semantic Volume

Dani Yogatama
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213, USA
dyogatama@cs.cmu.edu

Fei Liu
Electrical Engineering & Computer Science
University of Central Florida
Orlando, FL 32816, USA
feiliu@cs.ucf.edu

Noah A. Smith
Computer Science & Engineering
University of Washington
Seattle, WA 98195, USA
nasmith@cs.washington.edu

Abstract

The most successful approaches to extractive text summarization seek to maximize bigram coverage subject to a budget constraint. In this work, we propose instead to maximize semantic volume. We embed each sentence in a semantic space and construct a summary by choosing a subset of sentences whose convex hull maximizes volume in that space. We provide a greedy algorithm based on the Gram-Schmidt process to efficiently perform volume maximization. Our method outperforms the state-of-the-art summarization approaches on benchmark datasets.

1 Introduction

In artificial intelligence, changes in representation sometimes suggest new algorithms. For example, increased attention to distributed meaning representations suggests that existing combinatorial algorithms for NLP might be supplanted by alternatives designed specifically for embeddings. In this work, we consider summarization.

Classical approaches to extractive summarization represent each sentence as a bag of terms (typically bigrams) and seek a subset of sentences from the input document(s) that either (a) trade off between high relevance and low redundancy (Carbonell and Goldstein, 1998; McDonald, 2007), or (b) maximize bigram coverage (Yih et al., 2007; Gillick et al., 2008). The sentence representation is fundamentally *discrete*, and a range of greedy (Carbonell and Goldstein, 1998), approximate (Almeida and Martins, 2013), and exact optimization algorithms (McDonald, 2007; Martins and Smith, 2009; Berg-Kirkpatrick et al., 2011) have been proposed.

Recent studies have explored *continuous* sentence representations, including the paragraph vector (Le and Mikolov, 2014), a convolutional neural network architecture (Kalchbrenner et al., 2014), and a dictionary learning approach (Jenatton et al., 2011). If sentences are represented as low-dimensional embeddings in a distributed semantic space, then we begin to imagine a geometric relationship between a summary and a document. We propose that the **volume** of a summary (i.e., the semantic subspace spanned by the selected sentences) should ideally be large. We therefore formalize a new objective function for summarization based on semantic volume (§2), and we provide a fast greedy algorithm that can be used to maximize it (§3). We show that our method outperforms competing extractive baselines under similar experimental conditions on benchmark summarization datasets (§4).

2 Extractive Summarization Models

Assume we are given a set of N sentences: $\mathcal{D} = \{s_1, s_2, \ldots, s_N\}$ from one or many documents, and the goal is to produce a summary by choosing a subset $\mathcal{S}$ of M sentences, where $\mathcal{S} \subseteq \mathcal{D}$ and $M \leq N$, and the length of the summary is less than or equal to L words. In this work, we assume no summaries are available as training data. Denote a binary indicator vector $\mathbf{y} \in \mathbb{R}^N$, where sentence i is included if and only if $y_i = 1$ and 0 otherwise. Extractive summarization can be written as an optimization problem:

$$\begin{aligned} &\max && \text{score}(\mathcal{S}) = \text{score}(\mathcal{D}, \mathbf{y}) \\ &\text{with respect to} && \mathcal{S} \text{ equivalently } \mathbf{y} \\ &\text{subject to} && \text{length}(\mathcal{S}) \leq L \end{aligned}$$

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1961–1966,
Lisbon, Portugal, 17-21 September 2015.

with a scoring function $\text{score}(\mathcal{D}, \mathbf{y})$. A good scoring function should assign higher scores to better summaries. In the following, we describe two commonly used scoring functions and our proposed scoring function.

2.1 Maximal Marginal Relevance

The Maximal Marginal Relevance (MMR) method (Carbonell and Goldstein, 1998) considers the following scoring function:

$$\text{score}(\mathcal{D}, \mathbf{y}) = \sum_{i=1}^{N} y_i \text{Rel}(s_i) - \sum_{i,j=1}^{N} y_i y_j \text{Sim}(s_i, s_j)$$

where $\text{Rel}(s_i)$ measures the relevancy of sentence i and $\text{Sim}(s_i, s_j)$ measures the (e.g., cosine) similarity between sentence i and sentence j. The intuition is to choose sentences that are highly relevant to the document(s) and avoid redundancy. The above maximization problem has been shown to be NP-hard, solvable exactly using ILP (McDonald, 2007). A greedy algorithm that approximates the global solution by adding one sentence at a time to maximize the overall score (Lin and Bilmes, 2010) is often used in practice.

2.2 Coverage-Based Summarization

Another popular scoring function aims to give higher scores for covering more diverse concepts in the summary. Gillick et al. (2008) use bigrams as a surrogate for concepts. Following convention, we extract bigrams from each sentence $s_i \in \mathcal{D}$. Denote the number of unique bigrams extracted from all sentences by B. We introduce another binary vector $\mathbf{z} \in \mathbb{R}^B$ to indicate the presence or absence of a bigram in the summary, and a binary indicator matrix $\mathbf{M} \in \mathbb{R}^{N \times B}$, where $m_{i,j}$ is 1 if and only if bigram j is present in sentence i and 0 otherwise. The scoring function is:

$$\text{score}(\mathcal{D}, \mathbf{y}, \mathbf{z}) = \sum_{j=1}^{B} b_j z_j$$

and the two additional constraints are:

$$\forall j \in [B], \forall i \in [N] \qquad y_i m_{i,j} \leq z_j$$

$$\forall j \in [B] \qquad \sum_{i=1}^{N} y_i m_{i,j} \geq z_j$$

where we use $[B]$ as a shorthand for $\{1, 2, \ldots, B\}$. The first constraint makes sure that selecting a sentence implies selecting all its bigrams, whereas the second constraint makes sure that selecting a bigram implies selecting at least one of the sentences that contains it. In this formulation, there is no explicit penalty on redundancy. However, insofar as redundant sentences cover fewer bigrams, they are implicitly discouraged. Although the above scoring function also results in an NP-hard problem, an off-the-shelf ILP solver (Gillick et al., 2008) or a dual decomposition algorithm (Almeida and Martins, 2013) can be used to solve it in practice.

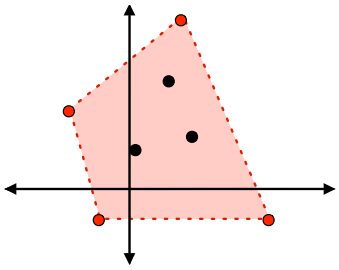


Figure 1: A toy example of seven sentences projected into a two-dimensional semantic space. Consider the case when the maximum summary length is four sentences. Our scoring function is optimized by chooseing the four sentences in red as the summary, since they maximize the volume (area in two dimensions).

2.3 Semantic Volume

We introduce a new scoring function for summarization. The main idea is based on the notion of coverage, but in a distributed semantic space: a good summary should have broad semantic coverage with respect to document contents. For every sentence $s_i, i \in [N]$, we denote its continuous semantic representation in a K-dimensional semantic space by $\Omega(s_i) = \mathbf{u}_i \in \mathbb{R}^K$, where Ω is a function that takes a sentence and returns its semantic vector representation. We denote embeddings of all sentences in $\mathcal{D}$ with the function Ω by $\Omega(\mathcal{D})$. We will return to the choice of Ω later. We propose to use a scoring function that maximizes the *volume* of selected sentences in this semantic space:

$$\text{score}(\mathcal{D}, \mathbf{y}) = \text{Volume}(\Omega(\mathcal{D}), \mathbf{y}) = \text{Volume}(\Omega(\mathcal{S}))$$

In the case when $K = 2$, this scoring function maximizes the area of a polytope, as illustrated in Figure 1. In the example, there exists a maximum number of sentences that can be selected such that adding more sentences does not increase the score, i.e., the set of selected sentences forms a convex hull of the set of all sentences. The sentences forming a convex hull may together be longer than

L words, so we seek to maximize the volume of the summary under this constraint.

There are many choices of Ω that we can use to produce sentence embeddings. As an exploratory study, we construct a vector of bigrams for each sentence, that is, $\mathbf{s}_i \in \mathbb{R}^B, \forall i \in [N]$. If bigram b is present in s_i, we let $s_{i,b}$ be the number of documents in the corpus that contain bigram b, and zero otherwise. We stack these vectors in columns to produce a matrix $\mathbf{S} \in \mathbb{R}^{N \times B}$, where N is the number of sentences in the corpus and B is the number of bigrams. We then perform singular value decomposition (SVD) on $\mathbf{S} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$. We use $\mathbf{U}_K \in \mathbb{R}^{N \times K}$ as the sentence representations, where K is a parameter that specifies the number of latent dimensions. Instead of performing SVD, we can also take $\mathbf{s}_i \in \mathbb{R}^B$ as our sentence representation, which makes our method resemble the bigram coverage-based summarization approach. However, this makes $\mathbf{s}_i$ a very sparse vector. Projecting to a lower dimensional space makes sense to allow the representation to incorporate information from (bigram) cooccurrences and share information across bigrams.

3 Volume Maximization

Given the semantic coverage scoring function in §2.3, our optimization problem is:

$$\begin{array}{ll} \max & \text{score}(\mathcal{S}) = \text{Volume}(\Omega(\mathcal{S})) \\ \text{with respect to} & \mathcal{S} \\ \text{subject to} & \text{length}(\mathcal{S}) \leq L \end{array}$$

For computational considerations, we propose to use a greedy algorithm that approximates the solution by iteratively adding a sentence that maximizes the current semantic coverage, given that the length constraint is still satisfied. The main steps in our algorithm are as follows. We first find the sentence that is farthest from the cluster centroid and add it to $\mathcal{S}$. Next, we find the sentence that is farthest from the first sentence and add it to $\mathcal{S}$. Given a set of already selected sentences, we choose the next one by finding the sentence farthest from the subspace spanned by sentences already in the set. We repeat this process until we have gone through all sentences, breaking ties arbitrarily and checking whether adding a sentence to $\mathcal{S}$ will result in a violation of the length constraint. This method is summarized in Algorithm 1. We note that related variants of our method for maximizing volume have appeared in other applications, such as remote sensing (Nascimento and Dias, 2005; Gomez et al., 2007) and topic modeling (Arora et al., 2012; Arora et al., 2013).

Algorithm 1 Greedy algorithm for approximately maximizing the semantic volume given a budget constraint.

Input: Budget constraint L, sentence representations $\mathcal{R} = \{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_N\}$
$\mathcal{S} = \{\}, \mathcal{B} = \{\}$
Compute the cluster centroid $\mathbf{c}$: $\frac{1}{N}\sum_{i=1}^{N} \mathbf{u}_i$.
$p \leftarrow$ index of sentence that is farthest from $\mathbf{c}$.
$\mathcal{S} = \mathcal{S} \cup \{s_p\}$. ▶ add first sentence
$q \leftarrow$ index of sentence that is farthest from $\mathbf{s}_p$.
$\mathcal{S} = \mathcal{S} \cup \{s_q\}$. ▶ add second sentence
$\mathbf{b}_0 = \frac{\mathbf{u}_q}{\|\mathbf{u}_q\|}, \mathcal{B} = \mathcal{B} \cup \{\mathbf{u}_0\}$
total length = length(s_p) + length(s_q)
for $i = 1, \ldots, N-2$ **do**
  $r \leftarrow$ index of sentence that is farthest from the subspace of Span($\mathcal{B}$). ▶ see text
  if total length + length(s_r) $\leq L$ **then**
    $\mathcal{S} = \mathcal{S} \cup \{s_r\}$.
    $\mathbf{b}_r = \frac{\mathbf{u}_r}{\|\mathbf{u}_r\|}, \mathcal{B} = \mathcal{B} \cup \{\mathbf{b}_r\}$.
    total length = total length + length(s_r)
  end if
end for

Computing Distance to a Subspace Our algorithm involves finding a point farthest from a subspace (except for the first and second sentences, which can be selected by computing pointwise distances). In order for this algorithm to be efficient, we need this operation to be fast, since it is executed frequently. There are several established methods to compute the distance between a point to a subspace spanned by sentences in $\mathcal{S}$. For completeness, we describe one method based on the Gram-Schmidt process (Laplace, 1812) here.

We maintain a set of basis vectors, denoted by $\mathcal{B}$. Our first basis vector consists of one element: $\mathbf{b}_0 = \frac{\mathbf{u}_q}{\|\mathbf{u}_q\|}$, where q is the second sentence chosen above. Next, we project each candidate sentence i to this basis vector:

$$\text{Proj}_{\mathbf{b}_0}(\mathbf{u}_i) = (\mathbf{u}_i^\top \mathbf{b}_0)\mathbf{b_0},$$

and find the distance by computing $\text{Distance}(\mathbf{u}_i, \mathcal{B}) = \|\mathbf{u}_i - \text{Proj}_{\mathbf{b}_0}(\mathbf{u}_i)\|$. Once we find the farthest sentence r, we add a new basis vector $\mathcal{B} = \mathcal{B} \cup \{\mathbf{b}_r\}$, where $\mathbf{b}_r = \frac{\mathbf{u}_r}{\|\mathbf{u}_r\|}$ and repeat this process. When there are more than one

basis vectors, we find the distance by computing:

$$\text{Distance}(\mathbf{u}_i, \mathcal{B}) = \left\| \mathbf{u}_i - \sum_{\mathbf{b}_j \in \mathcal{B}} \text{Proj}_{\mathbf{b}_j}(\mathbf{u}_i) \right\|.$$

4 Experiments

4.1 Setup

We evaluate our proposed method on the non-update portion of TAC-2008 and TAC-2009. The datasets contain 48 and 44 multi-document summarization problems, respectively. Each problem has 10 news articles as input; each is to be summarized in a maximum of $L = 100$ words. There are 4 human reference summaries for each problem, against which an automatically generated summary is compared. We compare our method with two baselines: Maximal Marginal Relevance (MMR, §2.1) and the coverage-based summarization method (CBS, §2.2). ROUGE (Lin, 2004) is used to evaluate the summarization results.

For preprocessing, we tokenize, stem with the Porter (1980) stemmer, and split documents into sentences. We remove bigrams consisting of only stopwords and bigrams which appear in less than 3 sentences. As a result, we have 2,746 and 3,273 bigrams for the TAC-2008 and TAC-2009 datasets respectively. Unlabeled data can help generate better sentence representations. For each summarization problem in each dataset, we use other problems in the same dataset as unlabeled data. We concatenate every problem in each dataset and perform SVD on this matrix (§2.3). Note that this also means we only need to do one SVD for each dataset.

4.2 Results

Table 1 shows results on the TAC-2008 and TAC-2009 datasets. We report results for our method with $K = 500$ (Volume 500), and $K = 600$ (Volume 600). We also include results for an oracle model that has access to the human reference summaries and extracts sentences that maximize bigram recall as an upper bound. Similar to previous findings, CBS is generally better than MMR. Our method outperforms other competing methods, although the optimal value of K is different in each dataset. The improvements with our proposed approach are small in terms of R-2. This is likely because the R-2 score computes bigram overlaps, and the CBS method that directly maximizes bigram coverage is already a resonable approach to optimizing this metric (although still worse than the best of our methods).

Methods	TAC-2008		TAC-2009	
	R-1	R-2	R-1	R-2
MMR	34.08	9.30	31.87	7.99
CBS	35.83	9.43	32.70	8.84
Volume 500	37.40	9.17	34.08	**8.91**
Volume 600	**37.50**	**9.58**	**34.37**	8.76
Oracle	46.06	19.33	46.77	16.99

Table 1: Results on the TAC-2008 and TAC-2009 datasets. "Volume" refers to our method, shown with two embedding sizes.

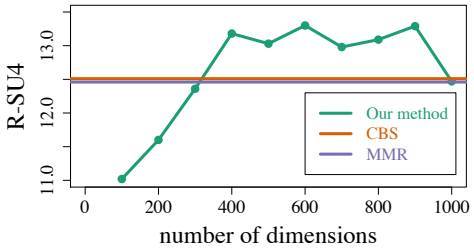


Figure 2: R-SU4 scores as we vary the number of dimensions (K) on the TAC-2008 datasets.

5 Discussion

Runtime comparisons In terms of inference running time, all methods perform reasonably fast. MMR is the slowest, on average it takes 0.38 seconds per problem, followed by our method at 0.17 seconds per problem, and CBS at 0.15 seconds per problem. However, our implementations of MMR and Algorithm 1 are in Python, whereas we use an optimzed solver from Gurobi for our CBS baseline. For preprocessing, our method is the slowest, since we need to compute sentence embeddings using SVD. There are about 10,000 sentences and 3,000 bigrams for each dataset. SVD takes approximately 2.5 minutes (150 seconds) using Matlab on our 12-core machine with 24GB RAM. Our method introduces another hyperparameter, the number of latent dimensions K for sentence embeddings. We observe that the optimal value depends on the dataset, although a value in the range of 400 to 800 seems best. Figure 2 shows R-SU4 scores on the TAC-2008 dataset as we vary K.

Other sentence projection methods We use SVD in this study for computing sentence embeddings. As mentioned previously, our summariza-

tion approach can benefit from advances in neural-network-based sentence representations (Jenatton et al., 2011; Le and Mikolov, 2014; Kalchbrenner et al., 2014). These models can also produce vector representations of sentences, so Algorithm 1 can be readily applied to the learned representations. Our work opens up a possibility to make summarization a future benchmark task for evaluating the quality of sentence representations.

Our method is related to determinantal point processes (DPPs; Gillenwater et al., 2012; Kulesza and Taskar, 2012) in that they both seek to maximize the volume spanned by sentence vectors to produce a summary. In DPP-based approaches, quality and selectional diversity correspond to vector magnitude and angle respectively. In this work, the length of a sentence vector is not tailored to encode quality in terms of representativeness directly. In contrast, we rely on sentence embedding methods to produce a semantic space and assume that a good summary should have a large volume in the semantic space. We show that a simple singular value decomposition embedding of sentences—one that is not especially tuned for this task—produces reasonably good results. We leave exploration of other sentence embedding methods to future work.

Future work Our method could be extended for compressive summarization, by simply including compressed sentences in the embedded space and running Algorithm 1 without any change. This resembles the summarization methods that jointly extracts and compresses (Berg-Kirkpatrick et al., 2011; Woodsend and Lapata, 2012; Almeida and Martins, 2013). Another alternative is a pipeline approach, where extractive summarization is followed or preceded by a sentence compression module, which can be built and tuned independent of our proposed extractive method (Knight and Marcu, 2000; Lin, 2003; Zajic et al., 2007; Wang et al., 2013; Li et al., 2013).

We are also interested in exploring volume as a relevance function within MMR. MMR avoids redundancy by penalizing redundant sentences, whereas in our method semantic redundancy is inherently discouraged since the method chooses sentences to maximize volume. Depending on the method used to embed sentences, this might not translate directly into avoiding n-gram redundancy. Plugging our scoring function to an MMR objective is a simple way to enforce diversity.

Finally, an interesting future direction is finding an exact tractable solution to the volume maximization problem (or demonstrating that one does not exist).

6 Conclusion

We introduced a summarization approach based on maximizing volume in a semantic vector space. We showed an algorithm to efficiently perform volume maximization in this semantic space. We demonstrated that our method outperforms existing state-of-the-art extractive methods on benchmark summarization datasets.

Acknowledgments

We thank anonymous reviewers for helpful suggestions. This work was supported by the Defense Advanced Research Projects Agency through grant FA87501420244 and by NSF grant SaTC-1330596. This work was completed while the authors were at CMU.

References

Miguel B. Almeida and Andre F. T. Martins. 2013. Fast and robust compressive summarization with dual decomposition and multi-task learning. In *Proc. of ACL*.

Sanjeev Arora, Rong Ge, Ravi Kannan, and Ankur Moitra. 2012. Computing a nonnegative matrix factorization – provably. In *Proc. of STOC*.

Sanjeev Arora, Rong Ge, Yoni Halpren, David Mimno, Ankur Moitra, David Sontag, Yichen Wu, and Michael Zu. 2013. A practical algorithm for topic modeling with provable guarantees. In *Proc. of ICML*.

Taylor Berg-Kirkpatrick, Dan Gillick, and Dan Klein. 2011. Jointly learning to extract and compress. In *Proc. of ACL*.

Jaime Carbonell and Jade Goldstein. 1998. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *Proc. of SIGIR*.

Jennifer Gillenwater, Alex Kulesza, and Ben Taskar. 2012. Discovering diverse and salient threads in document collections. In *Proc. of EMNLP-CoNLL*.

Dan Gillick, Benoit Favre, and Dilek Hakkani-Tur. 2008. The ICSI summarization system at TAC 2008. In *Proc. of TAC*.

C. Gomez, H. Le Borgne, P. Allemand, C. Delacourt, and P. Ledru. 2007. N-findr method versus independent component analysis for lithological identification in hyperspectral imagery. *International Journal of Remote Sensing*, 28(23):5315–5338.

Rodolphe Jenatton, Julien Mairal, Gullaume Obozinski, and Francis Bach. 2011. Proximal methods for hierarchical sparse coding. *Journal of Machine Learning Research*, 12:2297–2334.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proc. of ACL*.

Kevin Knight and Daniel Marcu. 2000. Statistics-based summarization – step one: Sentence compression. In *Proc. of AAAI*.

Alex Kulesza and Ben Taskar. 2012. Determinantal point processes for machine learning. *Foundations and Trends in Machine Learning*, 5(2–3):123–286.

Pierre-Simon Laplace. 1812. *Theorie analytique des probabilites*. Courcier, Paris.

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

Chen Li, Fei Liu, Fuliang Weng, and Yang Liu. 2013. Document summarization via guided sentence compression. In *Proc. of EMNLP*.

Hui Lin and Jeff Bilmes. 2010. Multi-document summarization via budgeted maximization of submodular functions. In *Proc. of NAACL-HLT*.

Chin-Yew Lin. 2003. Improving summarization performance by sentence compression – A pilot study. In *Proc. of Workshop on Information Retrieval with Asian Language*.

Chin-Yew Lin. 2004. Rouge: a package for automatic evaluation of summaries. In *Proc. of the ACL Workshop on Text Summarization Branches Out*.

Andre F. T. Martins and Noah A. Smith. 2009. Summarization with a joint model for sentence extraction and compression. In *Proc. of the ACL Workshop on Integer Linear Programming for Natural Language Processing*.

Ryan McDonald. 2007. A study of global inference algorithms in multi-document summarization. In *Proc. of ECIR*.

Jose M. P. Nascimento and Jose M. Bioucas Dias. 2005. Vertex component analysis: A fast algorithm to unmix hyperspectral data. *Proc. of IEEE Transaction on Geoscience and Remote Sensing*, 43(4):898–910.

M.F. Porter. 1980. An algorithm for suffix stripping. *Program*, 14(3):130–137.

Lu Wang, Hema Raghavan Vittorio Castelli Radu Florian, and Claire Cardie. 2013. A sentence compression based framework to query-focused multidocument summarization. In *Proc. of ACL*.

Kristian Woodsend and Mirella Lapata. 2012. Multiple aspect summarization using integer linear programming. In *Proc. of EMNLP*.

Wen-Tau Yih, Joshua Goodman, Lucy Vanderwende, and Hisami Suzuki. 2007. Multi-document summarization by maximizing informative content-words. In *Proc. of IJCAI*.

David Zajic, Bonnie J. Dorr, Jimmy Lin, and Richard Schwartz. 2007. Multi-candidate reduction: Sentence compression as a tool for document summarization tasks. *Information Processing and Management*, 43(6):1549–1570.

LCSTS: A Large Scale Chinese Short Text Summarization Dataset

Baotian Hu **Qingcai Chen** **Fangze Zhu**

Intelligent Computing Research Center
Harbin Institute of Technology, Shenzhen Graduate School
{baotianchina,qingcai.chen, zhufangze123}@gmail.com

Abstract

Automatic text summarization is widely regarded as the highly difficult problem, partially because of the lack of large text summarization data set. Due to the great challenge of constructing the large scale summaries for full text, in this paper, we introduce a large corpus of Chinese short text summarization dataset constructed from the Chinese microblogging website Sina Weibo, which is released to the public[1]. This corpus consists of over 2 million real Chinese short texts with short summaries given by the author of each text. We also manually tagged the relevance of 10,666 short summaries with their corresponding short texts. Based on the corpus, we introduce recurrent neural network for the summary generation and achieve promising results, which not only shows the usefulness of the proposed corpus for short text summarization research, but also provides a baseline for further research on this topic.

1 Introduction

Nowadays, individuals or organizations can easily share or post information to the public on the social network. Take the popular Chinese microblogging website (Sina Weibo) as an example, the People's Daily, one of the media in China, posts more than tens of weibos (analogous to tweets) each day. Most of these weibos are well-written and highly informative because of the text length limitation (less than140 Chinese characters). Such data is regarded as naturally annotated web resources (Sun, 2011). If we can mine these high-quality data from these naturally annotated web resources, it will be beneficial to the research that has been hampered by the lack of data.

【江西高考被曝替考 有关考生已被警方控制】人民日报记者吴齐强消息，江西高考被曝光替考，7日中午江西省教育厅发布消息称，接到有人组织替考的举报后，江西省教育厅、江西省教育考试院立即部署南昌市教育考试院，联合南昌市警方开展调查核实，有关考生已被警方控制。调查进展情况将及时向社会公布。

Figure 1: A Weibo Posted by People's Daily.

In the Natural Language Processing (NLP) community, automatic text summarization is a hot and difficult task. A good summarization system should understand the whole text and re-organize the information to generate coherent, informative, and significantly short summaries which convey important information of the original text (Hovy and Lin, 1998), (Martins, 2007). Most of traditional abstractive summarization methods divide the process into two phrases (Bing et al., 2015). First, key textual elements are extracted from the original text by using unsupervised methods or linguistic knowledge. And then, unclear extracted components are rewritten or paraphrased to produce a concise summary of the original text by using linguistic rules or language generation techniques. Although extensive researches have been done, the linguistic quality of abstractive summary is still far from satisfactory. Recently, deep learning methods have shown potential abilities to learn representation (Hu et al., 2014; Zhou et al., 2015) and generate language (Bahdanau et al., 2014; Sutskever et al., 2014) from large scale data by utilizing GPUs. Many researchers realize that we are closer to generate abstractive summarizations by using the deep learning methods. However, the publicly available and high-quality large scale summarization data set is still very rare and not easy to be constructed manually. For example, the popular document summarization dataset DUC[2], TAC[3] and TREC[4] have only hundreds of human written English text summarizations. The problem is even worse for Chinese. In this pa-

[1] http://icrc.hitsz.edu.cn/Article/show/139.html

[2] http://duc.nist.gov/data.html

[3] http://www.nist.gov/tac/2015/KBP/

[4] http://trec.nist.gov/

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1967–1972,
Lisbon, Portugal, 17-21 September 2015.

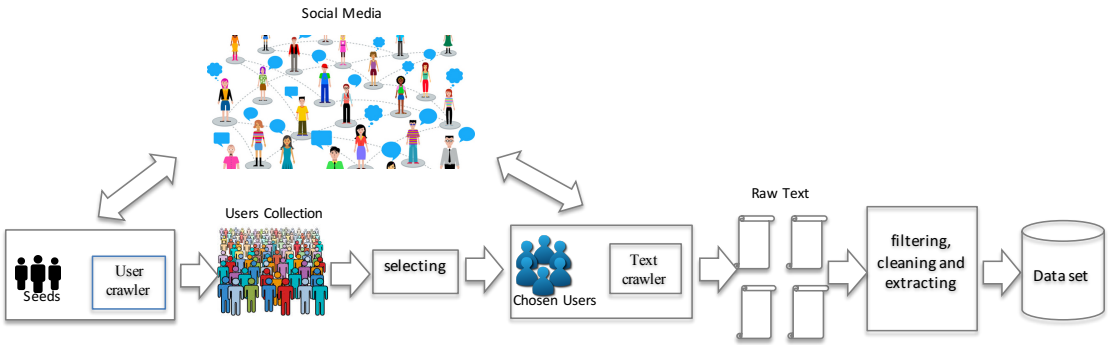


Figure 2: Diagram of the process for creating the dataset.

per, we take one step back and focus on constructing **LCSTS**, the **L**arge-scale **C**hinese **S**hort **T**ext **S**ummarization dataset by utilizing the naturally annotated web resources on Sina Weibo. Figure 1 shows one weibo posted by the People's Daily. In order to convey the import information to the public quickly, it also writes a very informative and short summary (in the blue circle) of the news. Our goal is to mine a large scale, high-quality short text summarization dataset from these texts.

This paper makes the following contributions: (1) We introduce a large scale Chinese short text summarization dataset. To our knowledge, it is the largest one to date; (2) We provide standard splits for the dataset into large scale training set and human labeled test set which will be easier for benchmarking the related methods; (3) We explore the properties of the dataset and sample 10,666 instances for manually checking and scoring the quality of the dataset; (4) We perform recurrent neural network based encoder-decoder method on the dataset to generate summary and get promising results, which can be used as one baseline of the task.

2 Related Work

Our work is related to recent works on automatic text summarization and natural language processing based on naturally annotated web resources, which are briefly introduced as follows.

Automatic Text Summarization in some form has been studied since 1950. Since then, most researches are related to extractive summarizations by analyzing the organization of the words in the document (Nenkova and McKeown, 2011) (Luhn, 1998); Since it needs labeled data sets for supervised machine learning methods and labeling dataset is very intensive, some researches focused on the unsupervised methods (Mihalcea, 2004). The scale of existing data sets are usually very small (most of them are less than 1000). For example, DUC2002 dataset contains 567 documents and each document is provided with two 100-words human summaries. Our work is also related to the headline generation, which is a task to generate one sentence of the text it entitles. Colmenares et.al construct a 1.3 million financial news headline dataset written in English for headline generation (Colmenares et al., 2015). However, the data set is not publicly available.

Naturally Annotated Web Resources based Natural Language Processing is proposed by Sun (Sun, 2011). Naturally Annotated Web Resources is the data generated by users for communicative purposes such as web pages, blogs and microblogs. We can mine knowledge or useful data from these raw data by using marks generated by users unintentionally. Jure et.al track 1.6 million mainstream media sites and blogs and mine a set of novel and persistent temporal patterns in the news cycle (Leskovec et al., 2009). Sepandar et.al use the users' naturally annotated pattern 'we feel' and 'i feel' to extract the 'Feeling' sentence collection which is used to collect the world's emotions. In this work, we use the naturally annotated resources to construct the large scale Chinese short text summarization data to facilitate the research on text summarization.

3 Data Collection

A lot of popular Chinese media and organizations have created accounts on the Sina Weibo. They use their accounts to post news and information. These accounts are verified on the Weibo and labeled by a blue 'V'. In order to guarantee the quality of the crawled text, we only crawl the verified organizations' weibos which are more likely to be clean, formal and informative. There are a lot of human intervention required in each step. The process of the data collection is shown as Figure 2 and

summarized as follows:

1) We first collect 50 very popular organization users as seeds. They come from the domains of politic, economic, military, movies, game and etc, such as People's Daily, the Economic Observe press, the Ministry of National Defense and etc. 2) We then crawl fusers followed by these seed users and filter them by using human written rules such as the user must be blue verified, the number of followers is more than 1 million and etc. 3) We use the chosen users and text crawler to crawl their weibos. 4) we filter, clean and extract (short text, summary) pairs. About 100 rules are used to extract high quality pairs. These rules are concluded by 5 peoples via carefully investigating of the raw text. We also remove those paris, whose short text length is too short (less than 80 characters) and length of summaries is out of [10,30].

4 Data Properties

The dataset consists of three parts shown as Table 1 and the length distributions of texts are shown as Figure 3.

Part I is the main content of LCSTS that contains 2,400,591 (short text, summary) pairs. These pairs can be used to train supervised learning model for summary generation.

Part II contains the 10,666 human labeled (short text, summary) pairs with the score ranges from 1 to 5 that indicates the relevance between the short text and the corresponding summary. '1' denotes " the least relevant " and '5' denotes "the most relevant". For annotating this part, we recruit 5 volunteers, each pair is only labeled by one annotator. These pairs are randomly sampled from Part I and are used to analysize the distribution of pairs in the Part I. Figure 4 illustrates examples of different scores. From the examples we can see that pairs scored by 3, 4 or 5 are very relevant to the corresponding summaries. These summaries are highly informative, concise and significantly short compared to original text. We can also see that many words in the summary do not appear in the original text, which indicates the significant difference of our dataset from sentence compression datasets. The summaries of pairs scored by 1 or 2 are highly abstractive and relatively hard to conclude the summaries from the short text. They are more likely to be headlines or comments instead of summaries. The statistics show that the percent of score 1 and 2 is less than 20% of the data, which can be filtered by using trained classifier.

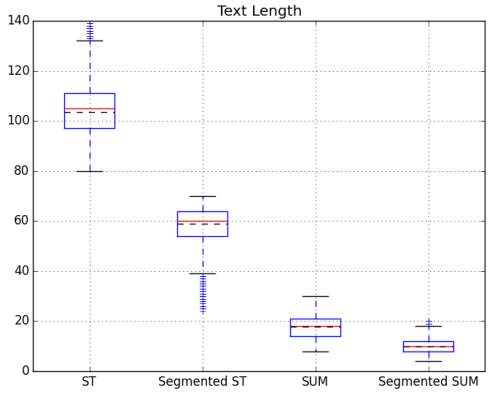


Figure 3: Box plot of lengths for short text(ST), segmented short text(Segmented ST), summary(SUM) and segmented summary(Segmented SUM). The red line denotes the median, and the edges of the box the quartiles.

Part III contains 1,106 pairs. For this part, 3 annotators label the same 2000 texts and we extract the text with common scores. This part is independent from Part I and Part II. In this work, we use pairs scored by 3, 4 and 5 of this part as the test set for short text summary generation task.

Part I	2,400,591	
Part II	Number of Pairs	10,666
	Human Score 1	942
	Human Score 2	1,039
	Human Score 3	2,019
	Human Score 4	3,128
	Human Score 5	3,538
Part III	Number of Pairs	1,106
	Human Score 1	165
	Human Score 2	216
	Human Score 3	227
	Human Score 4	301
	Human Score 5	197

Table 1: Data Statistics

5 Experiment

Recently, recurrent neural network (RNN) have shown powerful abilities on speech recognition (Graves et al., 2013), machine translation (Sutskever et al., 2014) and automatic dialog response (Shang et al., 2015). However, there is rare research on the automatic text summarization by using deep models. In this section, we use RNN as encoder and decoder to generate the summary of short text. We use the Part I as the training set

Short Text: 水利部水资源司司长陈明忠今日在新闻发布会上透露，根据刚刚完成的水资源管理制度的考核，有部分省接近了红线的指标，有部分省超过红线的指标。在一些超过红线的地方，将对一些取用水项目进行区域的限批，严格地进行水资源论证和取水许可的批准。

Mingzhong Chen, the Chief Secretary of the Water Devision of the Ministry of Water Resources, revealed today at a press conference, according to the just-completed assessment of water resources management system, some provinces are closed to the red line indicator, some provinces are over the red line indicator. In some places over the red line, It will enforce regional approval restrictions on some water projects, implement strictly water resources assessment and the approval of water licensing.

Summarization: 部分省超过年度用水红线指标 取水项目将被限批

Some provinces exceeds the red line indicator of annual water using, some water project will be. limited approved

Human Score: 5

Short Text: 各团购网站移动端销售额占比均在30%以下，用户通过PC端购物习惯短时间内难以转变。未来中国餐饮O2O市场，移动端将成为餐饮O2O的战略性发展方向，也将由线上驱动转变为线下驱动。一二线城市面临增长窘境，三四线城市O2O市场蕴含机会。

Groupons' sales on mobile terminals are below 30 percent. User's preference of shopping through PCs can not be changed in the short term. In the future Chinese O2O catering market, mobile terminals will become the strategic development direction. And also, it will become off-line driving from on-line driving. The first and second tier cities are facing growth difficulties. However, O2O market in the third and fourth tier cities contains opportunities.

Summarization: 移动端成餐饮O2O的战略性发展方向

The mobile terminals will become catering's strategic development direction.

Human Score: 4

Short Text: 7月百城住宅新建住宅平均价格为10347元/平方米，环比上涨0.87%，自去年6月以来连续14个月环比上涨。其中，广州、北京、深圳、南京涨幅均超过10%。中原张大伟认为，一二线城市因为集聚了过多资源，房价易涨难跌。

In July, 100-cities' average newly-built house prices is 10347 yuan per square, which rose 0.87%. It rises for the 14th consecutive month. Among them, Guangzhou, Beijing, Shenzhen, Nanjing rise more than 10%. Dawei Zhang, from Centaline Property Agency, said that because the first and second-tier city gathers too many resources, the price of house is likely to rise and hard to fall.

Summarization: 百城房价环比“14连涨”一二线城市涨幅扩大

100-cities' house prices gain “14th consecutive rising”, the first and second-tier cities rise more.

Human Score: 3

Short Text: 记者梳理发现，2009年至今有8起福彩开奖延迟事件，至少延迟2小时，2014年5月6日第2014050期延迟开奖达4小时。8起事件中福彩中心对其中3起给出了回应，理由有通讯故障及暴雨导致的数据上传延迟。另5起均未解释原因。

Reporters combed the information and found, from 2009 to now, there are at least 8 lottery delayed events and the delayed time are more than 2 hours. On May 6, 2014, the No. 2014050 delay more than 4 hours. The center of welfare lottery only respond to 3 of the 8 event. Their explanations are that a communications breakdown and heavy rain led to a data upload extension. There are no explanations for other 5 delay events.

Summarization:三问双色球开奖延迟：开奖为何要等数据汇总？

Ask about the lottery delay third times: why lottery should wait data collection?

Human Score: 2

Short Text: 商务部数据显示，中国7月实际利用外资同比大幅下降16.95%至78.1亿美元。外界有分析与近期官方对外资企业的密集反垄断调查有关。沈丹阳回应指出，“不能与对外资的反垄断调查挂钩，或者做其他没有根据的联想”。

According to China's Ministry of Commerce, China's actually utilized foreign capital in July fell sharply about 16.95% to 7.81 billion dollars, comparing to last year. Analysis of the outside world believe that it is related to the recent official intensive antitrust investigation. Danyang Shen responded, “It can not be linked to the antitrust investigation of foreign investment, or do other unfounded association”

Summarization: 商务部表态反垄断：几个案子不会把外商吓回去

China's Ministry of Commerce respond to antitrust investigation: Several cases will not scare foreign investors away.

Human Score: 1

Figure 4: Five examples of different scores.

and the subset of Part III, which is scored by 3, 4 and 5, as test set.

Two approaches are used to preprocess the data: 1) character-based method, we take the Chinese character as input, which will reduce the vocabulary size to 4,000. 2) word-based method, the

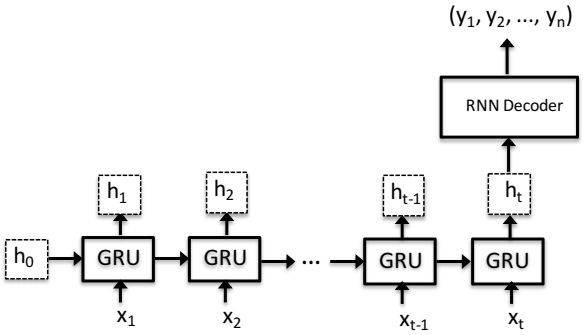


Figure 5: The graphical depiction of RNN encoder and decoder framework without context.

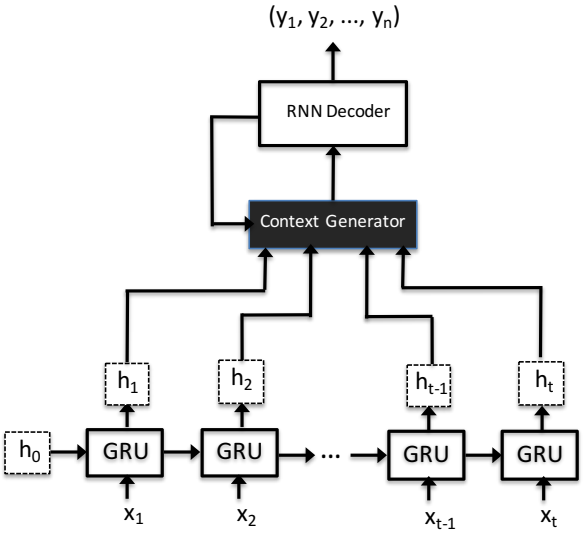


Figure 6: The graphical depiction of the RNN encoder and decoder framework with context.

text is segmented into Chinese words by using jieba[5]. The vocabulary is limited to 50,000. We adopt two deep architectures, 1) The local context is not used during decoding. We use the RNN as encoder and it's last hidden state as the input of decoder, as shown in Figure 5; 2) The context is used during decoding, following (Bahdanau et al., 2014), we use the combination of all the hidden states of encoder as input of the decoder, as shown in Figure 6. For the RNN, we adopt the gated recurrent unit (GRU) which is proposed by (Chung et al., 2015) and has been proved comparable to LSTM (Chung et al., 2014). All the parameters (including the embeddings) of the two architectures are randomly initialized and ADADELTA (Zeiler, 2012) is used to update the learning rate. After the model is trained, the beam search is used to generate the best summaries in the process of decoding and the size of beam is set to 10 in our experiment.

[5]https://pypi.python.org/pypi/jieba/

model	data	R-1	R-2	R-L
RNN	Word	0.043	0.025	0.043
	Char	0.061	0.028	0.057
RNN context	Word	0.087	0.054	0.085
	Char	**0.108**	**0.073**	**0.107**

Table 2: The experiment result: "Word" and "Char" denote the word-based and character-based input respectively.

Short Text: 9月3日，总部位于日内瓦的世界经济论坛发布了《2014-2015年全球竞争力报 告》，瑞士连续六年位居榜首，成为全球最具竞争力的国家，新加坡和美国分列第二位和第三位。中国排名第28 位，在金砖国家中排名最高。

On September 3, the Geneva-based World Economic Forum released " The Global Competitiveness Report 2014-2015 ". Switzerland topped the list for six consecutive years , becoming the world's most competitive country. Singapore and the United States are in the second and third place respectively. China is in the 28th place, ranking highest among the BRIC countries.

Human: 全球竞争力排行榜 中国居2 8 位 居金砖国家首位

the Global competitiveness ranking list, China is in the 28th place, the highest among BRIC countries.

RNN+Char: 瑞士总统9月30日在世界总统首次在中国开幕瑞士总裁2012年第一位第三大胜出国瑞士报告全球第六大国家5000亿美元

It is not a fluent sentence and can not be translated. There are only some segments related to original text.

RNN+Word: 球竞争力排名:瑞士居全球第3位中国第1位第1位第1位中国第1位瑞士第1位第1

Global competitiveness ranking: Switzerland ranks 3rd place, China 1st place, 1st place, China 1st place, Switzerland 1st place, first

RNN+Context+Char: 全球竞争力排名中国排名第28位金砖国最高

In the Global competitiveness ranking list, China is in the 28th place which is highest among the BRIC countries.

RNN+Context+Word: 全球竞争力报告出炉中国排名第28位金砖国家中排名第一 (第九

"The Global Competitiveness Report" is released. China is in the 28th place, rank highest among the BRIC countries (ninth

Figure 7: An example of the generated summaries.

For evaluation, we adopt the ROUGE metrics (Lin, 2004) proposed by (Lin and Hovy, 2003), which have been proved strongly correlated with human evaluations. ROUGE-1, ROUGE-2 and ROUGE-L are used. All the models are trained on the GPUs tesla M2090 for about one week.Table 2 lists the experiment results. As we can see in Figure 7, the summaries generated by RNN with context are very close to human written summaries, which indicates that if we feed enough data to the RNN encoder and decoder, it may generate summary almost from scratch.

The results also show that the RNN with context outperforms RNN without context on both character and word based input. This result indicates that the internal hidden states of the RNN encoder can be combined to represent the context of words in summary. And also the performances of the character-based input outperform the word-based input. As shown in Figure 8, the summary generated by RNN with context by inputing the character-based short text is relatively good, while the the summary generated by RNN with context

Short Text: 工厂，大门紧锁，约20名工人散坐在树荫下。"我们就是普通工人，在这里等工资。"其中一人说道。7月4日上午，记者抵达深圳龙华区清湖路上的深圳愿景光电子有限公司。正如传言一般，愿景光电子倒闭了，大股东邢毅不知所踪。

On The factory's door is locked. About 20 works are scattered to sit under the shade. "We are ordinary workers, we are waiting for our salary here." one of them said. On the morning of July 4th, reporters arrived at Shenzhen Yuanjing Photoelectron Corporation , located on Qinghu Road, Longhua District, Shenzhen. Just as the rumor, Yuanjing Photoelectron Corporation is closed down and the large shareholder Xing Yi is missing.

Human: 深圳亿元级LED企业倒闭烈日下工人苦等老板

Hundred-million-yuan-class LED enterprise is closed down and workers wait for the boss under the under the scorching sun.

RNN+Context+Char: 深圳愿景光电子倒闭了

Shenzhen Yuanjing PhotoElectron Corporation is closed down.

RNN+Context+Word: 深圳 UNK UNK UNK UNK UNK UNK UNK UNK UNK UNK UNK UNK Shenzhen UNK UNK UNK UNK UNK UNK UNK UNK UNK UNK UNK

Figure 8: An example of the generated summaries with UNKs.

on word-based input contains many UNKs. This may attribute to that the segmentation may lead to many UNKs in the vocabulary and text such as the person name and organization name. For example, "愿景光电子" is a company name which is not in the vocabulary of word-based RNN, the RNN summarizer has to use the UNKs to replace the "愿景光电子" in the process of decoding.

6 Conclusion and Future Work

We constructed a large-scale Chinese short text summarization dataset and performed RNN-based methods on it, which achieved some promising results. This is just a start of deep models on this task and there is much room for improvement. We take the whole short text as one sequence, this may not be very reasonable, because most of short texts contain several sentences. A hierarchical RNN (Li et al., 2015) is one possible direction. The rare word problem is also very important for the generation of the summaries, especially when the input is word-based instead of character-based. It is also a hot topic in the neural generative models such as neural translation machine(NTM) (Luong et al., 2014), which can benefit to this task. We also plan to construct a large document summarization data set by using naturally annotated web resources.

Acknowledgments

This work is supported by National Natural Science Foundation of China: 61473101, 61173075 and 61272383, Strategic Emerging Industry Development Special Funds of Shenzhen: JCYJ20140417172417105, JCYJ20140508161040764 and JCYJ20140627163809422. We thank to Baolin Peng, Lin Ma, Li Yu and the anonymous reviewers for their insightful comments.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Lidong Bing, Piji Li, Yi Liao, Wai Lam, Weiwei Guo, and Rebecca Passonneau. 2015. Abstractive multi-document summarization via phrase selection and merging. In *Proceedings of the ACL-IJCNLP*, pages 1587–1597, Beijing, China, July. Association for Computational Linguistics.

Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555.

Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. 2015. Gated feedback recurrent neural networks. *CoRR*, abs/1502.02367.

Carlos A. Colmenares, Marina Litvak, Amin Mantrach, and Fabrizio Silvestri. 2015. Heads: Headline generation as sequence prediction using an abstract feature-rich space. In *Proceddings of 2015 Conference of the North American Chapter of the Association for Computational Linguistics–Human Language Technologies (NAACL HLT 2015)*.

Alex Graves, Abdel-rahman Mohamed, and Geoffrey E. Hinton. 2013. Speech recognition with deep recurrent neural networks. *CoRR*, abs/1303.5778.

Eduard Hovy and Chin-Yew Lin. 1998. Automated text summarization and the summarist system. In *Proceedings of a Workshop on Held at Baltimore, Maryland: October 13-15, 1998*, TIPSTER '98, pages 197–214, Stroudsburg, PA, USA. Association for Computational Linguistics.

Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. In *Advances in Neural Information Processing Systems 27*, pages 2042–2050. Curran Associates, Inc.

Jure Leskovec, Lars Backstrom, and Jon Kleinberg. 2009. Meme-tracking and the dynamics of the news cycle. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, pages 497–506.

Jiwei Li, Minh-Thang Luong, and Dan Jurafsky. 2015. A hierarchical neural autoencoder for paragraphs and documents. In *Proceedings of ACL*.

Chin-Yew Lin and E.H. Hovy. 2003. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Proceedings of 2003 Language Technology Conference (HLT-NAACL 2003)*, Edmonton, Canada.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *In Proceedings of Workshop on Text Summarization Branches Out, Post-Conference Workshop of ACL 2004, Barcelona, Spain.*

H. P. Luhn. 1998. The automatic creation of literature abstracts. *IBM Journal of Research and Development*, 2(2):159–165.

Thang Luong, Ilya Sutskever, Quoc V. Le, Oriol Vinyals, and Wojciech Zaremba. 2014. Addressing the rare word problem in neural machine translation. *CoRR*, abs/1410.8206.

Dipanjan Das Andr F.T. Martins. 2007. A survey on automatic text summarization. Technical report, CMU.

Rada Mihalcea. 2004. Graph-based ranking algorithms for sentence extraction, applied to text summarization. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics, companion volume*, Spain.

Ani Nenkova and Kathleen McKeown. 2011. Automatic summarization. *Foundations and Trend in Information Retrieval*, 5(2-3):103–233.

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. *CoRR*, abs/1503.02364.

Mao Song Sun. 2011. Natural language procesing based on naturaly annotated web resources. *Journal of Chinese Information Processing*, 25(6):26–32.

Ilya Sutskever, Oriol Vinyals, and Quoc V. V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27*, pages 3104–3112.

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

Xiaoqiang Zhou, Baotian Hu, Qingcai Chen, Buzhou Tang, and Xiaolong Wang. 2015. Answer sequence learning with neural networks for answer selection in community question answering. In *Proceedings of the ACL-IJCNLP*, pages 713–718, Beijing, China, July. Association for Computational Linguistics.

Discourse Planning with an N-gram Model of Relations

Or Biran
Columbia University
orb@cs.columbia.edu

Kathleen McKeown
Columbia University
kathy@cs.columbia.edu

Abstract

While it has been established that transitions between discourse relations are important for coherence, such information has not so far been used to aid in language generation. We introduce an approach to discourse planning for concept-to-text generation systems which simultaneously determines the order of messages and the discourse relations between them. This approach makes it straightforward to use statistical transition models, such as n-gram models of discourse relations learned from an annotated corpus. We show that using such a model significantly improves the quality of the generated text as judged by humans.

1 Introduction

Discourse planning is a subtask of Natural Language Generation (NLG), concerned with determining the ordering of messages in a document and the discourse relations that hold among them (Reiter and Dale, 2000). Early approaches to discourse planning used manually written rules, often based on schemas (McKeown, 1985) or on Rhetorical Structure Theory (RST) (Mann and Thompson, 1987; Hovy, 1993; Power, 2000). In the past decade, various statistical approaches have emerged (Duboue and McKeown, 2001; Dimitromanolaki and Androutsopoulos, 2003; Soricut and Marcu, 2006; Konstas and Lapata, 2013). Other relevant statistical approaches to content ordering can also be found in the summarization literature (Barzilay et al., 2001; Lapata, 2003; Bollegala et al., 2005). These approaches overwhelmingly focus on determining the best order of messages using semantic conent, while discourse relations are in most cases either determined by manually-written derivation rules or completely ignored.

Meanwhile, researchers working on discourse relation disambiguation have observed that the sequence of discourse relations itself, independently of content, helps in disambiguating adjacent relations (Wellner et al., 2006; Pitler et al., 2008). Sequential discourse information has been used successfully in discourse parsing (Ghosh et al., 2011; Feng and Hirst, 2014), and discourse structure was shown to be as important for text coherence as entity-based content structure (Lin et al., 2011; Feng et al., 2014). Surprisingly, so far, discourse sequential information from existing discourse-annotated corpora, such as the Penn Discourse Treebank (PDTB) (Prasad et al., 2008) has not been used in generation.

In this paper, we present an NLG framework that generates texts from existing semantic web ontologies. We use an n-gram model of discourse relations to perform discourse planning for these stories. Through a crowd-sourced human evaluation, we show that the ordering of our documents and the choice of discourse relations is significantly better when using this model.

2 Generation Framework

In concept-to-text generation pipelines, discourse planning typically occurs after the content selection stage. The input, therefore, is an unordered set of messages that are not yet realized: instead of being represented as text, the messages have a structured semantic representation.

In this paper, we generate *comparison stories*, describing and comparing two similar entities, from an RDF ontology. The RDF semantic representation is commonly used in semantic web resources and free ontologies. An RDF message (called a *triple*) has three parts: a subject, a predicate and an object. For each story, we consider any triple whose subject is one of the participating entities as a potential message to be generated. We do only minimal processing on these messages:

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1973–1977,
Lisbon, Portugal, 17-21 September 2015.

where two triples have the same subject and predicate but different objects, we merge them into a single message with multiple objects; and where two triples have the same subject and object but different predicates, we merge them into a single message with multiple predicates.

Next, we build the set of potential discourse relations between all messages. We use the PDTB *class-level* relations, of which there are four: *expansion*, *comparison*, *contingency* and *temporal*. Each is an abstraction of a family of more specific relations, such as *cause*, *concession*, etc. We do not differentiate between explicit and implicit relations, and treat *entrel* as a type of *expansion*.

Potential discourse relations are implied in the semantics of the triples: messages that contain the same predicate and object may have an expansion relation among them (e.g. "John has a ball. Mary also has a ball"). Messages that contain the same predicate but different subjects and objects may have a comparison relation (e.g. "John likes apples but Mary likes oranges").

Specific predicate pairs will also have specific potential relations among them - for example, "birth place" and "residence" have a temporal relation (when applied to the same subject). The same is true for contingency relations (e.g., "city" and "country" for the same subject - if the subject is in a city, it implies which country it is in). We manually annotated the 59 predicate pairs that had potential temporal and contingency relations, as well as 8 pairs with special potential comparison relations (e.g., "birth place" and "residence" if the subject is the same but the object is not).

Once the potential relations are identified, we have a directed multigraph where each vertex is a message and each edge is a potential relation. There can be multiple edges between any two vertices, since messages may have more than one potential relation among them.

Once the graph is ready, we perform content selection. Given a desired number of messages to generate, we choose the set of messages that maximizes the number of edges in the resulting subgraph (thus ensuring that the selected messages are discourse-coherent). If there are multiple such sets, we choose one at random.

The task we are focused on in this paper is discourse planning, which in our formulation is the task of finding a Hamiltonian path through the selected subgraph, thus simultaneously selecting the order of the messages (nodes) as well as the relations (edges) that connect them. Our approach for choosing the best path is discussed in the next section. For the remainder of this section, we describe our simple implementations of the next stages of generation: sentence planning and realization.

For each of the four discourse relations we use, we selected a few explicit connectives from the PDTB that are often used to convey them. We specifically chose connectives that apply to the entire range of class-level relations (e.g., for *comparison* we chose "while" - since it applies to both *contrast* and *concession* in the PDTB, but not "in contrast" which applies only to the former). We also chose only those connectives which have the structure [ARG1 connective ARG2] or [ARG1. connective, ARG2]. During realization, we arbitrarily choose a connective to realize the relation.

Since the ordering and relations between messages is determined by the discourse plan, sentence planning falls naturally out of it: sentence breaks occur where the connective pattern creates them, or where there is no relation between adjacent messages.

To realize the messages themselves, we follow a single pattern: "the [predicate(s)] of [subject] (is/are) [object(s)]". Simple rules are used to pluralize the predicate when there are multiple objects and to create lists of multiples objects or predicates where needed.

These basic solutions for the various stages of NLG produce texts that are rich enough to be acceptable for human readers, but which have relatively little variation in grammatical and lexical quality. This crucial combination allows us to perform a human study to specifically evaluate the discourse planning component.

3 Discourse Planning

As explained in the previous section, we formulate the discourse planning task as finding a path through a multigraph of potential relations between messages. One major component of what makes a good path is the sequence of content: some content is more central and should appear earlier, for example; and some predicates and objects are semantically related and should appear near one another. In this paper we focus on a component that has so far been neglected in generation - the sequence of discourse relations - while trying to minimize the effect that content semantics

have on the evaluation (other than the semantics implicit in the relations). In order to quantify the likelihood of a sequence of relations, we build an n-gram model from a discourse-annotated corpus.

An n-gram model measures the transitional probabilities for sequences of the units that the n-grams are composed of. In this case, the units are discourse relations. The probability of a particular sequence of relations of length $n+1$ given an existing subsequence of length n is computed as a fraction of the number of times it appears in the corpus and the number of times the subsequence appears in the corpus, i.e.

$$P(r_i|r_{i-n},...,r_{i-1}) = \frac{C(r_{i-n},...,r_{i-1},r_i)}{C(r_{i-n},...,r_{i-1})}$$

Where $C(s)$ is the number of times sequence s appears in the corpus. Using this model to generate a discourse plan given a potential relation multigraph is a stochastic process: at each stage, we choose the next relation edge out of the last chosen message vertex (the first vertex is chosen at random) based on the selected sequence of relation edges and the probabilities for the next relation in the model. Once a vertex is added to the path edges leading to it can no longer be selected.

4 Evaluation

One method for evaluating a discourse plan independently of content is to produce pairs of generated short text documents, each containing the same content, but with different ordering and discourse relations (as dictated by the discourse plan). The only obvious way to decide which text is better is to have human judges make that decision. It is important to minimize the effects of other qualities of the texts (differences in content, word choice, grammatical style, etc.) as much as possible, so that the judgement is based only on the differences in order and discourse.

We used DBPedia (Auer et al., 2007) - an RDF ontology extracted from Wikipedia - to generate content. Each document generated was a *comparison story* of two entities in a single category (i.e., the messages in the stories were selected, as described in Section 2, from the set of triples where one of the entities was the subject). In order to experiment with different domains, we used four different categories: **Office Holder** (i.e., a person holding office such as a President or a Judge); **River**; **Television Show**; and **Military Unit**. The entity pairs from each category were chosen at random but were required to have at least 8 predicates and 3 objects in common, so that they were somewhat semantically related.

The birth place of Allen J. Ellender is Montegut, Louisiana, while the death place of Allen J. Ellender is Maryland. The birth place of Robert E. Quinn is Phoenix, Rhode Island. Subsequently, the death place of Robert E. Quinn is Rhode Island.
The birth place of Allen J. Ellender is Montegut, Louisiana. In comparison, the birth place of Robert E. Quinn is Phoenix, Rhode Island. The death place of Robert E. Quinn is Rhode Island, but the death place of Allen J. Ellender is Maryland.

Figure 1: Sample pair of comparison stories

To ensure that human judges can easily tell the differences between the stories on a sentential level, we limited the size of each story to 4 messages. For each pair of stories, everything but the discourse plan (i.e. the content selection, the realization of messages and the lexical choice of connectives) was identical. Figure 1 shows an example pair of stories from the Office Holder category.

4.1 Experiments

We conducted two crowd sourced experiments on the CrowdFlower platform. Each question consisted of two short stories that are completely identical in content, but each generated with a different discourse planner. The human judge was asked to decide which of the stories has a better flow (or whether they are equally good), and then to give each of the stories a score from 1 to 5, paying specific attention to the ordering of the prepositions and the relations between them. The stories were presented in a random order and were not given labels, to avoid bias. We generated 125 pairs of stories from each category - a total of 500 - for each experiment.

Each question was presented to three judges. In each experiment, there was complete disagreement among the three annotators in approximately 15% of the questions, and those were discarded. In approximately 20% there was complete agreement, and in the rest of the questions there were two judges who agreed and one who disagreed. We also computed inter-annotator agreement using Cohen's Kappa for 217 pairs of judges who

	Quality comparison			Avg. score	
	Base	Equal	Pdtb	Base	Pdtb
Of. Holder	27.4%	30.2%	**42.5%**	3.67	**3.76**
TV Show	34.3%	25.7%	**40%**	3.79	**3.8**
Mil. Unit	32.3%	23.2%	**44.4%**	3.69	**3.84**
River	**39.2%**	23.5%	37.3%	3.71	**3.72**
Total	34%	25%	**41%**	3.72	**3.78**

Table 1: Results for the comparison between the PDTB n-gram model and the baseline

	Quality comparison			Avg. score	
	Pdtb	Equal	Wiki	Pdtb	Wiki
Of. Holder	33.6%	14.5%	**51.8%**	3.51	**3.65**
TV Show	43.2%	8.1%	**48.6%**	3.62	**3.65**
Mil. Unit	40.4%	14.4%	**45.2%**	3.65	**3.67**
River	41.1%	11.2%	**47.7%**	3.68	**3.7**
Total	39.6%	12%	**48.4%**	3.61	**3.67**

Table 2: Results for the comparison between the Wikipedia model and the PDTB model

both answered at least 10 of the same questions. The average kappa value was 0.5, suggesting reasonable agreement.

In the first experiment, we compared stories generated by a planner using an n-gram model extracted from the PDTB with stories generated by a baseline planner, where all edges have identical probabilities. The results are shown in Table 1.

In the second experiment, we used a PDTB shallow discourse parser we developed (Biran and McKeown, 2015) to create a discourse-annotated version of the English Wikipedia. We then compared stories generated by a planner using an n-gram model extracted from the parsed Wikipedia corpus with those generated by a planner using the PDTB model. The results are shown in Table 2.

The total results in both tables are statistically significant ($p < 0.05$).

4.2 Discussion

The results in Table 1 show that the judges significantly preferred the stories created by the n-gram model-based planner to those created by the baseline planner, both in terms of the three-way decision and in terms of the numeric score. This is true for the total set as well as every specific topic, except for *River*. This may be because the predicates in the River category are much more cohesive than in other categories: virtually all predicates related to rivers describe an aspect of the location of the river. That fact may make it easier for a random planner to produce a story that seems coherent. Note, however, that while the judges preferred the baseline story more often in the *River* questions, the average score is higher for the model, which suggests that when the baseline was better it was only mildly so, while when the model was better is was significantly so.

The results in Table 2 show that the Wikipedia-based model produces better results than the PDTB-based model. We hypothesize that it is for two reasons. First, Wikipedia contains definitional texts and is closer in style and content to the stories we produce than the PDTB, which contains WSJ articles. Temporal relations constitute about 10% of both corpora, but contingency and comparison relations each make up almost 20% of the PDTB, while in Wikipedia they span only 10% and 12% of the corpus, respectively, making the share of expansion relations much larger. Second, since the PDTB is small, higher-order n-grams are sparsely found, which can add noise to the model. The Wikipedia corpus is significantly larger and does not suffer from this problem.

The differences in average scores seen in the experiments are relatively small. That is expected, since we have eliminated the content coherence factor, which is known to be significant. In addition, while judges were specifically asked to focus on the order of messages and relations between them, there is inevitably some noise due to accidental lexical or syntactic mismatches, ordering that is awkward content-wise, and other side-effects of the generation framework we employed.

5 Conclusion and Future Work

We introduced an approach to discourse planning that relies on a potential discourse multigraph, allowing for an n-gram model of relations to drive the discourse plan and efficiently determine both the ordering and the relations between messages.

We conducted two experiments, comparing stories generated with different discourse planners. The first shows that an n-gram model-based planner significantly outperforms the random baseline. The second suggests that using an n-gram model derived from a corpus that is larger and closer in style and content, though less accurately annotated, can further improve results.

In future work, we intend to combine this discourse-based view of coherence with a content-based view to create a unified statistical discourse planner. In addition, we will explore additional stochastic models of discourse that look at other, non-sequential collocational information.

References

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. Dbpedia: a nucleus for a web of open data. In *Proceedings of the 6th international The semantic web and 2nd Asian conference on Asian semantic web conference*, ISWC'07/ASWC'07, pages 722–735, Berlin, Heidelberg. Springer-Verlag.

Regina Barzilay, Noemie Elhadad, and Kathleen R. McKeown. 2001. Sentence ordering in multidocument summarization. In *Proceedings of the First International Conference on Human Language Technology Research*, HLT '01, pages 1–7, Stroudsburg, PA, USA. Association for Computational Linguistics.

Or Biran and Kathleen McKeown. 2015. Pdtb discourse parsing as a tagging task: The two taggers approach. In *Proceedings of the 16th Annual SIGdial Meeting on Discourse and Dialogue*, SIGDIAL 2015, Prague, Czech Republic.

Danushka Bollegala, Naoaki Okazaki, and Mitsuru Ishizuka. 2005. A machine learning approach to sentence ordering for multidocument summarization and its evaluation. In *Natural Language Processing–IJCNLP 2005*, pages 624–635. Springer.

Aggeliki Dimitromanolaki and Ion Androutsopoulos. 2003. Learning to order facts for discourse planning in natural language generation. *arXiv preprint cs/0306062*.

Pablo A. Duboue and Kathleen R. McKeown. 2001. Empirically estimating order constraints for content planning in generation. In *Proceedings of 39th Annual Meeting of the Association for Computational Linguistics*, pages 172–179, Toulouse, France, July. Association for Computational Linguistics.

Vanessa Wei Feng and Graeme Hirst. 2014. A Linear-Time Bottom-Up Discourse Parser with Constraints and Post-Editing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 511–521, Baltimore, Maryland, June. Association for Computational Linguistics.

Vanessa Wei Feng, Ziheng Lin, Graeme Hirst, and Singapore Press Holdings. 2014. The impact of deep hierarchical discourse structures in the evaluation of text coherence. In *Proceedings of the 25th International Conference on Computational Linguistics*.

Sucheta Ghosh, Richard Johansson, Giuseppe Riccardi, and Sara Tonelli. 2011. Shallow discourse parsing with conditional random fields. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 1071–1079.

Eduard H. Hovy. 1993. Automated discourse generation using discourse structure relations. *Artif. Intell.*, 63(1-2):341–385, October.

Ioannis Konstas and Mirella Lapata. 2013. Inducing document plans for concept-to-text generation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1503–1514, Seattle, Washington, USA, October. Association for Computational Linguistics.

Mirella Lapata. 2003. Probabilistic text structuring: Experiments with sentence ordering. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, ACL '03, pages 545–552, Stroudsburg, PA, USA. Association for Computational Linguistics.

Ziheng Lin, Hwee Tou Ng, and Min-Yen Kan. 2011. Automatically evaluating text coherence using discourse relations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 997–1006. Association for Computational Linguistics.

William C. Mann and Sandra A. Thompson. 1987. Rhetorical Structure Theory: A theory of text organization. Technical Report ISI/RS-87-190, ISI.

Kathleen R. McKeown. 1985. Discourse strategies for generating natural-language text. *Artif. Intell.*, 27(1):1–41, September.

Emily Pitler, Mridhula Raghupathy, Hena Mehta, Ani Nenkova, Alan Lee, and Aravind K Joshi. 2008. Easily identifiable discourse relations.

Richard Power. 2000. Planning texts by constraint satisfaction. In *Proceedings of the 18th Conference on Computational Linguistics - Volume 2*, COLING '00, pages 642–648, Stroudsburg, PA, USA. Association for Computational Linguistics.

Rashmi Prasad, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Livio Robaldo, Aravind Joshi, and Bonnie Webber. 2008. The penn discourse treebank 2.0. In *In Proceedings of LREC*.

Ehud Reiter and Robert Dale. 2000. *Building natural language generation systems*, volume 33. Cambridge university press.

Radu Soricut and Daniel Marcu. 2006. Discourse generation using utility-trained coherence models. In *Proceedings of the COLING/ACL on Main Conference Poster Sessions*, COLING-ACL '06, pages 803–810, Stroudsburg, PA, USA. Association for Computational Linguistics.

Ben Wellner, James Pustejovsky, Catherine Havasi, Anna Rumshisky, and Roser Saurí. 2006. Classification of discourse coherence relations: An exploratory study using multiple knowledge sources. In *Proceedings of the 7th SIGdial Workshop on Discourse and Dialogue*, SigDIAL '06, pages 117–125, Stroudsburg, PA, USA. Association for Computational Linguistics.

Experiments with Generative Models for Dependency Tree Linearization

Richard Futrell and Edward Gibson
Department of Brain and Cognitive Sciences
Massachusetts Institute of Technology
{futrell, egibson}@mit.edu

Abstract

We present experiments with generative models for linearization of unordered labeled syntactic dependency trees (Belz et al., 2011; Rajkumar and White, 2014). Our linearization models are derived from generative models for dependency structure (Eisner, 1996). We present a series of generative dependency models designed to capture successively more information about ordering constraints among sister dependents. We give a dynamic programming algorithm for computing the conditional probability of word orders given tree structures under these models. The models are tested on corpora of 11 languages using test-set likelihood, and human ratings for generated forms are collected for English. Our models benefit from representing local order constraints among sisters and from backing off to less sparse distributions, including distributions not conditioned on the head.

1 Introduction

We explore generative models for producing linearizations of unordered labeled syntactic dependency trees. This specific task has attracted attention in recent years (Filippova and Strube, 2009; He et al., 2009; Belz et al., 2011; Bohnet et al., 2012; Zhang, 2013) because it forms a useful part of a natural language generation pipeline, especially in machine translation (Chang and Toutanova, 2007) and summarization (Barzilay and McKeown, 2005). Closely related tasks are generation of sentences given CCG parses (White and Rajkumar, 2012), bags of words (Liu et al., 2015), and semantic graphs (Braune et al., 2014).

Here we focus narrowly on testing probabilistic generative models for dependency tree linearization. In contrast, the approach in most previous work is to apply a variety of scoring functions to trees and linearizations and search for an optimally-scoring tree among some set. The probabilistic linearization models we investigate are derived from generative models for dependency trees (Eisner, 1996), as most commonly used in unsupervised grammar induction (Klein and Manning, 2004; Gelling et al., 2012). Generative dependency models have typically been evaluated in a parsing task (Eisner, 1997). Here, we are interested in the inverse task: inferring a distribution over linear orders given unordered dependency trees.

This is the first work to consider generative dependency models from the perspective of word ordering. The results can potentially shed light on how ordering constraints are best represented in such models. In addition, the use of probabilistic models means that we can easily define well-motivated normalized probability distributions over orders of dependency trees. These distributions are useful for answering scientific questions about crosslinguistic word order in quantitative linguistics, where obtaining robust estimates has proven challenging due to data sparsity (Futrell et al., 2015).

The remainder of the work is organized as follows. In Section 2 we present a set of generative linearization models. In Section 3 we compare the performance of the different models as measured by test-set probability and human acceptability ratings. We also compare our performance with other systems from the literature. Section 4 concludes.

2 Generative Models for Projective Dependency Tree Linearization

We investigate head-outward projective generative dependency models. In these models, an ordered dependency tree is generated by the following kind

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1978–1983,
Lisbon, Portugal, 17-21 September 2015.

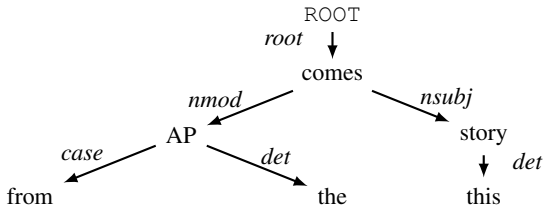
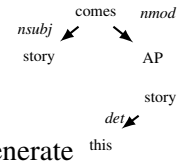


Figure 1: Example unordered dependency tree. Possible linearizations include (1) *This story comes from the AP* and (2) *From the AP comes this story*. Order 2 is the original order in the corpus, but order 1 is much more likely under our models.

of procedure. Given a head node, we use some generative process G to generate a depth-1 subtree rooted in that head node. Then we apply the procedure recursively to each of the dependent nodes. By applying the procedure starting at a ROOT node, we generate a dependency tree. For example, to generate the dependency tree in Figure 1 from the node *comes* down, we take the head *comes* and generate the subtree  , then we take the head *story* and generate , and so on. In this work, we experiment with different specific generative processes G which generate a local subtree conditioned on a head.

2.1 Model Types

Here we describe some possible generative processes G which generate subtrees conditioned on a head. These models contain progressively more information about ordering relations among sister dependents.

A common starting point for G is **Eisner Model C** (Eisner, 1996). In this model, dependents on one side of the head are generated by repeatedly sampling from a categorical distribution until a special stop-symbol is generated. The model only captures the propensity of dependents to appear on the left or right of the head, and does not capture any order constraints between sister dependents on one side of the head.

We consider a generalization of Eisner Model C which we call **Dependent N-gram** models. In a Dependent N-gram model, we generate dependents on each side the head by sampling a *sequence* of dependents from an N-gram model. Each dependent is generated conditional on the $N-1$ previously generated dependents from the head outwards. We have two separate N-gram sequence distributions for left and right dependents. Eisner Model C can be seen as a Dependent N-gram model with $N = 1$.

We also consider a model which can capture many more ordering relations among sister dependents: given a head h, sample a subtree whose head is h from a Categorical distribution over subtrees. We call this the **Observed Orders** model because in practice we are simply sampling one of the observed orders from the training data. This generative process has the capacity to capture the most ordering relations between sister dependents.

2.1.1 Distributions over Permutations of Dependents

We have discussed generative models for ordered dependency trees. Here we discuss how to use them to make generative models for word orders conditional on unordered dependency trees.

Suppose we have a generative process G for dependency trees which takes a head h and generates a sequence of dependents $\mathbf{w}_l$ to the left of h and a sequence of dependents $\mathbf{w}_r$ to the right of h. Let $\mathbf{w}$ denote the pair $(\mathbf{w}_l, \mathbf{w}_r)$, which we call the **configuration** of dependents. To get the probability of some $\mathbf{w}$ given an unordered subtree u, we want to calculate the probability of $\mathbf{w}$ given that G has generated the particular multiset $\mathbf{W}$ of dependents corresponding to u. To do this, we calculate:

$$\begin{aligned} p(\mathbf{w}|\mathbf{W}) &= \frac{p(\mathbf{w}, \mathbf{W})}{p(\mathbf{W})} \\ &= \frac{p(\mathbf{w})}{Z}, \end{aligned} \tag{1}$$

where

$$Z = \sum_{\mathbf{w}' \in \mathcal{W}} p(\mathbf{w}') \tag{2}$$

and $\mathcal{W}$ is the set of all possible configurations $(\mathbf{w}_l, \mathbf{w}_r)$ compatible with multiset $\mathbf{W}$. That is, $\mathcal{W}$ is the set of pairs of permutations of multisets $\mathbf{W}_l$ and $\mathbf{W}_r$ for all possible partitions of $\mathbf{W}$ into $\mathbf{W}_l$ and $\mathbf{W}_r$. The generative dependency model gives us the probability $p(\mathbf{w})$.

It remains to calculate the normalizing constant Z, the sum of probabilities of possible configurations. For the Observed Orders model, Z is the sum of probabilities of subtrees with the same dependents as subtree u. For the Dependent N-gram models of order N, we calculate Z using a dynamic programming algorithm, presented in Al-

gorithm 1 as memoized recursive functions. When $N = 1$ (Eisner Model C), Z is more simply:

$$Z_{\text{emc}} = p_L(\texttt{stop}) \times p_R(\texttt{stop}) \times \sum_{(\mathbf{W}_l, \mathbf{W}_r) \in \text{PARTS}(\mathbf{W})} |\mathbf{W}_l|! \times |\mathbf{W}_r|! \times \prod_{w \in \mathbf{W}_l} p_L(w) \prod_{w \in \mathbf{W}_r} p_R(w), \quad (3)$$

where $\text{PARTS}(\mathbf{W})$ is the set of all partitions of multiset $\mathbf{W}$ into two multisets $\mathbf{W}_l$ and $\mathbf{W}_r$, p_L is the probability mass function for a dependent to the left of the head, p_R is the function for a dependent to the right, and `stop` is a special symbol in the support of p_L and p_R which indicates that generation of dependents should halt. The probability mass functions may be conditional on the head h. These methods for calculating Z make it possible to transform a generative dependency model into a model of dependency tree *ordering* conditional on local subtree structure.

Algorithm 1 Compute the sum of probabilities of all configurations of dependents $\mathbf{W}$ under a Dependent N-gram model with two component N-gram models of order N: p_R for sequences to the right of the head and p_L for sequences to the left.

memoized function RIGHT_NORM($\mathbf{r}$, $\mathbf{c}$)
  if $|\mathbf{r}| = 0$ **then**
    return $p_R(\texttt{stop} \mid \mathbf{c})$
  end if
  $Z \leftarrow 0$
  for $i = 1 : |\mathbf{r}|$ **do**
    $\mathbf{r}' \leftarrow$ elements of $\mathbf{r}$ except the ith
    $\mathbf{c}' \leftarrow$ append r_i to $\mathbf{c}$ then truncate to length $N - 1$
    $Z \leftarrow Z + p_R(r_i|\mathbf{c}) \times \text{RIGHT\_NORM}(\mathbf{r}', \mathbf{c}')$
  end for
  return Z
end memoized function
memoized function LEFT_NORM($\mathbf{r}$, $\mathbf{c}$)
  $Z \leftarrow p_L(\texttt{stop} \mid \mathbf{c}) \times \text{RIGHT\_NORM}([\texttt{start}], \mathbf{r})$
  for $i = 1 : |\mathbf{r}|$ **do**
    $\mathbf{r}' \leftarrow$ elements of $\mathbf{r}$ except the ith
    $\mathbf{c}' \leftarrow$ append r_i to $\mathbf{c}$ then truncate to length $N - 1$
    $Z \leftarrow Z + p_L(r_i|\mathbf{c}) \times \text{LEFT\_NORM}(\mathbf{r}', \mathbf{c}')$
  end for
  return Z
end memoized function
Result is LEFT_NORM($\mathbf{W}$, [`start`])

2.2 Labelling

The previous section discussed the question of the structure of the generative process for dependency trees. Here we discuss an orthogonal modeling question, which we call **labelling**: what information about the *labels* on dependency tree nodes and edges should be included in our models. Dependency tree nodes are labeled with wordforms, lemmas, and parts-of-speech (POS) tags; and dependency tree edges are labeled with relation types. A model might generate orders of dependents conditioned on all of these labels, or a subset of them. For example, a generative dependnecy model might generate (relation type, dependent POS tag) tuples conditioned on the POS tag of the head of the phrase. When we use such a model for dependency linearization, we would say the model's labelling is relation type, dependent POS, and head POS. In this study, we avoid including wordforms or lemmas in the labelling, to avoid data sparsity issues.

2.3 Model Estimation and Smoothing

In order to alleviate data sparsity in fitting our models, we adopt two smoothing methods from the language modelling literature.

All categorical distributions are estimated using add-k smoothing where $k = 0.01$. For the Dependent N-gram models, this means adding k pseudocounts for each possible dependent in each context. For the Observed Orders model, this means adding k pseudocounts for each possible permutation of the head and its dependents.

We also experiment with combining our models into mixture distributions. This can be viewed as a kind of back-off smoothing (Katz, 1987), where the Observed Orders model is the model with the most context, and Dependent N-grams and Eisner Model C are backoff distributions with successively less context. Similarly, models with less information in the labelling can serve as backoff distributions for models with more information in the labelling. For example, a model which is conditioned on the POS of the head can be backed off to a model which does not condition on the head at all. We find optimal mixture weights using the Baum-Welch algorithm tuned on a held-out development set.

3 Evaluation

Here we empirically evaluate some options for model type and model labelling as described above. We are interested in how many of the possible orders of a sentence our model can generate (recall), and in how many of our generated orders really are acceptable (precision). As a recall-like measure, we quantify the probability of the word orders of held-out test sentences. Low probabil-

Labelling	Model	Basque	Czech	English	Finnish	French	German	Hebrew	Indonesian	Persian	Spanish	Swedish
HDR	oo	-6.83	-7.58	-5.23	-7.35	-10.86	-8.36	-9.74	-8.99	-10.39	-11.31	-8.83
	n1	-6.12	-8.97	-5.08	-7.15	-11.54	-9.81	-9.63	-8.68	-10.63	-13.19	-8.37
	n2	-4.86	-6.35	-2.87	-5.30	-6.86	-6.60	-5.91	-5.98	-5.54	-7.47	-4.92
	n3	-5.92	-6.59	-3.13	-5.68	-7.34	-7.02	-6.81	-6.69	-6.49	-8.06	-5.68
	n123	-4.58	-6.18	-2.60	-5.11	-6.67	-6.19	-5.77	-5.73	-5.51	-7.36	-4.72
	oo+n123	-4.52	-5.95	-2.57	-5.04	-6.58	-5.92	-5.68	-5.68	-5.47	-7.27	-4.68
HDR+R	oo	-5.56	-6.78	-3.94	-6.25	-9.63	-7.42	-7.95	-7.51	-9.19	-9.54	-7.28
	n1	-6.08	-8.97	-5.07	-7.16	-11.54	-9.79	-9.58	-8.67	-10.62	-13.17	-8.35
	n2	-4.49	-6.31	-2.62	-5.17	-6.79	-6.34	-5.62	-5.67	-5.42	-7.40	-4.67
	n3	-4.86	-6.41	-2.61	-5.20	-7.08	-6.43	-6.07	-6.02	-6.04	-7.70	-5.02
	n123	-4.41	-6.15	-2.48	-5.01	-6.59	-5.99	-5.54	-5.53	-5.42	-7.29	-4.53
	oo+n123	**-4.29**	**-5.84**	**-2.44**	**-4.88**	**-6.50**	**-5.74**	**-5.40**	**-5.47**	**-5.38**	**-7.09**	**-4.46**

Table 1: Average log likelihood of word order per sentence in test set under various models. Under "Labelling", **HDR** means conditioning on Head POS, Dependent POS, and Relation Type, and **R** means conditioning on Relation Type alone (see Section 2.2). Under "Model", **oo** is the Observed Orders model, **n1** is the Dependent 1-gram model (Eisner Model C), **n2** is the Dependent 2-gram model, and **n3** is the Dependent 3-gram model (see Section 2.1). In both columns, x**+**y means a mixture of model x and model y; **n123** means n1+n2+n3.

ities assigned to held-out sentences indicate that there are possible orders which our model is missing. As a precision-like measure, we get human acceptability ratings for sentence reorderings generated by our model.

We carry out our evaluations using the dependency corpora of the Universal Dependencies project (v1.1) (Agić et al., 2015), with the train/dev/test splits provided in that dataset. We remove nodes and edges dealing with punctuation. Due to space constraints, we only present results from 11 languages here.

3.1 Test-Set Probability

Here we calculate average probabilities of word orders per sentence in the test set. This number can be interpreted as the (negative) average amount of information contained in the word order of a sentence beyond information about dependency relations.

The results for selected languages are shown in Table 1. The biggest gains come from using Dependent N-gram models with $N > 1$, and from backing off the model labelling. The Observed Orders model does poorly on its own, likely due to data sparsity; its performance is much improved when backing off from conditioning on the head. Eisner Model C (n1) also performs poorly, likely because it cannot represent any ordering constraints among sister dependents. The fact it helps to back off to distributions not conditioned on the head suggests that there are commonalities among distributions of dependents of different heads, which could be exploited in further generative dependency models.

3.2 Human Evaluation

We collected human ratings for sentence reorderings sampled from the English models from 54 native American English speakers on Amazon Mechanical Turk. We randomly selected a set of 90 sentences from the test set of the English Universal Dependencies corpus. We generated a reordering of each sentence according to each of 12 model configurations in Table 1. Each participant saw an original sentence and a reordering of it, and was asked to rate how natural each version of the sentence sounded, on a scale of 1 to 5. The order of presentation of the original and reordered forms was randomized, so that participants were not aware of which form was the original and which was a reordering. Each participant rated 56 sentence pairs. Participants were also asked whether the two sentences in a pair meant the same thing, with "can't tell" as a possible answer.

Table 2 shows average human acceptability ratings for reorderings, and the proportion of sentence pairs judged to mean the same thing. The original sentences have an average acceptability rating of 4.48/5. The very best performing models are those which do not back off to a distribution not conditioned on the head. However, in the case of the Observed Orders and other sparse models, we see consistent improvement from this backoff.

Figure 2 shows the acceptability ratings (out of 5) plotted against test set probability. We see that

Labelling	Model	Acceptability	Same Meaning
HDR	oo	2.92	0.58
	n1	2.06	0.44
	n2	3.42	0.78
	n3	3.48	**0.85**
	n123	**3.56**	0.79
	oo+n123	3.45	0.75
HDR+R	oo	3.11	0.72
	n1	2.11	0.49
	n2	3.32	0.80
	n3	3.52	0.77
	n123	3.31	0.76
	oo+n123	3.43	0.80

Table 2: Mean acceptability rating out of 5, and proportion of reordered sentences with the same meaning as the original, for English models. Labels as in Table 1.

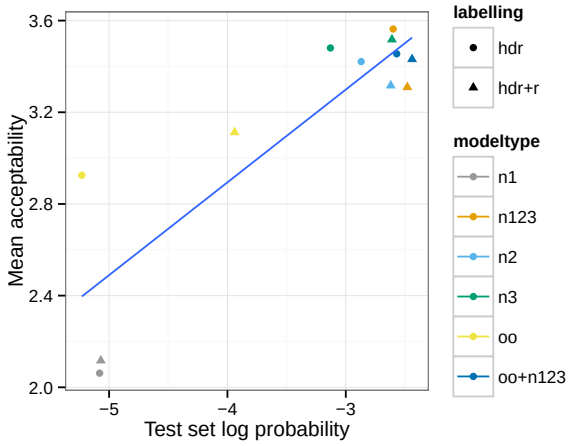


Figure 2: Comparison of test set probability (Table 1) and acceptability ratings (Table 2) for English across models. A least-squares linear regression line is shown. Labels as in Table 1.

the models which yield poor test set probability also have poor acceptability ratings.

3.3 Comparison with other systems

Previous work has focused on the ability to correctly reconstruct the word order of an observed dependency tree. Our goal is to explicitly model a distribution over possible orders, rather than to recover a single correct order, because many orders are often possible, and the particulator order that a dependency tree originally appeared in might not be the most natural. For example, our models typically reorder the sentence "From the AP comes this story" (in Figure 1) as "This story comes from the AP"; the second order is arguably more natural, though the first is idiomatic for this particular phrase. So we do not believe that BLEU scores and other metrics of similarity to a "correct" ordering are particularly relevant for our task.

Previous work uses BLEU scores (Papineni et al., 2002) and human ratings to evaluate generation of word orders. To provide some comparability with previous work, we report BLEU scores on the 2011 Shared Task data here. The systems reported in Belz et al. (2011) achieve BLEU scores ranging from 23 to 89 for English; subsequent work achieves BLEU scores of 91.6 on the same data (Bohnet et al., 2012). Drawing the highest-probability orderings from our models, we achieve a top BLEU score of 57.7 using the model configuration hdr/oo. Curiously, hdr/oo is typically the worst model configuration in the test set probability evaluation (Section 3.1). The BLEU performance is in the middle range of the Shared Task systems. The human evaluation of our models is more optimistic: the best score for Meaning Similarity in the Shared Task was 84/100 (Bohnet et al., 2011), while sentences ordered according to our models were judged to have the same meaning as the original in 85% of cases (Table 2), though these figures are based on different data. These comparisons suggest that these generative models do not provide state-of-the-art performance, but do capture some of the same information as previous models.

3.4 Discussion

Overall, the most effective models are the Dependent N-gram models. The naive approach to modeling order relations among sister dependents, as embodied in the Observed Orders model, does not generalize well. The result suggests that models like the Dependent N-gram model might be effective as generative dependency models.

4 Conclusion

We have discussed generative models for dependency tree linearization, exploring a path less traveled by in the dependency linearization literature. We believe this approach has value for answering scientific questions in quantitative linguistics and for better understanding the linguistic adequacy of generative dependency models.

Acknowledgments

We thank William P. Li, Kyle Mahowald, and Tim O'Donnell for helpful discussions, and Michael White for help accessing data.

References

Željko Agić, Maria Jesus Aranzabe, Aitziber Atutxa, Cristina Bosco, Jinho Choi, Marie-Catherine de Marneffe, Timothy Dozat, Richárd Farkas, Jennifer Foster, Filip Ginter, Iakes Goenaga, Koldo Gojenola, Yoav Goldberg, Jan Hajič, Anders Trærup Johannsen, Jenna Kanerva, Juha Kuokkala, Veronika Laippala, Alessandro Lenci, Krister Lindén, Nikola Ljubešić, Teresa Lynn, Christopher Manning, Héctor Alonso Martínez, Ryan McDonald, Anna Missilä, Simonetta Montemagni, Joakim Nivre, Hanna Nurmi, Petya Osenova, Slav Petrov, Jussi Piitulainen, Barbara Plank, Prokopis Prokopidis, Sampo Pyysalo, Wolfgang Seeker, Mojgan Seraji, Natalia Silveira, Maria Simi, Kiril Simov, Aaron Smith, Reut Tsarfaty, Veronika Vincze, and Daniel Zeman. 2015. Universal dependencies 1.1. LINDAT/CLARIN digital library at Institute of Formal and Applied Linguistics, Charles University in Prague.

Regina Barzilay and Kathleen R McKeown. 2005. Sentence fusion for multidocument news summarization. *Computational Linguistics*, 31(3):297–328.

Anja Belz, Michael White, Dominic Espinosa, Eric Kow, Deirdre Hogan, and Amanda Stent. 2011. The first surface realisation shared task: Overview and evaluation results. In *Proceedings of the 13th European Workshop on Natural Language Generation*, pages 217–226.

Bernd Bohnet, Simon Mille, Benoît Favre, and Leo Wanner. 2011. StuMaBa : from deep representation to surface. In *Proceedings of the 13th European workshop on natural language generation*, pages 232–235.

Bernd Bohnet, Anders Björkelund, Jonas Kuhn, Wolfgang Seeker, and Sina Zarrieß. 2012. Generating non-projective word order in statistical linearization. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 928–939.

Fabienne Braune, Daniel Bauer, and Kevin Knight. 2014. Mapping between english strings and reentrant semantic graphs. In *Int. Conf. on Language Resources and Evaluation (LREC)*.

Pi-Chuan Chang and Kristina Toutanova. 2007. A discriminative syntactic word order model for machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, page 9.

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

Jason M Eisner. 1997. An empirical comparison of probability models for dependency grammar. Technical report, IRCS Report 96–11, University of Pennsylvania.

Katja Filippova and Michael Strube. 2009. Tree linearization in English: Improving language model based approaches. In *Proceedings of NAACL-HLT (Short Papers)*.

Richard Futrell, Kyle Mahowald, and Edward Gibson. 2015. Quantifying word order freedom in dependency corpora. In *Proceedings of the Third International Conference on Dependency Linguistics (Depling 2015)*, pages 91–100, Uppsala, Sweden.

Douwe Gelling, Trevor Cohn, Phil Blunsom, and Joao Graça. 2012. The pascal challenge on grammar induction. In *Proceedings of the NAACL-HLT Workshop on the Induction of Linguistic Structure*, pages 64–80.

Wei He, Haifeng Wang, Yuqing Guo, and Ting Liu. 2009. Dependency based Chinese sentence realization. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 809–816.

Slava M Katz. 1987. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 35(3):400–401.

Dan Klein and Christopher D Manning. 2004. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, page 478.

Yijia Liu, Yue Zhang, Wanxiang Che, and Bing Qin. 2015. Transition-based syntactic linearization. In *Proceedings of NAACL*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318.

Rajakrishnan Rajkumar and Michael White. 2014. Better surface realization through psycholinguistics. *Language and Linguistics Compass*, 8(10):428–448.

Michael White and Rajakrishnan Rajkumar. 2012. Minimal dependency length in realization ranking. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 244–255.

Yue Zhang. 2013. Partial-tree linearization: generalized word ordering for text synthesis. In *Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*, pages 2232–2238. AAAI Press.

Summarization Based on Embedding Distributions

Hayato Kobayashi **Masaki Noguchi** **Taichi Yatsuka**
Yahoo Japan Corporation
9-7-1 Akasaka, Minato-ku, Tokyo 107-6211, Japan
{hakobaya, manoguch, tyatsuka}@yahoo-corp.jp

Abstract

In this study, we consider a summarization method using the document level similarity based on embeddings, or distributed representations of words, where we assume that an embedding of each word can represent its "meaning." We formalize our task as the problem of maximizing a submodular function defined by the negative summation of the nearest neighbors' distances on *embedding distributions*, each of which represents a set of word embeddings in a document. We proved the submodularity of our objective function and that our problem is asymptotically related to the KL-divergence between the probability density functions that correspond to a document and its summary in a continuous space. An experiment using a real dataset demonstrated that our method performed better than the existing method based on sentence-level similarity.

1 Introduction

Document summarization aims to rephrase a document in a short form called a summary while keeping its "meaning." In the present study, we aim to characterize the meaning of a document using embeddings or distributed representations of words in the document, where an embedding of each word is represented as a real valued vector in a Euclidean space that corresponds to the word (Mikolov et al., 2013a; Mikolov et al., 2013b).

Many previous studies have investigated summarization (Lin and Bilmes, 2010; Lin and Bilmes, 2011; Lin and Bilmes, 2012; Sipos et al., 2012; Morita et al., 2013), but to the best of our knowledge, only one (Kågebäck et al., 2014) considered a direct summarization method using embeddings, where the summarization problem was formalized as maximizing a submodular function defined by the summation of cosine similarities based on sentence embeddings. Essentially, this method assumes linear meanings since the objective function is characterized by the summation of sentence-level similarities. However, this assumption is not always valid in real documents, and thus there may be a better combination of two other sentences than the best and second best sentences in terms of similarity in a document.

In this study, we consider a summarization method based on document-level similarity, where we assume the non-linearity of meanings. First, we examine an objective function defined by a cosine similarity based on document embeddings instead of sentence embeddings. Unfortunately, in contrast to our intuition, this similarity is not submodular, which we disprove later. Thus, we propose a valid submodular function based on *embedding distributions*, each of which represents a set of word embeddings in a document, as the document-level similarity. Our objective function is calculated based on the nearest neighbors' distances on embedding distributions, which can be proved to be asymptotically related to KL-divergence in a continuous space. Several studies (Lerman and McDonald, 2009; Haghighi and Vanderwende, 2009) have addressed summarization using KL-divergence, but they calculated KL-divergence based on word distributions in a discrete space. In other words, our study is the first attempt to summarize by asymptotically estimating KL-divergence based on embedding distributions in a continuous space. In addition, they involved the inference of complex models, whereas our method is quite simple but still powerful.

2 Preliminaries

We treat a document as a bag-of-sentences and a sentence as a bag-of-words. Formally, let D be a document, and we refer to an element

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1984–1989,
Lisbon, Portugal, 17-21 September 2015.

$s \in D$ of a sentence and $w \in s$ of a word. We denote the size of a set S by $|S|$. Note that D and s are defined as multisets. For example, we can define a document such as $D := \{s_1, s_2\}$ with $s_1 := \{\texttt{just}, \texttt{do}, \texttt{it}\}$ and $s_2 := \{\texttt{never}, \texttt{say}, \texttt{never}\}$, which correspond to two sentences "Just do it" and "Never say never," respectively. From the definition, we have $|s_1| = 3$ and $|s_2| = 3$.

2.1 Submodularity

Submodularity is a property of set functions, which is similar to the convexity or concavity of continuous functions.

We formally define submodularity as follows.

Definition 1 (Submodularity)**.** Given a set X, a set function $f : 2^X \rightarrow \mathbb{R}$ is called *submodular* if for any two sets S_1 and S_2 such that $S_1 \subset S_2 \subset X$ and element $x \in X \setminus S_2$,

$$f(S_1 \cup \{x\}) - f(S_1) \geq f(S_2 \cup \{x\}) - f(S_2).$$

For simplicity, we define $f_S(x) := f(S \cup \{x\}) - f(S)$, which is called *the marginal value of x with respect to S*. A set function f is called *monotone* if $f_S(x) \geq 0$ for any set $S \subset X$ and element $x \in X \setminus S$.

If a set function f is monotone submodular, we can approximate the optimal solution efficiently by a simple greedy algorithm, which iteratively selects $x^* = \mathrm{argmax}_{x \in X \setminus S_i} f_{S_i}(x)$ where ties are broken arbitrarily, and we substitute $S_{i+1} = S_i \cup \{x^*\}$ in the i-th iteration beginning with $S_0 = \emptyset$. This algorithm is quite simple but it is guaranteed to find a near optimal solution within $1 - 1/\mathrm{e} \approx 0.63$ (Calinescu et al., 2007).

2.2 Embedding

An embedding or distributed representation of a word is a real valued vector in an m-dimensional Euclidean space $\mathbb{R}^m$, which expresses the "meaning" of the word. We denote an embedding of a word w by $\vec{w} \in \mathbb{R}^m$. If for any two words w_1 and w_2, the meaning of w_1 is similar to that of w_2, then $\vec{w}_1$ is expected to be near to $\vec{w}_2$.

A recent study (Mikolov et al., 2013a) showed that a simple log-bilinear model can learn high quality embeddings to obtain a better result than recurrent neural networks, where the concept of embeddings was originally proposed in studies of neural language models (Bengio et al., 2003). In the present study, we use the CW Vector[1] and W2V Vector[2] which are also used in the previous study (Kågebäck et al., 2014).

3 Proposed Method

In this study, we focus on a summarization task as sentence selection in a document. The optimization framework in our task is the same as in the previous study and formalized in Algorithm 1, where w_s represents the pre-defined weight or cost of a sentence s, e.g., sentence length, and r is its scaling factor. This algorithm, called modified greedy, was proposed in (Lin and Bilmes, 2010) and interestingly performed better than the state-of-the-art abstractive approach as shown in (Lin and Bilmes, 2011). Note that we have omitted the notation of D from f for simplicity because D is fixed in an optimization process.

Algorithm 1: Modified greedy algorithm.

Data: Document D, objective function f, and summary size ℓ.
Result: Summary $C \subset D$.

$C \leftarrow \emptyset$; $U \leftarrow D$;
while $U \neq \emptyset$ **do**
  $s^* \leftarrow \mathrm{argmax}_{s \in U} f_C(s)/(w_s)^r$;
  if $\sum_{s \in C} w_s + w_{s^*} \leq \ell$ **then** $C \leftarrow C \cup \{s^*\}$;
  $U \leftarrow U \setminus \{s^*\}$;
$s^* \leftarrow \mathrm{argmax}_{s \in D : w_s \leq \ell} f(\{s\})$;
return $C \leftarrow \mathrm{argmax}_{C' \in \{C, \{s^*\}\}} f(C')$;

Similarity Based on Document Embeddings

First, we examine an objective function f^{Cos} defined by a cosine similarity based on document embeddings. An embedding of a document D is defined as $\mathbf{v}_D := \sum_{s \in D} \sum_{w \in s} \vec{w}$. We formalize the objective function f^{Cos} as follows.

$$f^{Cos}(C) := \frac{\mathbf{v}_C \cdot \mathbf{v}_D}{\|\mathbf{v}_C\| \, \|\mathbf{v}_D\|}.$$

Note that the optimal solution does not change, if we use an average embedding $\mathbf{v}_D / \sum_{s \in D} |s|$ instead of $\mathbf{v}_D$. The next theorem shows that a solution of f^{Cos} by Algorithm 1 is not guaranteed to be near optimal.

Theorem 1. *f^{Cos} is not submodular.*

[1] http://metaoptimize.com/projects/wordreprs

[2] https://code.google.com/p/word2vec

Proof. A simple counterexample is sufficient to prove the theorem. Let us consider $D := \{s_1 := \{w_1\},\ s_2 := \{w_2\},\ s_3 := \{w_3\},\ s_4 := \{w_4\}\}$ with corresponding vectors $\vec{w}_1 := (1,1)$, $\vec{w}_2 := (1,2)$, $\vec{w}_3 := (1,-1)$, and $\vec{w}_4 := (1,-2)$, respectively. In this case, the document embedding $\mathbf{v}_D$ is $(4,0)$. We set $C_1 := \{s_1\}$ and $C_2 := \{s_1, s_2\}$. Clearly, $C_1 \subset C_2$. However, we obtain $f^{Cos}_{C_1}(s_4) = f^{Cos}(\{s_1, s_4\}) - f^{Cos}(\{s_1\}) \approx 0.187$ and $f^{Cos}_{C_2}(s_4) = f^{Cos}(\{s_1, s_2, s_4\}) - f^{Cos}(\{s_1, s_2\}) \approx 0.394$. Therefore, we have $f^{Cos}_{C_2}(s_4) > f^{Cos}_{C_1}(s_4)$. □

Similarity Based on Embedding Distributions

We propose a valid submodular objective function f^{NN} based on embedding distributions. The key observation is that for any two embedding distributions A and B, when A is similar to B, each embedding in A should be near to some embedding in B. In order to formalize this idea, we define *the nearest neighbor of a word w in a summary C* as $n(w, C) := \mathrm{argmin}_{v \in s : s \in C, \vec{w} \neq \vec{v}}\, d(\vec{w}, \vec{v})$, where d is the Euclidian distance in the embedding space, i.e., $d(\vec{w}, \vec{v}) := \|\vec{w} - \vec{v}\|$. We denote the distance of w to its nearest neighbor $n := n(w, C)$ by $N(w, C) := d(\vec{w}, \vec{n})$. Finally, we define f^{NN} as follows:

$$f^{NN}(C) := -\sum_{s \in D} \sum_{w \in s} g(N(w, C)),$$

where g is a non-decreasing scaling function. The function f^{NN} represents the negative value $-\delta$ of dissimilarity δ between a document and summary based on embedding distributions. Note that we can use sentence embeddings instead of word embeddings as embedding distributions, although we focus on word embeddings in this section.

The next theorem shows the monotone submodularity of our objective function, which means that a solution of f^{NN} by Algorithm 1 is guaranteed to be near optimal.

Theorem 2. *f^{NN} is monotone submodular.*

Proof. (Monotonicity) First, we prove the monotonicity. For simplicity, we use the following two abbreviations: $C^s := C \cup \{s\}$ and $\sum_w^D := \sum_{s \in D} \sum_{w \in s}$. For any set $C \subset D$ of sentences and sentence $s \in D \setminus C$, we have $f^{NN}_C(s) = f^{NN}(C^s) - f^{NN}(C) = \sum_w^D (g(N(w, C)) - g(N(w, C^s)))$. Since $C \subset C^s$, obviously $N(w, C) \geq N(w, C^s)$ holds. Therefore, we obtain $f^{NN}_C(s) \geq 0$ from the non-decreasing property of g.

(Submodularity) Next, we prove the submodularity. For any two sets C_1 and C_2 of sentences such that $C_1 \subset C_2 \subset D$, and sentence $s \in D \setminus C_2$, we have $f^{NN}_{C_1}(s) - f^{NN}_{C_2}(s) = f^{NN}(C^s_1) - f^{NN}(C_1) - (f^{NN}(C^s_2) - f^{NN}(C_2)) = \sum_w^D (g(N(w, C_1)) - g(N(w, C^s_1)) - g(N(w, C_2)) + g(N(w, C^s_2)))$. Let $\alpha := g(N(w, C_1)) - g(N(w, C^s_1)) - g(N(w, C_2)) + g(N(w, C^s_2))$.

If $n(w, C^s_2) \in s$, then $n(w, C^s_1) \in s$ holds, since $C^s_1 \subset C^s_2$. This means that $N(w, C^s_2) = N(w, C^s_1) = N(w, \{s\})$. Clearly, $N(w, C_1) \geq N(w, C_2)$, since $C_1 \subset C_2$. Therefore, we obtain $\alpha \geq 0$ from the non-decreasing property of g.

If $n(w, C^s_2) \notin s$ and $n(w, C^s_1) \notin s$, we have $N(w, C^s_1) = N(w, C_1)$ and $N(w, C^s_2) = N(w, C_2)$. This indicates that $\alpha = 0$.

If $n(w, C^s_2) \notin s$ and $n(w, C^s_1) \in s$, so similarly $N(w, C^s_1) \leq N(w, C_1)$ and $N(w, C^s_2) = N(w, C_2)$ hold. Therefore, we obtain $\alpha \geq 0$. □

The objective function f^{NN} is simply heuristic for small documents, but the next theorem shows that f^{NN} is asymptotically related to an approximation of KL-divergence in a continuous space, if g is a logarithmic function. This result implies that we can use mathematical techniques of a continuous space for different NLP tasks, by mapping a document into a continuous space based on word embeddings.

Theorem 3. *Suppose that we have a document D and two summaries C_1 and C_2 such that $|C_1| = |C_2|$, which are samples drawn from some probability density functions p, q, and r, i.e., $D \sim p$, $C_1 \sim q$, and $C_2 \sim r$, respectively. If the scaling function g of f^{NN} is a logarithmic function, the order relation of the expectations of $f^{NN}(C_1)$ and $f^{NN}(C_2)$ is asymptotically the same as that of the KL-divergences $\mathcal{D}_{KL}(p \,||\, r)$ and $\mathcal{D}_{KL}(p \,||\, q)$, i.e., $\mathbb{E}[f^{NN}(C_1)] - \mathbb{E}[f^{NN}(C_2)] > 0 \Leftrightarrow \mathcal{D}_{KL}(p \,||\, r) - \mathcal{D}_{KL}(p \,||\, q) > 0$, as $|D| \to \infty$, $|C_1| \to \infty$, and $|C_2| \to \infty$.*

Proof. Let m be the dimension on embeddings. Using a divergence estimator based on nearest neighbor distances in (Pérez-Cruz, 2009; Wang et al., 2009), we can approximate $\mathcal{D}_{KL}(p \,||\, q)$ by $\widehat{\mathcal{D}}_{KL}(D, C_1) := \frac{m}{|D|} \sum_w^D \ln \frac{N(w,C_1)}{N(w,D)} + \ln \frac{|C_1|}{|D|-1}$. Therefore, we obtain $\widehat{\mathcal{D}}_{KL}(D, C_2) - \widehat{\mathcal{D}}_{KL}(D, C_1) \propto \sum_w^D \ln N(w, C_2) - \sum_w^D \ln N(w, C_1)$. Since $g(x) = \ln(x)$, we have $f^{NN}(C_1) - f^{NN}(C_2) > 0$ if

and only if $\widehat{\mathcal{D}}_{KL}(D, C_2) - \widehat{\mathcal{D}}_{KL}(D, C_1) > 0$ holds. The fact that $\mathbb{E}[\widehat{\mathcal{D}}_{KL}(D, C_1)] \rightarrow \mathcal{D}_{KL}(p \mid\mid q)$ as $|C_1| \rightarrow \infty$ and $|D| \rightarrow \infty$ concludes the theorem. □

4 Experiments

We compared our two proposed methods DocEmb and EmbDist with two state-of-the-art methods SenEmb and TfIdf. The first two methods DocEmb and EmbDist represent Algorithm 1 with our proposed objective functions f^{Cos} and f^{NN}, respectively. TfIdf represents Algorithm 1 with an objective function based on the sum of cosine similarities of tf-idf vectors that correspond to sentences, which was proposed in (Lin and Bilmes, 2011). SenEmb uses a cosine similarity measure based on embeddings instead of tf-idf vectors in the same framework as TfIdf, which was proposed in (Kågebäck et al., 2014).

We conducted an experiment with almost the same setting as in the previous study, where they used the Opinosis dataset (Ganesan et al., 2010). This dataset is a collection of user reviews in 51 different topics such as hotels, cars, and products; thus, it is more appropriate for evaluating summarization of user-generated content than well-known DUC datasets, which consist of formal news articles. Each topic in the collection comprises 50–575 sentences and includes four and five gold standard summaries created by human authors, each of which comprises 1–3 sentences.

We ran an optimization process to choose sentences within 100 words[3] by setting the summary size and weights as $\ell = 100$ and $w_s = |s|$ for any sentence s, respectively. As for TfIdf and SenEmb, we set a cluster size of k-means as $k = |D|/5$ and chose the best value for a threshold coefficient α, trade-off coefficient λ, and the scaling factor r, as in (Lin and Bilmes, 2011). Note that our functions DocEmb and EmbDist have only one parameter r, and we similarly chose the best value of r. Regarding DocEmb, EmbDist, and SenEmb, we used the best embeddings from the CW Vector and W2V Vector for each method, and created document and sentence embeddings by averaging word embeddings with tf-idf weights since it performed better in this experiment. In the case of EmbDist, we used a variant of f^{NN} based on distributions of sentence embeddings. In addition, we examined three scaling functions: logarithmic, linear, and exponential functions, i.e., $\ln x$, x, e^x, respectively.

[3]The previous work used a sentence-based constraint as $\ell = 2$ and $w_s = 1$, but we changed the setting since the variation in length has a noticeable impact on ROUGE scores as suggested in (Hong et al., 2014).

	R-1	R-2	R-3	R-4
ApxOpt	62.22	21.60	8.71	4.56
EmbDist ($\ln x$)	56.00	16.70	4.93	**1.89**
EmbDist (x)	55.70	15.73	4.59	1.84
EmbDist (e^x)	**56.29**	15.96	4.43	1.39
DocEmb	55.80	13.59	3.23	0.90
SenEmb	53.96	15.42	3.97	1.10
TfIdf	52.97	**17.24**	**5.40**	1.49

Table 1: ROUGE-N (R-N) metrics of DocEmb, EmbDist, SenEmb, and TfIdf.

We calculated the ROUGE-N metric (Lin, 2004)[4], which is a widely-used evaluation metric for summarization methods. ROUGE-N is based on the co-occurrence statistics of N-grams, and especially ROUGE-1 has been shown to have the highest correlation with human summaries (Lin and Hovy, 2003). ROUGE-N is similar to the BLEU metric for machine translation, but ROUGE-N is a recall-based metric while BLEU is a precision-based metric.

Table 1 shows the results obtained for ROUGE-N (N ≤ 4) using DocEmb, EmbDist, SenEmb, and TfIdf. ApxOpt represents the approximation results of the optimal solution in our problem, where we optimized ROUGE-1 with the gold standard summaries by Algorithm 1. The obtained results indicate that our proposed method EmbDist with exponential scaling performed the best for ROUGE-1, which is the best metric in terms of correlation with human summaries. The W2V Vector was the best choice for EmbDist. Furthermore, the other proposed method DocEmb performed better than the state-of-the-art methods SenEmb and TfIdf, although DocEmb is not theoretically guaranteed to obtain a near optimal solution. These results imply that our methods based on the document-level similarity can capture more complex meanings than the sentence-level similarity. On the other hand, TfIdf with tf-idf vectors performed the worst for ROUGE-1. A possible reason is that a wide variety of expressions by users made it difficult to calculate similarities. This also suggests that embedding-based methods

[4]We used their software ROUGE version 1.5.5 with the parameters: -n 4 -m -a -l 100 -x -c 95 -r 1000 -f A -p 0.5 -t 0.

naturally have robustness for user-generated content.

In the case of N $\geq$ 2, `TfIdf` performed the best for ROUGE-2 and ROUGE-3, while `EmbDist` with logarithmic scaling is better than `TfIdf` for ROUGE-4. According to (Lin and Hovy, 2003), the higher order ROUGE-N is worse than ROUGE-1 since it tends to score grammaticality rather than content. Conversely, Rankel et al. (2013) reports that there is a dataset where the higher order ROUGE-N is correlated with human summaries well. We may need to conduct human judgments to decide which metric is the best in this dataset for more accurate comparison. However, it is still important that our simple objective functions can obtain good results competing with the state-of-the-art methods.

5 Conclusion

In this study, we proposed simple but powerful summarization methods using the document-level similarity based on embeddings, or distributed representations of words. Our experimental results demonstrated that the proposed methods performed better than the existing state-of-the-art methods based on the sentence-level similarity. This implies that the document-level similarity can capture more complex meanings than the sentence-level similarity.

Recently, Kusner et al. (2015) independently discovered a similar definition to our objective function f^{NN} through a different approach. They constructed a dissimilarity measure based on a framework using Earth Mover's Distance (EMD) developed in the image processing field (Rubner et al., 1998; Rubner et al., 2000). EMD is a consistent measure of distance between two distributions of points. Interestingly, their heuristic lower bound of EMD is exactly the same as $-f^{NN}$ with a linear scaling function, i.e., $g(x) = x$. Moreover, they showed that this bound appears to be tight in real datasets. This suggests that our intuitive framework can theoretically connect the two well-known measures, KL-divergence and EMD, based on the scaling of distance. Note that, to the best of our knowledge, there is currently no known study that considers such a theoretical relationship.

In future research, we will explore other scaling functions suitable for our problem or different problems. A promising direction is to consider a relative scaling function to extract a biased summary of a document. This direction should be useful for query-focused summarization tasks.

Acknowledgments

The authors would like to thank the reviewers for their helpful comments, especially about Earth Mover's Distance.

References

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A Neural Probabilistic Language Model. *Journal of Machine Learning Research (JMLR 2003)*, 3:1532–4435.

Gruia Calinescu, Chandra Chekuri, Martin Pál, and Jan Vondrák. 2007. Maximizing a Submodular Set Function Subject to a Matroid Constraint (Extended Abstract). In *Proceedings of the 12th International Conference on Integer Programming and Combinatorial Optimization (IPCO 2007)*, pages 182–196. Springer-Verlag.

Kavita Ganesan, ChengXiang Zhai, and Jiawei Han. 2010. Opinosis: A Graph-based Approach to Abstractive Summarization of Highly Redundant Opinions. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING 2010)*, pages 340–348. Association for Computational Linguistics.

Aria Haghighi and Lucy Vanderwende. 2009. Exploring Content Models for Multi-document Summarization. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT 2009)*, pages 362–370. Association for Computational Linguistics.

Kai Hong, John Conroy, Benoit Favre, Alex Kulesza, Hui Lin, and Ani Nenkova. 2014. A Repository of State of the Art and Competitive Baseline Summaries for Generic News Summarization. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC 2014)*, pages 26–31. European Language Resources Association (ELRA).

Mikael Kågebäck, Olof Mogren, Nina Tahmasebi, and Devdatt Dubhashi. 2014. Extractive Summarization using Continuous Vector Space Models. In *Proceedings of the 2nd Workshop on Continuous Vector Space Models and their Compositionality (CVSC 2014)*, pages 31–39. Association for Computational Linguistics.

Matt J. Kusner, Yu Sun, Nicholas I. Kolkin, and Kilian Q. Weinberger. 2015. From Word Embeddings To Document Distances. In *Proceedings of the 32nd International Conference on Machine Learning (ICML 2015)*, pages 957–966. JMLR.org.

Kevin Lerman and Ryan McDonald. 2009. Contrastive Summarization: An Experiment with Consumer Reviews. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT 2009)*, pages 113–116. Association for Computational Linguistics.

Hui Lin and Jeff Bilmes. 2010. Multi-document Summarization via Budgeted Maximization of Submodular Functions. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT 2010)*, pages 912–920. Association for Computational Linguistics.

Hui Lin and Jeff Bilmes. 2011. A Class of Submodular Functions for Document Summarization. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL 2011)*, pages 510–520. Association for Computational Linguistics.

Hui Lin and Jeff Bilmes. 2012. Learning Mixtures of Submodular Shells with Application to Document Summarization. In *Proceedings of the 28th Conference on Uncertainty in Artificial Intelligence (UAI 2012)*, pages 479–490. AUAI.

Chin-Yew Lin and Eduard Hovy. 2003. Automatic Evaluation of Summaries Using N-gram Co-Occurrence Statistics. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology (NAACL-HLT 2003)*. Association for Computational Linguistics.

Chin-Yew Lin. 2004. ROUGE: A Package for Automatic Evaluation of summaries. In *Proceedings of the Workshop on Text Summarization Branches Out (WAS 2004)*. Association for Computational Linguistics.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013a. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems 26 (NIPS 2013)*, pages 3111–3119. Curran Associates, Inc.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013b. Linguistic Regularities in Continuous Space Word Representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2013)*, pages 746–751. Association for Computational Linguistics.

Hajime Morita, Ryohei Sasano, Hiroya Takamura, and Manabu Okumura. 2013. Subtree Extractive Summarization via Submodular Maximization. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL 2013)*, pages 1023–1032. Association for Computational Linguistics.

Fernando Pérez-Cruz. 2009. Estimation of Information Theoretic Measures for Continuous Random Variables. In *Advances in Neural Information Processing Systems 21 (NIPS 2009)*, pages 1257–1264. Curran Associates, Inc.

Peter A. Rankel, John M. Conroy, Hoa Trang Dang, and Ani Nenkova. 2013. A Decade of Automatic Content Evaluation of News Summaries: Reassessing the State of the Art. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL 2013)*, pages 131–136. Association for Computational Linguistics.

Yossi Rubner, Carlo Tomasi, and Leonidas J. Guibas. 1998. A Metric for Distributions with Applications to Image Databases. In *Proceedings of the Sixth International Conference on Computer Vision (ICCV 1998)*, pages 59–66. IEEE Computer Society.

Yossi Rubner, Carlo Tomasi, and Leonidas J. Guibas. 2000. The Earth Mover's Distance as a Metric for Image Retrieval. *International Journal of Computer Vision*, 40(2):99–121.

Ruben Sipos, Adith Swaminathan, Pannaga Shivaswamy, and Thorsten Joachims. 2012. Temporal Corpus Summarization Using Submodular Word Coverage. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management (CIKM 2012)*, pages 754–763. ACM.

Qing Wang, S.R. Kulkarni, and S. Verdu. 2009. Divergence Estimation for Multidimensional Densities Via k-Nearest-Neighbor Distances. *IEEE Transactions on Information Theory*, 55(5):2392–2405.

Reversibility reconsidered: finite-state factors for efficient probabilistic sampling in parsing and generation

Marc Dymetman[†], **Sriram Venkatapathy**[‡], **Chunyang Xiao**[†]

[†]Xerox Research Centre Europe, Grenoble, France
[‡]Amazon, Machine Learning Team, Bangalore, India*
[†]{*first.last*}@xrce.xerox.com, [‡]vesriram@amazon.com

Abstract

We restate the classical logical notion of generation/parsing reversibility in terms of feasible probabilistic sampling, and argue for an implementation based on finite-state factors. We propose a modular decomposition that reconciles generation accuracy with parsing robustness and allows the introduction of dynamic contextual factors. *(Opinion Piece)*

1 Introduction

The objective of Natural Language Understanding (NLU) is to map linguistic utterances to semantic representations, that of Natural Language Generation (NLG) to map semantic representations to linguistic utterances. In most of NLP practice, these two objectives are handled by different processes, and computational linguists rarely operate at the intersection of the two subdomains.

For a few years around the early nineties, based both on cognitive, linguistic, and engineering considerations, there was a surge of interest in so called *reversible grammar* approaches to NLP, where one and the same grammatical specification could serve both for parsing utterance x into logical form z, but also for generating x from z (Strzalkowski, 1994).

We start by a brief review of this historical non-probabilistic notion of reversibility and point out certain of its weaknesses, in particular regarding robustness; we then give in section 3 a new probabilistic definition of reversibility; then, in section 4 we argue for a reversibility model based on modular weighted finite-state transducers. We end with a discussion of recent related work.

*Work done while at XRCE.

2 Classical reversibility

The most direct approaches to NLU attempt to design procedures for semantic parsing that, given an input utterance x, produce a semantic representation z, by following a number of intermediate steps where the surface form is gradually transformed into semantic structure. Such "procedural" approaches to semantic parsing are typically very hard or impossible to invert: starting from a semantic representation z, there is no simple process that is able to find an x which, when given to the parser, would produce z. Formally, a Boolean relation $r(x, z)$ can be such that the question $?\exists z\ r(x, z)$ is decidable for all x's, while the reciprocal question $?\exists x\ r(x, z)$ is undecidable for some z's (Dymetman, 1991).[1] One of the motivations for the emerging paradigm of unification grammars at the end of the eighties was the clean separation they promised between specifying well-formed linguistic structures, both on the syntactic and semantic levels, through a formal description of the relation $r(x, z)$, and producing efficient implementations of the specification; in particular, there was much hope that such formalisms would be conductive to effective reversibility (by contrast to variable assignment, variable unification is inherently symmetrical), that is, to feasible (and if possible efficient) implementations of the parsing problem $r(x, ?)$ and of the generation problem $r(?, z)$.

To some extent, this hope was validated through a number of works at the time, mostly involving machine translation applications, and constraining in more or less explicit ways the specification of r (van Noord, 1990). However, for the non-statistical approaches to parsing then strongly dominant, *robustness* was an issue: a parser had to

[1]Some intuition into the issue may be gained by considering typical techniques of public key cryptography, which rely on the difficulty of inverting some simple arithmetic computations.

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1990–1995, Lisbon, Portugal, 17-21 September 2015.

either accept or reject a given input x, with no intermediary options, and in order to be able to parse actual utterances, with all their empirical diversity, parsers had to be rather tolerant. In the procedural view of parsing, such robustness issues could often be mitigated through engineering tricks such as ordering the rules from strict to lax, where grammatical constructions were given preference over less conventional ones; however, when trying to move to reversible grammars, these tricks could not be reproduced: if the grammar was able to parse an x into z, then, by design, it was also able to generate x from z, and there was no obvious way, in these non-probabilistic approaches, to distinguish between producing a linguistically correct x or producing a deviant or incorrect one.

3 Probabilistic reversibility

In the classical non-probabilistic case, a (relative) consensus existed around the fact that a reversible grammar should be, as we indicated above, a formal specification of the relation $r(x, z)$ such that the problems $r(x, ?)$ and $r(?, z)$ were effectively solvable.

Transposing this to the probabilistic world, we propose the following semi-formal Definition:

A probabilistic reversible grammar is a formal specification of a joint probability distribution $p(x, z)$ over logical forms z and utterance strings x such that the conditional distributions $p(z|x) \stackrel{\text{def}}{=} \frac{p(x,z)}{\sum_{z'} p(x,z')}$ (parsing) and $p(x|z) \stackrel{\text{def}}{=} \frac{p(x,z)}{\sum_{x'} p(x',z)}$ (generation) can be efficiently sampled from.[2]

Why such focus on *sampling*? We could have chosen other definitions of parsing (and similarly for generation), for instance the ability to return the *most probable* z given x, i.e. to return $\text{argmax}_z\, p(z|x)$; however sampling is the most direct way of providing a concrete view of the underlying probabilistic distribution, and has many applications to learning, so we think the definition above is reasonable (see also footnote[4]).

[2]We note the "semi-formal" aspect of this definition: contrarily to the classical case, which has a formal notion of effective computation, there is no universally accepted notion of effective sampling from a probability distribution. For many probability distributions, the only feasible sampling approaches are the *MCMC* techniques (Robert and Casella, 2004), which typically do not come with convergence guarantees; in some situations, *exact sampling* techniques are applicable, which come with much better guarantees. We will see that the approach proposed in section 4 allows such exact sampling to take place.

4 Finite-state models for reversibility

Finite-state transducers have properties which make them uniquely suited to implementing reversible linguistic specifications in the above sense. Consider a simple weighted string-to-string transducer $\tau(s, t)$, where s, t are strings, and where the underlying semiring is the "probabilistic semiring" over the nonnegative reals, addition and multiplication having their usual interpretations. Such a transducer *preserves regularity*, both in the forward (resp. reverse) directions, meaning that the image through τ of any weighted regular language over s (resp. over t) is again a weighted regular language over t (resp. over s). In particular the forward (resp. reverse) image of a fixed string s_0 (resp t_0) can be computed *in a compact form* as a weighted finite-state automaton (FSA) over t (resp. s), which we can denote by $\tau(s_0, \cdot)$ (resp. $\tau(\cdot, t_0)$). A weighted FSA can be easily normalized into a probabilistic FSA[3] and, from this probabilitic FSA exact samplers for the "parser" $\tau(s_0, \cdot)$ and for the "generator" $\tau(\cdot, t_0))$ are directly obtained.[4]

In general, some of the properties that make weighted FSAs and FSTs — over strings or trees — specially relevant for probabilistic models of language are the following: (i) they allow compact representations of complex probability distributions over linguistic objects (automata) or pairs of linguistic objects (transducers), (ii) they permit efficient exact sampling (and efficient optimization over derivations (but not always over strings)), (iii) they support modularity: intersection of automata, composition of transducers, projections of an automaton through a transducer.[5]

Conceptual architecture Armed with these general considerations, let us now propose a conceptual architecture based on a small number of

[3]That is, into a weighted FSA such the weights of the transitions from each state sum to 1.

[4] While *sampling* strings from a weighted finite-state automaton is simple, finding the most probable *string* (not *path*) in a probabilistic FSA is an NP-hard problem (Casacuberta and de la Higuera, 2000), and one has to resort to the so-called Viterbi approximation (assuming that the most probable path projects into the most probable string). Contrary to popular belief, sampling can sometimes be simpler than optimization.

[5]Outside of the realm of finite-state machines, this modularity is typically impossible to obtain. Thus, in general, the availability of a sampler for a distribution $p(x)$ (resp. a distribution $q(x)$) does not imply that we can efficiently sample from the product (i.e. intersection) $p(x).q(x)$, but we can in case p and q are both represented by weighted FSAs.

finite-state modules, which attempts to satisfy the definition given above for probabilistic reversibility, to address the problem of robustness that we described earlier, and can also support contextual preferences. We illustrate the approach with some simple examples of human-machine dialogues (between a customer and a virtual agent), a domain for which reversibility has high relevance, due to effects such as self-monitoring (Neumann, 1998; Levelt, 1983), interleaving of understanding and generation (Otsuka and Purver, 2003), and lexical entrainment (Brennan, 1996).

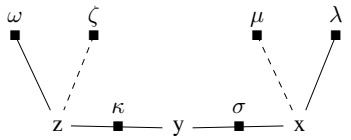


Figure 1: Reversible specification through finite-state factors.

The conceptual architecture is shown in Figure 1. Formally, the figure represents a probabilistic graphical model in so-called factor form, where the factors are $\omega, \kappa, \sigma, \lambda$ (we have also indicated for future reference the "contextual" factors ζ, μ, that we ignore for now). The factors take as arguments three types of objects: z is a logical form, that is, a structured object which can be naturally represented as a tree, x is a surface string, and y is a latent "underlying" string that corresponds to one of a small collection of "canonical" texts for realizing the logical form z (more about that later).

Each factor is realized through a weighted finite-state machine (acceptor or transducer) over strings or trees (Mohri, 2009; Fülöp and Vogler, 2009; Maletti, 2010; Graehl et al., 2008).

The λ factor is a string automaton that represents a standard ngram language model (typically specific to domain), in other words a probability distribution over utterances x. *Symmetrically*, the regular tree automaton ω represents a distribution over logical forms z, which can be seen as playing a similar role to the language model, but at the semantic level, namely telling us what are the possible/likely logical forms in a certain domain.[6]

The "canonical factor" κ is a weighted tree-to-string transducer (Graehl et al., 2008), which implements a relation between logical forms z and a small number of latent "canonical" texts y realizing these logical forms. For example, κ may associate the logical form (dialog act) $z = \texttt{wad}(\texttt{batLife}, \texttt{iphone6})$ — with `wad` an abbreviation for "what is the value of this attribute on this device?", and `batLife` an abbreviation for "battery life" —, with such a canonical text (among a few others) as: *What is the battery life of the Iphone 6?*.

The "similarity factor" σ is a weighted string-to-string finite state transducer which gives scores to x, y according to a notion of similarity. It has the role of "bridging" the gap between the actual utterances x and the latent canonical utterances y. The intention behind the similarity factor is to "decouple" the task of modeling some possible realizations of a given logical form from the task of recognizing that a given more or less well-formed input is a variant of such a realization. This factor relates the two strings y and x, where y is a possible canonical utterance in the limited repertory produced by κ, and x is an actual utterance, in particular any utterance that could be produced by a human speaker. So for instance suppose that the user's utterance is $x =$ *What about battery duration on this Iphone 6?*, we would like this x to have a significant similarity with the canonical utterance $y =$ *What is the battery life of the Iphone 6?* but a negligible similarity with another canonical utterance such as $y' =$ *What is the screen size of the Galaxy Trend?*.

Overall, the canonical factor $\kappa(z, y)$ concentrates more on a core "generation model", namely on producing some well-formed output y from a logical form z, while the similarity factor $\sigma(y, x)$ allows relating an actual user input x to a possible output y of the κ model. The main import of σ is then to allow to use the core generation *model defined by κ to be exploited for* robust semantic parsing.

Different instantiations of this scheme can be employed. In some preliminary experiments that we have performed,[7] σ is a simple edit-distance transducer (Mohri, 2003) which penalizes differently the discrepancies between x and y: strongly for some salient content words or named entities of the domain, weakly for less relevant content words and for non-content words, with limited use of local paraphrases (which can also be implemented

[6] In particular, the ω factor makes explicit the notion of a well-formed semantic representation, a notion often left implicit in semantic parsing.

[7] In these experiments, we used string-based approximations of the logical forms, and only employed string-based transducers from the OpenFST library.

through σ). This strategy seems to work reasonably well when the semantical repertory of the domain is restricted, because a large number of possible variants for x are "attracted" to the same underlying semantics. In domains where small nuances of expression may result in distinct semantics, the division of work between κ and σ may be different.

Parsing and Generation To understand the reversibility properties of the model of Figure 1, let us first simplify the description by assuming that z, instead of being a tree, is actually a string. Then both ω and λ are string automata, and both κ and σ string-to-string transducers. Such a specification satisfies our definition of probabilistic reversibility, exploiting well-known compositionality properties of weighted finite-state machines over strings (Mohri, 2009). For parsing, we start from a fixed x_0, and can project it through σ into a weighted FSA over y; in turn we can project this automaton onto an FSA over z, and finally intersect this automaton with ω, obtaining a final weighted "x_0-parser" automaton over z, representing a probability distribution from which we can draw exact samples as explained above.[8] Generation works in exactly the reverse way, starting from a z_0 and eventually building a "z_0-generator" automaton over x.

In the actual proposal, z is a tree, meaning that ω is a tree automaton, and κ a tree-to-string transducer. While finite-state tree automata correspond to a single concept, and share all the nice properties of string automata (Comon et al., 2007), the situation with tree-to-tree or tree-to-string transducers is more complicated (Maletti, 2010; Graehl et al., 2008): several variants exist, only some of which support the operations that our conceptual model requires (composition with the string transducer σ and intersection with the tree automaton ω). In particular, the "linear non-deleting top-down tree transducers" defined in (Maletti, 2010)[9] have the requisite properties.

Contextual factors We now briefly come back to the factors ζ (tree automaton) and μ (string automaton) of Figure 1, which highlight the usefulness of our modular finite-state architecture. These factors play similar roles to ω and λ, but they evolve dynamically with the context. In dialogue applications, utterances can often only be interpreted by reference to the current dialogue state (e.g. "ten hours" in the context of a question about battery life), and the ζ factor can be used as a compact representation of the current expectations of the dialogue manager about the next logical form, to be combined with the actual customer's utterance. Symmetrically, the μ factor can be used to represent such phenomena as lexical entrainment (Brennan, 1996), where the agent's utterance is oriented towards using similar wordings to the customer's.

5 Related work

The unique formal properties of finite-state machines, which favor modular decompositions of complex tasks, have long been exploited in Computational Linguistics. Tree transducers in particular have gained popularity in Statistical Machine Translation, starting with (Yamada and Knight, 2001), as described in the surveys (Maletti, 2010; Razmara, 2011).

The *reversibility* properties of finite-state transducers have been exploited to a more limited extent, starting with applications of non-weighted string-to-string transducers to morphological analysis and generation (Beesley, 1996).

Concerning the application of weighted finite-state tree machines to NLU/NLG reversibility, our proposal is strongly related on the one hand to the approach of (Jones et al., 2012), who explicitely proposes tree-to-string transducers as a tool for modelling semantic parsing and for training on semantically annotated data, and on the other hand to (Wong, 2007; Wong and Mooney, 2007), who focus more directly on the problem of inverting a semantic parser into a generator. Wong et al. do not explicitely use tree-based transducers, but rather a formalism inspired by SCFGs (synchronous context-free grammars), which essentially corresponds to a form of tree-to-string transducer. In relation to reversibility considerations, presentations in terms of synchronous formalisms have the interest that they are intrinsically symmetrical. Such formalisms have tight relations to tree-transducers (Shieber, 2004); one recently proposed generalization, "Interpreted Regular Tree Grammars" (Koller and Kuhlmann, 2011), allows

[8] We could also have precompiled a generic parser for all x's by first marginalizing the latent variable y through a composition of the transducers κ and σ, and then intersecting the resulting transducer with the automaton ω.

[9] The paper only defines tree-to-tree transducers, but tree-to-string variants can be derived easily.

multiple (possibly more than two) synchronized views of an underlying abstract derivation tree, and has the advantage of permitting a uniform treatment of strings and trees.

One important aspect in which our proposal differs from these previous approaches is in proposing to decouple the "core" task of mapping logical forms to well-formed latent canonical realizations from the task of relating these realizations to actual utterances, through an additional "similarity" transducer acting as a bridge.

This idea of a bridge is however close to another line of work in semantic parsing, not transducer based, namely (Berant and Liang, 2014; Wang et al., 2015). There, a simple generic grammar is used to generate canonical realizations from a repertory of possible logical forms (expressed in a variant of lambda calculus). Given an input to parse, simple heuristics are used to select a finite list of potential logical forms which are then ranked according to the (paraphrase-based) similarity of their associated canonical realization with the input. Thus in this approach, a form of generation plays an important role, not for its own sake, but as a tool for semantic parsing.

6 Conclusion

Because of their unique compositional properties, finite-state modules are a natural choice for implementing our definition of reversibility as efficient bidirectional sampling from a common specification. In this piece we have argued in favor of an architecture realizing this definition and displaying robustness and contextuality.

Acknowledgments We thank the anonymous reviewers for their detailed comments and for pointing us to some relevant literature that we had overlooked.

References

Kenneth Beesley. 1996. Arabic Finite-State Morphological Analysis and Generation. In *Coling*, pages 89–94.

Jonathan Berant and Percy Liang. 2014. Semantic parsing via paraphrasing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1415–1425, Baltimore, Maryland, June. Association for Computational Linguistics.

S.E. Brennan. 1996. Lexical entrainment in spontaneous dialog. In *Proceedings of International Symposium on Spoken Dialogue (ISSD-96)*.

Francisco Casacuberta and Colin de la Higuera. 2000. Computational complexity of problems on probabilistic grammars and transducers. In *ICGI*, pages 15–24.

H. Comon, M. Dauchet, R. Gilleron, C. Löding, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. 2007. Tree automata techniques and applications. Available on: `http://www.grappa.univ-lille3.fr/tata`. release October, 12th 2007.

Marc Dymetman. 1991. Inherently reversible grammars, logic programming and computability. In *In Proceedings of the ACL Workshop: Reversible Grammar in Natural Language Processing*.

Zoltán Fülöp and Heiko Vogler. 2009. Weighted tree automata and tree transducers. In Manfred Droste, Werner Kuich, and Heiko Vogler, editors, *Handbook of Weighted Automata*, Monographs in Theoretical Computer Science. An EATCS Series, pages 313–403. Springer Berlin Heidelberg.

Jonathan Graehl, Kevin Knight, and Jonathan May. 2008. Training tree transducers. *Comput. Linguist.*, 34(3):391–427, September.

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

Alexander Koller and Marco Kuhlmann. 2011. A generalized view on parsing and translation. In *Proceedings of the 12th International Conference on Parsing Technologies*, IWPT '11, pages 2–13, Stroudsburg, PA, USA. Association for Computational Linguistics.

W.J.M Levelt. 1983. Monitoring and self-repair in speech. *Cognition*, 14(1):41–104.

Andreas Maletti. 2010. Survey: Tree transducers in machine translation. Technical report, Universitat Rovira i Virgili.

Mehryar Mohri. 2003. Edit-Distance of Weighted Automata: General Definitions and Algorithms. *International Journal of Foundations of Computer Science*, 14:957–982.

Mehryar Mohri. 2009. Weighted automata algorithms. In Manfred Droste, Werner Kuich, and Heiko Vogler, editors, *Handbook of Weighted Automata*, pages 213–254. Springer.

Günter Neumann. 1998. Interleaving natural language parsing and generation through uniform processing. *Artificial Intelligence*, 99(1):121–163.

M. Otsuka and M. Purver. 2003. Incremental generation by incremental parsing. In *Proceedings 6th CLUK Colloquium*.

Majid Razmara. 2011. Applications of tree transducers in statistical machine translation. Technical report, Simon Fraser University.

Christian P. Robert and George Casella. 2004. *Monte Carlo Statistical Methods (Springer Texts in Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Stuart M. Shieber. 2004. Synchronous grammars as tree transducers. In *Proceedings of the Seventh International Workshop on Tree Adjoining Grammar and Related Formalisms (TAG+ 7)*, Vancouver, Canada, 20–22 May.

Tomek Strzalkowski, editor. 1994. *Reversible Grammar in Natural Language Processing*. Springer.

Gertjan van Noord. 1990. Reversible unification based machine translation. In *Proceedings of the 13th Conference on Computational Linguistics - Volume 2*, COLING '90, pages 299–304, Stroudsburg, PA, USA. Association for Computational Linguistics.

Y. Wang, J. Berant, and P. Liang. 2015. Building a semantic parser overnight. In *Association for Computational Linguistics (ACL)*.

Yuk Wah Wong and Raymond J Mooney. 2007. Generation by inverting a semantic parser that uses statistical machine translation. In *HLT-NAACL*, pages 172–179.

Yuk Wah Wong. 2007. *Learning for Semantic Parsing and Natural Language Generation Using Statistical Machine Translation Techniques*. Ph.D. thesis, Department of Computer Sciences, University of Texas at Austin, Austin, TX, August. Also appears as Technical Report AI07-343, Artificial Intelligence Lab, University of Texas at Austin, August 2007.

Kenji Yamada and Kevin Knight. 2001. A syntax-based statistical translation model. In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, ACL '01, pages 523–530, Stroudsburg, PA, USA. Association for Computational Linguistics.

A quantitative analysis of gender differences in movies using psycholinguistic normatives

Anil Ramakrishna[1], Nikolaos Malandrakis[1], Elizabeth Staruk[1], Shrikanth Narayanan[1, 2]
[1]Department of Computer Science,
[2]Department of Electrical Engineering,
University of Southern California, Los Angeles, USA
{akramakr,malandra,staruk}@usc.edu, shri@sipi.usc.edu

Abstract

Direct content analysis reveals important details about movies including those of gender representations and potential biases. We investigate the differences between male and female character depictions in movies, based on patterns of language used. Specifically, we use an automatically generated lexicon of linguistic norms characterizing gender ladenness. We use multivariate analysis to investigate gender depictions and correlate them with elements of movie production. The proposed metric differentiates between male and female utterances and exhibits some interesting interactions with movie genres and the screenplay writer gender.

1 Introduction

Gender has been an important research topic in the social sciences, with studies conducted on the effect of gender on various aspects of human perception and expression (Benshoff and Griffin, 2011) as well as investigations of the societal (Behm-Morawitz and Mastro, 2008) and career implications of gender and possible underlying biases. Previous studies report significant implications of gender on career progress in medicine (Sidhu et al., 2009), information technology (Cohoon and Aspray, 2006), politics (Niven, 2006) and show-business (Smith, 2010).

In this paper we investigate the depictions of the genders in feature films, through the analysis of their respective dialogues. The differences in depiction are a contentious subject, since aspects of these can be viewed as the result of stereotyping or gender bias, with the relative presence of women being a well investigated subject (Bielby and Bielby, 1996; Lincoln and Allen, 2004). We are interested in the existing gender depictions, regardless of relative frequencies, as well as any factors that may affect them. While popular tools such as the Bechdel test provide a test for detecting female presence in the movies, we hope to identify more subtle forms of gender differences across character gender from the dialogues. Our aim is to devise a non-binary metric that can be used to compare or rank movies, characters and perhaps individual utterances.

To analyze the dialogues we propose using a metric of language gender ladenness, a number representing a normative rating of the "perceived feminine or masculine association" (Paivio et al., 1968) of language. The metric, as originally defined, is meant to provide an indication of gender-specificity of individual words, with extreme values assigned to highly stereotypical concepts. Generating this rating for male and female character dialogues and comparing the character gender with this rating of "language gender" should allow us to observe stereotypical behavior.

Word based ratings such as the gender ladenness are referred to as linguistic norms (or psycholinguistic norms when corresponding to psychological constructs) and are popular in cognitive psychology (Clark and Paivio, 2004) and some computational disciplines, such as sentiment analysis (Nielsen, 2011) and opinion mining. To utilize gender ladenness, we follow an approach similar to simple sentiment analysis, with word-level norms automatically generated based on a small starting set of manually annotated norms and sentence (and higher) level norms estimated through word-level norm statistics. The resulting algorithm allows us to estimate gender ladenness at any arbitrary granularity.

We use these ratings of dialogue language to quantify the depictions of male and female characters and attempt to relate the observed gender ladenness with objective factors.

In section 2 and 3 we describe the data corpus

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1996–2001,
Lisbon, Portugal, 17-21 September 2015.

Word	*Norm*
Infantry	-0.97
Truck	-0.73
Dictator	-0.56
Strider	-0.36
United	-0.18
Volunteerism	0.04
Hygiene	0.22
Candle	0.45
Radiant	0.66
Bride	0.84
Gorgeous	0.96

Table 1: Sample word norms($\in [-1, 1]$); -1: Most masculine and $+1$: Most feminine.

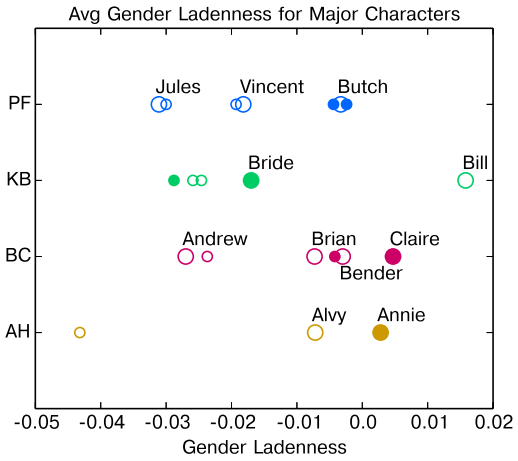


Figure 1: Average Gender Ladenness for a few sample movies, marker size proportional to number of utterances. Filled markers: Female characters, Hollow markers: Male characters; PF: Pulp Fiction, KB: Kill Bill, BC: Breakfast Club, AH: Annie Hall.

and the feature extraction process respectively. We detail the experimental procedure in section 4 and analysis in section 5 and conclude with future extensions in section 6.

2 Estimating Gender Ladenness

Gender Ladenness, as defined in (Clark and Paivio, 2004) represents the degree of perceived "feminine or masculine association" on a numerical scale ranging from very masculine to very feminine. It is important to note that there was no restriction to what "association" may mean: while it is reasonable to assume that associations of the form "A is B" or "A has B" would dominate annotator perception, that does not preclude other forms of association. Because of that, referring to the norms as indicators of how masculine or feminine the words are is not entirely accurate, though it is a reasonable approximation. The original ratings were re-scaled to $[-1, 1]$ for our purposes, with lower values indicating a masculine association and high values indicating a feminine association. Some sample words, utterances and their corresponding ratings are presented in Table 1 and Table 3. Figure 1 shows the average gender ladenness across all utterances for the major characters of a few movies. The annotations as a whole are reflective of stereotypical views of gender roles, e.g., words related to war and violence have a strong masculine association, whereas words related to family or positive emotions carry strong feminine associations.

The manual annotations from (Clark and Paivio, 2004) contain ratings for only 925 words, which are not enough to provide sufficient coverage. Therefore we use a lexicon expansion method, inspired by the work of (Malandrakis et al., 2013) to estimate the gender ladenness $\hat{g}(w_i)$ of word w_i using the semantic similarities $s()$ between w_i and reference words or concepts c_j, as

$$\hat{g}(w_i) = \theta_0 + \sum_{j=1}^{N} \theta_j s(w_i, c_j), \qquad (1)$$

where the terms θ_i are trained model parameters.

Given a manually annotated lexicon and a set of reference words, this equation can be used to create a linear system. Solving the system via Least Squares Estimation (LSE) gives us the parameters θ and an equation that can be used to generate gender ladenness for any new set of words.

Gender ladenness for larger lexical units is generated via simple statistics, as the average of word gender ladenness over all content words (adjectives, nouns, verbs and adverbs).

3 Data

Our primary data source is the Movie DiC corpus (Banchs, 2012) which includes 619 movie scripts parsed from The Internet Movie Script Database (IMSDb, 2015). The xml formatted scripts contain transcripts with speaker information as well as some structural information. Additional metadata for each movie were collected from the Internet Movie Database (IMDb, 2015).

Since our goal was to analyze gender depictions, we had to annotate each script utterance with a gender label. The process was complicated by inconsistencies between the information contained in the IMDb and Movie DiC corpora, like mismatched names, particularly for minor characters. Initially the script character names were cleaned using simple heuristics, such as the removal of all instances of the possessive "'s". The IMDb api (IMDbPY, 2015) was used to recover candidate movies matching the script movie name and, in the case of multiple candidates, the best candidate was selected based on the number of character names matching the script. Character names were compared using the Jaro-Winkler distance (Winkler, 1990). Having achieved a one to one mapping between IMDb and Movie DiC, we assigned a gender label to each matched character, using the gender predictor (NamSor Applied Onomastics, 2015). To make these predictions, we first use the name of the corresponding actor portraying that role; if there was no character match, we use the name of the character. Finally, we calculate a *confidence* score of our gender assignment per utterance for each movie, equal to the percentage of utterances with perfectly matched character name and a high confidence by the gender predictor. For the movies for which the confidence scores are not satisfactory, we manually match the script characters with IMDb's characters and annotate genders. In our experiments, we did this manual annotation with roughly 75 movies.

Having a mapping of scripts to IMDb entries, we collected more information about the movie such as the list of genres it belongs to and the members of the production team (producers, scriptwriters, directors), and followed a similar process as described above to assign genders to all persons. While movies may be created by multiple scriptwriters and directors, we retain only the first name, the primary credit, in each category. We removed infrequent genres and movies which belonged only to the removed genres. We also filtered out utterances with missing or incorrect character information and the utterances corresponding to characters for which the gender predictor fails to make confident predictions.

Movies with missing fields were also removed, leaving us with a total of 568 movies after the aforementioned pre-processing steps. Table 2 lists some descriptive statistics of the processed movie

Property Name	*Female*	*Male*	*Total*
Movie Utterances	107372	256274	363646
Producers	746	2974	3720
Directors	33	572	605
Assistant Directors	846	2739	3585
Screenplay Writers	130	1217	1347
Casting Directors	548	195	743

Table 2: Movie Dataset statistics

Utterance	*Avg. GL*
Flowers for the Diva.	0.77
Yeah, what a woman.	0.47
Got the house to yourselves?	-0.01
What about the crew?	-0.51
Yeah? You and what army?	-0.85

Table 3: Average gender ladenness for sample utterances from the dataset

corpus. At least in terms of raw frequencies, the gender ratio is clearly skewed towards male, particularly in the case of directors and with the exception of casting directors.

The norm generating equation (1) requires a semantic similarity estimate $s()$, which for the purposes of this paper is the cosine of context vectors generated over a large corpus of raw web text. The corpus was created by posing a query to the Yahoo! search engine for every word in the English version of the aspell spell-checker and collecting the top 500 result previews. Each preview is composed of a title and a sample of the content, each being a single sentence. Overall the collected corpus contains approximately 117 million sentences.

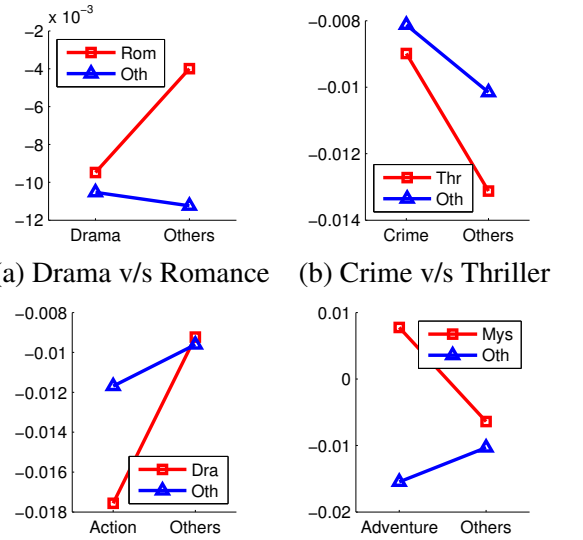
4 Experimental Procedure

The descriptive feature in this method is gender ladenness, so we extracted an estimate for each utterance of every movie. Initially, all utterances were part-of-speech tagged and non-content words were removed. Then, word-level gender landenness norms were generated for every remaining word.

To generate word-level norms, we used equation (1) with the intermediate seed words w_i being the top 10000 most frequent words in our corpus of web text with length longer that 3 characters. For each word in our corpus, we generated a binary weighted context vector (of window size 1) of size $\sim$ 125000. Then, for each word

of interest we calculated a 10000 place similarity vector, containing the cosine similarity scores between the context vector of said word and the context vectors of the 10000 intermediate seeds. Using the training set we generated a $K \times 10000$ matrix of similarities to the seed words and applied dimensionality reduction via Principal Component Analysis (PCA), keeping the first $N = 300$ components. These transformed similarities became the similarity terms $s()$ of equation (1) and were used to train the model. For any word in the scripts, a 10000 place similarity vector is generated and transformed using the pre-calculated PCA matrix, then equation (1) is used to create the gender ladenness estimate.

Ratings were generated at the utterance level, and collective ratings (per character, gender or movie) were calculated as utterance rating averages.



(a) Drama v/s Romance (b) Crime v/s Thriller

(c) Action v/s Drama (d) Adventure v/s Mystery

Figure 2: Interactions between genres

5 Results

To evaluate the word norm generation algorithm, we performed a 10-fold cross-validation experiment on the 925 manually annotated norms in (Paivio et al., 1968). The generated norms were evaluated against the ground truth and the method achieved a 0.801 Pearson correlation to the ground truth. While there is no comparable result in literature, the resulting performance appears sufficiently high.

We first investigated the overall gender ladenness of movies, represented as the average of all utterance level scores, with respect to the genre(s) the movie belongs to. The independent variables for this analysis were nine binary indicator variables, one for each of the most frequent genre labels in our movie corpus, with values of zero if the movie does not belong to the specific genre and one if it does. The particular representation of genres as separate variables was chosen because each movie can belong to multiple genres. Interaction terms were included. Running n-way ANOVA with the aggregate gender ladenness across both character genders as the dependent variable revealed significant differences between genres, with *Action* movies leaning towards the masculine ($p = 0.013$) compared to *Non-Action* movies, a not surprising result.

A few significant interactions between genres are shown in figure 2. Fig. 2a indicates that among non-drama movies, romantic movies tend to include more feminine language compared to non-romantic movies. However, if a movie belongs to the genre drama, its mean gender ladenness scores remain fairly constant, irrespective of its other genres. Similar interpretations can be drawn from the other plots in figure 2.

To analyze the effect of character gender on the gender ladenness scores, we next ran ANOVA with the character gender and the movie writer's gender as additional independent variables. The dependent variable in this case was the aggregate gender ladenness score across all utterances for male and female characters, so two scores per movie. The interaction of character gender and movie genre is shown in figure 3. The scores of male and female characters are correlated, which can be attributed to the underlying concepts in the utterances from these movies. The difference between genders is significant ($p = 0.034$), with male characters consistently using significantly more masculine language than their female counterparts, a finding that lends some credence to the metric used. Looking at the binary genre variables revealed that

Action movies contained significantly more masculine language than *Non-Action* movies ($p < 10^{-5}$) and the same holds for *Crime* movies ($p < 10^{-5}$). Conversely, *Romantic* movies leaned towards the more feminine language than non-*Romantic* movies ($p < 10^{-5}$) and similarly for *Comedy* movies compared to non-*Comedy* movies ($p = 0.02$). The male - female character gender

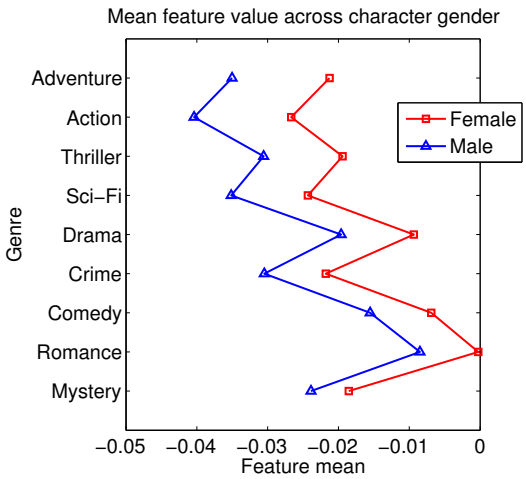


Figure 3: Interaction of movie genre with character gender

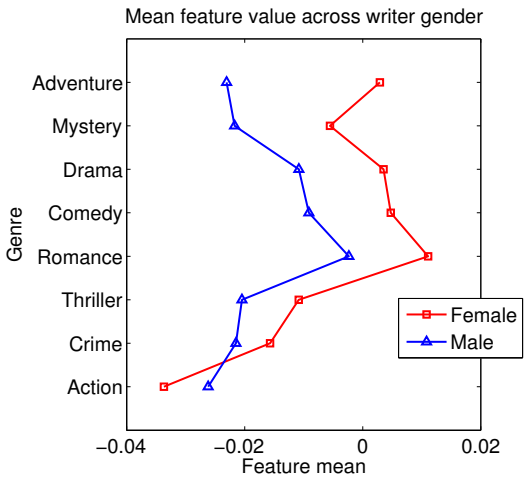


Figure 4: Interaction of screenplay writer's gender with genre

ladenness distance however is not affected in any significant way by the genre.

We include only the screenplay writer's gender in our analysis, though both the directors and screenplay writers influence the dialog lines (utterances), since the writers are more likely to directly influence the actual language used. In addition, the very small number of female directors in the data, as seen in table 2, leads to a violation of ANOVA's homoscedasticity assumption. Though the writer gender itself was not a significant factor, the interaction of writer's gender with the *Action* genre was significant ($p = 0.005$). The plot illustrating this interaction is shown in figure 4. It appears that female script writers write more masculine utterances compared to their male colleagues, at least for *Action* movies. We also investigated interactions between the writer and character gender, but none proved significant.

6 Conclusions and Future Work

We used regression to extrapolate manually annotated psycholinguistic normatives to movie utterances and investigated the use of these metrics to describe gender depictions. The metric proved successful, showing significant differences between the genders and predictable patterns with respect to movie genres.

Future work will include the use of further metrics, with those describing emotions being the first candidates. We also intend to collect more movie and character level metadata to be used in analysis. Finally, it is worth remembering that language provides only a partial description of depicted characters, so we should aim to augment with aural/visual information.

7 Acknowledgements

The authors gratefully acknowledge support from NSF, Geena Davis Institute on Gender in Media and Google.org.

References

Rafael E Banchs. 2012. Movie-Dic: a movie dialogue corpus for research and development. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pages 203–207. Association for Computational Linguistics.

Elizabeth Behm-Morawitz and Dana E Mastro. 2008. Mean girls? the influence of gender portrayals in teen movies on emerging adults' gender-based attitudes and beliefs. *Journalism & Mass Communication Quarterly*, 85(1):131–146.

Harry M Benshoff and Sean Griffin. 2011. *America on film: Representing race, class, gender, and sexuality at the movies*. John Wiley & Sons.

Denise D Bielby and William T Bielby. 1996. Women and men in film gender inequality among writers in a culture industry. *Gender & Society*, 10(3):248–270.

James M Clark and Allan Paivio. 2004. Extensions of the Paivio, Yuille, and Madigan (1968) norms. *Behavior Research Methods, Instruments, & Computers*, 36(3):371–383.

J McGrath Cohoon and William Aspray. 2006. *Women and information technology: Research on underrepresentation*, volume 1. The MIT Press.

IMDb. 2015. Internet movie database. [Online; accessed 10-June-2015].

IMDbPY. 2015. [Online; accessed 10-June-2015].

IMSDb. 2015. Internet movie script database. [Online; accessed 10-June-2015].

Anne E Lincoln and Michael Patrick Allen. 2004. Double jeopardy in hollywood: Age and gender in the careers of film actors, 1926–1999. *Sociological Forum*, 19(4):611–631.

Nikolaos Malandrakis, Alexandros Potamianos, Elias Iosif, and Shrikanth Narayanan. 2013. Distributional semantic models for affective text analysis. *Audio, Speech, and Language Processing, IEEE Transactions on*, 21(11):2379–2392, Nov.

NamSor Applied Onomastics. 2015. NamSor gender API. [Online; accessed 10-June-2015].

Finn Årup Nielsen. 2011. A new ANEW: Evaluation of a word list for sentiment analysis in microblogs. *arXiv preprint arXiv:1103.2903*.

David Niven. 2006. Throwing your hat out of the ring: Negative recruitment and the gender imbalance in state legislative candidacy. *Politics & Gender*, 2(04):473–489.

Allan Paivio, John C Yuille, and Stephen A Madigan. 1968. Concreteness, imagery, and meaningfulness values for 925 nouns. *Journal of experimental psychology*, 76(1p2):1.

Reena Sidhu, Praveen Rajashekhar, Victoria L Lavin, Joanne Parry, James Attwood, Anita Holdcroft, and David S Sanders. 2009. The gender imbalance in academic medicine: a study of female authorship in the united kingdom. *Journal of the Royal Society of Medicine*, 102(8):337–342.

Stacy L Smith. 2010. Gender oppression in cinematic content? a look at females on-screen & behind-the-camera in top-grossing 2007 films. *Retrieved September*, 2:2010.

William E Winkler. 1990. String comparator metrics and enhanced decision rules in the fellegi-sunter model of record linkage.

EMNLP versus ACL: Analyzing NLP Research Over Time

Sujatha Das Gollapalli, Xiao-Li Li
Institute for Infocomm Research, A*STAR, Singapore
{gollapallis,xlli}@i2r.a-star.edu.sg

Abstract

The conferences ACL (Association for Computational Linguistics) and EMNLP (Empirical Methods in Natural Language Processing) rank among the premier venues that track the research developments in Natural Language Processing and Computational Linguistics. In this paper, we present a study on the research papers of approximately two decades from these two NLP conferences. We apply keyphrase extraction and corpus analysis tools to the proceedings from these venues and propose probabilistic and vector-based representations to represent the topics published in a venue for a given year. Next, similarity metrics are studied over pairs of venue representations to capture the progress of the two venues with respect to each other and over time.

1 Introduction

Scientific findings in a subject-area are typically published in conferences, journals, patents, and books in that domain. These research documents constitute valuable resources from the perspective of data mining applications. For instance, the citation links among research documents are used in computing bibliometric quantities for authors (Alonso et al., 2009) whereas topic models on research corpora are used to distinguish between influential and impactful researchers (Kataria et al., 2011) and to capture temporal topic trends (He et al., 2009).

Despite several potential benefits mentioned above and the free availability of most research proceedings in NLP through the ACL Anthology[1], the topical and temporal aspects of this corpus are yet to be fully studied in current literature. In this paper, we present our study on research proceedings of approximately two decades from two leading NLP conferences, namely ACL and EMNLP, to complement a previous study on this topic by Hall et al (2008). To the best of our knowledge, we are the first to characterize the developments in the NLP domain using a comparative study of two of its leading publication venues. Our contributions are summarized below:

1. We represent the NLP research corpus from approximately two decades as a keyphrase-document matrix and apply Latent Dirichlet Allocation (Blei et al., 2003) to extract coherent topics from it (Newman et al., 2010).

2. We propose two novel representations for summarizing the venue proceedings in a given year. (1) The **probabilistic** representation expresses each venue as a probability distribution over topics, whereas (2) the **TP-ICP** representation captures topics that are the major focus in the venue for a particular year via *Topic Proportion* (TP) as well as topic importance as measured with *inverse corpus proportion* (ICP).

3. We apply Jensen-Shannon divergence and cosine similarity on our proposed venue representations to analyze the venues over time. Specifically, we ask the following questions: What are the popular topics in ACL and EMNLP in a particular year? Is the topical focus in EMNLP different from ACL? How

[1] https://aclweb.org/anthology/

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 2002–2006,
Lisbon, Portugal, 17-21 September 2015.

did the topical focus in each venue change over time?

Organization: We describe our novel venue representations and the measures used to compare them in Section 2. The details of our datasets and experiments are presented in Section 3 along with results and observations. We summarize related research in Section 4 before concluding the paper in Section 5.

2 Methods

Let $Y = \{y_1, y_2 \dots y_T\}$ be the consecutive years in which the research proceedings are available from V, set of publication venues under consideration ($V = \{$"ACL", "EMNLP"$\}$ in this paper). If D is the set of all documents over the years, each document $d \in D$ is associated with $\{K_d, y, v\}$ where K_d refers to the content of d whereas v and y refer to the venue and year in which d was published.

2.1 Venues as Probability Distributions

Let $t_1, t_2 \dots t_k$ denote the topics capturing the content of documents in D. Using probabilistic topic modeling and dimension reduction tools such as Latent Dirichlet Allocation or pLSA (Hofmann, 1999; Blei et al., 2003), we extract for each $d \in D$, $P(t_i|d), i = 1 \dots k$, the multinomial distribution over the topics associated with d.

The venue-topic probability distribution $P(t_i|v_y)$ for a given (venue, year) pair $(v = l, y = m)$ can be computed using $D_{l,m}$, the set of documents published in venue l, in the year m. That is,

$$P_{l,m}(t_i) = \frac{1}{|D_{l,m}|} \sum_{d \in D_{l,m}} P(t_i|d) \qquad (1)$$

Note that the above probabilistic representation facilitates a quantitative measure to compare two venues: the divergence between the probability distributions of the two venues. The Kullback−Leibler divergence (KLD) between two (discrete) probability distributions P and Q is given by: $D_{KL}(P||Q) = \sum_i P(i) log \frac{P(i)}{Q(i)}$. Due to the unsymmetric nature of KLD, we use the Jensen-Shannon divergence, a symmetric and finite measure ($0 \leq JSD(P||Q) \leq 1$) based on KLD. Let $M = \frac{1}{2}(P + Q)$,

$$JSD(P||Q) = \frac{1}{2}[D_{KL}(P||M) + D_{KL}(Q||M)]$$

2.2 Venues as TP-ICP Vectors

Discrete probability distributions are often represented in computations as normalized vectors. For instance, the $P(t_i|v)$ values comprise the components of a k-dimensional vector. This topic proportion (TP) vector is similar to the normalized term frequency vector commonly used in Information Retrieval (IR) (Manning et al., 2008). TP values are fractions indicating the percentage of a given topic among all topics covered in a particular year. Thus these values are higher for topics that are the major focus in the venue for a particular year .

We also extend *inverse document frequency*, a popular concept that is used to weigh terms in IR (Manning et al., 2008) to describe **Inverse Corpus Proportion** or ICP. Our objective in defining ICP is to capture the importance of a topic by diminishing the effect of topics that are common across all years. Let $\mathbf{TP}_{v,y}(i)$ represents the proportion of topic t_i in venue v for year y, then $ICP(t_i) =$

$$\log\left(\frac{\sum_{y=1}^{|Y|}\sum_{v=1}^{|V|}\sum_{j=1}^{k}\mathbf{TP}_{v,y}(j)}{\sum_{y=1}^{|Y|}\sum_{v=1}^{|V|}\mathbf{TP}_{v,y}(i)}\right) = \left(\frac{|D|}{\sum_{y=1}^{|Y|}\sum_{v=1}^{|V|}\mathbf{TP}_{v,y}(i)}\right)$$

since $\sum_{j=1}^{k}\mathbf{TP}(j) = 1$, $\mathbf{TP}$ being a probability distribution vector and $|Y| \times |V| = |D|$. The TP-ICP vector for a venue is defined as: $[TP(1) \times ICP(1), \dots TP(k) \times ICP(k)]$ and captures in each component the weighted proportion of a topic in that venue for a year. This novel representation can be considered the topic-level counterpart of the popular TF-IDF representation in IR. Given two TP-ICP vectors $P = [p_1, p_2, \dots p_k]$ and $Q = [q_1, q_2, \dots q_k]$, the similarity between them using the cosine measure is given by:

$$cosine(P, Q) = \frac{\sum_{i=1}^{k} p_i.q_i}{||P||_2.||Q||_2}$$

2.3 Keyphrases for representing documents

Corpus analysis tools often use bag-of-words models and term frequencies for representing documents (Heinrich, 2005). However, research documents are often well-structured, and contain various sections with author information, citations,

Topic ID	Top Words
0	System, Dialogue, Dialogue System, Information, Speech Recognition, Speech, Dialogue Manager, Data Collection, User Utterances
1	Model, Training Data, Language Model, Model Parameters, Models, Generative Model, Probabilistic Model
2	Noun Phrases, Other Hand, Head Noun, Future Work, Corpus, Sentence, Language, Method, Japanese Language, Syntactic Structure
3	Evaluation, Evaluation Metrics, Automatic Evaluation, Machine Translation, Human Judgments, Translation Quality
4	Error, Error Correction, Error Types, Spelling Errors, Category, Errors, Lexical Category, Error Rate, Ccg Parser
5	Sentence, First Sentence, Summarization, Sentence Length, Summarization Task, Document Summarization Text Summarization
6	Algorithm, Search Space, Objective Function, Function, Search Algorithm, Model, Optimization Problem, Large Number
7	Language, Information, Sentence, System, Results, Corpus, Approach, Research, Learning, Language Processing, Systems, Machine
8	Rules, Parse Tree, Grammar, Tree Structure, Root Node, Parse Trees, Grammar Rules, Rule Extraction, Rule Set, Elementary Tree
9	Natural Language, System, Language Generation, Information, Generation System, Language, Sentence, System Architecture
10	Sentiment Analysis, Sentiment, Event, Discourse, Sentiment Classification, Discourse Structure, Discourse Relations
11	Dependency Parsing, Dependency, Parsing, Parser, Dependency Tree, Parse Tree, Dependency Parser, Parsing Model, Dependency Trees
12	Words, Other Hand, Natural Language, Other Words, Corpus, Language Processing, Model, Information, TestSet, Language
13	Pos Tagging, Pos Tags, Word Segmentation, Pos Tag, Words, Model, Word Boundaries, Unknown Words, Chinese Word, Pos Tagger
14	Training Data, Training Set, Test Data, Training, Test Set, Data Sets, Data Set, Labeled Data, Training Examples, Unlabeled Data
15	Language, Target Language, Source Language, Machine Translation, Translation, Different Languages, Language Pairs
16	Features, Feature Set, Training Data, Feature Vector, Training Set, Lexical Features, Model, Feature Space, Test Data
17	Clustering Algorithm, Annotation, Same Cluster, Clustering, Clustering Method, Clustering Methods, Annotation Scheme
18	Query, Information Retrieval, Search Engine, Web Search, Search, Search Results, Information, Query Terms, Search Engines
19	Relation, Relation Extraction, Relations, Information Extraction, Semantic Relations, Relation Types, Semantic Relation
20	Topic, Topic Model, Topic Models, Topic Distribution, Same Topic, Topics, Model, Distribution, Topic Modeling, Word Distribution
21	Coreference Resolution, Entity, Same Entity, Coreference, Resolution System, Pronoun Resolution, Anaphora Resolution, Entity Type
22	Machine Translation, Translation, Language Model, Word Alignment, Translation Model, Target Language, Model, Translation Quality
23	Word Sense, Sense Disambiguation, Sense, Word Senses, Words, Different Senses, Target Word, Semantic Relations, Lexical Resources
24	Question, Question Answering, Answer, Questions, Correct Answer, Question Types, System, Textual Entailment, Answer Type
25	Semantic Role, Semantic Roles, Semantic Information, Syntactic Structure, Syntactic Information, Semantic, Parse Tree
26	Machine Learning, Learning, Classification Task, Features, Supervised Learning, Text Classification, Learning Algorithms, Feature
27	Semantic Similarity, Vector Space, Similarity Measure, Word Vectors, Vector, Similarity Measures, Similarity, Semantic Space
28	Language Model, Speech Recognition, Language Models, Word Error, Language, Model, Automatic Speech, Speech
29	Grammar, Language, Natural Language, Lexical Entries, Feature Structure, Feature Structures, Finite Set, Other Hand, Lexical Items
Topics ranked by *Inverse Corpus Proportions*: 3 7 4 17 24 20 21 27 9 28 0 19 6 10 23 2 5 18 15 25 26 29 8 13 14 11 16 1 22 12 Maximum ICP: 4.3533, Minimum ICP: 2.1809, Average ICP: 3.5591	

Table 1: The top words for each topic are shown here after modeling the ACL+EMNLP publications over the years with #topics=30. The topics ranked by their ICP values are shown in the last row to illustrate that ICP values indeed capture the specificity of a topic across the years.

and content-related sections such as *abstract*, *related work*, and *experiments*. To best represent the topical content of these documents, we harness the latest work on keyphrase extraction for research documents and represent documents using keyphrases (Hasan and Ng, 2014).

We use the ExpandRank algorithm (Wan and Xiao, 2008) to extract top n-grams $\forall d \in D$. ExpandRank effectively combines PageRank values on word graphs with text similarity scores between documents to score n-grams for a document and was shown to outperform other unsupervised keyphrase extraction methods on research documents in absence of other information such as citations (Gollapalli and Caragea, 2014).

3 Experiments

Datasets and setup: We crawled the ACLWeb for research papers from EMNLP and ACL from the year 1996 through 2014[2] using the Java-based crawler, Heritrix[3]. The text from the PDF documents was extracted using the PDFBox software[4] after which simple rules similar to the ones used in CiteSeer (Li et al., 2006) were employed to extract the "body" of the research document[5]. The numbers of papers for each year at the end of this process are listed in Table 2. From these numbers, it appears that the paper "intake" in each conference has gone up overall during the last decade although occasionally the increase is due to co-location with related conferences such as IJCNLP and HLT[6].

We construct the keyphrase-document matrix using top-100 keyphrases of each document extracted with ExpandRank. The LDA implementation provided in Mallet (McCallum, 2002) was used to extract topics from this matrix. The LDA algorithm was run along with hyperparameter optimization (Minka, 2003) for different numbers of topics between $10 \ldots 100$ in increments of 10. We use the average corpus likelihood over ten randomly-initialized runs to choose the optimal number of topics that best "explain" the corpus (Heinrich, 2005). As indicated by the left plot in Figure 1 this optimum is obtained when the number of topics is 30.

3.1 Results and Observations

The top phrases that reflect the "theme" captured by a topic are shown in Table 1. As indicated in this table, we are able to extract coherent topics from the corpus using LDA on a document-keyphrase matrix (AlSumait et al., 2009; Newman et al., 2010).

Top research topics in NLP: We select five timepoints $\{1996, 2000, 2005, 2010, 2014\}$ by splitting the 1996-2014 period into roughly-

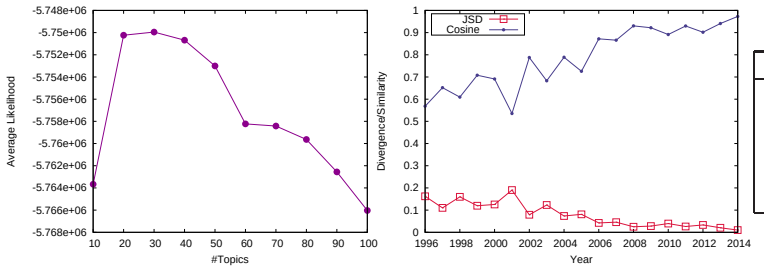
[2] Since our goal is to compare the two venues, we start from 1996 when EMNLP branched off into a full conference from a workshop on Very Large Corpora although ACL proceedings are available from 1979.

[3] https://webarchive.jira.com/wiki/display/Heritrix/Heritrix

[4] https://pdfbox.apache.org/

[5] Processed data available upon request.

[6] ACL was co-located with related conferences in the years 1997, 1998, 2006, 2008, and 2009 and EMNLP in the years 2005, 2007, and 2012.



Year	ACL	EMNLP
2014	27, 10, 20	27, 10, 19
2010	0, 5, 6	20, 19, 18
2005	9, 0, 19	24, 18, 0
2000	7, 9, 5	18, 23, 20
1996	9, 2, 7	28, 2, 5

Figure 1: *Left*: #topics vs. Average Corpus Likelihood, *Middle*: EMNLP vs. ACL, *Right*: Top topics in EMNLP and ACL

Venue	2014	2013	2012	2011	2010	2009	2008	2007	2006	2005	2004	2003	2002	2001	2000	1999	1998	1997	1996
ACL	330	399	227	349	272	244	213	207	310	137	129	103	160	65	45	83	244	73	58
EMNLP	226	207	141	149	125	164	115	131	73	28	53	27	33	10	27	34	13	23	14

Table 2: Number of papers for each venue for different years

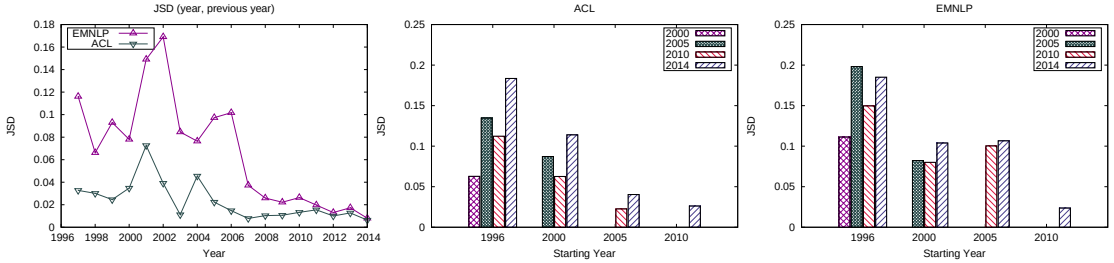


Figure 2: Comparing EMNLP and ACL over the years. Each point in the *Left* plot shows the JSD between a given year y and the year $y-1$. The *Middle* (ACL) and *Right* (EMNLP) plots show the JSD between a timepoint with preceding timepoints in the set $\{1996, 2000, 2005, 2010, 2014\}$.

equal parts and examine the top topics for ACL and EMNLP at these time points. We rank the topics in each conference by their TP-ICP values and list the top 3 topics in the right table of Figure 1. "Semantic relation extraction", "sentiment analysis", and "topic models" are the top research topics in NLP last year (2014) whereas in the year 1996, the topics "noun phrase extraction", "summarization", "corpus modeling", and "speech recognition" dominated the NLP research arena. From the table, it can be seen that "information retrieval" (topicID: 18) ranks among the top topics in EMNLP for all three timepoints during 2000-2010 whereas "natural language generation" (topicID: 9) was consistently addressed during 1996-2005 in ACL.

EMNLP versus ACL: We compare the venues using JSD and Cosine similarity measures in the middle plot of Figure 1. The plot shows decreasing divergence between the topical distributions over the years and increasing cosine similarity between the TP-ICP vectors for the venues. These trends indicate that over the two decades the two venues ACL and EMNLP seem to have "become like each other" although their topical focus was different during the initial years. Increasingly, both venues seem to publish papers on similar topics. This behavior could be interpreted to mean that the NLP research field is more stable now with two of its leading conferences addressing problems on similar topics.

Changing topical focus over the years: In the first plot of Figure 2, we show the Jensen-Shannon divergence between the topic distributions of a particular venue for a given year y and $(y-1)$, the year preceding it. The curve indicates that in the years between 1997-2008, the rate of change from year to year is higher than in the years following 2008. We split the time period 1996-2014 into five roughly-equal parts to form the set $\{1996, 2000, 2005, 2010, 2014\}$. The JSD between the distribution in a particular year and the years preceding it in the above set is shown for ACL (middle plot) and EMNLP (right plot) in Figure 2. For example, the first cluster in the middle plot, shows the JSD values between the distributions for the years 2000, 2005, 2010, 2014 with the starting year 1996 for ACL. For both venues, the divergences of a given year are higher with the early starting years 1996 and 2000 than with the later starting years 2005 and 2010, indicating that the topics being addressed currently in NLP research are significantly different from those addressed a decade back.

4 Related Work

Temporal analysis of corpora is an upcoming research topic in text mining groups. Topic models were particularly investigated for detecting activity patterns in corpora annotated with time information (Huynh et al., 2008; Shen et al., 2009). Evolution of topics and their trends were studied on research corpora from NIPS (Wang and McCallum, 2006) as well as CiteSeer (He et al., 2009).

In contrast with existing approaches that seek to model the detection of new topics and their evolution, we focus on representing different venues pertaining to a research field and examine their development over time by comparing them against each other. In a similar study, Hall et al. (2008) examined the emergence of topics in NLP literature. They proposed "topic entropy" to measure the diversity in three conferences from the ACL Anthology during the years 1978-2006. They also noted that all the venues seem to converge in the topics they cover over the years based on the JSD between their topic distributions.

5 Conclusions

We presented our study on research proceedings of approximately two decades from the leading NLP conference venues: EMNLP and ACL. We extracted coherent topics from this corpus by applying topic modeling on the corresponding keyphrase-document matrix. Next, the extracted topics were used to characterize each venue through probabilistic and vector representations and compared against each other and over the years using various similarity measures. To the best of our knowledge, we are the first to present insights related to the development of a research field by studying two leading conferences in the area using various techniques from NLP and IR.

References

S. Alonso, F.J. Cabrerizo, E. Herrera-Viedma, and F. Herrera. 2009. h-index: A review focused in its variants, computation and standardization for different scientific fields. *Journal of Informetrics*, 3(4):273 – 289.

Loulwah AlSumait, Daniel Barbar, James Gentle, and Carlotta Domeniconi. 2009. Topic significance ranking of lda generative models. In *Machine Learning and Knowledge Discovery in Databases*, Lecture Notes in Computer Science.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *J. Mach. Learn. Res.*, pages 993–1022.

Sujatha Das Gollapalli and Cornelia Caragea. 2014. Extracting keyphrases from research papers using citation networks. In *AAAI*.

David Hall, Daniel Jurafsky, and Christopher D. Manning. 2008. Studying the history of ideas using topic models. In *EMNLP*.

Kazi Saidul Hasan and Vincent Ng. 2014. Automatic keyphrase extraction: A survey of the state of the art. *ACL*.

Qi He, Bi Chen, Jian Pei, Baojun Qiu, Prasenjit Mitra, and C. Lee Giles. 2009. Detecting topic evolution in scientific literature: how can citations help? In *CIKM*, pages 957–966.

G. Heinrich. 2005. Parameter estimation for text analysis. Technical report.

Thomas Hofmann. 1999. Probabilistic latent semantic indexing. In *SIGIR*, pages 50–57.

Tâm Huynh, Mario Fritz, and Bernt Schiele. 2008. Discovery of activity patterns using topic models. In *International Conference on Ubiquitous Computing*.

Saurabh Kataria, Prasenjit Mitra, Cornelia Caragea, and C. Lee Giles. 2011. Context sensitive topic models for author influence in document networks. In *IJCAI*, pages 2274–2280.

Huajing Li, Isaac G. Councill, Levent Bolelli, Ding Zhou, Yang Song, Wang-Chien Lee, Anand Sivasubramaniam, and C. Lee Giles. 2006. Citeseerx: a scalable autonomous scientific digital library. In *InfoScale*.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schutze. 2008. *Introduction to Information Retrieval*. Cambridge University Press.

Andrew Kachites McCallum. 2002. Mallet: A machine learning for language toolkit. http://mallet.cs.umass.edu.

Thomas P. Minka. 2003. Estimating a dirichlet distribution. Technical report, Microsoft Research.

David Newman, Jey Han Lau, Karl Grieser, and Timothy Baldwin. 2010. Automatic evaluation of topic coherence. In *Human Language Technologies*.

Zhiyong Shen, Ping Luo, Yuhong Xiong, Jun Sun, and Yidong Shen. 2009. Topic modeling for sequences of temporal activities. In *ICDM*, pages 980–985.

Xiaojun Wan and Jianguo Xiao. 2008. Single document keyphrase extraction using neighborhood knowledge. In *AAAI*.

Xuerui Wang and Andrew McCallum. 2006. Topics over time: A non-markov continuous-time model of topical trends. In *SIGKDD*.

Answering Elementary Science Questions by Constructing Coherent Scenes using Background Knowledge

Yang Li*
UC Santa Barbara
Santa Barbara, CA, USA
yangli@cs.ucsb.edu

Peter Clark
Allen Institute for Artificial Intelligence
Seattle, WA, USA
peterc@allenai.org

Abstract

Much of what we understand from text is not explicitly stated. Rather, the reader uses his/her knowledge to fill in gaps and create a coherent, mental picture or "scene" depicting what text appears to convey. The scene constitutes an understanding of the text, and can be used to answer questions that go beyond the text.

Our goal is to answer elementary science questions, where this requirement is pervasive; A question will often give a partial description of a scene and ask the student about implicit information. We show that by using a simple "knowledge graph" representation of the question, we can leverage several large-scale linguistic resources to provide missing background knowledge, somewhat alleviating the knowledge bottleneck in previous approaches. The coherence of the best resulting scene, built from a question/answer-candidate pair, reflects the confidence that the answer candidate is correct, and thus can be used to answer multiple choice questions. Our experiments show that this approach outperforms competitive algorithms on several datasets tested. The significance of this work is thus to show that a simple "knowledge graph" representation allows a version of "interpretation as scene construction" to be made viable.

1 Introduction

Elementary grade science tests are challenging as they test a wide variety of commonsense knowledge that human beings largely take for granted, yet are very difficult for machines (Clark, 2015). For example, consider a question from a NY Regents 4th Grade science test:

Question 1 *"When a baby shakes a rattle, it makes a noise. Which form of energy was changed to sound energy?" [Answer: mechanical energy]*

Science questions are typically quite different from the entity-centric factoid questions extensively studied in the question answering (QA) community, e.g., "In which year was Bill Clinton born?" (Ferrucci et al., 2010; Yao and Van Durme, 2014). While factoid questions are usually answerable from text search or fact databases, science questions typically require deeper analysis. A full understanding of the above question involves not just parsing and semantic interpretation; it involves adding implicit information to create an overall picture of the "scene" that the text is intended to convey, including facts such as: noise is a kind of sound, the baby is holding the rattle, shaking involves movement, the rattle is making the noise, movement involves mechanical energy, etc. This mental ability to create a scene from partial information is at the heart of natural language understanding (NLU), which is essential for answering these kinds of question. It is also very difficult for a machine because it requires substantial world knowledge, and there are often many ways a scene can be elaborated.

We present a method for answering multiple-choice questions that implements a simple version of this. A scene is represented as a "knowledge graph" of nodes (words) and relations, and the scene is elaborated with (node,relation,node) tuples drawn from three large-scale linguistic knowledge resources: WordNet (Miller, 1995), DART (Clark and Harrison, 2009), and the Free-Association database (Nelson et al., 2004). These elaborations reflect the mental process of "filling in the gaps", and multiple choice questions can then be answered by finding which answer option creates the most coherent scene.

The notion of NLU as constructing a most coherent scene is not new, and has has been studied in several contexts including work on scripts (Schank and Abelson, 1977), interpretation as ab-

*Work was done while the author was an intern at Allen Institute for Artificial Intelligence.

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 2007–2012,
Lisbon, Portugal, 17-21 September 2015.

duction (Hobbs et al., 1988; Hobbs, 1979; Ovchinnikova et al., 2014), bridging anaphora (Asher and Lascarides, 1998; Fan et al., 2005), and paragraph understanding (Zadrozny and Jensen, 1991; Harabagiu and Moldovan, 1997). These methods are inspiring, but have previously been limited by the lack of background knowledge to supply implicit information, and with the complexity of their representations. To make progress, we have chosen to work with a simple "knowledge graph" representation of nodes (words) and edges (relations). Although we lose some subtlety of expression, we gain the ability to leverage several vast resources of world knowledge to supply implicit information. The significance of this work is thus to show that, by working with a simple "knowledge graph" representation, we can make a viable version of "interpretation as scene construction". Although the approach makes several simplifying assumptions, our experiments show that it outperforms competitive algorithms on several datasets of (real) elementary science questions.

2 Approach

The input to our question-answering system is a multiple choice question Q, a set of answer options a_k, and one or more background knowledge base(s) each containing a set of $(word_i, relation, word_j)$ tuples, each denoting that $word_1$ is plausibly related to $word_j$ by $relation$. The output is a ranked list of the K answer options.

We define a *scene* S as a "knowledge graph" of *nodes* (words) and *edges* (relations between words), where all $(word_i, relation, word_j)$ edges are sanctioned by (contained in) at least one of the background knowledge bases. Each scene node has an associated measure of $coherence$ (described shortly), denoting how well-connected it is. The question-answering objective is, for each answer option a_k, to find the most coherent scene containing (at least) the question keywords $kw_i \in Q$ and answer option a_k, and then return the answer option with the overall highest coherence score. Our implementation approximates this objective using a simple elaborate-and-prune algorithm, illustrated in Figure 1[1] and now described.

[1]The system constructs 4 alternative graphs, each contains only one answer option plus some additional related nodes. Figure 1 shows just one of these 4 graphs, namely the graph containing answer option "food".

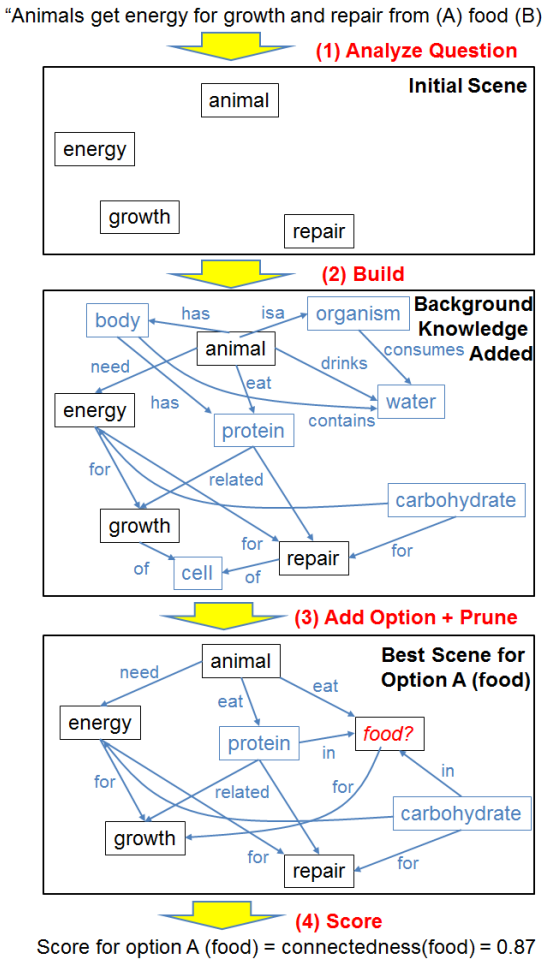


Figure 1: (1) Question keywords are extracted to form the initial scene. (2) The scene is elaborated with background knowledge to add plausible relationships. (3) For each answer option, it is added into the scene and connected with additional relationships. Then the scene is pruned. (4) A score is derived from the final scene, reflecting confidence that the answer option is correct.

2.1 Question Analyzer

The initial scene is simply the keywords (non-stop words) $KW = \{kw_i\}$ in the question Q, along with a measure of importance $IS(kw_i)$ for each word kw_i. For our purposes we compute importance by sending the question to Google, grouping the top 20 result snippets into a document d, and computing:

$$IS(kw) = \frac{tf_d(kw)}{df_Q(kw)}, \quad (1)$$

where $tf_d(kw)$ is the term frequency of kw in document d, and $df_Q(kw)$ is the document frequency of kw in question set Q containing all the available elementary science questions. The intuition here is

KB	Size (# tuples)	Examples
WordNet	235k	(dog,isa,animal) (sunlight,isa,energy)
DART	2.3M	(nutrients,in,food) (animal,eat,food)
FreeAssoc	64k	(car,relate_to,tire) (ice,relate_to,cold)

Table 1: Knowledge Bases Used

that the words frequently mentioned in variations of the question should be important (reflected by "tf"), while the descriptive words (e.g. "following", "example") which are widely used in many questions should be penalized (reflected by "idf"). Other methods could equally be used to compute importance.

2.2 Builder

In this step our goal is to inject implicit knowledge from the background KBs to form an elaborated knowledge graph. To do this, we first fetch all the triples $(kw, relation, w)$ that are directly connected with any keyword $kw \in KW$ from the background KBs, as well as all $(kw_i, relation, kw_j)$ triples between keywords. In our experiments we use three background knowledge bases to supply implicit knowledge, although in principle any triple store could be used: WordNet (Miller, 1995), DART (Clark and Harrison, 2009), and the FreeAssociation database (Nelson et al., 2004). Table 1 shows examples of each [2].

These triples introduce new nodes w into the graph. As we may get a large number of such nodes, we score them and retain only the top scoring ones (and all edges connecting to it). Informally, a new word w is preferred if it is connected to many important keywords kw with strong relationships. Formally, the scoring function is:

$$score(w) = \sum_{kw \in K} IS(kw) * rel(kw, w) \quad (2)$$

where $IS(kw)$ is the importance score of keyword kw and $rel(kw, w)$ is the relatedness score between kw and w. In this work we use the cosine similarity between word2vec (Mikolov et al., 2013) word vectors to measure two words' relatedness, as a rough proxy for the strength of relatedness in the KBs (the KBs themselves do not provide meaningful strengths of relationship). After the ranking, the top $N \times |KW|$ neighbor words w are retained[3], along with their edges to keywords kw and each other.

Note that at this point the elaboration process is independent of any answer option; rather, the graph depicts the question scenario.

2.3 Elaborate and Prune

To score the K different answer options, the system now builds K alternative elaborations of the scene so far, each one with answer option a_k added, and assesses the coherence of the addition. The answer option a_k that fits "most coherently" with the scene is returned as the answer to the question.

To do this for a given option a_k, we add a_k to the graph along with all triples $(w_i, relation, a_k)$ in the KBs that relate any node w_i in the graph to a_k. Now that the focus a_k of the question is known, some of the earlier added nodes w in the graph may be only weakly relevant to the question and answer, and so we add a pruning step to remove these nodes. The goal of this pruning is to find a dense subgraph (i.e. the coherent scene) that would ideally contain all the question keywords kw, the answer option a_k, and extra words w_k that are highly connected with them.

Inspired by Sozio et al's work (Sozio and Gionis, 2010) on finding strongly interconnected subgroups in social networks, we have developed an iterative node removal algorithm for extracting this subgraph. We define the coherence of a node as the summed weight of its incident edges:

$$coherence(w) = \sum_{w' \in \{(w,r,w')\}} rel(w, w') \quad (3)$$

where $rel(w, w')$ is the weight of edge (w, r, w') in the graph, again computed using cosine similarity between w and w'. We then iteratively remove the non-keyword node (and attached edges) with least coherence until the answer option a_k is about to removed. The resulting graph is thus maximally pruned, subject to the constraint it must still describe the question plus answer option.

Finally, we use the coherence of the answer option a_k in this final scene as the confidence that a_k is the correct answer. The system repeats this

[2] WordNet: all relationships types are used. DART: The NVN and NPN databases with frequency counts > 10 are used. FreeAssoc: The top 3 associations per word were used.

[3] The optimal N (here 6) was selected using an independent set of training questions

for all K answer options and selects the a_k with highest confidence.

3 Evaluation

The system was developed using a dataset of natural (unedited) elementary science exam questions, and then tested on three similar, unseen (hidden) datasets. Its performance was compared with two other state-of-the-art systems for this task. As our system only fields questions where the answer options are all single words, we evaluate it, and the other systems, only on these subsets. These subsets are in general easier than other questions, but this advantage is the same for all systems being compared so it is still a fair test.

3.1 Evaluation Datasets

The datasets used are the non-diagram, multiple-choice questions with single-word answer options drawn from the following exams:

- **Dev (System Development):** New York Regents 4th Grade Science [4] (47 questions in 6 years)
- **Test1:** New York Regents 4th Grade Science (23 questions in 3 years)
- **Test2:** Additional 4th Grade Science (from multiple States) (26 questions)
- **Test3:** 5th Grade Science (from multiple States) (197 questions)

Although these datasets are small (real exam questions of this type are in limited supply), the numbers are large enough to draw conclusions.

3.2 Experiments

We compared our system (called **SceneQA**) with two other state-of-the-art systems for this task:

- **LSModel** (Lexical semantics model): SVM combination of several language models (likelihood of answer given question) and information retrieval scores (score of top retrieved sentence matching question plus answer), trained on a set of questions plus answers. (An expanded version of Section 4.3 of (Jansen et al., 2014))
- **A*Rules**: "Prove" the answer option from the question by applying lexical inference rules automatically extracted from science texts. Select the option with the strongest "proof". (Clark et al., 2014)

[4] http://www.nysedregents.org/Grade4/Science/home.html

	Dev	Test1	Test2	Test3
LSModel	65.96	58.70	28.85	30.08
A*Rules	65.96	67.00	47.00	29.22
SceneQA	**83.51**	66.30	**65.38**	**55.20**

Table 2: SceneQA outperforms two competitive systems on two of the three test sets. The highlighted improvements are statistically significant.

The results (% scores, Table 2) show SceneQA significantly outperforms the two other systems on two of the three test sets, including the largest (Test3, 197 questions), suggesting the approach has merit.

We also performed some case studies to identify what kinds of questions SceneQA does well on, relative to the baselines. In general, SceneQA performs well when the question words and the (correct) answer can be tightly related by background knowledge, including through intermediate nodes (words). For example, in Question 2 below:

Question 2 *Which type of energy does a person use to pedal a bicycle? (A) light (B) sound (C) mechanical (D) electrical*

the KB relates the correct answer "mechanical" to the question words "energy", "pedal", "bicycle", and the intermediate node "power" forming a tight graph. In contrast, the other algorithms select the wrong answer "light" due to frequent mentions of "bicycle lights" in their supporting text corpora that confuses their algorithms.

3.3 Ablations

We also performed ablations to assess which parts of our method are contributing the most:

- **-NewNodes**: Only add edges but no new nodes w during the Build step.
- **-Prune**: Do not prune nodes during the Elaborate and Prune step.
- **-Both**: No new nodes, no pruning

	Dev	Test1	Test2	Test3
SceneQA	**83.51**	66.30	**65.38**	**55.20**
-NewNodes	65.96	69.57	42.31	51.78
-Prune	70.74	57.61	47.12	50.13
-Both	59.57	65.22	42.31	50.25

Table 3: SceneQA outperforms all the ablations on two of the three test sets. The highlighted improvements are statistically significant.

The results (% scores, Table 3) suggest that the two most important algorithmic features - adding

concepts implied but not explicitly stated in the text (NewNodes), and later removing implied information that is of low relevance to the answer (Prune) - are important for answering the questions correctly. (The small gain without adding NewNodes on Test1 is not statistically significant).

3.4 Error Analysis

We also examined cases where SceneQA gave the wrong answer. Two problems were particularly common:

(1) There were two answer options with opposite meanings, and one of them was correct. For example:

Question 3 *An animal that has a backbone is called a(n) (A) invertebrate (B) vertebrate (C) exoskeleton (D) sponge*

Since the relatedness measure we use (i.e. word2vec) cannot distinguish words with similar distributional semantics (a common property of antonyms), our method cannot confidently identify which of the opposites (e.g., here, vertebrate vs. invertebrate) is correct.

(2) The word ordering in the question is particularly important, e.g., questions about processes or sequences. For example:

Question 4 *The process that changes a gas to liquid is called (A) condensation (B) melting (C) evaporation (D) vaporization*

Because our method ignores word order (the knowledge graph is initially populated with keywords in the question), the representation is inherently incapable of capturing sequential information (e.g., here, gas to liquid vs. liquid to gas). As a result, it struggles with such questions.

4 Discussion and Conclusion

Our goal is to answer simple science questions. Unlike entity-centric factoid QA tasks, science questions typically involve general concepts, and answering them requires identifying implicit relationships in the question. Our approach is to view question-answering as constructing a coherent scene. While the notion of scene construction is not new, our insight is that this can be done with a simple "knowledge graph" representation, allowing several massive background KBs to be applied, somewhat alleviating the knowledge bottleneck. Our contribution is to show this works well in the elementary science domain.

Despite this, there are clearly many limitations with our approach: we are largely ignoring syntactic structure in the questions; the KBs are noisy, contributing errors to the scenes; the graph representation has limited expressivity (e.g., no quantification or negation); the word2vec measure of relationship strength does not account for the question context; and contradictions are not detected within the scene. These all contributed to QA failures in the tests. However, the approach is appealing as it takes a step towards a richer picture of language understanding, the empirical results are encouraging, and there are many ways these initial limitations can be addressed going forward. We are confident that this is a rich and exciting space, worthy of further exploration.

Acknowledgments

The research was supported by the Allen Institute for Artificial Intelligence (AI2). We thank the Aristo team at AI2 for invaluable discussions, and the anonymous reviewers for helpful comments.

References

Nicholas Asher and Alex Lascarides. 1998. Bridging. *Journal of Semantics*, 15(1):83–113.

Peter Clark and Phil Harrison. 2009. Large-scale extraction and use of knowledge from text. In *Proceedings of the fifth international conference on Knowledge capture*, pages 153–160.

Peter Clark, Niranjan Balasubramanian, Sumithra Bhakthavatsalam, Kevin Humphreys, Jesse Kinkead, Ashish Sabharwal, and Oyvind Tafjord. 2014. Automatic construction of inference-supporting knowledge bases. In *Proceedings of AKBC*.

Peter Clark. 2015. Elementary school science and math tests as a driver for ai: Take the aristo challenge! In *Twenty-Seventh IAAI Conference*.

James Fan, Ken Barker, and Bruce Porter. 2005. Indirect anaphora resolution as semantic path search. In *Proceedings of the 3rd international conference on Knowledge capture*, pages 153–160. ACM.

David Ferrucci, Eric Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya A Kalyanpur, Adam Lally, J William Murdock, Eric Nyberg, John Prager, et al. 2010. Building watson: An overview of the deepqa project. *AI magazine*, 31(3):59–79.

Sanda M Harabagiu and Dan I Moldovan. 1997. Textnet–a text-based intelligent system. *Natural Language Engineering*, 3(02):171–190.

Jerry R Hobbs, Mark Stickel, Paul Martin, and Douglas Edwards. 1988. Interpretation as abduction. In *Proceedings of the 26th annual meeting on Association for Computational Linguistics*, pages 95–103. Association for Computational Linguistics.

Jerry R Hobbs. 1979. Coherence and coreference. *Cognitive science*, 3(1):67–90.

Peter Jansen, Mihai Surdeanu, and Peter Clark. 2014. Discourse complements lexical semantics for non-factoid answer reranking. In *Proceedings of ACL*, pages 977–986.

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

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Douglas L Nelson, Cathy L McEvoy, and Thomas A Schreiber. 2004. The university of south florida free association, rhyme, and word fragment norms. *Behavior Research Methods, Instruments, & Computers*, 36(3):402–407.

Ekaterina Ovchinnikova, Niloofar Montazeri, Theodore Alexandrov, Jerry R Hobbs, Michael C McCord, and Rutu Mulkar-Mehta. 2014. Abductive reasoning with a large knowledge base for discourse processing. In *Computing Meaning*, pages 107–127. Springer.

Roger C Schank and Robert P Abelson. 1977. *Scripts, Plans, Goals and Understanding*. Erlbaum, Hillsdale, NJ.

Mauro Sozio and Aristides Gionis. 2010. The community-search problem and how to plan a successful cocktail party. In *Proceedings of SIGKDD*, pages 939–948.

Xuchen Yao and Benjamin Van Durme. 2014. Information extraction over structured data: Question answering with freebase. In *Proceedings of ACL*.

Wlodek Zadrozny and Karen Jensen. 1991. Semantics of paragraphs. *Computational Linguistics*, 17(2):171–209.

WIKIQA: A Challenge Dataset for Open-Domain Question Answering

Yi Yang*
Georgia Institute of Technology
Atlanta, GA 30308, USA
yiyang@gatech.edu

Wen-tau Yih **Christopher Meek**
Microsoft Research
Redmond, WA 98052, USA
{scottyih,meek}@microsoft.com

Abstract

We describe the WIKIQA dataset, a new publicly available set of question and sentence pairs, collected and annotated for research on open-domain question answering. Most previous work on answer sentence selection focuses on a dataset created using the TREC-QA data, which includes editor-generated questions and candidate answer sentences selected by matching content words in the question. WIKIQA is constructed using a more natural process and is more than an order of magnitude larger than the previous dataset. In addition, the WIKIQA dataset also includes questions for which there are no correct sentences, enabling researchers to work on answer triggering, a critical component in any QA system. We compare several systems on the task of answer sentence selection on both datasets and also describe the performance of a system on the problem of answer triggering using the WIKIQA dataset.

1 Introduction

Answer sentence selection is a crucial subtask of the open-domain question answering (QA) problem, with the goal of extracting answers from a set of pre-selected sentences (Heilman and Smith, 2010; Yao et al., 2013; Severyn and Moschitti, 2013). In order to conduct research on this important problem, Wang et al. (2007) created a dataset, which we refer to by QASENT, based on the TREC-QA data. The QASENT dataset chose questions in TREC 8-13 QA tracks and selected sentences that share one or more non-stopwords from the questions. Although QASENT has since become the benchmark dataset for the answer selection problem, its creation process actually introduces a strong bias in the types of answers that are included. The following example illustrates an answer that does not share any content words with the question and would not be selected:

Q: *How did Seminole war end?*
A: *Ultimately, the Spanish Crown ceded the colony to United States rule.*

One significant concern with this approach is that the lexical overlap will make sentence selection easier for the QASENT dataset and might inflate the performance of existing systems in more natural settings. For instance, Yih et al. (2013) find that simple word matching methods outperform many sophisticated approaches on the dataset. We explore this possibility in Section 3.

A second, more subtle challenge for question answering is that it normally assumes that there is at least one correct answer for each question in the candidate sentences. During the data construction procedures, all the questions without correct answers are manually discarded.[1] We address a new challenge of *answer triggering*, an important component in QA systems, where the goal is to detect whether there exist correct answers in the set of candidate sentences for the question, and return a correct answer if there exists such one.

We present WIKIQA, a dataset for open-domain question answering.[2] The dataset contains 3,047 questions originally sampled from Bing query logs. Based on the user clicks, each question is associated with a Wikipedia page presumed to be the topic of the question. In order to eliminate answer sentence biases caused by keyword matching, we consider all the sentences in

*Work conducted while interning at Microsoft Research.

[1] The policy is adopted both by the official QASENT tracks (Voorhees and Tice, 1999) and by Wang et al. (2007).

[2] The data and evaluation script can be downloaded at http://aka.ms/WikiQA.

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 2013–2018,
Lisbon, Portugal, 17-21 September 2015.

the summary paragraph of the page as the candidate answer sentences, with labels on whether the sentence is a correct answer to the question provided by crowdsourcing workers. Among these questions, about one-third of them contain correct answers in the answer sentence set.

We implement several strong baselines to study model behaviors in the two datasets, including two previous state-of-the-art systems (Yih et al., 2013; Yu et al., 2014) on the QASENT dataset as well as simple lexical matching methods. The results show that lexical semantic methods yield better performance than sentence semantic models on QASENT, while sentence semantic approaches (e.g., convolutional neural networks) outperform lexical semantic models on WIKIQA. We propose to evaluate answer triggering using question-level precision, recall and F_1 scores. The best F_1 scores are slightly above 30%, which suggests a large room for improvement.

2 WIKIQA Dataset

In this section, we describe the process of creating our WIKIQA dataset in detail, as well as some comparisons to the QASENT dataset.

2.1 Question & Sentence Selection

In order to reflect the true information need of general users, we used Bing query logs as the question source. Taking the logs from the period of May 1st, 2010 to July 31st, 2011, we first selected question-like queries using simple heuristics, such as queries starting with a WH-word (e.g., "what" or "how") and queries ending with a question mark. In addition, we filtered out some entity queries that satisfy the rules, such as the TV show "how I met your mother." In the end, approximately 2% of the queries were selected. To focus on factoid questions and to improve the question quality, we then selected only the queries issued by at least 5 unique users and have clicks to Wikipedia. Among them, we sampled 3,050 questions based on query frequencies.

Because the summary section of a Wikipedia page provides the basic and usually most important information about the topic, we used sentences in this section as the candidate answers. Figure 1 shows an example question, as well as the summary section of a linked Wikipedia page.

Question: Who wrote second Corinthians?

Second Epistle to the Corinthians The Second Epistle to the Corinthians, often referred to as Second Corinthians (and written as 2 Corinthians), is the eighth book of the New Testament of the Bible. Paul the Apostle and "Timothy our brother" wrote this epistle to "the church of God which is at Corinth, with all the saints which are in all Achaia".

Figure 1: An example question and the summary paragraph of a Wikipedia page.

2.2 Sentence Annotation

We employed crowdsourcing workers through a platform, which is similar to Amazon MTurk, to label whether the candidate answer sentences of a question are correct. We designed a cascaded Web UI that consists of two stages. The first stage shows a testing question, along with the title and the summary paragraph of the associated Wikipedia page, asking the worker "Does the short paragraph answer the question?" If the worker chooses "No", then equivalently all the sentences in this paragraph are marked incorrect and the UI moves to the next question. Otherwise, the system enters the second stage and puts a checkbox along each sentence. The worker is then asked to check the sentences that can answer the question *in isolation*, assuming coreference is resolved. To ensure the label quality, each question was labeled by three workers. Sentences with inconsistent labels would be verified by a different set of crowdsourcing workers. The final decision was based on the majority vote of all the workers. In the end, we included 3,047 questions and 29,258 sentences in the dataset, where 1,473 sentences were labeled as answer sentences to their corresponding questions.

Although not used in the experiments, each of these answer sentence is associated with the *answer phrase*, which is defined as the shortest substring of the sentence that answers the question. For instance, the second sentence in the summary paragraph shown in Figure 1 is an answer sentence. Its substring "Paul the Apostle and Timothy our brother" can be treated as the answer phrase. The annotations of the answer phrases were given by the authors of this paper. Because the answer phrase boundary can be highly ambiguous, each sentence is associated with at most two answer phrases that are both acceptable, given by two different labelers. We hope this addition to the WIKIQA data can be beneficial to future researchers for building or evaluating an end-to-end question answering system.

	Train	Dev	Test	Total
# of ques.	94	65	68	227
# of sent.	5,919	1,117	1,442	8,478
# of ans.	475	205	248	928
Avg. len. of ques.	11.39	8.00	8.63	9.59
Avg. len. of sent.	30.39	24.90	25.61	28.85

Table 1: Statistics of the QASENT dataset.

	Train	Dev	Test	Total
# of ques.	2,118	296	633	3,047
# of sent.	20,360	2,733	6,165	29,258
# of ans.	1,040	140	293	1,473
Avg. len. of ques.	7.16	7.23	7.26	7.18
Avg. len. of sent.	25.29	24.59	24.95	25.15
# of ques. w/o ans.	1,245	170	390	1,805

Table 2: Statistics of the WIKIQA dataset.

2.3 WIKIQA vs. QASENT

Our WIKIQA dataset differs from the existing QASENT dataset in both question and candidate answer sentence distributions. Questions in QASENT were originally used in TREC 8-13 QA tracks and were a mixture of questions from query logs (e.g., Excite and Encarta) and from human editors. The questions might be outdated and do not reflect the true information need of a QA system user. By contrast, questions in WIKIQA were sampled from real queries of Bing without editorial revision. On the sentence side, the candidate sentences in QASENT were selected from documents returned by past participating teams in the TREC QA tracks, and sentences were only included if they shared content words from the questions. These procedures make the distribution of the candidate sentence skewed and unnatural. In comparison, 20.3% of the answers in the WIKIQA dataset share no content words with questions. Candidate sentences in WIKIQA were chosen from relevant Wikipedia pages directly, which could be closer to the input of an answer sentence selection module of a QA system.

To make it easy to compare results of different QA systems when evaluated on the WIKIQA dataset, we randomly split the data to training (70%), development (10%) and testing (20%) sets. Some statistics of the QASENT and WIKIQA datasets are presented in Tables 1 and 2.[3] WIKIQA contains an order of magnitude more questions and three times more answer sentences compared to QASENT. Unlike QASENT, we did not filter questions with only incorrect answers, as they are still valuable for model training and more importantly, useful for evaluating the task of answer triggering, as described in Section 3. Specifically, we find nearly two-thirds of questions contain no correct answers in the candidate sentences.

Class	QASENT	WIKIQA
Location	37 (16%)	373 (12%)
Human	65 (29%)	494 (16%)
Numeric	70 (31%)	658 (22%)
Abbreviation	2 (1%)	16 (1%)
Entity	37 (16%)	419 (14%)
Description	16 (7%)	1087 (36%)

Table 3: Question classes of the QASENT and WIKIQA datasets.

The distributions of question types in these two datasets are also different, as shown in Table 3.[4] WIKIQA contains more description or definition questions, which could be harder to answer.

3 Experiments

Many systems have been proposed and tested on the QASENT dataset, including lexical semantic models (Yih et al., 2013) and sentence semantic models (Yu et al., 2014). We investigate the performance of several systems on WIKIQA and QASENT. As discussed in Section 2, WIKIQA offers us the opportunity to evaluate QA systems on answer triggering. We propose simple metrics and perform a feature study on the new task. Finally, we include some error analysis and discussion at the end of this section.

3.1 Baseline Systems

We consider two simple word matching methods: Word Count and Weighted Word Count. The first method counts the number of non-stopwords in the question that also occur in the answer sentence. The second method re-weights the counts by the IDF values of the question words.

We reimplement LCLR (Yih et al., 2013), an answer sentence selection approach that achieves very competitive results on QASENT. LCLR

[3] We follow experimental settings of Yih et al. (2013) on the QASENT dataset. Although the training set in the original data contains more questions, only 94 of them are paired with sentences that have human annotations.

[4] The classifier is trained using a logistic regression model on the UIUC Question Classification Datasets (`http://cogcomp.cs.illinois.edu/Data/QA/QC`). The performance is comparable to (Li and Roth, 2002).

Model	QASENT		WIKIQA	
	MAP	MRR	MAP	MRR
Word Cnt	0.5919	0.6662	0.4891	0.4924
Wgt Word Cnt	0.6095	0.6746	0.5099	0.5132
LCLR	**0.6954**	0.7617	0.5993	0.6086
PV	0.5213	0.6023	0.5110	0.5160
CNN	0.5590	0.6230	0.6190	0.6281
PV-Cnt	0.6762	0.7514	0.5976	0.6058
CNN-Cnt	0.6951	**0.7633**	**0.6520**	**0.6652**

Table 4: Baseline results on both QASENT and WIKIQA datasets. Questions without correct answers in the candidate sentences are removed in the WIKIQA dataset. The best results are in **bold**.

Model	Prec	Rec	F_1
CNN-Cnt	26.09	37.04	30.61
+QLen	27.96	37.86	32.17
+SLen	26.14	37.86	30.92
+QClass	27.84	33.33	30.34
+All	28.34	35.80	31.64

Table 5: Evaluation of answer triggering on the WIKIQA dataset. Question-level precision, recall and F_1 scores are reported.

makes use of rich lexical semantic features, including word/lemma matching, WordNet and vector-space lexical semantic models. We do not include features for Named Entity matching.[5]

We include two sentence semantic methods, Paragraph Vector[6] (PV; Le and Mikolov, 2014) and Convolutional Neural Networks (CNN; Yu et al., 2014). The model score of PV is the cosine similarity score between the question vector and the sentence vector. We follow Yu et al. (2014) and employ a bigram CNN model with average pooling. We use the pre-trained word2vec embeddings provided by Mikolov et al. (2013) as model input.[7] For computational reasons, we truncate sentences up to 40 tokens for our CNN models.

Finally, we combine each of the two sentence semantic models with the two word matching features by training a logistic regression classifier, referring as PV-Cnt and CNN-Cnt. CNN-Cnt has been shown to achieve state-of-the-art results on the QASENT dataset (Yu et al., 2014).

3.2 Evaluation of Answer Triggering

The task of answer sentence selection assumes that there exists at least one correct answer in the candidate answer sentence set. Although the assumption simplifies the problem of question answering, it is unrealistic for practical QA systems. Modern QA systems rely on an independent component to pre-select candidate answer sentences, which utilizes various signals such as lexical matching and user behaviors. However, the candidate sentences are not guaranteed to contain the correct answers, no matter what kinds of pre-selection components are employed. We propose the answer triggering task, a new challenge for the question answering problem, which requires QA systems to: (1) detect whether there is at least one correct answer in the set of candidate sentences for the question; (2) if yes, select one of the correct answer sentences from the candidate sentence set.

Previous work adopts MAP and MRR to evaluate the performance of a QA system on answer sentence selection. Both metrics evaluate the relative ranks of correct answers in the candidate sentences of a question, and hence are not suitable for evaluating the task of answer triggering. We need metrics that consider both the presence of answers with respect to a question and the correctness of system predictions.

We employ precision, recall and F_1 scores for answer triggering, at the question level. In particular, we compute these metrics by aggregating all the candidate sentences of a question. A question is treated as a positive case only if it contains one or more correct answer sentences in its candidate sentence pool. For the prediction of a question, we only consider the sentence in the candidate set that has the highest model score. If the score is above a predefined threshold and the sentence is labeled as a correct answer to the question, then it means that the prediction is correct and the question is answered correctly.

3.3 Results

WIKIQA vs. QASENT The MAP and MRR results are presented in Table 4. We only evaluate questions with answers in the WIKIQA dataset under these metrics. On the QASENT dataset, as found by prior work, the two word matching methods are very strong baselines, in which they sig-

[5]The improvement gains from the features are marginal on the QASENT dataset.

[6]We choose the Distributed Bag of Words version of Paragraph Vector, as we found it significantly outperforms the Distributed Memory version of Paragraph Vector.

[7]Available at https://code.google.com/p/word2vec/

nificantly outperform sentence semantic models. By incorporating rich lexical semantic information, LCLR further improves the results. CNN-Cnt gives results that match LCLR, and PV-Cnt performs worse than CNN-Cnt.[8]

The story on the WIKIQA dataset is different. First, methods purely rely on word matching are not sufficient to achieve good results. Second, CNN significantly outperforms simple word matching methods and performs slightly better than LCLR, which suggests that semantic understanding beyond lexical semantics is important for obtaining good performance on WIKIQA. Finally, word matching features help to further boost CNN results by approximately 3 to 4 points in both MAP and MRR.

Evaluation of answer triggering on WIKIQA We evaluate the best system CNN-Cnt on the task of answer triggering, and the results are shown in Table 5. We tune the model scores for making predictions with respect to F_1 scores on the dev set, due to the highly skewed class distribution in training data. The absolute F_1 scores are relative low, which suggests a large room for improvement.

We further study three additional features: the length of question (QLen), the length of sentence (SLen), and the class of the question (QClass). The motivation for adding these features is to capture the hardness of the question and comprehensiveness of the sentence. Note that the two question features have no effects on MAP and MRR. As shown in Table 5, the question-level F_1 score is substantially improved by adding a simple QLen feature. This suggests that designing features to capture question information is very important for this task, which has been ignored in the past. SLen features also give a small improvement in the performance, and QClass feature has slightly negative influence on the results.

3.4 Error Analysis & Discussion

The experimental results show that for the same model, the performance on the WIKIQA dataset is inferior to that on the QASENT dataset, which suggests that WIKIQA is a more challenging dataset. Examining the output of CNN-Cnt, the best performing model, on the WIKIQA dev set seems to suggest that deeper semantic understanding and answer inference are often required. Below are two examples selected that CNN-Cnt does not correctly rank as the top answers:

Q1: *What was the GE building in Rockefeller Plaza called before?*
A1: [GE Building] *Known as the RCA Building until 1988, it is most famous for housing the headquarters of the television network NBC.*

Q2: *How long was I Love Lucy on the air?*
A2: [I Love Lucy] *The black-and-white series originally ran from October 15, 1951, to May 6, 1957, on the Columbia Broadcasting System (CBS).*

Answering the first question may require a better semantic representation that captures the relationship between "called before" and "known ... until". As for the second question, knowing that on a TV channel (e.g., CBS) implies "on the air" and a time span between two dates is legitimate to a "how long" question is clearly beneficial.

4 Conclusion

We present WIKIQA, a new dataset for open-domain question answering. The dataset is constructed in a natural and realistic manner, on which we observed different behaviors of various methods compared with prior work. We hope that WIKIQA enables research in the important problem of answer triggering and enables further research in answer sentence selection in more realistic settings. We also hope that our empirical results will provide useful baselines in these efforts.

References

Michael Heilman and Noah A. Smith. 2010. Tree edit models for recognizing textual entailments, paraphrases, and answers to questions. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1011–1019.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 1188–1196.

Xin Li and Dan Roth. 2002. Learning question classifiers. In *Proceedings the International Conference on Computational Linguistics (COLING)*, pages 556–562.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the Conference on Advances*

[8] Our CNN reimplementation performs slightly worse than (Yu et al., 2014).

in Neural Information Processing Systems (NIPS), pages 3111–3119.

Aliaksei Severyn and Alessandro Moschitti. 2013. Automatic feature engineering for answer selection and extraction. In *Proceedings of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*, pages 458–467.

Ellen M. Voorhees and Dawn M. Tice. 1999. The TREC-8 question answering track evaluation. In *Proceedings of the Text Retrieval Conference TREC-8*, page 82.

Mengqiu Wang, Noah A Smith, and Teruko Mitamura. 2007. What is the Jeopardy model? A quasi-synchronous grammar for QA. In *Proceedings of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*, pages 22–32.

Xuchen Yao, Benjamin Van Durme, Chris Callison-Burch, and Peter Clark. 2013. Answer extraction as sequence tagging with tree edit distance. In *Proceedings of the Annual Meeting of the North American Association of Computational Linguistics (NAACL)*, pages 858–867.

Wen-tau Yih, Ming-Wei Chang, Christopher Meek, and Andrzej Pastusiak. 2013. Question answering using enhanced lexical semantic models. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Lei Yu, Karl Moritz Hermann, Phil Blunsom, and Stephen Pulman. 2014. Deep learning for answer sentence selection. In *Proceedings of the Deep Learning and Representation Learning Workshop: NIPS-2014*.

Personalized Machine Translation: Predicting Translational Preferences

Shachar Mirkin*
IBM Research - Haifa
Mount Carmel, Haifa
31905, Israel
shacharm@il.ibm.com

Jean-Luc Meunier
Xerox Research Centre Europe
6 chemin de Maupertuis
38240 Meylan, France
jean-luc.meunier@xrce.xerox.com

Abstract

Machine Translation (MT) has advanced in recent years to produce better translations for clients' specific domains, and sophisticated tools allow professional translators to obtain translations according to their prior edits. We suggest that MT should be further personalized to the end-user level – the receiver or the author of the text – as done in other applications. As a step in that direction, we propose a method based on a recommender systems approach where the user's preferred translation is predicted based on preferences of similar users. In our experiments, this method outperforms a set of non-personalized methods, suggesting that user preference information can be employed to provide better-suited translations for each user.

1 Introduction

Technologies are increasingly personalized, accommodating their behavior for each user. Such personalization is done through user modeling where the goal is to "get to know" the user. To that end, personalization is based on users' attributes, such as demographics (gender, age etc.), personalities, and preferences. For example, in Information Retrieval, results are customized according to the user's information and search history (Speretta and Gauch, 2005), performance of Automatic Speech Recognition substantially improves when adapted to a specific speaker (Neumeyer et al., 1995), and Targeted Advertising makes use of the user's location and prior purchases (Kölmel and Alexakis, 2002).

Personalization in machine translation has a somewhat different nature. Providers of MT tools and services offer means to "customize" or "personalize" the translation engine for each client, mostly through domain adaptation techniques, and a great deal of effort is made to make the human-involved translation process more efficient (see Section 2.2). Most of the focus, though, goes to customization for companies or professional translators. We argue that *Personalized Machine Translation* (PMT below) should and can take the next step and directly address individual end-users.

The difficulty to objectively determine whether one (automatic) translation is better than another has been repeatedly revealed in the MT literature. Our conjecture is that one reason is individual preferences, to which we refer as *Translational Preferences* (TP). TP come into play both when the alternative translations are all correct, and when each of them is wrong in a different way. In the former case, a preference may be a stylistic choice, and in the latter, a matter of comprehension or a selection of the least intolerable error in one's opinion. For instance, one user may prefer shorter sentences than others; she may favor a more formal style, while another would rather have it casual. A user could be fine with some reordering errors but be more picky concerning punctuations. One user will not be bothered if some words are left untranslated (perhaps because the source language belongs to the same language family as the target language that he speaks), while another will find it utterly displeasing. Such differences may be the result of the type of translation system being employed (e.g. syntax- vs. phrased-based), the specific training data or many other factors. On the user's side, a preference may be attributed, for example, to her mother tongue, her age or her personality.

Two aspects of end-user PMT may be considered: (i) Personalized translation of texts written by a specific user, and (ii) PMT to provide better translations for a specific reader. In this work we address the second task, aiming to identify translations each user is more likely to prefer.[1] Specifically, we consider a setting where at least two MT systems are available, and the goal is to predict which of the translation systems the user would choose, assuming we have no knowledge about her preference between them. Benchmarking the systems in advance with respect to a reference set, or estimating the quality of the translations (Specia et al., 2009) are viable alternatives for translation selection; these, however, are not personalized to the target user. Instead, we employ a user-user Collaborative Filtering approach, common in Recommender Systems, which we map to the TP prediction task.

We assess this approach using a collection of user rankings of MT systems from a shared translation task

*This work was done while the first author was at Xerox Research Centre Europe.

[1] In (Mirkin et al., 2015) we investigate the first task, assessing whether the author's demographic and personality traits are preserved over machine translation.

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 2019–2025,
Lisbon, Portugal, 17-21 September 2015.

(see Section 3). Our results show that the personalized method modestly, but consistently, outperforms several other approaches that rank the systems in general, disregarding the specific user. We consider this as an indication that user feedback can be employed towards a more personalized approach to machine translation.

2 Background

2.1 Collaborative filtering

Collaborative filtering (CF) is a common approach employed by recommender systems for suggesting to users items, such as books or movies. A recommender system may simply suggest to all users the most popular items; often, however, the recommendations are personalized for each individual user to fit her taste or preferences. User-user CF relies on community preferences. The idea is to recommend to the user items that are liked by users similar to her, as manifested, for example, by high rating. Similar users are those that agree with the current user on previously-rated items. In k-nearest-neighbors CF, a user is typically represented by a vector of her preferences, where each entry of the vector is, e.g., a rating of a movie. k similar users are then identified by measuring the similarity between the users' vectors. Cosine similarity is a popular function for that purpose, and we also use it in our work (Resnick et al., 1994; Sarwar et al., 2001; Ricci et al., 2011). An alternative to cosine, Pearson's correlation coefficient (Pearson, 1895), allows addressing different rating patterns across users. In comparison to cosine, here vector entries are normalized with respect to the user's average rating. In our case, such normalization is not very meaningful since the entries of the users vectors represent comparisons rather than absolute ratings, as will be made clear in Section 4. Nevertheless, we have experimented with Pearson correlation as well, and found no advantage in using it instead of cosine.

2.2 Customization, personalization and adaptation in MT

Various means of customization and personalization are available, in both academic and commercial MT. Many of them target the company, rather than the individual user, and much of the effort is invested in designing tools for professional translators, aiming to improve their productivity, through intelligent Computer Aided Translation (CAT).

Domain adaptation methods are commonly used to adapt to the topic, the genre and even the style of the translated material. Using the company's own corpora is one of the simplest techniques to do so, but many more approaches have been proposed, including data-selection (Axelrod et al., 2011; Gascó et al., 2012; Mirkin and Besacier, 2014), mixture models (Foster and Kuhn, 2007) and table fill-up (Bisazza et al., 2011). Clients can utilize their own glossaries (Federico et al., 2014), corpora (parallel or monolingual) and translation memories (TM), either shared or private ones (Caskey and Maskey, 2014; Federico et al., 2014). Through Adaptive and Interactive MT (Nepveu et al., 2004), the system learns from the translator's edits, in order to avoid repeating errors that have already been corrected. Post-editions can continuously be added to the translator's TM or be used as additional training material, for tighter adaptation to the domain of interest, through batch or incremental training.

2.3 User preferences in MT

Many tasks that require annotation by humans are affected by the annotator and not only by the item being judged. Metrics for inter-rater reliability or inter-annotator agreement, such as Cohen's Kappa (Cohen, 1960), help measuring the extent to which annotators disagree. Disagreement may be due to untrained or inattentive annotators, a result of a task that is not well defined, or when there is no obvious "truth". Such is the case with the evaluation of translation quality – it is not always straightforward to tell whether one translation is better than another. A single sentence can be translated in multiple correct ways. The decision becomes even harder when the translations are automatically produced and are imperfect: Is one error worse than another? The answer is in the eye of the beholder. MT papers regularly report rather low Kappa levels, even when measured on simpler tasks, such as short segments (Macháček and Bojar, 2015).

Turchi et al. (2013) refer to the issue of "subjectivity" of human annotators. They address the task of binary classification of "good" vs. "bad" translations, and show that relying on human annotation for training a binary quality estimator is less effective than using automatically-generated labels. This subjectivity is exactly what we are after. We treat it as a *preference*, trying to identify the systems or specific translations which the user subjectively prefers.

Kichhoff et al. (2012) analyze user preferences with respect to MT errors. They show that some types, e.g. word order errors, are the most dis-preferred by users, and that this is a more important factor than the number of errors. While very relevant for our research, their analysis is aggregated over all users participating in the study, and is not focusing on individuals' preferences.

3 Data

In this work we used the data provided for the MT Shared Task in the 2013 Workshop on Statistical Machine Translation (WMT) (Bojar et al., 2013).[2] This data was of a particularly large scale, with crowd-sourced human judges, either volunteer researchers or paid Amazon Turkers. For each source sentence, a judge was presented with the source sentence itself, a reference translation, and the outputs of five machine translation systems. The five systems were randomly selected from the pool of participating systems,

[2] http://www.statmt.org/wmt13/translation-task.html

and were anonymized and randomly-ordered when presented to the judge. The judge had to rank the translations, with ties allowed (i.e. two system can receive the same ranking). Hence, each annotation point provided with 10 pairwise rankings between systems. Translations of 10 language pairs were assessed, with 11 to 19 systems for each pair. In total, over 900K non-tied pairwise rankings were collected. The Turkers' annotation included a control task for quality assurance, rejecting Turkers failing more than 50% of the control points. The inter-annotator score showed on average a fair to moderate level of agreement.

4 Translational preferences with collaborative filtering

Our method, denoted CTP (*Collaborative Translational Preferences*), is based on a k-nearest-neighbors approach for user-user CF. That is, we predict the translational preferences of a user based on those of similar users. In our setting, a user preference is the choice between two translation systems – which system's translations does the user prefer. Given two systems (or models of the same system) we wish to predict which one the user would prefer, without assuming the user has ever expressed her preference between these two specific systems. It is important to emphasize that the method presented here considers the users' overall preferences of systems, and does not regard the specific sentence that is being translated. In future work we intend to make use of this information as well.

4.1 Representation

As mentioned in Section 3, each annotation consists of a ranking of five systems. From that, we extract pairwise rankings for every pair of systems that were ranked for a given language pair. For each user $u \in \mathcal{U}$ (where $\mathcal{U}$ are all users who annotated the language pair), we create a user-preference vector, p_u, that contains an entry for each pair of translation systems. Denoting the set of systems with $\mathcal{S}$, we have $\frac{|\mathcal{S}|\cdot(|\mathcal{S}|-1)}{2}$ system pairs. E.g., for Czech-English, with 11 participating systems, the user vector size is 55. Each entry (i, j) of the vector is assigned the following value:

$$p_u{}^{(i,j)} = \frac{w_u^{(i,j)} - l_u^{(i,j)}}{w_u^{(i,j)} + l_u^{(i,j)}} \quad (1)$$

where $w_u^{(i,j)}$ and $l_u^{(i,j)}$ are the number of wins and loses of system s_i vs. system s_j as judged by user u.[3]

With this representation, a user vector contains values between -1 (if s_i always lost to s_j) and 1 (if s_i always won). If the user always ranked the two systems identically, the value is 0, and if she has never evaluated the pair, the entry is regarded as a missing value (*NA*). Altogether, we have a matrix of users by system pairs, as depicted in Figure 1.

[3]We have also considered including ties in the denominator of the equation; discarding them was found superior.

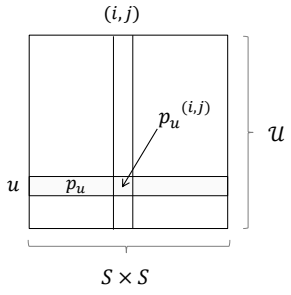


Figure 1: The user-preferences matrix.

4.2 Finding similar users

Given a user preference to predict for a pair of systems (s_i, s_j), we compute the similarity between p_u and each one of $p_{u'}$ for all other $u' \in \mathcal{U}$. In our experiments we used cosine as the similarity measure. The k most-similar-users (MSU) are then selected. To be included in $MSU(u)$, we require that u and u' have judged at least 2 common system pairs.

4.3 Preference prediction

Given the similarity scores, to predict the user's preference for the target system pair, we compute a weighted average of the predictions of the users in $MSU(u)$.

We include in the average only users with similarity scores above a certain positive threshold (0.05). We then require that a minimum number of users meet the above criteria of common annotations and minimum similarity (we used 5). If not enough such similar users are found, we turn to a fallback, where we use the non-weighted average preference across all users (AVPF presented in Section 5).[4] The prediction is then the sign of the weighted average. A positive value means s_i is the preferred system; a negative one means it is s_j, and a zero is a draw. In our evaluation we compare this prediction to the sign of the actual preference of the user, $p_u{}^{(i,j)}$. Formally, CTP computes the following prediction function f for a given user u and a system pair (s_i, s_j):

$$f_{\text{CTP}}(u)^{(i,j)} = sign\left(\frac{\sum_{u'} p_{u'}{}^{(i,j)} \cdot sim(u, u')}{\sum_{u'} sim(u, u')}\right) \quad (2)$$

where $u' \in MSU(u)$ are the most similar users (the nearest neighbors) of u; $p_{u'}{}^{(i,j)}$ are the preferences of user u' for (s_i, s_j) and $sim(u, u')$ is the similarity score between the two users.[5]

5 Experiments and results

5.1 Evaluation methodology

In our experiments we try to predict which one of two translation systems would be preferred by a given user.

[4]The fallback was used 0.1% of the times.

[5]The denominator is not required as long as we predict only the sign since all used similarity scores are positive. We keep it in order to obtain a normalized score that can be used for other decisions, e.g. ranking multiple systems.

We evaluate our method, as well as several other prediction functions, when compared with the user's pairwise system preference according to the annotation – ${p_u}^{(i,j)}$, shown in Equation 1. For each user this is an aggregated figure over all her pairwise rankings for the pair, determining the preferred system as the one chosen by the user (i.e. ranked higher) more times.

We conduct a leave-one-out experiment. For each language pair, we iterate over all non-*NA* entries in the user-preferences matrix, remove the entry and try to predict it. User similarity scores are re-computed for each evaluation point, to ensure they do not consider the target pair. The "gold" preference is positive when the user prefers s_i, negative when she prefers s_j and 0 when she has no preference between them. Hence, each of the assessed methods is measured by the accuracy of predicting the sign of the preference.

5.2 Non-personalized methods

We compare CTP to the following prediction methods:

Always i (ALI) This is a naïve baseline showing the score when always predicting that system i wins. Note that the baseline is not simply 50% due to ties.

Average rank (RANK) Here, two systems are compared by the average of their rankings across all annotations ($r \in \{1, 2, 3, 4, 5\}$):

$$f_{\text{RANK}}(u)^{(i,j)} = sign(\overline{r}_j - \overline{r}_i) \tag{3}$$

$\overline{r}_j$ and $\overline{r}_i$ are the average ranks of s_j and s_i respectively. Since a smaller value of r corresponds to a higher rank, we subtract the rank of s_i from s_j and not the other way around. This way, if for instance, s_i is ranked on averaged higher than s_j, the prediction would be positive, as desired.

Expected (EXPT) This metric, proposed by Koehn (2012) and used by Bojar et al. (2013) in order to rank the participating systems in the WMT benchmark, compares the *expected wins* of the two systems. Its intuition is explained as follows: "If the system is compared against a randomly picked opposing system, on a randomly picked sentence, by a randomly picked judge, what is the probability that its translation is ranked higher?" The expected wins of s_i, $e(s_i)$, is the probability of s_i to win when compared to another system, estimated as the total number of wins of s_i relative to the total number of comparisons involving it, excluding ties, and normalized by the total number of systems excluding s_i, $|\{s_k\}|$:

$$e(i) = \frac{1}{|\{s_k\}|} \sum_{k \neq i} \frac{w^{(i,k)}}{w^{(i,k)} + l^{(i,k)}} \tag{4}$$

where $w^{(i,k)}$ and $l^{(i,k)}$ are summed over all users. The preference prediction is therefore:

$$f_{\text{EXPT}}(u)^{(i,j)} = sign(e(i) - e(j)) \tag{5}$$

RANK and EXPT predict preferences based on a system's performance in general, when compared to **all** other systems. We propose an additional prediction function for comparison which uses only the information concerning the system pair under consideration.

Average user preference (AVPF) This method takes into account only the specific system pair and averages the user preferences for the pair. Formally:

$$f_{\text{AVPF}}(u)^{(i,j)} = sign(\frac{\sum_{u'} p_{u'}^{(i,j)}}{|\{u'\}|}) \tag{6}$$

where $u' \neq u$, and $\{u'\}$ are all users except u.

This method can be viewed as a non-personalized version of CTP, with two differences:

(1) It considers all users, and not only similar ones.
(2) It does not weight the preferences of the other users by their similarity to the target user.

5.3 Results

Table 1 shows the results of an experiment comparing the performance of the various methods in terms of prediction accuracy. Figure 2 shows the micro-average scores, when giving each of the 97,412 test points an equal weight in the average. CTP outperforms all others for 9 out of 10 language pairs, and in the overall micro-averaged results. The difference between CTP and each of the other metrics was found statistically significance with $p < 5 \cdot 10^{-6}$ at worse, as measured with a paired Wilcoxon signed rank test (Wilcoxon, 1945) on the predictions of the two methods. The significance test captures in this case the fact that the methods disagreed in many more cases than is visible by the score difference.

Our method was found superior to all others also when computing macro-average, taking the average of the scores of each language pair, as well as when the ties are included in the computation of p_u.

The parameters with which the above results were obtained are found within the method's description in Section 4. Yet, in our experiments, CTP turned out to be rather insensitive to their values. In this experiment we used a global set of parameters and did not tune them for each language pair separately. It is reasonable to assume that such tuning would improve results. For instance, choosing k, the number of users to include in the average, depends on the total number of users. E.g., for *en-es*, where there are only 57 users in total, reducing k's value from 50 to 25, improves results of CTP from 62.6% to 63.2%, higher than all other methods (whose scores are not affected).

Specifically in comparison to AVPF, weighting by the similarity scores was found to be a more significant factor than selecting a small subset of the users. This may not come as a surprise, since less similar users that are added to $MSU(u)$ have a smaller impact on the final decision since their weight in the average is smaller.

Lang.	f	Acc.
cs-en	ALI	31.6
	RANK	62.9
	EXPT	63.5
	AVPF	63.5
	CTP	**64.4**
en-cs	ALI	36.2
	RANK	67.8
	EXPT	67.9
	AVPF	67.4
	CTP	**68.2**

Lang.	f	Acc.
de-en	ALI	41.7
	RANK	62.6
	EXPT	62.6
	AVPF	62.6
	CTP	**63.5**
en-de	ALI	42.0
	RANK	67.2
	EXPT	66.9
	AVPF	66.5
	CTP	**67.6**

Lang.	f	Acc.
es-en	ALI	35.5
	RANK	61.0
	EXPT	61.2
	AVPF	61.4
	CTP	**63.0**
en-es	ALI	35.9
	RANK	62.3
	EXPT	**63.0**
	AVPF	61.5
	CTP	62.6

Lang	f	Acc.
fr-en	ALI	35.0
	RANK	61.3
	EXPT	61.2
	AVPF	61.2
	CTP	**61.8**
en-fr	ALI	35.0
	RANK	65.0
	EXPT	65.1
	AVPF	64.4
	CTP	**65.3**

Lang.	f	Acc.
ru-en	ALI	43.5
	RANK	57.6
	EXPT	57.8
	AVPF	56.6
	CTP	**58.2**
en-ru	ALI	44.6
	RANK	70.2
	EXPT	72.1
	AVPF	71.4
	CTP	**72.4**

Table 1: Results in accuracy percentage for the 10 language pairs, including the languages: Czech (*cs*), English (*en*), German (*de*), Spanish (*es*), French (*fr*) and Russian (*ru*). The best results is in bold. The difference between CTP and each of the other methods is highly statistically significant. Figure 2 shows a micro-average of these results.

One weakness of CTP, as well as of other methods, is that it poorly predict ties. In the above experiment, approximately 13.5% of the preferences were 0, none of them was correctly identified. Our analysis showed that numerical accuracy is not the main cause; setting any prediction that is smaller than some values of $|\varepsilon|$ to 0 was not found helpful. Arguably, ties need not be predicted, since if the user has no preference between two systems, any choice is just as good. Still, we believe that better ties prediction could lead to general improvement of our method and we wish to address it in future work.

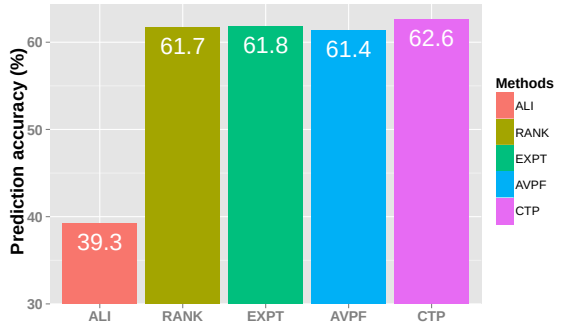


Figure 2: Micro-average over all 97,412 test points.

6 Discussion

We addressed the task of predicting user preference with respect to MT output via a collaborative filtering approach whose prediction is based on preferences of similar users. This method predicts TP better than a set of non-personalized methods. The gain is modest in absolute numbers, but the results are highly statistically significant and stable over parameter values.

We consider this work as a step towards more personalized MT. This line of research can be extended in multiple ways. First and foremost, as mentioned, we did not consider the actual content of the sentences, but rather identified a general preference for one system over another. It is plausible, however, that one system is better – from the user's perspective – at translating one type of text, while another is preferred for other texts. Taking the actual texts into account seems therefore essential. Content-based methods for recommender systems may be useful for this purpose. Another factor that may be affecting preferences is translation quality: when compared translations are all poor, preferences play a less significant role. Hence, it may be informative to assess TP prediction separately across different levels of translation quality.

Large parallel corpora are typically required for training reasonable statistical translation models. Yet, parallel corpora, and even more so in-domain ones, are hard to gather. It is virtually impossible to find a user-specific parallel corpus, and methods for monolingual domain adaptation are easier to envisage if one wishes to address author-aware PMT (the first PMT task mentioned in Section 1). Collecting user feedback is another challenge, especially since most end-users do not speak the source language. For that and other reasons, it currently seems more feasible to collect preference information from professional translators, explicitly or implicitly.Yet, in this research we aim at end-users rather than translators whose preferences are often driven by the ease of correction more than anything else. We believe that one way to tackle this issue is to exploit other kinds of feedback, from which we can infer user preferences and similarity. Online MT providers are recently collecting end-user feedback for their proposed translations which may be useful for TP prediction. For instance, in early 2015 Facebook introduced a feature letting users rate (Bing) translations, and Google Translate asks for suggested improvements. We are hopeful that such data becomes publicly available. Nevertheless, it remains unlikely to obtain feedback from each and every user. A potential direction for both corpora and feedback collection is personalizing models and identifying preferences for groups of users based on socio-demographic traits, such as gender, age or mother tongue, or based on (e.g. Big 5) personality traits. These can even be inferred by automatically analyzing user texts.

Acknowledgments

We wish to thank Hervé Déjean and the EMNLP reviewers for their valuable feedback on this work.

References

Amittai Axelrod, Xiaodong He, and Jianfeng Gao. 2011. Domain adaptation via pseudo in-domain data selection. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 355–362, Stroudsburg, PA, USA. Association for Computational Linguistics.

Arianna Bisazza, Nick Ruiz, and Marcello Federico. 2011. Fill-up versus interpolation methods for phrase-based SMT adaptation. In *Proceedings of the International Workshop on Spoken Language Translation (IWSLT)*, San Francisco, California, USA.

Ondřej Bojar, Christian Buck, Chris Callison-Burch, Christian Federmann, Barry Haddow, Philipp Koehn, Christof Monz, Matt Post, Radu Soricut, and Lucia Specia. 2013. Findings of the 2013 Workshop on Statistical Machine Translation. In *Proceedings of the Eighth Workshop on Statistical Machine Translation (WMT)*, pages 1–44, Sofia, Bulgaria, August. Association for Computational Linguistics.

Sasha P. Caskey and Sameer Maskey. 2014. Translation cache prediction, August 12. US Patent 8,805,672.

Jacob Cohen. 1960. A Coefficient of Agreement for Nominal Scales. *Educational and Psychological Measurement*, 20(1):37.

Marcello Federico, Nicola Bertoldi, Mauro Cettolo, Matteo Negri, Marco Turchi, Marco Trombetti, Alessandro Cattelan, Antonio Farina, Domenico Lupinetti, Andrea Martines, Alberto Massidda, Holger Schwenk, Loïc Barrault, Frederic Blain, Philipp Koehn, Christian Buck, and Ulrich Germann. 2014. The matecat tool. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: System Demonstrations*, pages 129–132, Dublin, Ireland, August. Dublin City University and Association for Computational Linguistics.

George Foster and Roland Kuhn. 2007. Mixture-model adaptation for SMT. In *Proceedings of the Second Workshop on Statistical Machine Translation (WMT)*, Prague, Czech Republic, June.

Guillem Gascó, Martha-Alicia Rocha, Germán Sanchis-Trilles, Jesús Andrés-Ferrer, and Francisco Casacuberta. 2012. Does more data always yield better translations? In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 152–161, Avignon, France, April. Association for Computational Linguistics.

Katrin Kirchhoff, Daniel Capurro, and Anne Turner. 2012. Evaluating user preferences in machine translation using conjoint analysis. *Proceedings of the European Association of Machine Translation*, 16:119–126.

Philipp Koehn. 2012. Simulating human judgment in machine translation evaluation campaigns. In *Proceedings of the International Workshop on Spoken Language Translation (IWSLT)*, pages 179–184, Hong Kong.

Bernhard Kölmel and Spiros Alexakis. 2002. Location based advertising. In *Proceedings of the First International Conference on Mobile Business, Athens, Greece*.

Matouš Macháček and Ondřej Bojar. 2015. Evaluating machine translation quality using short segments annotations. *The Prague Bulletin of Mathematical Linguistics*, No. 103:85–110, April.

Shachar Mirkin and Laurant Besacier. 2014. Data selection for compact adapted SMT models. In *Proceedings of the eleventh biennial conference of the Association for Machine Translation in the Americas (AMTA-2014)*, Vancouver, Canada, Oct.

Shachar Mirkin, Scott Nowson, Caroline Brun, and Julien Perez. 2015. Motivating personality-aware machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Lisbon, Portugal. Association for Computational Linguistics.

Laurent Nepveu, P Langlais, G Lapalme, and George Foster. 2004. Adaptive language and translation models for interactive machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 190–197, Barcelona, Spain.

Leonardo Neumeyer, Ananth Sankar, and Vassilios Digalakis. 1995. A comparative study of speaker adaptation techniques. In *Fourth European Conference on Speech Communication and Technology, EUROSPEECH 1995, Madrid, Spain, September 18-21, 1995*.

Karl Pearson. 1895. Note on regression and inheritance in the case of two parents. *Proceedings of the Royal Society of London*, 58(347-352):240–242.

Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. 1994. Grouplens: an open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM conference on Computer supported cooperative work*, pages 175–186. ACM.

Francesco Ricci, Lior Rokach, and Bracha Shapira. 2011. Introduction to recommender systems handbook.

Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295. ACM.

Lucia Specia, Marco Turchi, Nicola Cancedda, Marc Dymetman, and Nello Cristianini. 2009. Estimating the sentence-level quality of machine translation systems. In *13th Conference of the European Association for Machine Translation*, pages 28–37.

Mirco Speretta and Susan Gauch. 2005. Personalized search based on user search histories. In *Web Intelligence, 2005. Proceedings. The 2005 IEEE/WIC/ACM International Conference on*, pages 622–628. IEEE.

Marco Turchi, Matteo Negri, and Marcello Federico. 2013. Coping with the subjectivity of human judgements in MT quality estimation. In *Proceedings of the Eighth Workshop on Statistical Machine Translation (WMT)*, pages 240–251, Sofia, Bulgaria, August. Association for Computational Linguistics.

Frank Wilcoxon. 1945. Individual Comparisons by Ranking Methods. *Biometrics Bulletin*, 1(6):80–83, December.

Talking to the crowd: What do people react to in online discussions?

Aaron Jaech, Vicky Zayats, Hao Fang, Mari Ostendorf and Hannaneh Hajishirzi
Dept. of Electrical Engineering
University of Washington
{ajaech,vzayats,hfang,ostendor,hannaneh}@uw.edu

Abstract

This paper addresses the question of how language use affects community reaction to comments in online discussion forums, and the relative importance of the message vs. the messenger. A new comment ranking task is proposed based on community annotated karma in Reddit discussions, which controls for topic and timing of comments. Experimental work with discussion threads from six subreddits shows that the importance of different types of language features varies with the community of interest.

1 Introduction

Online discussion forums are a popular platform for people to share their views about current events and learn about issues of concern to them. Discussion forums tend to specialize on different topics, and people participating in them form communities of interest. The reaction of people within a community to comments posted provides an indication of community endorsement of opinions and value of information. In most discussions, the vast majority of comments spawn little reaction. In this paper, we look at whether (and how) language use affects the reaction, compared to the relative importance of the author and timing of the post.

Early work on factors that appear to influence crowd-based judgments of comments in the Slashdot forum (Lampe and Resnick, 2004) indicate that timing, starting score, length of the comment, and poster anonymity/reputation appear to play a role (where anonymity has a negative effect). Judging by differences in popularity of various discussion forums, topic is clearly important. Evidence that language use also matters is provided by recent work (Danescu-Niculescu-Mizil et al., 2012; Lakkaraju et al., 2013; Althoff et al., 2014; Tan et al., 2014). Teasing these different factors apart, however, is a challenge. The work presented in this paper provides additional insight into this question by controlling for these factors in a different way than previous work and by examining multiple communities of interest. Specifically, using data from Reddit discussion forums, we look at the role of author reputation as measured in terms of a karma k-index, and control for topic and timing by ranking comments in a constrained window within a discussion.

The primary contributions of this work include findings about the role of author reputation and variation across communities in terms of aspects of language use that matter, as well as the problem formulation, associated data collection, and development of a variety of features for characterizing informativeness, community response, relevance and mood.

2 Data

Reddit[1] is the largest public online discussion forum with a wide variety of subreddits, which makes it a good data source for studying how textual content in a discussion impacts the response of the crowd. On Reddit, people initiate a discussion thread with a post (a question, a link to a news item, etc.), and others respond with comments. Registered users vote on which posts and comments are important. The total amount of up votes minus the down votes (roughly) is called *karma*; it provides an indication of community endorsement and popularity of a comment, as used in (Lakkaraju et al., 2013). Karma is valued as it impacts the order in which the posts or comments are displayed, with the high karma content rising to the top. Karma points are also accumulated by members of the discussion forum as a function of the karma associated with their comments.

[1] http://www.reddit.com

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 2026–2031,
Lisbon, Portugal, 17-21 September 2015.

subreddit	# Posts	# Comments/Post
FITNESS	3K	16.3
ASKSCIENCE	4K	8.8
POLITICS	7K	23.7
ASKWOMEN	4K	50.5
ASKMEN	4K	58.3
WORLDNEWS	12K	26.1

Table 1: Data collection statistics.

	Top1	Top3
ASKSCIENCE	9.3	25.9
FITNESS	1.4	12.3
POLITICS	0.3	7.4
ASKWOMEN	2.2	13.5
ASKMEN	3.9	11.9
WORLDNEWS	3.1	6.4

Table 2: Percentage of discussions where the top comment is made by the top k-index person (or top 3 people) in the discussion.

The Reddit data is highly skewed. Although there are thousands of active communities, only a handful of them are large. Similarly, out of the more than a million comments made per day[2], most of them receive little to no attention; the distributions of positive comment karma and author karma are Zipfian. Slightly more than half of all comments have exactly one karma point (no votes beyond the author), and only 5% of comments have less than one karma point.

For this study, we downloaded all the posts and associated comments made to six subreddits over a few weeks, as summarized in Table 1, as well as karma of participants in the discussion[3]. All available comments on each post were downloaded at least 48 hours after the post was made.[4]

3 Uptake Factors

Factors other than the language use that influence whether a comment will have uptake from the community include the topic, the timing of the message, and the messenger. These factors are all evident in the Reddit discussions. Some subreddits are more popular and thus have higher karma comments than others, reflecting the influence of topic. Comments that are posted early in the discussion are more likely to have high karma, since they have more potential responses.

Previous studies on Twitter show that the reputation of the author substantially increases the chances of the retweet (Suh et al., 2010; Cha et al., 2010), and reputation is also raised as a factor in Slashdot (Lampe and Resnick, 2004). On Reddit most users are anonymous, but it is possible that members of a forum become familiar with particular usernames associated with high karma comments. In order to see how important personal reputation is, we looked at how often the top karma comments are associated with the top karma participants in the discussion. Since an individual's karma can be skewed by a few very popular posts, we measure reputation instead using a measure we call the *k-index*, defined to be equal to the number of comments in each user's history that have karma $\geq k$. The k-index is analgous to the h-index (Hirsch, 2005) and arguably a better indicator of extended impact than total karma.

The results in Table 2 address the question of whether the top karma comments always come from the top karma person. The Top1 column shows the percentage of threads where the top karma comment in a discussion happens to be made by the highest k-index person participating in the discussion; the next column shows the percentage of threads where the comment comes from any one of the top 3 k-index people. We find that, in fact, the highest karma comment in a discussion is rarely from the highest k-index people. The highest percentage is in ASKSCIENCE, where expertise is more highly valued. If we consider whether any one of the multiple comments that the top k-index person made is the top karma comment in the discussion, then the frequency is even lower.

4 Methods

4.1 Tasks

Having shown that the reputation of the author of a post is not a dominating factor in predicting high karma comments, we propose to control for topic and timing by ranking a set of 10 comments that were made consecutively in a short window of time within one discussion thread according to the karma they finally received. The ranking has access to the comment history about these posts. This simulates the view of an early reader of these posts, i.e., without influence of the ratings of oth-

[2]http://www.redditblog.com/2014/12/reddit-in-2014.html

[3]Our data collection is available online at https://ssli.ee.washington.edu/tial/data/reddit

[4]Based on our initial look at the data, we noticed that most posts receive all of their comments within 48 hours. Some comments are deleted before we are able to download them.

ers, so that the language content of the post is more likely to have an impact. Very long threads are sampled, so that these do not dominate the set of lists. Approximately 75% of the comment lists are designated for training and the rest is for testing, with splits at the discussion thread level. Here, feature selection is based on mean precision of the top-ranked comment (P@1), so as to emphasize learning the rare high karma events. (Note that P@1 is equivalent to accuracy but allows for any top-ranking comment to count as correct in the case of ties.) The system performance is evaluated using both P@1 and normalized discounted cumulative gain (NDCG) (Burges et al., 2005), which is a standard criterion for ranking evaluation when the samples to be ranked have meaningful differences in scores, as is the case for karma of the comments.

In addition, for analysis purposes, we report results for three surrogate tasks that can be used in the ranking problem: i) the binary ranker trained on all comment pairs within each list, in which low karma comments dominate, ii) a positive vs. negative karma classifier, and iii) a high vs. medium karma classifier. All use class-balanced data; the second two are trained and tested on a biased sampling of the data, where the pairs need not be from the same discussion thread.

4.2 Classifier

We use the support vector machine (SVM) rank algorithm (Joachims, 2002) to predict a rank order for each list of comments. The SVM is trained to predict which of a pair of comments has higher karma. The error term penalty parameter is tuned to maximize P@1 on a held-out validation set (20% of the training samples).

Since much of the data includes low-karma comments, there will be a tendancy for the learning to emphasize features that discriminate comments at the lower end of the scale. In order to learn features that improve P@1, and to understand the relative importance of different features, we use a greedy automatic feature selection process that incrementally adds one feature whose resulting feature set achives the highest P@1 on the validation set. Once all features have been used, we select the model with the subset of features that obtains the best P@1 on the validation set.

4.3 Features

The features are designed to capture several key attributes that we hypothesize are predictive of comment karma motivated by related work. The features are categorized in groups as summarized below, with details in supplementary material.

- **Graph and Timing (G&T):** A baseline that captures discourse history (response structure) and comment timing, but no text content.
- **Authority and Reputation (A&R):** K-index, whether the commenter was the original poster, and in some subreddits "flair" (display next to a comment author's username that is subject to a cursory verification by moderators).
- **Informativeness (Info.):** Different indicators suggestive of informative content and novelty, including various word counts, named entity counts, urls, and unseen n-grams.
- **Lexical Unigrams (Lex.):** Miscellaneous word class indicators, puncutation, and part-of-speech counts
- **Predicted Community Response (Resp.):** Probability scores from surrogate classification tasks (reply vs. no reply, positive vs. negative sentiment) to measure the community response of a comment using bag-of-words predictors.
- **Relevance (Rel.):** Comment similarity to the parent, post and title in terms of topic, computed with three methods: i) a distributed vector representation of topic using a non-negative matrix factorization (NMF) model (Xu et al., 2003), ii) the average of skip-gram word embeddings (Mikolov et al., 2013), and iii) word set Jaccard similarity (Strehl et al., 2000).
- **Mood:** Mean and std. deviation of sentence sentiment in the comment; word list indicators for politeness, argumentativeness and profanity.
- **Community Style (Comm.):** Posterior probability of each subreddit given the comment using a bag-of-words model.

The various word lists are motivated by feature exploration studies in surrogate tasks. For example, projecting words to a two dimensional space of positive vs. negative and likelihood of reply showed that self-oriented pronouns were more likely to have no response and second-person pronouns were more likely to have a negative response. The politeness and argumentativeness/profanity lists are generated by starting with hand-specified seed lists used to train an SVM to classify word embeddings (Mikolov et al., 2013)

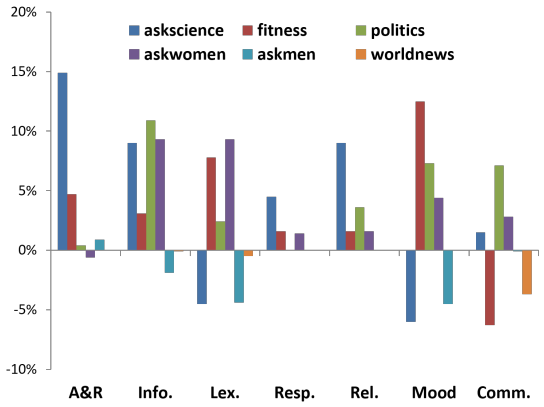


Figure 1: Relative improvement in P@1 over G&T for individual feature groups.

into these categories, and expanding the lists with 500 words farthest from the decision boundary.

Both the NMF and the skip-gram topic models use a cosine distance to determine topic similarity, with 300 as the word embedding dimension. Both are trained on approximately 2 million comments in high karma posts taken across a wide variety of subreddits. We use topic models in various measures of comment relevance to the discussion, but we do not use topic of the comment on its own since topic is controlled for by ranking within a thread.

5 Ranking Experiments

We present three sets of experiments on comment karma ranking, all of which show very different behavior for the different subreddits. Fig. 1 shows the relative gain in P@1 over the G&T baseline associated with using different feature groups. The importance of the different features reflect the nature of the different communities. The authority/reputation features help most for ASKSCIENCE, consistent with our k-index study. Informativeness and relevance help all subreddits except ASKMEN and WORLDNEWS. Lexical, mood and community style features are useful in some cases, but hurt others. The predicted probability of a reply was least useful, possibly because of the low-karma training bias.

Tables 3 and 4 summarize the results for the P@1 and NDCG criteria using the greedy selection procedure (which optimizes P@1) compared to a random baseline and the G&T baseline. The random baseline for P@1 is greater than 10% because of ties. The G&T baseline results show that the graph and timing features alone obtain 21-32%

subreddit	Random	G&T	All
ASKSCIENCE	15.9	21.8	**25.3**
FITNESS	19.4	22.1	**27.3**
POLITICS	18.5	24.7	**26.4**
ASKWOMEN	17.6	24.9	**28.0**
ASKMEN	18.2	**31.4**	29.1
WORLDNEWS	15.4	**24.5**	23.3
Improvement	-	42.9%	52.1%

Table 3: Test set precision of top one prediction (P@1) performance for specific subreddits.

subreddit	Random	G&T	All
ASKSCIENCE	0.53	**0.60**	**0.60**
FITNESS	0.57	0.61	**0.62**
POLITICS	0.55	0.61	**0.62**
ASKWOMEN	0.56	0.62	**0.65**
ASKMEN	0.56	**0.66**	**0.66**
WORLDNEWS	0.54	**0.61**	0.60
Improvement	-	12.5%	13.2%

Table 4: Test set ranking NDCG performance for specific subreddits.

of top karma comments depending on subreddits. Adding the textual features gives an improvement in P@1 performance over the G&T baseline for all subreddits except ASKMEN and WORLDNEWS. The trends for performance measured with NDCG are similar, but the benefit from textual features is smaller. The results in both tables show different ways of reporting performance of the same system, but the system has been optimized for P@1 in terms of feature selection. In initial exploratory experiments, this seems to have a small impact: when optimizing for NDCG in feature selection we obtain 0.61 vs. 0.60 with the P@1-optimized features.

A major challenge with identifying high karma comments (and negative karma comments) is that

subreddit	Pos/Neg	High/Mid	Ranking
ASKSCIENCE	44.5	63.7	61.3
FITNESS	74.7	43.9	57.5
POLITICS	95.5	59.1	58.0
ASKWOMEN	82.5	67.6	59.7
ASKMEN	87.0	66.2	60.6
WORLDNEWS	93.3	69.9	57.3
Average	79.6	61.7	59.1

Table 5: Accuracy of binary classifiers trained on balanced data to distinguish: positive vs. negative karma (Pos/Neg), high vs. mid-level karma (High/Mid), and ranking between any pair (Ranking).

they are so rare. Although our feature selection tunes for high rank precision, it is possible that the low-karma data dominate the learning. Alternatively, it may be that language cues are mainly useful for identifying distinguishing the negative or mid-level karma comments, and that the very high karma comments are a matter of timing. To better understand the role of language for these different types, we trained classifiers on balanced data for positive vs. negative karma and high vs. mid levels of karma. For these models, the training pairs could come from different threads, but topic is controlled for in that all topic features are relative (similarity to original post, parent, etc.). We compared the results to the binary classifier used in ranking, where all pairs are considered. In all three cases, random chance accuracy is 50%.

Table 5 shows the pairwise accuracy of these classifiers. We find that distinguishing positive from negative classes is fairly easy, with the notable exception of the more information-oriented subreddit ASKSCIENCE. Averaging across the different subreddits, the high vs. mid task is slightly easier than the general ranking task, but the variation across subreddits is substantial. The high vs. mid distinction for FITNESS falls below chance (likely overtraining), whereas it seems to be an easier task for the ASKWOMEN, ASKMEN, and WORLDNEWS.

6 Related Work

Interest in social media is rapidly growing in recent years, which includes work on predicting the popularity of posts, comments and tweets. Danescu-Niculescu-Mizil et al. (2012) investigate phrase memorability in the movie quotes. Cheng et al. (2014) explore prediction of information cascades on Facebook. Weninger et al. (2013) analyze the hierarchy of the Reddit discussions, topic shifts, and popularity of the comment, using among the others very simple language analysis. Lampos et al. (2014) study the problem of predicting a Twitter user impact score (determined by combining the numbers of user's followers, followees, and listings) using text-based and non-textual features, showing that performance improves when user participation in particular topics is included.

Most relevant to this paper are studies of the effect of language in popularity predictions. Tan et al. (2014) study how word choice affects the popularity of Twitter messages. As in our work, they control for topic, but they also control for the popularity of the message authors. On Reddit, we find that celebrity status is less important than it is on Twitter since on Reddit almost everyone is anonymous. Lakkaraju et al. (2013) study how timing and language affect the popularity of posting images on Reddit. They control for content by only making comparisons between reposts of the same image. Our focus is on studying comments within a discussion instead of standalone posts, and we analyze a vast majority of language features. Althoff et al. (2014) use deeper language analysis on Reddit to predict the success of receiving a pizza in the Random Acts of Pizza subreddit. To our knowledge, this is the first work on ranking comments in terms of community endorsement.

7 Conclusion

This paper addresses the problem of how language affects the reaction of community in Reddit comments. We collect a new dataset of six subredit discussion forums. We introduce a new task of ranking comments based on karma in Reddit discussions, which controls for topic and timing of comments. Our results show that using language features improve the comment ranking task in most of the subreddits. Informativeness and relevance are the most broadly useful feature categories; reputation matters for ASKSCIENCE, and other categories could either help or hurt depending on the community. Future work involves improving the classification algorithm by using new approaches to learning about rare events.

References

Tim Althoff, Cristian Danescu-Niculescu-Mizil, and Dan Jurafsky. 2014. How to ask for a favor: A case study on the success of altruistic requests. In *Proc. ICWSM*.

Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. 2005. Learning to rank using gradient descent. In *Proceedings of the International Conference on Machine Learning*, pages 89–96.

Meeyoung Cha, Hamed Haddadi, Fabricio Benevenuto, and P Krishna Gummadi. 2010. Measuring user influence in twitter: The million follower fallacy. *ICWSM*, 10(10-17):30.

Justin Cheng, Lada Adamic, P Alex Dow, Jon Michael Kleinberg, and Jure Leskovec. 2014. Can cascades be predicted? In *Proc. WWW*.

Cristian Danescu-Niculescu-Mizil, Justin Cheng, Jon Kleinberg, and Lillian Lee. 2012. You had me at hello: How phrasing affects memorability. In *Proc. ACL*.

Jorge E Hirsch. 2005. An index to quantify an individual's scientific research output. *Proceedings of the National Academy of Sciences of the United States of America*, 102(46):16569–16572.

Thorsten Joachims. 2002. Optimizing search engines using clickthrough data. In *Proc. SIGKDD*.

Himabindu Lakkaraju, Julian J McAuley, and Jure Leskovec. 2013. What's in a name? understanding the interplay between titles, content, and communities in social media. In *Proc. ICWSM*.

Cliff Lampe and Paul Resnick. 2004. Slash(dot) and burn: distributed moderation in a large online conversation space. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 543–550.

Vasileios Lampos, Nikolaos Aletras, Daniel Preotiuc-Pietro, and Trevor Cohn. 2014. Predicting and characterizing user impact on Twitter. In *Proceedings of the Conference of the European Chapter of the ACL*, pages 405–413.

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

Alexander Strehl, Joydeep Ghosh, and Raymond Mooney. 2000. Impact of similarity measures on web-page clustering. In *Workshop on Artificial Intelligence for Web Search*.

Bongwon Suh, Lichan Hong, Peter Pirolli, and Ed H Chi. 2010. Want to be retweeted? Large scale analytics on factors impacting retweet in Twitter network. In *Proc. SocialCom*, pages 177–184. IEEE.

Chenhao Tan, Lillian Lee, and Bo Pang. 2014. The effect of wording on message propagation: Topic-and author-controlled natural experiments on Twitter. In *Proc. ACL*.

Tim Weninger, Xihao Avi Zhu, and Jiawei Han. 2013. An exploration of discussion threads in social news sites: A case study of the reddit community. In *Proc. ASONAM*.

Wei Xu, Xin Liu, and Yihong Gong. 2003. Document clustering based on non-negative matrix factorization. In *Proc. SIGIR*.

What Your Username Says About You

Aaron Jaech and Mari Ostendorf
Dept. of Electrical Engineering
University of Washington
{ajaech,ostendor}@uw.edu

Abstract

Usernames are ubiquitous on the Internet, and they are often suggestive of user demographics. This work looks at the degree to which gender and language can be inferred from a username alone by making use of unsupervised morphology induction to decompose usernames into sub-units. Experimental results on the two tasks demonstrate the effectiveness of the proposed morphological features compared to a character n-gram baseline.

1 Introduction

There is much interest in automatic recognition of demographic information of Internet users to improve the quality of online interactions. Researchers have looked into identifying a variety of factors about users, including age, gender, language, religious beliefs and political views. Most work leverages multiple sources of information, such as search query history, Twitter feeds, Facebook likes, social network links, and user profiles. However, in many situations, little of this information is available. Conversely, usernames are almost always available.

In this work, we look specifically at classifying gender and language based only on the username. Prior work by sociologists has established a link between usernames and gender (Cornetto and Nowak, 2006), and studies have linked usernames to other attributes, such as individual beliefs (Crabill, 2007; Hassa, 2012) and shown how usernames shape perceptions of gender and ethnicity in the absence of common nonverbal cues (Pelletier, 2014). The connections to ethnicity motivate the exploration of language identification.

Gender identification based on given names is very effective for English (Liu and Ruths, 2013), since many names are strongly associated with a particular gender, like "Emily" or "Mark". Unfortunately, the requirement that each username be unique precludes use of given names alone. Instead, usernames are typically a combination of component words, names and numbers. For example, the Twitter name @taylorswift13 might decompose into "taylor", "swift" and "13". The sub-units carry meaning and, importantly, they are shared with many other individuals. Thus, our approach is to leverage automatic decomposition of usernames into sub-units for use in classification.

We use the Morfessor algorithm (Creutz and Lagus, 2006; Virpioja et al., 2013) for unsupervised morphology induction to learn the decomposition of the usernames into sub-units. Morfessor has been used successfully in a variety of language modeling frameworks applied to a number of languages, particularly for learning concatenative morphological structure. The usernames that we analyze are a good match to the Morfessor framework, which allows us to push the boundary of how much can be done with only a username.

The classifier design is described in the next section, followed by a description of experiments on gender and language recognition that demonstrate the utility of morph-based features compared to character n-gram features. The paper closes with a discussion of related work and a summary of key findings.

2 General Classification Approach

2.1 Unsupervised Morphology Learning

In linguistics, a morpheme is the "minimal linguistic unit with lexical or grammatical meaning" (Booij, 2012). Morphemes are combined in various ways to create longer words. Similarly, usernames are frequently made up of a concatenated sequence of smaller units. These sub-units will be referred to as u-morphs to highlight the fact that they play an analogous role to morphemes but for

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 2032–2037,
Lisbon, Portugal, 17-21 September 2015.

purposes of encoding usernames rather than standard words in a language. The u-morphs are subunits that are small enough to be shared across different usernames but retain some meaning.

Unsupervised morphology induction using Morfessor (Creutz and Lagus, 2006) is based on a minimum description length (MDL) objective, which balances two competing goals: maximizing both the likelihood of the data and of the model. The likelihood of the data is maximized by longer tokens and a bigger lexicon whereas the likelihood of the model is maximized by a smaller lexicon with shorter tokens. A parameter controls the trade-off between the two parts of the objective function, which alters the average u-morph length. We tune this parameter on held-out data to optimize the classification performance of the demographic tasks.

Maximizing the Morfessor objective exactly is computationally intractable. The Morfessor algorithm searches for the optimal lexicon using an iterative approach. First, the highest probability decomposition for each training token is found given the current model. Then, the model is updated with the counts of the u-morphs. A u-morph is added to the lexicon when it increases the weighted likelihood of the data by more than the cost of increasing the size of the lexicon.

Usernames can be mixed-case, e.g. "JohnDoe". The case change gives information about a likely u-morph boundary, but at the cost of doubling the size of the character set. To more effectively leverage this cue, all characters are made lowercase but each change from lower to uppercase is marked with a special token, e.g. "john$doe". Using this encoding reduces the u-morph inventory size, and we found it to give slightly better results in language identification.

Character 3-grams and 4-grams are used as baseline features. Before extracting the n-grams a "#" token is placed at the start and end of each username. The n-grams are overlapping to give them the best chance of finding a semantically meaningful sub-unit.

2.2 Classifier Design

Given a decomposition of the username into a sequence of u-morphs (or character n-grams), we represent the relationship between the observed features and each class with a unigram language model. If a username u has decomposition $m_1, \ldots, m_n$ then it is assigned to the class c_i for which the unigram model gives it the highest posterior probability, or equivalently:

$$\text{argmax}_i \; p_C(c_i) \prod_{k=1}^{n} p(m_k|c_i),$$

where $p_C(c_i)$ is the class prior and $p(m_k|c_i)$ is the class-dependent unigram.[1]

For some demographics, the class prior can be very skewed, as in the case of language detection where English is the dominant language. The choice of smoothing algorithm can be important in such cases, since minority classes have much less training data for estimating the language model and benefit from having more probability mass assigned to unseen words. Here, we follow the approach proposed in (Frank and Bouckaert, 2006) that normalizes the token count vectors for each class to have the same L_1 norm, specifically:

$$p(m_k|c_i) = \frac{1}{Z}\left(1 + \frac{\beta \cdot n(m_k, c_i)}{n(c_i)}\right),$$

where $n(\cdot)$ indicates counts and β controls the strength of the smoothing. Setting β equal to the number of training examples approximately matches the strength of the smoothing to the add-one-smoothing algorithm. $Z = \beta + |M|$ is a constant to make the probabilities sum to one.

Only a small portion of usernames on the Internet come with gender labels. In these situations, semi-supervised learning algorithms can use the unlabeled data to improve the performance of the classifier. We use a self-training expectation-maximization (EM) algorithm similar to that described in (Nigam et al., 2000). The algorithm first learns a classifier on the labeled data. In the E-step, the classifier assigns probabilistic labels to the unlabeled data. In the M-step, the labeled data and the probabilistic labels are combined to learn a new classifier. These steps are iterated until convergence, which usually requires three iterations for our tasks.

[1] Note that the unigram model used here, which considers only the observed u-morphs or n-grams, is not the same as using a Naive Bayes (NB) classifier based on a vector of u-morph counts. In the former, unobserved u-morphs do not impact the class-dependent probability, whereas the zero counts do impact the probability for the NB classifier. Since the vast majority of possible u-morphs are unobserved in a username, it is better to base the decision only on the observed u-morphs. The n-gram model is actually a unigram with an n-gram "vocabulary" rather than an n-gram language model.

Nigam et al. (2000) call their method EM-λ because it uses a parameter λ to reduce the weight of the unlabeled examples relative to the labeled data. This is important because the independence assumptions of the unigram model lead to over-confident predictions. We used another method that directly corrects the estimated posterior probabilities. Using a small validation set, we binned the probability estimates and calculated the true class probability for each bin. The EM algorithm used the corrected probabilities for each bin for the unlabeled data during the maximization step. Samples with a prediction confidence of less than 60% are not used for training.

3 Experiments

3.1 Gender Identification

Data was collected from the OkCupid dating site by downloading up to 1,000 profiles from 27 cities in the United States, first for men seeking women and again for women seeking men to obtain a balanced set of 44,000 usernames. The data is partitioned into three sets with 80% assigned to training and 10% each to validation and test. We also use 3.5M usernames from the photo messaging app Snapchat (McCormick, 2014): 1.5M are used for u-morph learning and 2M are for self-training. All names in this task used only lower case, due to the nature of the available data.

	Male	**Female**
u-morph	guy, mike, matt, josh	girl, marie, lady, miss
trigram	guy, uy#, kev, joe	irl, gir, grl, emm

Table 1: Top Gender Identification Features

The top features ranked by likelihood ratios are given in Table 1. The u-morphs clearly carry semantic meaning, and the trigram features appear to be substrings of the top u-morph features. The trigram features have an advantage when the u-morphs are under-segmented such as if the u-morph "niceguy" or "thatguy" is included in the lexicon. Conversely, the n-grams can suffer from over-segmentation. For example, the trigram "guy" is inside the surname "Nguyen" even though it is better to ignore that substring in this context. Many other tokens suffer from this problem, e.g. "miss" is in "mission".

The variable-length u-morphs are longer on average than the character n-grams (4.9 characters). The u-morph inventory size is similar to that for 3-grams but 5-10 times smaller than the 4-gram inventory, depending on the amount of data used since the inventory is expanded in semi-supervised training. By using the MDL criterion in unsupervised morphology learning, the u-morphs provide a more efficient representation of usernames than n-grams and make it easier to control the trade-off between vocabulary size and average segment length. The smaller inventory is less sensitive to sparse data in language model training.

	Error Rate	
Features	Supervised	Self-Training
3-gram	28.7%	32.0%
4-gram	28.7%	29.4%
u-morph	27.8%	25.8%

Table 2: Gender Classification Results

The experiment results are presented in Table 2. For the supervised learning method, the character 3-gram and 4-gram features give equivalent performance, and the u-morph features give the lowest error rate by a small amount (3% relative). More significantly, the character n-gram systems do not benefit from semi-supervised learning, but the u-morph features do. The semi-supervised u-morph features obtain an error rate of 25.8%, which represents a 10% relative reduction over the baseline character n-gram results.

3.2 Language Identification on Twitter

This experiment takes usernames from the Twitter streaming API. Each username is associated with a tweet, for which the Twitter API identifies a language. The language labels are noisy, so we remove approximately 35% of the tweets where the Twitter API does not agree with the langid.py classifier (Lui and Baldwin, 2012). Both training and test sets are restricted to the nine languages that comprise at least 1% of the training set. These languages cover 96% of the observed tweets (see Table 4). About 110,000 usernames were reserved for testing and 430,000 were used for training both u-morphs and the classifier. Semi-supervised methods are not used because of the abundant labeled data. For each language, we train a one-vs.-all classifier. The mixed case encoding technique (see sec. 2.1) gives a small increase (0.5%) in the

Model	Prec.	Recall	F1
4-gram	.67	.75	.70
u-morph	.77	.67	.71
Combination	.73	.73	.73

Table 3: Precision, recall and F1 score for language identification using the 4-gram, u-morph representations and a combination system, averaging over all users.

accuracy of the model and reduces the u-morph model size by 5%.

The results in Tables 3 and 4 contrast systems using 4-grams, u-morphs, and a combination model, showing precision-recall trade-offs for all users together and F1 scores broken down by specific languages, respectively. The combination system simply uses the average of the posterior log-probabilities for each class giving equal weight to each model. While the overall F1 scores are similar for the 4-gram and u-morph systems, their precision and recall trade-offs are quite different, making them effective in combination. The 4-gram system has higher recall, and the u-morph system has higher precision. With the combination, we obtain a substantial gain in precision over the 4-gram system with a modest loss in recall, resulting in a 3% absolute improvement in average F1 score.

Looking at performance on the different languages, we find that the F1 score for the combination model is higher than the 4-gram for every language, with precision always improving. For the dominant languages, the difference in recall is negligible. The infrequent languages have a 4-8% drop in recall, but the gains in precision are substantial for these languages, ranging from 50-100% relative. The greatest contrast between the 4-gram and the combination system can be seen for the least frequent languages, i.e. the languages with the least amount of training data. In particular, for French, the precision of the combination system (0.36) is double that of the 4-gram model (0.18) with only a 34% loss in recall (0.24 to 0.16).

Looking at the most important features from the classifier highlights the ability of the morphemes to capture relevant meaning. The presence of the morpheme "juan", "jose" or "flor" increase the probability of a Spanish language tweet by five times. The same is true for Portuguese and the morpheme "bieber". The morpheme "q8" increases the odds of an Arabic language tweet by thirteen times due to its phonetic similarity to the name of the Arabic speaking country Kuwait. Other features may simply reflect cultural norms. For example, having an underscore in the username makes it five percent less likely to observe an English tweet. These highly discriminative morphemes are both long and short. It is hard for the fixed-length n-grams to capture this information as well as the morphemes do.

Language	Freq	4-gr	u-m	Comb
English	43.5	.78	.78	.79
Japanese	21.6	.75	.76	.77
Spanish	15.1	.66	.65	.68
Arabic	7.9	.66	.65	.68
Portuguese	6.2	.50	.55	.58
Russian	1.8	.40	.58	.45
Turkish	1.8	.59	.36	.65
French	1.1	.18	.13	.22
Indonesian	1.1	.34	.30	.43

Table 4: Language identification performance (F1 Scores) and relative frequency in the corpus for 4-gram (4-gr) and u-morph (u-m) representations and the combination system (Comb).

4 Related Work

Of the many studies on automatic classification of online user demographics, few have leveraged names or usernames at all, and the few that do mainly explore their use in combination with other features. The work presented here differs in its use of usernames alone, but more importantly in the introduction of morphological analysis to handle a large number of usernames.

Two studies on gender recognition are particularly relevant. Burger *et al.* (2011) use the Twitter username (or screen name) in combination with other profile and text features to predict gender, but they also look at the use of username features alone. The results are not directly comparable to ours, because of differences in the data set used (150k Twitter users) and the classifier framework (Winnow), but the character n-gram performance is similar to ours (21-22% different from the majority baseline). The study uses over 400k character n-grams (n=1-5) for screen names alone; our study indicatess that the u-morphs can reduce this number by a factor of 10. Burger *et al.* (2011) used the same strategy with the self-identified full

name of the user as entered into their profile, obtaining 89% gender recognition (vs. 77% for screen names). Later, Liu and Ruths (2013) use the full first name from a user's profile for gender detection, finding that for the names that are highly predictive of gender, performance improves by relying on this feature alone. However, more than half of the users have a name that has an unknown gender association. Manual inspection of these cases indicated that the majority includes strings formed like usernames, nicknames or other types of word concatenations. These examples are precisely what the u-morph approach tries to address.

Language identification is an active area of research (Bergsma et al., 2012; Zubiaga et al., 2014), but the username has not been used as a feature. Again, results are difficult to compare due to the lack of a common test set, but it is notable that the average F1 score for the combination model approaches the scores obtained on a similar Twitter language identification task where the algorithm has access to the full text of the tweet (Lui and Baldwin, 2014): 73% vs. 77% .

A study that is potentially relevant to our work is automatic classification of ethnicity of Twitter users, specifically whether a user is African-American (Pennacchiotti and Popescu, 2011). Again, a variety of content, profile and behavioral features are used. Orthographic features of the username are used (e.g. length, number of numeric/alpha characters), and names of users that a person retweets or replies to. The profile name features do not appear to be useful, but examples of related usernames point to the utility of our approach for analysis of names in other fields.

5 Conclusions

In summary, this paper has introduced the use of unsupervised morphological analysis of usernames to extract features (u-morphs) for identifying user demographics, particularly gender and language. The experimental results demonstrate that usernames contain useful personal information, and that the u-morphs provide a more efficient and complementary representation than character n-grams.[2] The result for language identification is particularly remarkable because it comes close to matching the performance achieved by using the full text of a tweet. The work is complementary to other demographic studies in that the username prediction can be used together with other features, both for the user and members of his/her social network.

The methods proposed here could be extended in different directions. The unsupervised morphology learning algorithm could incorporate priors related to capitalization and non-alphabetic characters to better model these phenomena than our simple text normalization approach. More sophisticated classifiers could also be used, such as variable-length n-grams or neural-network-based n-gram language models, as opposed to the unigram model used here. Of course the sophistication of the classifier will be limited by the amount of training data available.

A large amount of data is not necessary to build a high precision username classifier. For example, less than 7,000 training examples were available for Turkish in the language identification experiment and the classifier had a precision of 76%. Since little data is required, there may be many more applications of this type of model.

Prior work on unsupervised morphological induction focused on applying the algorithm to natural language input. By using those techniques with a new type of input, this paper shows that there are other applications of morphology learning.

[2] In order to allow the replicability of the experiments, software and data for building and evaluating our classifiers using pre-trained Morfessor models is available at `http://github.com/ajaech/username_analytics`.

References

Shane Bergsma, Paul McNamee, Mossaab Bagdouri, Clayton Fink, and Theresa Wilson. 2012. Language identification for creating language-specific twitter collections. In *Proc. second workshop on language in social media*, pages 65–74.

G. Booij. 2012. *The Grammar of Words: An Introduction to Linguistic Morphology*. Oxford Textbooks in Linguistics. OUP Oxford.

John D Burger, John Henderson, George Kim, and Guido Zarrella. 2011. Discriminating gender on Twitter. In *Proc. Conference on Empirical Methods in Natural Language Processing*, pages 1301–1309. Association for Computational Linguistics.

Karen M. Cornetto and Kristine L. Nowak. 2006. Utilizing usernames for sex categorization in computer-mediated communication: Examining perceptions and accuracy. *CyberPsychology & Behavior 9.4*, pages 377–387.

Scott L Crabill. 2007. *Comparative content analysis of social identity cues within a white supremacist dis-*

cussion board and a social activist discussion board. Ph.D. thesis, Wayne State University.

Mathias Creutz and Krista Lagus. 2006. Morfessor in the morpho challenge. In *Proc. PASCAL Challenge Workshop on Unsupervised Segmentation of Words into Morphemes*.

Eibe Frank and Remco R Bouckaert. 2006. Naive Bayes for text classification with unbalanced classes. In *Proc. KDD*, pages 503–510.

Samira Hassa. 2012. Projecting, exposing, revealing self in the digital world: Usernames as a social practice in a Moroccan chatroom. *Names*, 60(4):201–209.

Wendy Liu and Derek Ruths. 2013. What's in a name? using first names as features for gender inference in Twitter. In *Proc. AAAI Spring Symposium*, pages 10–16.

Marco Lui and Timothy Baldwin. 2012. langid. py: An off-the-shelf language identification tool. In *Proc. ACL 2012 System Demonstrations*, pages 25–30.

Marco Lui and Timothy Baldwin. 2014. Accurate language identification of Twitter messages. *Proc. 5th Workshop on Language Analysis for Social Media (LASM)@ EACL*, pages 17–25.

Rich McCormick. 2014. 4.6 million snapchat phone numbers and usernames leaked, January.

Kamal Nigam, Andrew Kachites McCallum, Sebastian Thrun, and Tom Mitchell. 2000. Text classification from labeled and unlabeled documents using EM. *Machine learning*, 39(2-3):103–134.

Laura Pelletier. 2014. You've got mail: Identity perceptions based on email usernames. *Journal of Undergraduate Research at Minnesota State University, Mankato*, 9.1(13).

Marco Pennacchiotti and Ana-Maria Popescu. 2011. A machine learning approach to Twitter user classification. In *Proc. AAAI Conference on Weblogs and Social Media*, pages 281–288.

Sami Virpioja, Peter Smit, Stig-Arne Grönroos, Mikko Kurimo, et al. 2013. Morfessor 2.0: Python implementation and extensions for morfessor baseline.

Arkaitz Zubiaga, Inaki San Vicente, Pablo Gamallo, José Ramom Pichel, Inaki Alegria, Nora Aranberri, Aitzol Ezeiza, and Vıctor Fresno. 2014. Overview of TweetLID: Tweet language identification at SEPLN 2014. *Proceedings of the Tweet Language Identification Workshop*.

Knowledge Base Inference using Bridging Entities

Bhushan Kotnis
Indian Institute of Science
bkotnis@dese.iisc.ernet.in

Pradeep Bansal
Indian Institute of Science
pradeepb@ee.iisc.ernet.in

Partha Talukdar
Indian Institute of Science
ppt@serc.iisc.in

Abstract

Large-scale Knowledge Bases (such as NELL, Yago, Freebase, etc.) are often sparse, i.e., a large number of valid relations between existing entities are missing. Recent research have addressed this problem by augmenting the KB graph with additional edges mined from a large text corpus while keeping the set of nodes fixed, and then using the Path Ranking Algorithm (PRA) to perform KB inference over this augmented graph. In this paper, we extend this line of work by augmenting the KB graph not only with edges, but also with *bridging entities*, where both the edges and bridging entities are mined from a 500 million web text corpus. Through experiments on real-world datasets, we demonstrate the value of bridging entities in improving the performance and running time of PRA in the KB inference task.

1 Introduction

Large-scale knowledge bases (KB) like Freebase (Bollacker et al., 2008), Yago (Suchanek et al., 2007), NELL (Mitchell et al., 2015) can be useful in a variety of applications like natural language question answering, semantic search engines, etc. These knowledge bases consist of millions of real world entities and relationships between them which are stored in the form of a directed graph where links represent relations and nodes represent the entities. Although such KBs contain millions of entities, they are still very sparse, i.e., they are missing a large number of relations between existing entities (West et al., 2014).

Performing inference over the knowledge graph for predicting relations between two entities is one way of densifying the KB graph. For example, from *(Germany, playsinTournament, FIFA)* and *(FIFA, tournamentofSport, Soccer)*, we can infer *(Germany, playsSport, Soccer)*. The Path Ranking Algorithm (PRA) (Lao and Cohen, 2010), (Lao et al., 2011) performs such an inference by learning inference rules over the knowledge graph.

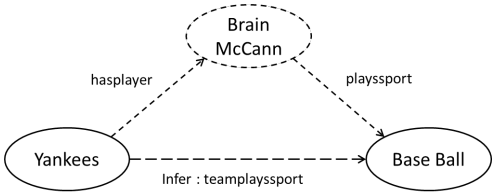


Figure 1: Example showing how addition of the *bridging entity*, *Brian McCain*, and the two edges incident on it can help the PRA algorithm (Lao and Cohen, 2010) to infer the initially missing relation instance *teamPlaysSport(Yankees, BaseBall)*. The original KB graph consisted only of two nodes, *Yankees* and *Baseball*, and no edges.

If the knowledge graph is sparse, i.e., if there are a very few or no paths between source and target entities, then PRA is unable to predict the existence of a relation. To address this shortcoming, (Lao et al., 2012) augmented the knowledge graph with paths obtained from an external corpus. The added paths consisted of unlexicalized dependency labels obtained from a dependency parsed external corpus. To improve the expressivity of the added paths, instead of the unlexicalized labels, (Gardner et al., 2013) augmented the KB graph with verbs (surface relations) from a corpus containing over 600 million Subject-Verb-Object (SVO) triples. These verbs act as edges that connect previously unconnected entities thereby increasing the connectivity of the KB graph which can potentially improve PRA performance.

However, naïvely adding these edges increases the feature sparsity which degrades the discriminative ability of the logistic regression classifier

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 2038–2043,
Lisbon, Portugal, 17-21 September 2015.

used in PRA. This can be addressed by adding latent relations obtained by clustering the surface relations, instead of directly adding the surface relations. This reduces feature sparsity and has been shown to improve PRA inference (Gardner et al., 2013) , (Gardner et al., 2014).

In this article we propose a scheme for augmenting the KB using paths obtained by mining noun phrases that connect two SVO triples from an external corpus. We term these noun phrases as *bridging entities* since they bridge two KB relations to form a path. This is different from the scheme in (Gardner et al., 2013) and (Gardner et al., 2014), which adds edges between KB nodes by mining surface relations from an external corpus. We search for such bridging entities in the corpus by performing a limited depth DFS (depth first search) on the corpus graph in an *on-demand* fashion.

We term this procedure as **On-Demand Augmentation (ODA)**, because the search can be performed during test time in an on-demand manner. In contrast, the previous approaches of adding edges or embeddings to the KB (Gardner et al., 2013), and vector space random walk PRA (Gardner et al., 2014) are batch procedures. As we shall see in Section 4, due to a limited search space, on-demand augmentation is much faster compared to algorithms in (Gardner et al., 2013; Gardner et al., 2014). Furthermore, since edges are not added blindly, on-demand augmentation does not increase feature sparsity which is responsible for performance degradation. Our experiments suggest that ODA provides better performance than (Gardner et al., 2013) and nearly the same prediction performance as provided by (Gardner et al., 2014), but in both cases with the added advantage of faster running time and greater flexibility due to its online and on-demand nature. The code along with the results can be obtained at https://github.com/malllabiisc/pra-oda.

2 Related Work

Using surface level relations and noun phrases for extracting meaningful relational facts is not a new idea (Hearst, 1992),(Brin, 1999), (Etzioni et al., 2004). However, none of them make use of Knowledge Bases for improving information extraction.

The Path Ranking Algorithm (PRA) first proposed in (Lao and Cohen, 2010) was used for performing inference over a KB in (Lao et al., 2011). It was extended by (Lao et al., 2012), to improve the inference by augmenting the KB with syntactic information obtained from a dependency parsed corpus. Augmenting the KB for improving PRA inference using surface relations mined from an external corpus and using latent edge labels obtained by performing PCA on the surface relations was explored in (Gardner et al., 2013). Instead of hard mapping of surface relations to latent embeddings, (Gardner et al., 2014) perform a 'soft' mapping using vector space random walks. This allows the random walker to traverse an edge semantically similar to the current edge type more frequently than other edges.

Although, like others, we too use an external corpus to augment the KB, the crucial difference in our approach is that apart from adding surface relations, we also add bridging entities that enable us to create new paths in the KB. Furthermore, the procedure is targeted so that only paths that play a part in inferring the relations that are of interest are added. Thus, the number of paths added in this manner is much lower than the number of surface relations added using the procedure in (Gardner et al., 2013). As we shall see in Section 4, this results in a more effective algorithm with faster runtime.

3 Method

3.1 Background: Path Ranking Algorithm (PRA)

We first present a brief overview of the Path Ranking Algorithm (PRA) (Lao and Cohen, 2010). The PRA uses paths as features for a logistic regression classifier which predicts if the given relation exists between a pair of entities. For a given pair of entities s and t, the *path type* connecting s to t form the feature vector. A *path types* π is an ordered set of relations. Paths with the same ordered relations but different intermediate or terminal entities belong to the same path type. For example, $s_1 \xrightarrow{v_0} x_1 \xrightarrow{v_1} t_1$ and $s_2 \xrightarrow{v_0} x_2 \xrightarrow{v_1} t_2$ belong to path type $\xrightarrow{v_0}\xrightarrow{v_1}$. The value of a feature, is taken to be $P(s \rightarrow t; \pi)$, where $P(s \rightarrow t; \pi)$ is the probability of reaching t from s by traversing paths of type π. PRA approximates these probabilities by running a random walk (RW) on the KB graph. Let $F = \{\pi_1, \pi_2, ..., \pi_k\}$ be the set of all path types. For predicting the existence of relation r between entities s and t, the logistic regression classifier outputs a score which is a measure of the

Query	Candidate Answer	Path added by PRA-ODA with bridging entity (in bold)
sportsteamPositionForSport(right handed pitcher, ?)	baseball	right handed pitcher $\overset{playsfor}{\longrightarrow}$ **Chicago Cubs** $\overset{play}{\longrightarrow}$ baseball
riverFlowsThroughCity(Moselle, ?)	Koblenz	Moselle $\overset{flows\ into}{\longrightarrow}$ **Rhine** $\overset{meet\ at}{\longrightarrow}$ Koblenz
teamPlaysInLeague(Cleveland Indians, ?)	MLB	Cleveland Indians $\overset{play}{\longrightarrow}$ **Detroit Tigers** $\overset{blew}{\longrightarrow}$ MLB

Table 1: Examples of paths involving bridging entities (marked in bold) added to the KB by PRA-ODA.

confidence that r exists between s and t. It does so by first assigning weights to the features in the training phase. The score is given by

$$S(s,t,r) = \sum_{\pi \in F} P(s \rightarrow t; \pi) \times \theta_{\pi}^{r} \qquad (1)$$

where θ_{π}^{r} is the weight learned by the logistic regression classifier during training specially for relation r and path type π. During the test phase, since targets are not available, the PRA gathers candidate targets by performing a random walk and then computes feature vectors and the score.

3.2 PRA-SVO and PRA-VS

PRA-SVO and PRA-VS are the systems proposed in (Gardner et al., 2013) and (Gardner et al., 2014) respectively, where the KB graph is augmented with edges mined from a large subject-verb-object (SVO) triple corpus. In these two systems, only new edges are added over the fixed set of nodes, and the augmentation happens in a batch, offline setting. In contrast, PRA-ODA, the method proposed in the paper, also expands the set of nodes through bridging entities, and performs the augmentation in an on-demand manner.

3.3 PRA On-Demand Augmentation (PRA-ODA)

Training: Let s and t be any two KB entities and let $s^{(n)}$ and $t^{(n)}$ be their corresponding noun phrase representations or aliases. We search for bridging entities $x_1, x_2, ..x_n$ by performing limited depth first search (DFS) starting with s^n such that we obtain a path $s \overset{ALIAS}{\longrightarrow} s^{(n)} \overset{v_0}{\longrightarrow} x_1 \overset{v_1}{\longrightarrow} ... \overset{v_{n-1}}{\longrightarrow} x_n \overset{v_n}{\longrightarrow} t^{(n)} \overset{ALIAS}{\longrightarrow} t$, where v_i are verbs present in the corpus graph. This is done for all $n \leq d_{max} - 1$, where d_{max} is the maximum depth of DFS. We add an 'ALIAS' edge between the KB entity and its noun phase representation. The usefulness of bridging entities is illustrated in Fig. 1.

We mine bridging entities from a corpus containing over 600 million SVO triples which were obtained from the ClueWeb09 corpus (Callan et al., 2009) parsed using the MALT parser (Nivre et al., 2007). We use Mongo DB to store the triples as an adjacency list. During training time, for any relation that is being inferred, both the source and its corresponding target entities are known. A limited depth DFS is performed for *all depths* less then d_{max} on the SVO graph with the aliases of subject entity acting as the starting points. Such aliases are available for the NELL and Freebase knowledge bases. The DFS is said to discover a path if the terminating entity of the path matches any alias of the target entity. We choose to use aliases to perform string match, since it is easy to change the softness of the match by simply adding more aliases. This is done for all training source-target pairs. A few examples of added paths are shown in Table 1.

The SVO graph is noisy since it is obtained by parsing the ClueWeb corpus which was obtained by scraping the web. To reduce noise, we add the top K most frequent discovered SVO path types, where K is a tunable parameter. By SVO path type we refer to a set of ordered verbs mined from the SVO corpus. There is a possibility that the bridging entities, extracted from the corpus, may be present in the KB. If the bridging entity matches any alias, then it is treated as an alias to an existing KB entity. If not, then the bridging entity is added to the KB as a new entity. To avoid overfitting we add negative data to the training set. Furthermore, only high quality expressive bridging entities result in meaningful and discriminative paths. Although the quality of bridging entities depend on the corpus, low quality bridging entities can be filtered out by adding negative training data. Low quality bridging entities connect source target pairs from both positive and negative training sets, and hence are eliminated by the sparse logistic regression classifier. The negative dataset is generated using the closed world assumption by performing a random walk.

After augmenting the KB, we run the training phase of the PRA algorithm to obtain the feature (path) weights computed by the logistic regression

KB Relations	PRA	PRA-SVO	PRA-VS	PRA-ODA
actorstarredinmovie	0.0	**1.0**	**1.0**	**1.0**
atheleteplaysforteam	1.0	1.0	1.0	1.0
citylocatedincountry	0.166	0.25	**1.0**	**1.0**
journalistwritesfor publication	1.0	1.0	1.0	1.0
riverflowsthroughcity	0.333	0.25	**1.0**	**1.0**
sportsteamposition forsport	1.0	1.0	1.0	1.0
stadiumlocatedincity	1.0	1.0	1.0	1.0
statehaslake	0.0	0.0	0.0	0.0
teamplaysinleague	1.0	1.0	1.0	1.0
writerwrotebook	1.0	1.0	1.0	1.0
Average (MRR)	0.649	0.75	**0.9**	**0.9**

Table 2: Comparison of Mean Reciprocal Rank (MRR) metric for 10 relations from NELL (higher is better). PRA-SVO, PRA-VS are the systems proposed in (Gardner et al., 2013; Gardner et al., 2014). PRA-ODA is the approach proposed in this paper. Improvements in PRA-ODA over PRA-SVO is statistically significant with $p < 0.007$, with PRA-SVO as null hypothesis.

classifier.

Query Time: The set of target entities corresponding to a source entity and the relation being predicted is not available during query (test) time. We use all the entities included in the range of the relation being predicted as candidate target entities. For example, if the relation is *riverFlowsThroughCity*, the candidate target set would include entities in the KB that are *cities*. The DFS is now performed starting from source entities as during training, but this time only restricting to paths with positive weights learned during training. Any path (along with bridging entities) found during this search are added to the KB, and the PRA algorithm is now run over this augmented graph.

4 Experiments

We used the implementation of PRA provided by the authors of (Gardner et al., 2014). For our experiments, we used the same 10 NELL relation data as used in (Gardner et al., 2014). The augmentation resulted in the addition of 1086 paths during training and 1430 paths during test time.

We split the NELL data into 60% training data, 15 % development data and 25% test data. Values for d_{max}, and K, the most frequent paths, were obtained by tuning on a development set for 4 relations (athleteplaysforsport,actorstarredinmovie,citylocatedincountry and journalistwritesforpublication). The hyperparameter values $d_{max} = 2$, $K = 10$ reported the highest MRR and were used for the rest of the relations. For the L_1 and L_2 regularization parameters in the logistic regression classifier, we used the same values as used in (Gardner et al., 2013; Gardner et al., 2014), viz., $L_1 = 0.005$, and $L_2 = 1.0$. This is because the parameters were reported to be robust, and seemed to work well even when the knowledge base was augmented.

Timings (seconds)	PRA	PRA-SVO	PRA-VS	PRA-ODA
Training	635.6	574.5	564.2	913.3
Test	354.3	322.0	301.2	436.7
Batch augmentation	n/a	797	797	n/a
Embedding computation	n/a	n/a	812	n/a
Total Time	989.9	1693.5	2474.4	1350

Table 3: Runtime comparison for the entire experiment (lower is better). PRA-SVO, PRA-VS are the systems proposed in (Gardner et al., 2013; Gardner et al., 2014). PRA-ODA is the approach proposed in this paper. Between the two top performing systems, i.e., PRA-ODA and PRA-VS, PRA-ODA is faster by a factor of 1.8.

We compare the results (PRA-ODA) with the PRA algorithm executed on the NELL KB, NELL KB augmented with surface relations (PRA-SVO) (Gardner et al., 2013) and vector space random walk PRA (PRA-VS) (Gardner et al., 2014). The run times, i.e, the time taken to perform an entire experiment for PRA-SVO and PRA-VS includes the time taken to augment NELL KB with SVO edges. The PRA-VS runtime also includes the time taken for generating embeddings to perform the vector space random walk. As can be seen from Table 2 and Table 3, our scheme, PRA-ODA, provides performance equivalent to PRA-VS with faster running time (speed up of 1.8). In addition to the time taken for the full SVO augmentation, PRA-VS takes additional time to generate embeddings (13 minutes) from the added verbs. We note that the batch augmentation in case of PRA-SVO and PRA-VS, and embedding computation in case of PRA-VS are all specific to the relations in the evaluation set, and hence can't be ignored as a one-time offline cost. In other words, these costs are likely to increase as more relations (and their instances) are included during training and testing. Runtime gains with PRA-ODA are likely to be even more pronounced in such settings.

An additional advantage of the proposed algorithm is that it can also be run on the top of any PRA based algorithm such as the PRA-SVO and PRA-VS.

5 Conclusion

In this paper, we investigated the usefulness of adding paths to a Knowledge Base for improving its connectivity by mining *bridging entities* from an external corpus. While previous KB augmentation methods focused only on augmentation using mined surface verbs while keeping the node set fixed, we extended these approaches by also adding *bridging entities* in an online fashion. We used a large corpus of 500 million web text corpus to mine these additional edges and bridging entities. Through experiments on real-world datasets, we demonstrate that the proposed approach is not only comparable or better than other state-of-the-art baselines, but more importantly provides faster overall runtime compared with the alternatives.

Acknowledgment

This work is supported in part by a gift from Google.

References

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, pages 1247–1250, New York, NY, USA. ACM.

Sergey Brin. 1999. Extracting patterns and relations from the world wide web. In Paolo Atzeni, Alberto Mendelzon, and Giansalvatore Mecca, editors, *The World Wide Web and Databases*, volume 1590 of *Lecture Notes in Computer Science*, pages 172–183. Springer Berlin Heidelberg.

J. Callan, M. Hoy, C. Yoo, and L. Zhao. 2009. Clueweb09 data set. boston.lti.cs.cmu.edu.

Oren Etzioni, Michael Cafarella, Doug Downey, Stanley Kok, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates. 2004. Web-scale information extraction in knowitall: (preliminary results). In *Proceedings of the 13th International Conference on World Wide Web*, WWW '04, pages 100–110, New York, NY, USA. ACM.

Matt Gardner, Partha Pratim Talukdar, Bryan Kisiel, and Tom Mitchell. 2013. Improving learning and inference in a large knowledge-base using latent syntactic cues. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 833–838.

Matt Gardner, Partha Pratim Talukdar, Jayant Krishnamurthy, and Tom Mitchell. 2014. Incorporating vector space similarity in random walk inference over knowledge bases. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 397–406.

Marti A. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th Conference on Computational Linguistics - Volume 2*, COLING '92, pages 539–545, Stroudsburg, PA, USA. Association for Computational Linguistics.

Ni Lao and William W. Cohen. 2010. Relational retrieval using a combination ofpath constrained random walks. *Machine Learning*, 81(1):53–67.

Ni Lao, Tom Mitchell, and William W. Cohen. 2011. Random walk inference and learning in a large scale knowledge base. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 529–539, Stroudsburg, PA, USA. Association for Computational Linguistics.

Ni Lao, Amarnag Subramanya, Fernando Pereira, and William W. Cohen. 2012. Reading the web with learned syntactic-semantic inference rules. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 1017–1026, Stroudsburg, PA, USA. Association for Computational Linguistics.

T. Mitchell, W. Cohen, E. Hruschka, P. Talukdar, J. Betteridge, A. Carlson, B. Dalvi, M. Gardner, B. Kisiel, J. Krishnamurthy, N. Lao, K. Mazaitis, T. Mohamed, N. Nakashole, E. Platanios, A. Ritter, M. Samadi, B. Settles, R. Wang, D. Wijaya, A. Gupta, X. Chen, A. Saparov, M. Greaves, and J. Welling. 2015. Never-ending learning. In *Proceedings of AAAI*.

Joakim Nivre, Johan Hall, Jens Nilsson, Atanas Chanev, Gülsen Eryigit, Sandra Kübler, Svetoslav Marinov, and Erwin Marsi. 2007. Maltparser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(02):95–135.

Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: A core of semantic knowledge. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, pages 697–706, New York, NY, USA. ACM.

Robert West, Evgeniy Gabrilovich, Kevin Murphy, Shaohua Sun, Rahul Gupta, and Dekang Lin. 2014. Knowledge base completion via search-based question answering. In *Proceedings of the 23rd International Conference on World Wide Web*, WWW '14, pages 515–526, New York, NY, USA. ACM.

Specializing Word Embeddings for Similarity or Relatedness

Douwe Kiela, Felix Hill and Stephen Clark
Computer Laboratory
University of Cambridge
douwe.kiela|felix.hill|stephen.clark@cl.cam.ac.uk

Abstract

We demonstrate the advantage of specializing semantic word embeddings for either similarity or relatedness. We compare two variants of retrofitting and a joint-learning approach, and find that all three yield specialized semantic spaces that capture human intuitions regarding similarity and relatedness better than unspecialized spaces. We also show that using specialized spaces in NLP tasks and applications leads to clear improvements, for document classification and synonym selection, which rely on either similarity or relatedness but not both.

1 Introduction

Most current models of semantic word representation exploit the *distributional hypothesis*: the idea that words occurring in similar contexts have similar meanings (Harris, 1954; Turney and Pantel, 2010; Clark, 2015). Such representations (or embeddings) can reflect human intuitions about similarity and relatedness (Turney, 2006; Agirre et al., 2009), and have been applied to a wide variety of NLP tasks, including bilingual lexicon induction (Mikolov et al., 2013b), sentiment analysis (Socher et al., 2013) and named entity recognition (Turian et al., 2010; Guo et al., 2014).

Arguably, one of the reasons behind the popularity of word embeddings is that they are "general purpose": they can be used in a variety of tasks without modification. Although this behavior is sometimes desirable, it may in other cases be detrimental to downstream performance. For example, when classifying documents by topic, we are particularly interested in related words rather than similar ones: knowing that *dog* is associated with *cat* is much more informative of the topic than knowing that it is a synonym of *canine*. Conversely, if our embeddings indicate that *table* is closely related to *chair*, that does not mean we should translate *table* into French as *chaise*.

This distinction between "genuine" similarity and associative similarity (i.e., relatedness) is well-known in cognitive science (Tversky, 1977). In NLP, however, semantic spaces are generally evaluated on how well they capture *both* similarity and relatedness, even though, for many word combinations (such as *car* and *petrol*), these two objectives are mutually incompatible (Hill et al., 2014b). In part, this oversight stems from the distributional hypothesis itself: *car* and *petrol* do not have the same, or even very similar, meanings, but these two words may well occur in similar contexts. Corpus-driven approaches based on the distributional hypothesis therefore generally learn embeddings that capture both similarity and relatedness reasonably well, but neither perfectly.

In this work we demonstrate the advantage of specializing semantic spaces for either similarity or relatedness. Specializing for similarity is achieved by learning from both a corpus and a thesaurus, and for relatedness by learning from both a corpus and a collection of psychological association norms. We also compare the recently-introduced technique of graph-based retrofitting (Faruqui et al., 2015) with a skip-gram retrofitting and a skip-gram joint-learning approach. All three methods yield specialized semantic spaces that capture human intuitions regarding similarity and relatedness significantly better than unspecialized spaces, in one case yielding state-of-the-art results for word similarity. More importantly, we show clear improvements in downstream tasks and applications: specialized similarity spaces improve synonym detection, while association spaces work better than both general-purpose and similarity-specialized spaces for document classification.

2 Approach

The underlying assumption of our approach is that, during training, word embeddings can be "nudged" in a particular direction by including information from an additional semantic data

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 2044–2048,
Lisbon, Portugal, 17-21 September 2015.

source. For directing embeddings towards genuine similarity, we use the MyThes thesaurus developed by the OpenOffice.org project[1]. It contains synonyms for almost 80,000 words in English. For directing embeddings towards relatedness, we use the University of South Florida (USF) free association norms (Nelson et al., 2004). This dataset contains scores for free association (an experimental measure of cognitive association) of over 10,000 concept words. For raw text data we use a dump of the English Wikipedia plus newswire text (8 billion words in total)[2].

2.1 Evaluations (Intrinsic and Extrinsic)

For instrinsic comparisons with human judgements, we evaluate on SimLex (Hill et al., 2014b) (999 pairwise comparisons), which explicitly measures similarity, and MEN (Bruni et al., 2014) (3000 comparisons), which explicitly measures relatedness. We also consider two downstream tasks and applications. In the TOEFL synonym selection task (Landauer and Dumais, 1997), the objective is to select the correct synonym for a target word from a multiple-choice set of possible answers. For a more extrinsic evaluation, we use a document classification task based on the Reuters Corpus Volume 1 (RCV1) (Lewis et al., 2004). This dataset consists of over 800,000 manually categorized news articles.[3]

2.2 Joint Learning

The standard skip-gram training objective for a sequence of training words $w_1, w_2, ..., w_T$ and a context size c is the log-likelihood criterion:

$$\frac{1}{T}\sum_{t=1}^{T} J_\theta(w_t) = \frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c} \log p(w_{t+j}|w_t)$$

where $p(w_{t+j}|w_t)$ is obtained via the softmax:

$$p(w_{t+j}|w_t) = \frac{\exp^{u_{w_{t+j}}^\top v_{w_t}}}{\sum_{w'}\exp^{u_{w'}^\top v_{w_t}}}$$

where u_w and v_w are the context and target vector representations for word w, respectively, and w' ranges over the full vocabulary (Mikolov et al., 2013a). For our *joint learning* approach, we supplement the skip-gram objective with additional contexts (synonyms or free-associates) from an external data source. In the **sampling** condition, for target word w_t, we modify the objective to include an additional context w^a sampled uniformly from the set of additional contexts A_{w_t}:

$$\frac{1}{T}\sum_{t=1}^{T}\left(J_\theta(w_t) + [w^a \sim \mathcal{U}_{A_{w_t}}]\log p(w^a|w_t)\right)$$

In the **all** condition, all additional contexts for a target word are added at each occurrence:

$$\frac{1}{T}\sum_{t=1}^{T}\left(J_\theta(w_t) + \sum_{w^a\in A_{w_t}} \log p(w^a|w_t)\right)$$

The set of additional contexts A_{w_t} contains the relevant contexts for a word w_t; e.g., for the word *dog*, A_{dog} for the thesaurus is the set of all synonyms of dog in the thesaurus.

2.3 Retrofitting

Faruqui et al. (2015) introduced retrofitting as a post-hoc graph-based learning objective that improves learned word embeddings. We experiment with their method, calling it *graph-based retrofitting*. In addition, we introduce a similar approach that instead uses the same objective function that was used to learn the original skip-gram embeddings. In other words, we first train a standard skip-gram model, and then learn from the additional contexts in a second training stage as if they form a separate corpus:

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{w^a\in A_{w_t}} \log p(w^a|w_t)$$

We call this approach *skip-gram retrofitting*. In all cases, our embeddings have 300 dimensions, which has been found to work well (Mikolov et al., 2013a; Baroni et al., 2014)

3 Results for Intrinsic Evaluation

We compare standard skip-gram embeddings with retrofitted and jointly learned specialized embeddings, as well as with "fitted" embeddings that were randomly initialized and learned only from the additional semantic resource. In each case, the

[1] https://www.openoffice.org/lingucomponent/thesaurus.html

[2] The script for obtaining this corpus is available from http://word2vec.googlecode.com/svn/trunk/demo-train-big-model-v1.sh

[3] We exclude articles with multiple topic labels in order to avoid multi-class document classification. The dataset contains a total of 78 topic labels and 33,226 news articles.

Method	SimLex-999	MEN
Skip-gram	0.31	0.68
Fit-Norms	0.08	0.14
Fit-Thesaurus	0.26	0.14
Joint-Norms-Sampled	0.43	**0.72**
Joint-Norms-All	0.42	0.67
Joint-Thesaurus-Sampled	0.38	0.69
Joint-Thesaurus-All	0.44	0.60
GB-Retrofit-Norms	0.32	0.71
GB-Retrofit-Thesaurus	0.38	0.68
SG-Retrofit-Norms	0.35	0.71
SG-Retrofit-Thesaurus	**0.47**	0.69

Table 1: Spearman ρ on a genuine similarity (SimLex-999) and relatedness (MEN) dataset.

training algorithm was run for a single iteration (results from more iterations are presented later).

As shown in Table 1, embeddings that were specialized for similarity using a thesaurus perform better on SimLex-999, and those specialized for relatedness using association data perform better on MEN. Fitting, or learning only from the additional semantic resource without access to raw text, does not perform well. Skip-gram retrofitting with the thesaurus performs best on SimLex-999; joint learning with sampling from the USF norms performs best on MEN, although the two retrofitting approaches are close. There is an interesting difference between the two joint learning approaches: while **sampling** a single free associate as additional context works best for relatedness, presenting **all** additional contexts (synonyms) works best for similarity. In both cases, skip-gram retrofitting matches or outperforms graph-based retrofitting.

More training iterations All the results above were obtained using a single training iteration. When retrofitting, however, it is easy to learn from multiple iterations of the thesaurus or the USF norms. The results are shown in Figure 1, where the dashed lines are the joint learning and standard skip-gram results for comparison with retrofitting scores. As would be expected, too many iterations leads to overfitting on the semantic resource, with performance eventually decreasing after the initial increase. The results show that retrofitting is particularly useful for similarity, as indicated by the large increase in performance on SimLex-999. The highest performance obtained, at 5 iterations, is a Spearman ρ_s correlation of 0.53, which, as far as we know, matches the current state-of-the-art.[4]

For relatedness-specific embeddings, the effect is less clear: joint learning performs comparatively much better. Retrofitting does outperform it, at around 2-10 iterations on the USF norms, but the improvement is marginal. The highest retrofitting score is 0.74; the highest joint learning score is 0.72. Both are highly competitive results on MEN, and outperform e.g. GloVe at 0.71 (Pennington et al., 2014). Joint learning with a thesaurus, however, leads to poor performance on MEN, as expected: the embeddings get dragged away from relatedness and towards similarity.

3.1 Curriculum learning?

The fact that joint learning works better when supplementing raw text input with free associates, but skip-gram retrofitting works better with additional thesaurus information, could be due to *curriculum learning* effects (Bengio et al., 2009). Unlike the USF norms, many of the words from the thesaurus are unusual and have low frequency. This suggests that the thesaurus is more 'advanced' (from the perspective of the learning model) than the USF norms as an information source. Its information may be detrimental to model optimization when encountered early in training (in the joint learning condition) because the model has not acquired the basic concepts on which it builds. However, with retrofitting the model first acquires good representations for frequent words from the raw text, after which it can better understand, and learn from, the information in the thesaurus.

4 Downstream Tasks and Applications

4.1 TOEFL Synonym Task

Unsupervised synonym selection has many applications including the generation of thesauri and other lexical resources from raw text (Kageura et al., 2000). In the well-known TOEFL evaluation (Freitag et al., 2005) models are required to identify true synonyms to question words from a selection of possible answers. To test our models on this task, for each question in the dataset, we rank the multiple-choice answers according to the cosine similarity between their word embeddings and that of the target word, and choose the highest-ranked option.

[4]Hill et al. (2014a) obtain a score of 0.52 using neural translation embeddings.

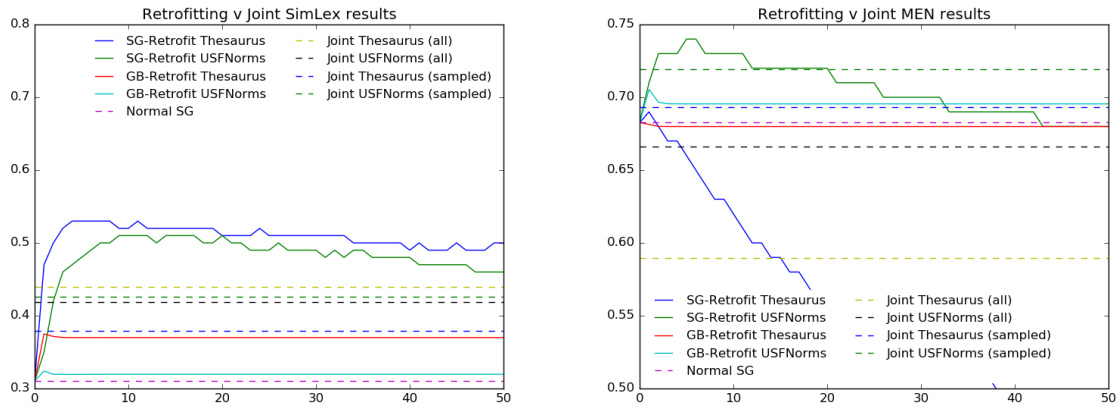


Figure 1: Varying the number of iterations when retrofitting

Method	TOEFL		Doc
Skip-gram	77.50		83.96
Joint-Norms-Sampled	78.75		84.46
Joint-Norms-All	66.25		**84.82**
Joint-Thesaurus-Sampled	81.25		83.90
Joint-Thesaurus-All	80.00		83.56
GB-Retrofit-Norms	80.00		80.58
GB-Retrofit-Thesaurus	83.75		80.24
SG-Retrofit-Norms	80.00		84.56
SG-Retrofit-Thesaurus	**88.75**		84.55

Table 2: TOEFL synonym selection and document classification accuracy (percentage of correctly answered questions/correctly classified documents).

As Table 2 shows, similarity-specialized embeddings perform much better than standard embeddings and relatedness-specialized embeddings. Retrofitting outperforms joint learning, and skip-gram retrofitting matches or outperforms graph-based retrofitting.

4.2 Document Classification

To investigate how well the various semantic spaces perform on document classification, we first construct document-level representations by summing the vector representations for all words in a given document. After setting aside a small development set for tuning the hyperparameters of the supervised algorithm, we train a support vector machine (SVM) classifier with a linear kernel and evaluate document topic classification accuracy using ten-fold cross-validation.

The results are reported in the rightmost column of Table 2. Relatedness-specialized embeddings perform better on document topic classification than similarity embeddings, except with graph-based retrofitting, which in fact performs below the standard skip-gram model. The joint-learning model with all relevant free association norms presented as context for each target word is the best performing model. The differences in the table appear small, but the dataset contains more than 10,000 documents, so every percentage point is worth more than 100 documents. Joint learning while presenting all relevant association norms for each target word performs best on this task.

5 Conclusion

We have demonstrated the advantage of specializing embeddings for the tasks of genuine similarity and relatedness. In doing so, we compared two retrofitting methods and a joint learning approach. Specialized embeddings outperform standard embeddings by a large margin on instrinsic similarity and relatedness evaluations. We showed that the difference in how embeddings are specialized carries to downstream NLP tasks, demonstrating that similarity embeddings are better at the TOEFL synonym selection task and relatedness embeddings at a document topic classification task. Lastly, we varied the number of iterations that we use for retrofitting, showing that performance could be improved even further by going over several iterations of the semantic resource.

Acknowledgments

DK is supported by EPSRC grant EP/I037512/1. FH is supported by St Johns College, Cambridge. SC is supported by ERC Starting Grant DisCoTex

(306920) and EPSRC grant EP/I037512/1. We thank Yoshua Bengio, Kyunghyun Cho and Ivan Vulić for useful discussions and the anonymous reviewers for their helpful comments.

References

Eneko Agirre, Enrique Alfonseca, Keith B. Hall, Jana Kravalova, Marius Pasca, and Aitor Soroa. 2009. A study on similarity and relatedness using distributional and WordNet-based approaches. In *NAACL*, pages 19–27.

Marco Baroni, Georgiana Dinu, and Germán Kruszewski. 2014. Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors. In *ACL*, pages 238–247.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48. ACM.

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

Stephen Clark. 2015. Vector Space Models of Lexical Meaning. In Shalom Lappin and Chris Fox, editors, *Handbook of Contemporary Semantics*, chapter 16. Wiley-Blackwell, Oxford.

Manaal Faruqui, Jesse Dodge, Sujay K Jauhar, Chris Dyer, Eduard Hovy, and Noah A Smith. 2015. Retrofitting word vectors to semantic lexicons. In *Proceedings of NAACL*.

Dayne Freitag, Matthias Blume, John Byrnes, Edmond Chow, Sadik Kapadia, Richard Rohwer, and Zhiqiang Wang. 2005. New experiments in distributional representations of synonymy. In *Proceedings of the Ninth Conference on Computational Natural Language Learning*, pages 25–32.

Jiang Guo, Wanxiang Che, Haifeng Wang, and Ting Liu. 2014. Revisiting embedding features for simple semi-supervised learning. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 110–120.

Zelig Harris. 1954. Distributional Structure. *Word*, 10(23):146—162.

Felix Hill, Kyunghyun Cho, Sébastien Jean, Coline Devin, and Yoshua Bengio. 2014a. Embedding word similarity with neural machine translation. *CoRR*, abs/1412.6448.

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

Kyo Kageura, Keita Tsuji, and Akiko N Aizawa. 2000. Automatic thesaurus generation through multiple filtering. In *Proceedings of the 18th conference on Computational linguistics-Volume 1*, pages 397–403.

Thomas K Landauer and Susan T Dumais. 1997. A solution to plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological review*, 104(2):211.

David D Lewis, Yiming Yang, Tony G Rose, and Fan Li. 2004. RCV1: A new benchmark collection for text categorization research. *The Journal of Machine Learning Research*, 5:361–397.

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

Tomas Mikolov, Quoc V Le, and Ilya Sutskever. 2013b. Exploiting similarities among languages for machine translation. In *Proceedings of ICLR*, Scottsdale, Arizona, USA.

Douglas L Nelson, Cathy L McEvoy, and Thomas A Schreiber. 2004. The University of South Florida free association, rhyme, and word fragment norms. *Behavior Research Methods, Instruments, & Computers*, 36(3):402–407.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. *Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014)*, 12:1532–1543.

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

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of ACL*, pages 384–394.

Peter D. Turney and Patrick Pantel. 2010. From Frequency to Meaning: vector space models of semantics. *Journal of Artifical Intelligence Research*, 37(1):141–188, January.

Peter D. Turney. 2006. Similarity of semantic relations. *Computational Linguistics*, 32(3):379–416.

Amos Tversky. 1977. Features of similarity. *Psychological Review*, 84(4).

Evaluation of Word Vector Representations by Subspace Alignment

Yulia Tsvetkov **Manaal Faruqui** **Wang Ling** **Guillaume Lample** **Chris Dyer**
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA, 15213, USA
{ytsvetko, mfaruqui, lingwang, glample, cdyer}@cs.cmu.edu

Abstract

Unsupervisedly learned word vectors have proven to provide exceptionally effective features in many NLP tasks. Most common intrinsic evaluations of vector quality measure correlation with similarity judgments. However, these often correlate poorly with how well the learned representations perform as features in downstream evaluation tasks. We present QVEC—a computationally inexpensive intrinsic evaluation measure of the quality of word embeddings based on alignment to a matrix of features extracted from manually crafted lexical resources—that obtains strong correlation with performance of the vectors in a battery of downstream semantic evaluation tasks.[1]

1 Introduction

A major attraction of vector space word representations is that they can be derived from large unannotated corpora, and they are useful as a source of features for downstream NLP tasks that are learned from small amounts of supervision. Unsupervised word vectors have been shown to benefit parsing (Lazaridou et al., 2013; Bansal et al., 2014), chunking (Turian et al., 2010), named entity recognition (Guo et al., 2014) and sentiment analysis (Socher et al., 2013), among others.

Despite their ubiquity, there is no standard scheme for intrinsically evaluating the quality of word vectors: a vector quality is traditionally judged by its utility in downstream NLP tasks. This lack of standardized evaluation is due, in part, to word vectors' major criticism: word vectors are linguistically opaque in a sense that it is still not clear how to interpret individual vector dimensions, and, consequently, it is not clear how to score a non-interpretable representation. Nevertheless, to facilitate development of better word vector models and for better error analysis of word vectors, it is desirable (1) to compare word vector models easily, without recourse to multiple extrinsic applications whose implementation and runtime can be costly; and (2) to understand how features in word vectors contribute to downstream tasks.

We propose a simple intrinsic evaluation measure for word vectors. Our measure is based on component-wise correlations with manually constructed "linguistic" word vectors whose components have well-defined linguistic properties (§2). Since vectors are typically used to provide features to downstream learning problems, our measure favors *recall* (rather than precision), which captures our intuition that meaningless dimensions in induced vector representations are less harmful than important dimensions that are missing. We thus align dimensions in a distributional word vector model with the linguistic dimension vectors to maximize the cumulative correlation of the aligned dimensions (§3). The resulting sum of correlations of the aligned dimensions is our evaluation score. Since the dimensions in the linguistic vectors are linguistically-informed, the alignment provides an "annotation" of components of the word vector space being evaluated.

To show that our proposed score is meaningful, we compare our intrinsic evaluation model to the standard (semantic) extrinsic evaluation benchmarks (§4). For nine off-the-shelf word vector representation models, our model obtains high correlation ($0.34 \leq r \leq 0.89$) with the extrinsic tasks (§5).

2 Linguistic Dimension Word Vectors

The crux of our evaluation method lies in quantifying the similarity between a distributional word vector model and a (gold-standard) linguistic re-

[1] The evaluation script and linguistic vectors described in this paper are available at
https://github.com/ytsvetko/qvec

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 2049–2054,
Lisbon, Portugal, 17-21 September 2015.

source capturing human knowledge. To evaluate the semantic content of word vectors, we exploit an existing semantic resource—SemCor (Miller et al., 1993). From the SemCor annotations we construct a set of linguistic word vectors, details are given in the rest of this section; table 1 shows an example of the vectors.

WordNet (Fellbaum, 1998, WN) partitions nouns and verbs into coarse semantic categories known as supersenses (Ciaramita and Altun, 2006; Nastase, 2008).[2] There are 41 supersense types: 26 for nouns and 15 for verbs, for example, NOUN.BODY, NOUN.ANIMAL, VERB.CONSUMPTION, or VERB.MOTION. SemCor is a WordNet-annotated corpus that captures, among others, supersense annotations of WordNet's 13,174 noun lemmas and 5,686 verb lemmas at least once. We construct term frequency vectors normalized to probabilities for all nouns and verbs that occur in SemCor at least 5 times. The resulting set of 4,199 linguistic word vectors has 41 interpretable columns.

WORD	NN.ANIMAL	NN.FOOD	...	VB.MOTION
fish	0.68	0.16	...	0.00
duck	0.31	0.00	...	0.69
chicken	0.33	0.67	...	0.00

Table 1: Oracle linguistic word vectors, constructed from a linguistic resource containing semantic annotations.

3 Word Vector Evaluation Model

We align dimensions of distributional word vectors to dimensions (linguistic properties) in the linguistic vectors described in §2 to maximize the cumulative correlation of the aligned dimensions. By projecting linguistic annotations via the alignments, we also obtain plausible annotations of dimensions in the distributional word vectors. In this section, we formally describe the model, which we call the QVEC.

Let the number of common words in the vocabulary of the distributional and linguistic word vectors be N. We define, the distributional vector matrix $\mathbf{X} \in \mathbb{R}^{D \times N}$ with every row as a dimension vector $\mathbf{x} \in \mathbb{R}^{1 \times N}$. D denotes word vector dimensionality. Similarly, $\mathbf{S} \in \mathbb{R}^{P \times N}$ is the linguistic property matrix with every row as a linguistic property vector $\mathbf{s} \in \mathbb{R}^{1 \times N}$. P denotes linguistic properties obtained from a manually-annotated linguistic resource. We obtain an alignment between the word vector dimensions and the linguistic dimensions which maximizes the correlation between the aligned dimensions of the two matrices. This is 1:n alignment: one distributional dimension is aligned to at most one linguistic property, whereas one linguistic property can be aligned to n distributional dimensions; see figure 1.

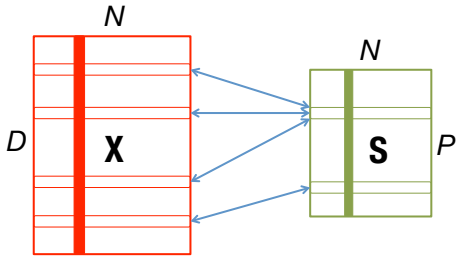


Figure 1: The filled vertical vectors represent the word vector in the word vector matrix $\mathbf{X}$ and the linguistic property matrix $\mathbf{S}$. The horizontal hollow vectors represent the "distributional dimension vector" in $\mathbf{X}$ and "linguistic dimension vector" in $\mathbf{S}$. The arrows show mapping between distributional and linguistic vector dimensions.

Let $\mathbf{A} \in \{0,1\}^{D \times P}$ be a matrix of alignments such that $a_{ij} = 1$ iff $\mathbf{x}_i$ is aligned to $\mathbf{s}_j$, otherwise $a_{ij} = 0$. If $r(\mathbf{x}_i, \mathbf{s}_j)$ is the Pearson's correlation between vectors $\mathbf{x}_i$ and $\mathbf{s}_j$, then our objective is defined as:

$$\text{QVEC} = \max_{\mathbf{A} | \sum_j a_{ij} \leq 1} \sum_{i=1}^{L} \sum_{j=1}^{P} r(\mathbf{x}_i, \mathbf{s}_j) \times a_{ij} \quad (1)$$

The constraint $\sum_j a_{ij} \leq 1$, warrants that one distributional dimension is aligned to at most one linguistic dimension. The total correlation between two matrices QVEC is our intrinsic evaluation measure of a set of word vectors relative to a set of linguistic properties.

The QVEC's underlying hypothesis is that dimensions in distributional vectors correspond to linguistic properties of words. It is motivated, among others, by the effectiveness of word vectors in linear models implying that linear combinations of features (vector dimensions) produce relevant, salient content. Via the alignments a_{ij} we obtain labels on dimensions in the distributional word vectors. The magnitude of the correlation $r(\mathbf{x}_i, \mathbf{s}_j)$ corresponds to the annotation confidence: the higher the correlation, the more salient the linguistic content of the dimension. Clearly, dimensions in the linguistic matrix S do not capture every possible linguistic property, and low correlations often correspond to the missing information in the linguistic matrix. Thus, QVEC is a recall-oriented measure: highly-

[2] Supersenses are known as "lexicographer classes" in WordNet documentation, `http://wordnet.princeton.edu/man/lexnames.5WN.html`

correlated alignments provide evaluation and annotation of vector dimensions, and missing information or noisy dimensions do not significantly affect the score since the correlations are low.

4 Experimental Setup

4.1 Word Vector Models

To test the QVEC, we select a diverse suite of popular/state-of-the-art word vector models. All vectors are trained on 1 billion tokens (213,093 types) of English Wikipedia corpus with vector dimensionality 50, 100, 200, 300, 500, 1000.

CBOW and Skip-Gram (SG). The WORD2VEC tool (Mikolov et al., 2013) is fast and widely-used. In the SG model, each word's Huffman code is used as an input to a log-linear classifier with a continuous projection layer and words within a given context window are predicted. In the CBOW model a word is predicted given the context words.[3]

CWindow and Structured Skip-Gram (SSG). Ling et al. (2015b) propose a syntactic modification to the WORD2VEC models that accounts for word order information, obtaining state-of-the-art performance in syntactic downstream tasks.[4]

CBOW with Attention (Attention). Ling et al. (2015a) further improve the WORD2VEC CBOW model by employing an attention model which finds, within the contextual words, the words that are relevant for each prediction. These vectors have been shown to benefit both semantically and syntactically oriented tasks.

GloVe. Global vectors for word representations (Pennington et al., 2014) are trained on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations show interesting linear substructures of the vector space.[5]

Latent Semantic Analysis (LSA). We construct word-word co-occurrence matrix $\mathbf{X}$; every element in the matrix is the pointwise mutual information between the two words (Church and Hanks, 1990). Then, truncated singular value decomposition is applied to factorize $\mathbf{X}$, where we keep the k largest singular values. Low dimensional word vectors of dimension k are obtained from $\mathbf{U_k}$ where $\mathbf{X} \approx \mathbf{U_k}\Sigma\mathbf{V_k}^\mathsf{T}$ (Landauer and Dumais, 1997).

GloVe+WN, GloVe+PPDB, LSA+WN, LSA+PPDB. We use retrofitting (Faruqui et al., 2015) as a post-processing step to enrich GloVe and LSA vectors with semantic information from WordNet and Paraphrase database (PPDB) (Ganitkevitch et al., 2013).[6]

4.2 Semantic Evaluation Benchmarks

We compare the QVEC to six standard extrinsic semantic tasks for evaluating word vectors; we now briefly describe the tasks.

Word Similarity. We use three different benchmarks to measure word similarity. The first one is the **WS-353** dataset (Finkelstein et al., 2001), which contains 353 pairs of English words that have been assigned similarity ratings by humans. The second is the **MEN** dataset (Bruni et al., 2012) of 3,000 words pairs sampled from words that occur at least 700 times in a large web corpus. The third dataset is **SimLex-999** (Hill et al., 2014) which has been constructed to overcome the shortcomings of WS-353 and contains 999 pairs of adjectives, nouns and verbs. Word similarity is computed using cosine similarity between two words and the performance of word vectors is computed by Spearman's rank correlation between the rankings produced by vector model against the human rankings.[7]

Text Classification. We consider four binary categorization tasks from the 20 Newsgroups **(20NG)** dataset.[8] Each task involves categorizing a document according to two related categories with training/dev/test split in accordance with Yogatama and Smith (2014). For example, a classification task is between two categories of Sports: baseball vs hockey. We report the average classification accuracy across the four tasks. Our next downstream semantic task is the sentiment analysis task **(Senti)** (Socher et al., 2013) which is a binary classification task between positive and negative movie reviews using the standard training/dev/test split and report accuracy on the test set. In both cases, we use the average of the word vectors of words in a document (and sentence, respectively) and use them as features in an ℓ_2-regularized logistic regression classifier. Finally, we evaluate vectors on the metaphor detection **(Metaphor)** (Tsvetkov et al.,

[3] https://code.google.com/p/word2vec

[4] https://github.com/wlin12/wang2vec

[5] http://www-nlp.stanford.edu/projects/glove/

[6] https://github.com/mfaruqui/retrofitting

[7] We employ an implementation of a suite of word similarity tasks at wordvectors.org (Faruqui and Dyer, 2014).

[8] http://qwone.com/~jason/20Newsgroups

2014a).[9] The system uses word vectors as features in a random forest classifier to label adjective-noun pairs as literal/metaphoric. We report the system accuracy in 5-fold cross validation.

5 Results

To test the efficiency of QVEC in capturing the semantic content of word vectors, we evaluate how well QVEC's scores correspond to the scores of word vector models on semantic benchmarks. We compute the Pearson's correlation coefficient r to quantify the linear relationship between the scorings. We begin with comparison of QVEC with one extrinsic task—Senti—evaluating 300-dimensional vectors.

Model	QVEC	Senti
CBOW	40.3	90.0
SG	35.9	80.5
CWindow	28.1	76.2
SSG	40.5	81.2
Attention	40.8	80.1
GloVe	34.4	79.4
GloVe+WN	42.1	79.6
GloVe+PPDB	39.2	79.7
LSA	19.7	76.9
LSA+WN	29.4	77.5
LSA+PPDB	28.4	77.3
Correlation (r)	**0.87**	

Table 2: Intrinsic (QVEC) and extrinsic scores of the 300-dimensional vectors trained using different word vector models and evaluated on the Senti task. Pearson's correlation between the intrinsic and extrinsic scores is $r = 0.87$.

As we show in table 2, the Pearson's correlation between the intrinsic and extrinsic scores is $r = 0.87$. To account for variance in WORD2VEC representations (due to their random initialization and negative sampling strategies, the representations are different for each run of the model), and to compare QVEC to a larger set of vectors, we now train three versions of vector sets per model. This results in 21 word vector sets: three vector sets per five WORD2VEC models plus GloVe, LSA, and retrofitting vectors shown in table 2. The Pearson's correlation computed on the extended set of comparison points (in the same experimental setup as in table 2) is $r = 0.88$. In the rest of this section we report results on the extended suite of word vectors.

We now extend the table 2 results, and show correlations between the QVEC and extrinsic scores across all benchmarks for 300-dimensional vectors. Table 3 summarizes the results. The QVEC obtains high positive correlation with all the semantic tasks.

Table 4 shows, for the same 300-dimensional vectors, that QVEC's correlation with the downstream text classification tasks is on par with or higher than the correlation between the word similarity and text classification tasks. Higher correlating methods—in our experiments, QVEC and MEN—are better predictors of quality in downstream tasks.

	20NG	Metaphor	Senti
WS-353	0.55	0.25	0.46
MEN	**0.76**	0.49	0.55
SimLex	0.56	0.44	0.51
QVEC	0.74	**0.75**	**0.88**

Table 4: Pearson's correlations between word similarity/QVEC scores and the downstream text classification tasks.

Next, we measure correlations of QVEC with the extrinsic tasks across word vector models with different dimensionality. The results are shown in figure 2.

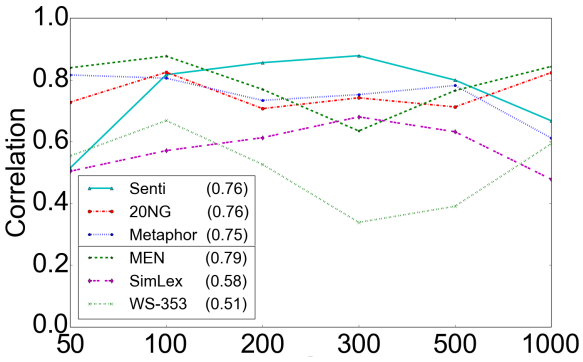


Figure 2: Pearson's correlation between QVEC scores and the semantic benchmarks across word vector models on vectors of different dimensionality. The scores at dimension 300 correspond to the results shown in table 3. The scores in the legend show average correlation across dimensions.

To summarize, we observe high positive correlation between QVEC and the downstream tasks, consistent across the tasks and across different models with vectors of different dimensionalities.

Since QVEC favors recall over precision, larger numbers of dimensions will *ceteris paribus* result in higher scores—but not necessarily higher correlations with downstream tasks. We therefore impose the restriction that QVEC only be used to compare vectors of the same size, but we now show that its correlation with downstream tasks is stable, conditional on the size of the vectors being compared. We aggregate rankings by individual

[9] https://github.com/ytsvetko/metaphor

	WS-353	MEN	SimLex	20NG	Metaphor	Senti
r	0.34	0.63	0.68	0.74	0.75	0.88

Table 3: Pearson's correlations between QVEC scores of the 300-dimensional vectors trained using different word vector models and the scores of the downstream tasks on the same vectors.

	50	100	200	300	500	1000
ρ(QVEC, Senti)	0.32	0.57	0.73	0.78	0.72	0.60
ρ(QVEC, All)	0.66	0.59	0.63	0.65	0.62	0.59

Table 5: Spearman's rank-order correlation between the QVEC ranking of the word vector models and the ranking produced by (1) the Senti task, or (2) the aggregated ranking of all tasks (All). We rank separately models of vectors of different dimensionality (table columns).

downstream tasks into a global ranking using the Kemeny–Young rank aggregation algorithm, for each dimension separately (Kemeny, 1959). The algorithm finds a ranking which minimizes pairwise disagreement of individual rankers. Table 5 shows Spearman's rank correlation between the rankings produced by the QVEC and the Senti task/the aggregated ranking. For example, ranking of 300-dimensional models produced by Senti is *{SSG, CBOW, SG, Attention, GloVe+PPDB, GloVe+WN, GloVe, LSA+WN, LSA+PPDB, LSA, CWindow}*, and the QVEC's ranking is *{GloVe+WN, Attention, SSG, CBOW, GloVe+PPDB, SG, GloVe, LSA+WN, LSA+PPDB, CWindow, LSA}*. The Spearman's ρ between the two rankings is 0.78. We note, however, that there is a considerable variation between rankings across all models and across all dimensions, for example the SimLex ranking produced for the same 300-dimensional vectors is *{GloVe+PPDB, GloVe+WN, SG, LSA+PPDB, SSG, CBOW, Attention, CWindow, LSA+WN, GloVe, LSA}*, and ρ(Senti, SimLex) $= 0.46$. In a recent related study, Schnabel et al. (2015) also observe that existing word similarity and text categorization evaluations yield different orderings of word vector models. This task-specifity of rankings emphasizes the deficiency of evaluating word vector models solely on downstream tasks, and the need of a standardized intrinsic evaluation approach that quantifies linguistic content of word vectors.

6 Future Work

Aligning dimensions of linguistic and distributional vectors enables projection of linguistic annotations via the alignments, and thereby facilitates qualitative analysis of individual dimensions in distributional vectors. Albeit noisy, we find correspondence between the projected labels of distributional columns and the column content. For example, in the 50-dimensional SG model top-10 ranked words in a dimension aligned to NOUN.BODY with r=0.26 are *amputated, sprained, palsy, semenya, lacerations, genital, cervical, concussion, congenital, abdominal*. This interesting by-product of our method will be addressed in future work.

While we experiment with linguistic vectors capturing semantic concepts, our methodology is generally applicable to other linguistic resources (Faruqui and Dyer, 2015). For example, part-of-speech annotations extracted from a treebank would yield linguistic vectors capturing syntactic content of vectors. Thus, QVEC can be used as a task-specific evaluator; we will investigate this in future work.

A useful property of supersenses (features in our linguistic vectors) is that they are stable across languages (Schneider et al., 2013; Tsvetkov et al., 2014b). Cross-lingual vector evaluation and evaluation of multilingual word vectors with QVEC is thus an additional promising research avenue.

7 Conclusion

We propose a method for intrinsic evaluation of word vectors which shows strong relationship—both linear and monotonic—with the scores/rankings produced by the downstream tasks.

Acknowledgments

We are grateful to the anonymous reviewers for constructive feedback. This work was supported by the U.S. Army Research Laboratory and the U.S. Army Research Office under contract/grant number W911NF-10-1-0533.

References

Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2014. Tailoring continuous word representations for dependency parsing. In *Proc. of ACL*.

Elia Bruni, Gemma Boleda, Marco Baroni, and Nam-Khanh Tran. 2012. Distributional semantics in technicolor. In *Proc. of ACL*.

Kenneth Ward Church and Patrick Hanks. 1990. Word association norms, mutual information, and lexicography. *Computational Linguistics*, 16(1):22–29.

Massimiliano Ciaramita and Yasemin Altun. 2006. Broad-coverage sense disambiguation and information extraction with a supersense sequence tagger. In *Proc. of EMNLP*, pages 594–602.

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

Manaal Faruqui and Chris Dyer. 2015. Non-distributional word vector representations. In *Proc. ACL*.

Manaal Faruqui, Jesse Dodge, Sujay Kumar Jauhar, Chris Dyer, Noah A. Smith, and Eduard Hovy. 2015. Retrofitting word vectors to semantic lexicons. In *Proc. of NAACL*.

Christiane Fellbaum, editor. 1998. *WordNet: an electronic lexical database*. MIT Press.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2001. Placing search in context: the concept revisited. In *Proc. of WWW*.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: The paraphrase database. In *Proc. of NAACL*.

Jiang Guo, Wanxiang Che, Haifeng Wang, and Ting Liu. 2014. Revisiting embedding features for simple semi-supervised learning. In *Proc. of EMNLP*.

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

John G. Kemeny. 1959. Mathematics without numbers. 88(4):577–591.

Thomas K Landauer and Susan T. Dumais. 1997. A solution to Plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological review*.

Angeliki Lazaridou, Eva Maria Vecchi, and Marco Baroni. 2013. Fish transporters and miracle homes: How compositional distributional semantics can help NP parsing. In *Proc. of EMNLP*.

Wang Ling, Lin Chu-Cheng, Yulia Tsvetkov, Silvio Amir, Ramon Fermandez, Chris Dyer, Alan W Black, and Isabel Trancoso. 2015a. Not all contexts are created equal: Better word representations with variable attention. In *Proc. of EMNLP*.

Wang Ling, Chris Dyer, Alan Black, and Isabel Trancoso. 2015b. Two/too simple adaptations of `word2vec` for syntax problems. In *Proc. of NAACL*.

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

George A. Miller, Claudia Leacock, Randee Tengi, and Ross T. Bunker. 1993. A semantic concordance. In *Proc. of HLT*, pages 303–308.

Vivi Nastase. 2008. Unsupervised all-words word sense disambiguation with grammatical dependencies. In *Proc. of IJCNLP*, pages 7–12.

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

Tobias Schnabel, Igor Labutov, David Mimno, and Thorsten Joachims. 2015. Evaluation methods for unsupervised word embeddings. In *Proc. of EMNLP*.

Nathan Schneider, Behrang Mohit, Chris Dyer, Kemal Oflazer, and Noah A. Smith. 2013. Supersense tagging for Arabic: the MT-in-the-middle attack. In *Proc. NAACL-HLT*, pages 661–667.

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

Yulia Tsvetkov, Leonid Boytsov, Anatole Gershman, Eric Nyberg, and Chris Dyer. 2014a. Metaphor detection with cross-lingual model transfer. In *Proc. ACL*, pages 248–258.

Yulia Tsvetkov, Nathan Schneider, Dirk Hovy, Archna Bhatia, Manaal Faruqui, and Chris Dyer. 2014b. Augmenting English adjective senses with supersenses. In *Proc. LREC*.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proc. of ACL*.

Dani Yogatama and Noah A Smith. 2014. Linguistic structured sparsity in text categorization. In *Proc. of ACL*.

Higher-order logical inference with compositional semantics

Koji Mineshima[1,3]
mineshima.koji@ocha.ac.jp

Pascual Martínez-Gómez[1,2,3]
pascual@nii.ac.jp

Yusuke Miyao[2]
yusuke@nii.ac.jp

Daisuke Bekki[1,2,3]
bekki@is.ocha.ac.jp

[1]Ochanomizu University
Tokyo, Japan

[2]National Institute of Informatics
Tokyo, Japan

[3]CREST, JST
Saitama, Japan

Abstract

We present a higher-order inference system based on a formal compositional semantics and the wide-coverage CCG parser. We develop an improved method to bridge between the parser and semantic composition. The system is evaluated on the FraCaS test suite. In contrast to the widely held view that higher-order logic is unsuitable for efficient logical inferences, the results show that a system based on a reasonably-sized semantic lexicon and a manageable number of non-first-order axioms enables efficient logical inferences, including those concerned with generalized quantifiers and intensional operators, and outperforms the state-of-the-art first-order inference system.

1 Introduction

Entailment relations are of central importance in the enterprise of both formal and computational semantics. Traditionally, formal semanticists have concentrated on a relatively small set of linguistic inferences. However, since the emergence of statistical parsers based on sophisticated syntactic theories (Clark and Curran, 2007), an open domain system has been developed that supports certain degree of robust semantic interpretation with wide coverage (Bos et al., 2004). It is then reasonable to expect that a state-of-the-art formal semantics provides an accurate computational basis of natural language inferences.

However, there are still obstacles in the way of achieving this goal. One is that the statistical parsers on which semantic interpretations rely do not necessarily reflect the best syntactic analysis as assumed in the formal semantics literature (Honnibal et al., 2010). Another persistent problem is the gap between the logics employed in the two communities; while it is generally assumed among formal semanticists that adequate semantic representations for natural language demand higher-order logic or type theory (Carpenter, 1997), the dominant view in computational linguistics is that inferences based on higher-order logic are hopelessly inefficient for practical applications (Bos, 2009a). Accordingly, it is claimed that some approximation of higher-order representations in terms of first-order logic (Hobbs, 1985), or a more efficient "natural logic" system based on surface structures is needed. However, it is often not a trivial task to give an approximation of rich higher-order information within a first-order language (Pulman, 2007). Moreover, the coverage of existing natural logic systems is limited to single-premise inferences (MacCartney and Manning, 2008).

In this paper, we first present an improved compositional semantics that fills the gap between the parser syntax and a composition derivation. We then develop an inference system that is capable of higher-order inferences in natural languages. We combine a state-of-the-art higher-order proof system (Coq) with a wide-coverage parser based on a modern syntactic theory (Combinatory Categorial Grammar, CCG). The system is designed to handle multi-premise inferences as well as single-premise ones.

We test our system on the FraCaS test suite (Cooper et al., 1994), which is suitable for evaluating the linguistic coverage of an inference system. The experiments show that our higher-order system outperforms the state-of-the-art first-order system with respect to the speed and accuracy of making logical inferences.

2 CCG and Compositional Semantics

As an initial step of compositional semantics, we use the C&C parser (Clark and Curran, 2007), a statistical CCG parser trained on CCGbank (Hockenmaier and Steedman, 2007). Parser out-

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 2055–2061,
Lisbon, Portugal, 17-21 September 2015.

category : $S\backslash NP$
semantics : $\lambda Q.Q(\lambda x.\mathsf{True})(\lambda x.\mathsf{E}(x))$

Figure 1: Schematic lexical entry (semantic template) for intransitive verbs. E is a position in which a particular lexical item appears.

category : NP/N
semantics : $\lambda F\lambda G\lambda H.\forall x(Fx \wedge Gx \rightarrow Hx)$
surf : every

Figure 2: The lexical entry for determiner *every*

puts are mapped onto semantic representations in a standard way (Bos, 2008), using λ-calculus as an interface between syntax and semantics.

The strategy we use to build a semantic lexicon is similar to that of Bos et al. (2004). A lexical entry for each open word class consists of a syntactic category in CCG (possibly with syntactic features) and a semantic representation encoded as a λ-term. Fig. 1 gives an example.[1] For a limited number of closed words such as logical or functional expressions, a λ-term is directly assigned to a surface form (see Fig. 2). The output formula is obtained by combining each λ-term in accordance with meaning composition rules and then by applying β-conversion.

There is a non-trivial gap between the parser output and the standard CCG-syntax as presented in Steedman (2000). Due to this gap, it is not straightforward to obtain desirable semantic representations for a wide range of constructions. One major difference from the standard CCG-syntax is the treatment of post-NP modifiers; for instance, the relative clause *who works* is assigned not the category $N\backslash N$, but the category $NP\backslash NP$, which applies to the whole NP. To derive correct truth-conditions for quantificational sentences, we assign to determiners a semantic term having an extra predicate variable as shown in Fig. 2, namely, $\lambda F\lambda G\lambda H.\forall x(Fx \wedge Gx \rightarrow Hx)$, in a similar way to the continuation semantics for event predicates (Bos, 2009b; Champollion, 2015). The extra predicate variable G can be filled by the semantically empty predicate $\lambda x.\mathsf{True}$ in a verb phrase (see Fig. 1). Fig. 3 gives an example derivation.

Note that the changes in the lexical entries as illustrated in Fig. 1 and Fig. 2 are made for the correct semantic parsing, namely, the compositional

[1]Here we use a non-standard semantics for intransitive verbs. We will explain it in the next paragraph.

Examples	Semantic Types
most	$(\mathsf{E}\rightarrow\mathsf{Prop})\rightarrow(\mathsf{E}\rightarrow\mathsf{Prop})\rightarrow\mathsf{Prop}$
might	$\mathsf{Prop}\rightarrow\mathsf{Prop}$
true	$\mathsf{Prop}\rightarrow\mathsf{Prop}$
manage	$\mathsf{Prop}\rightarrow\mathsf{E}\rightarrow\mathsf{Prop}$
believe	$\mathsf{Prop}\rightarrow\mathsf{E}\rightarrow\mathsf{Prop}$

Table 1: A classification of key linguistic elements having higher-order denotations.

derivation of semantic representations. Usually, inferences are conducted on those output semantic representations in which additional complexities, such as lambda operators and extra predicate variables, disappear. Accordingly, the changes in the lexical entries do not affect the efficiency of inferences.

The present analysis of post NP-modifiers can also handle non-restrictive relative clauses such as "*the president, who ...*". In this case, the modifier "*who ...*" can be taken to apply to the whole NP *the president*, thus its syntactic category can be regarded as $NP\backslash NP$, not as $N\backslash N$. Thus, although the $NP\backslash NP$ analysis of relative clauses is a non-standard one, it has an advantage in that it provides a unified treatment of restrictive and non-restrictive relative clauses.

3 Representation and Inference in HOL

We present a higher-order representation language and describe apparently higher-order phenomena that have received attention in formal semantics.

3.1 Semantic representations in HOL

We use the language of higher-order logic (HOL) with two basic types, E for entities and Prop for propositions. Here we distinguish between propositions and truth-values, as is standard in modern type theory (Ranta, 1994; Luo, 2012). Key higher-order constructs are summarized in Table 1.[2] A first-order language can be taken as a fragment of this language. Thus, adopting a higher-order language does not lead to the loss of the expressive power of the first-order language.

Apart from sub-sentential utterances such as short answers to *wh*-questions (Ginzburg, 2005), there are important constructions that are naturally

[2]We write a function from objects of type A to objects of type B as $A\rightarrow B$. Here $\rightarrow$ is right-associative: $A\rightarrow B\rightarrow C$ means $A\rightarrow(B\rightarrow C)$. We use the symbol $\rightarrow$ both for logical implication and function-type constructor, following the so-called Curry-Howard isomorphism (Carpenter, 1997).

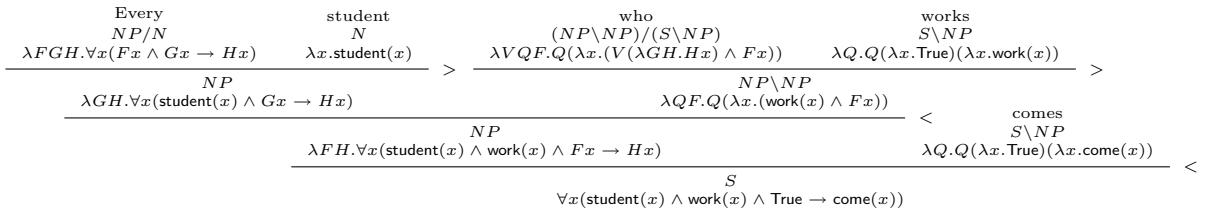


Figure 3: A CCG derivation of the semantic representation for the sentence *Every student who works comes*. $\lambda FGH.X$ is an abbreviation for $\lambda F \lambda G \lambda H.X$. "True" denotes the tautology, hence the final formula is equivalent to $\forall x(\mathsf{student}(x) \wedge \mathsf{work}(x) \rightarrow \mathsf{come}(x))$.

represented in higher-order languages.[3]

Generalized quantifiers A classical example of non-first-orderizable expressions is a proportional generalized quantifier like *most* and *half of* (Barwise and Cooper, 1981). Model-theoretically, they denote relations between sets. We represent them as a two-place higher-order predicate taking first-order predicates as arguments. For instance, *Most students work* is represented as follows.

(1) $\mathsf{most}(\lambda x.\mathsf{student}(x), \lambda x.\mathsf{work}(x))$

Here, most is a higher-order predicate in the sense that it takes first-order predicates $\lambda x.\mathsf{student}(x)$ and $\lambda x.\mathsf{work}(x)$ as arguments. We take the entailment patterns governing *most* as axioms, along the same lines of natural logic and monotonicity calculus (Icard and Moss, 2014), where determiners are taken as primitive two-place operators.

Standard quantifiers like *every* and *some* could also be treated as binary operators in the same way as the binary most in (1). But we choose to adopt the first-order decomposition in such cases (see Fig. 2 for the lexical entry of *every*).

Modals Modal auxiliary expressions like *might*, *must* and *can* are represented as unary sentential operators. For instance, the sentence *Some student might come* is represented as:

(2) $\exists x(\mathsf{student}(x) \wedge \mathsf{might}(\mathsf{come}(x)))$.

An important inference role of such a modal operator is to distinguish modal contexts from actual contexts and thus block an inference from one context to another (might A does not entail A).

Alternatives to the higher-order approach include the first-order decomposition of modal operators using world variables (Blackburn et al., 2001) and the first-order modal semantic representations implemented in Boxer (Bos, 2005). We prefer the higher-order approach, because the first-order approaches introduce additional quantifiers and variables at the level of the semantic representations on which one makes inferences.

Veridical and anti-veridical predicates A sentential operator O is *veridical* if $O(A)$ entails A, and *anti-veridical* if $O(A)$ entails $\neg A$. While modal auxiliary verbs like *might* are neither veridical nor anti-veridical, there is a class of expressions licensing these patterns of inference. Typical examples are adjectives taking an embedded proposition, such as *true/correct* and *false/incorrect*. Note that sentences like *Everything/what he said is false* involve a quantification over propositions, which is problematic for the first-order approach.

The so-called implicative verbs like *manage* and *fail* (Nairn et al., 2006) are also an instance of this class. For example, *Some student manages to come* is formalized as

(3) $\exists x(\mathsf{student}(x) \wedge \mathsf{manage}(x, \mathsf{come}(x)))$

where *manage* is a veridical predicate taking a proposition as the second argument; it licenses an inference to $\exists x(\mathsf{student}(x) \wedge \mathsf{come}(x))$.

Attitude verbs A wide range of propositional attitude verbs such as *believe* and *hope* are similar to modals in that they do not license an inference from attitude contexts to actual contexts. But factives like *know* and *remember* are an exception; they are veridical.[4]

A first-order translation can be given along the lines of Hintikka (1962). (4) is translated as (5).

(4) $\mathsf{know}(\mathsf{john}, \exists x(\mathsf{student}(x) \wedge \mathsf{come}(x)))$

(5) $\forall w_1(R_{\mathsf{john}} w_0\, w_1 \rightarrow \exists x(\mathsf{student}(w_1, x) \wedge \mathsf{come}(w_1, x)))$

[3] See also Blackburn and Bos (2005) for some discussion on inferences that go beyond first-order logic.

[4] Factive predicates show the important inference patterns known as *presupposition projection* (van der Sandt, 1992), which are beyond the scope of this paper.

Inference pattern	Axiom
Existential import	$\forall F \forall G\,(\mathsf{most}(F,G) \rightarrow \exists x(Fx \wedge Gx))$
Conservativity	$\forall F \forall G\,(\mathsf{most}(F,G) \rightarrow \mathsf{most}(F, \lambda x.(Fx \wedge Gx)))$
Monotonicity (right-upward)	$\forall F \forall G \forall H\,(\mathsf{most}(F,G) \rightarrow (\forall x(Gx \rightarrow Hx) \rightarrow \mathsf{most}(F,H)))$
Veridicality	$\forall P(\mathsf{true}(P) \rightarrow P)$ $\forall x \forall P(\mathsf{manage}(x,P) \rightarrow P)$ $\forall x \forall P(\mathsf{know}(x,P) \rightarrow P)$
Anti-veridicality	$\forall P(\mathsf{false}(P) \rightarrow \neg P)$ $\forall x \forall P(\mathsf{fail}(x,P) \rightarrow \neg P)$

Table 2: Axioms for non-first-order constructions.

However, one drawback is that the compositional semantics becomes complicated, so we prefer the non-decomposition approach for attitude verbs.

3.2 Inferences in HOL

Following Chatzikyriakidis and Luo (2014), we use a proof-assistant Coq (Castéran and Bertot, 2004) to implement a specialized prover for higher-order features in natural languages, and combine it with efficient first-order inferences. We use Coq's built-in tactics for first-order inferences. Coq also has a language called *Ltac* for user-defined automated tactics (Delahaye, 2000). The additional axioms and tactics specialized for natural language constructions are written in Ltac. We ran Coq fully automated, by feeding to its interactive mode a set of predefined tactics combined with user-defined proof-search tactics.

Table 2 shows the axioms we implemented. Modals and non-veridical predicates (by which we mean predicates that are neither veridical nor anti-veridical) do not have particular axioms, with the consequence that actual and hypothetical contexts are correctly distinguished.

4 Experiments

We evaluated our system on the FraCaS test suite (Cooper et al., 1994), a set of entailment problems that is designed to evaluate theories of formal semantics.[5] We used the version provided by MacCartney and Manning (2007). The whole data set is divided into nine sections, each devoted to linguistically challenging problems. Of these, we used six sections, excluding three sections (nominal anaphora, ellipsis and temporal reference) that involve a task of resolving context-dependency, a task beyond the scope of this paper. Each problem consists of one or more premises, followed by a hypothesis. There are three types of answer: *yes* (the premise set entails the hypothesis), *no* (the premise set entails the negation of the hypothesis), and *unknown* (the premise set entails neither the hypothesis nor its negation). Fig. 4 shows some examples.

[5]Our system will be publicly available at `https://github.com/mynlp/ccg2lambda`.

Section	#	Ours	Nut	L&S 13	Tian 14
Quantifiers	74	.77	.53	.62	**.80**
Plurals	33	**.67**	.52	–	–
Adjectives	22	**.68**	.32	–	–
Comparatives	31	**.48**	.45	–	–
Verbs	8	.62	.62	–	–
Attitudes	13	**.77**	.46	–	–
Total	181	**.69**	.50	–	–

Table 3: Accuracy on the FraCaS test suite. The first column shows the number of problems. Of the total 188 problems, we excluded seven problems that lack a well-defined answer.

Currently, our system has 57 templates for general syntactic categories and 80 lexical entries for closed words. In a similar way to Bos et al. (2004), closed words are confined to a limited range of logical and functional expressions such as quantifiers and connectives. These templates and lexical entries are not specific with respect to the FraCaS test suite. We use WordNet (Miller, 1995) as the knowledge base for antonymy; logical axioms relevant to given inferences are extracted from this knowledge base.

We compared our system with the state-of-the-art CCG-based first-order system Boxer (Bos, 2008), which is one of the most well-known logic-based approaches to textual entailment. We used the Nutcracker system based on Boxer that utilizes a first-order prover (*Bliksem*) and a model builder (*Mace*) with the option enabling access to WordNet. We did not use the option enabling modal semantics, since it did not improve the results. All experiments were run on a 4-core@1.8Ghz, 8GB RAM and SSD machine with Ubuntu.

Experimental results are shown in Table 3. Our system improved on Nutcracker. We set a time-out of 30 seconds, after which we output the label "unknown". Nutcracker timed-out in one third of the problems (57 out of 181), whereas there was no time-out in our system.

Table 4 shows parse times and inference times for the FraCaS test suite. The inference speed

fracas-067	
Premise 1	All residents of the North American continent can travel freely within Europe.
Premise 2	Every Canadian resident is a resident of the North American continent.
Hypothesis	All Canadian residents can travel freely within Europe.
Answer	Yes
fracas-074	
Premise 1	Most Europeans can travel freely within Europe.
Hypothesis	Most Europeans who are resident outside Europe can travel freely within Europe.
Answer	Unknown

Figure 4: Examples of entailment problems from the FraCaS test suite

Parsing and inference	sec /problem
CCG Parsing (C&C parser)	3.76
Our system with higher-order inference	3.72
Our system with higher-order rules ablated	3.46
Nutcracker with first-order inference (first-order prover + model builder)	11.23

Table 4: Comparison of inference time on the FraCaS test suite. CCG parsing is common to both our system and Nutcracker.

of our system is significantly higher than that of Nutcracker. Our system's total accuracy with higher-order rules is 69%, and drops to 59% when ablating the higher-order rules.

We are aware of two other systems tested on FraCaS that are capable of multiple-premise inferences: the CCG-based first-order system of Lewis and Steedman (2013) and the dependency-based compositional semantics of Tian et al. (2014). These systems were only evaluated on the Quantifier section of FraCaS. As shown in Table 3, our results improve on the former and are comparable with the latter.

Other important studies on FraCaS are those based on natural logic (MacCartney and Manning, 2008; Angeli and Manning, 2014). These systems are designed solely for single-premise inferences and hence are incapable of handling the general case of multiple-premise problems (which cover about 45% of the problems in FraCaS). Our system improves on these natural-logic-based systems by making multiple-premise inferences as well.

Main errors we found are due to various parse errors caused by the CCG parser, including the failure to handle multiwords like *a lot of*. The performance of our system will be further improved with correct syntactic analyses. Our experiments on FraCaS problems do not constitute an evaluation on real texts nor on unseen test data. Note, however, that a benefit of using a linguistically controlled set of entailment problems is that one can check not only whether, but also *how* each semantic phenomenon is handled by the system. In contrast to the widely held view that higher-order logic is less useful in computational linguistics, our results demonstrate the logical capacity of a higher-order inference system integrated with the CCG-based compositional semantics.

5 Conclusion

We have presented a framework for a compositional semantics based on the wide-coverage CCG parser, combined with a higher-order inference system. The experimental results on the FraCaS test suite have shown that a reasonable number of lexical entries and non-first-order axioms enable various logical inferences in an efficient way and outperform the state-of-the-art first-order system. Future work will focus on incorporating a robust model of lexical knowledge (Lewis and Steedman, 2013; Tian et al., 2014) to our framework.

Acknowledgments

We are grateful to Chris Worth, Ribeka Tanaka, and the three anonymous reviewers for their helpful comments and suggestions. This research has been supported by the JST CREST program (Establishment of Knowledge-Intensive Structural Natural Language Processing and Construction of Knowledge Infrastructure).

References

Gabor Angeli and Christopher D. Manning. 2014. NaturalLI: Natural logic inference for common sense reasoning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP-2014)*, pages 534–545.

Jon Barwise and Robin Cooper. 1981. Generalized quantifiers and natural language. *Linguistics and Philosophy*, 4(2):159–219.

Patrick Blackburn and Johan Bos. 2005. *Representation and Inference for Natural Language: A First Course in Computational Semantics*. CSLI.

Patrick Blackburn, Maarten de Rijke, and Yde Venema. 2001. *Modal Logic*. Cambridge University Press.

Johan Bos, Stephen Clark, Mark Steedman, James R. Curran, and Julia Hockenmaier. 2004. Wide-coverage semantic representations from a CCG parser. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING-2004)*, pages 104–111.

Johan Bos. 2005. Towards wide-coverage semantic interpretation. In *Proceedings of Sixth International Workshop on Computational Semantics (IWCS-6)*, pages 42–53.

Johan Bos. 2008. Wide-coverage semantic analysis with Boxer. In *Proceedings of the 2008 Conference on Semantics in Text Processing*, pages 277–286.

Johan Bos. 2009a. Applying automated deduction to natural language understanding. *Journal of Applied Logic*, 7(1):100–112.

Johan Bos. 2009b. Towards a large-scale formal semantic lexicon for text processing. In *Proceedings of the Biennal GSCL Conference From Form to Meaning: Processing Texts Automatically*, pages 3–14.

Bob Carpenter. 1997. *Type-Logical Semantics*. MIT Press.

Pierre Castéran and Yves Bertot. 2004. *Interactive Theorem Proving and Program Development. Coq'Art: The Calculus of Inductive Constructions.* Springer.

Lucas Champollion. 2015. The interaction of compositional semantics and event semantics. *Linguistics and Philosophy*, 38(1):31–66.

Stergios Chatzikyriakidis and Zhaohui Luo. 2014. Natural language inference in Coq. *Journal of Logic, Language and Information*, 23(4):441–480.

Stephen Clark and James R Curran. 2007. Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics*, 33(4):493–552.

Robin Cooper, Richard Crouch, Jan van Eijck, Chris Fox, Josef van Genabith, Jan Jaspers, Hans Kamp, Manfred Pinkal, Massimo Poesio, Stephen Pulman, et al. 1994. FraCaS: A Framework for Computational Semantics. *Deliverable*, D6.

David Delahaye. 2000. A Tactic Language for the System Coq. In *Logic for Programming and Automated Reasoning (LPAR)*, volume 1955 of *Lecture Notes in Computer Science*, pages 85–95. Springer.

Jonathan Ginzburg. 2005. Abstraction and ontology: Questions as propositional abstracts in type theory with records. *Journal of Logic and Computation*, 15(2):113–130.

Jaakko Hintikka. 1962. *Knowledge and Belief*. Cornell University Press.

Jerry R. Hobbs. 1985. Ontological promiscuity. In *Proceedings of the 23rd annual meeting on Association for Computational Linguistics (ACL-1985)*, pages 60–69.

Julia Hockenmaier and Mark Steedman. 2007. CCGbank: a corpus of CCG derivations and dependency structures extracted from the Penn Treebank. *Computational Linguistics*, 33(3):355–396.

Matthew Honnibal, James R. Curran, and Johan Bos. 2010. Rebanking CCGbank for improved NP interpretation. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL-2010)*, pages 207–215.

Thomas Icard and Lawrence Moss. 2014. Recent progress in monotonicity. *Linguistic Issues in Language Technology (LiLT)*, 9.

Mike Lewis and Mark Steedman. 2013. Combining distributional and logical semantics. *Transactions of the Association for Computational Linguistics*, 1:179–192.

Zhaohui Luo. 2012. Formal semantics in modern type theories with coercive subtyping. *Linguistics and Philosophy*, 35(6):491–513.

Bill MacCartney and Christopher D. Manning. 2007. Natural logic for textual inference. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pages 193–200.

Bill MacCartney and Christopher D. Manning. 2008. Modeling semantic containment and exclusion in natural language inference. In *Proceedings of the 22nd International Conference on Computational Linguistics*, pages 521–528.

George A Miller. 1995. WordNet: a lexical database for English. *Communications of the ACM*, 38(11):39–41.

Rowan Nairn, Cleo Condoravdi, and Lauri Karttunen. 2006. Computing relative polarity for textual inference. *Inference in Computational Semantics (ICoS-5)*, pages 67–76.

Stephen Pulman. 2007. Formal and computational semantics: a case study. In *Proceedings of the Seventh International Workshop on Computational Semantics (IWCS-7)*, pages 181–196.

Aarne Ranta. 1994. *Type-Theoretical Grammar*. Oxford University Press.

Mark Steedman. 2000. *The Syntactic Process*. MIT Press.

Ran Tian, Yusuke Miyao, and Takuya Matsuzaki. 2014. Logical inference on dependency-based compositional semantics. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL-2014)*, pages 79–89.

Rob van der Sandt. 1992. Presupposition projection as anaphora resolution. *Journal of Semantics*, 9(4):333–377.

Any-language frame-semantic parsing

Anders Johannsen, Héctor Martínez Alonso, Anders Søgaard
Center for Language Technology
University of Copenhagen, Denmark
Njalsgade 140, DK-2300 Copenhagen S
{ajohannsen,alonso,soegaard}@hum.ku.dk

Abstract

We present a multilingual corpus of Wikipedia and Twitter texts annotated with FRAMENET 1.5 semantic frames in nine different languages, as well as a novel technique for weakly supervised cross-lingual frame-semantic parsing. Our approach only assumes the existence of linked, comparable source and target language corpora (e.g., Wikipedia) and a bilingual dictionary (e.g., Wiktionary or BABELNET). Our approach uses a truly interlingual representation, enabling us to use the *same* model across all nine languages. We present average error reductions over running a state-of-the-art parser on word-to-word translations of 46% for target identification, 37% for frame identification, and 14% for argument identification.

1 Introduction

Frame-semantic parsing is the task of automatically finding semantically salient targets in text, disambiguating the targets by assigning a sense (frame) to them, identifying their arguments, and labeling these arguments with appropriate roles. The FRAMENET 1.5 lexicon[1] provides a fixed repository of semantic frames and roles, which we use in the experiments below.

Several learning and parsing algorithms have been developed for frame-semantic analysis (Johansson and Nugues, 2007; Das et al., 2014; Täckström et al., 2015), and frame semantics has been successfully applied to question-answering (Shen and Lapata, 2007), information extraction (Surdeanu et al., 2003) and knowledge extraction (Søgaard et al., 2015b).

[1] https://framenet.icsi.berkeley.edu/

In contrast to Propbank-style semantic-role labeling (Titov and Klementiev, 2012), only very limited frame-semantic resources exist for languages other than English. We therefore focus on *multilingual* or cross-language frame-semantic parsing, leveraging resources for English and other major languages to build *any*-language parsers. We stress that we learn frame-semantic parsing models that can be applied to *any* language, rather than cross-lingual transfer models for specific target languages. Our approach relies on inter-lingual word embeddings (Søgaard et al., 2015a), which are built from topic-aligned documents. Word embeddings have previously been used for monolingual frame-semantic parsing by Hermann et al. (2014).

Contributions This paper makes the following three contributions. We present a new multilingual frame-annotated corpus covering five topics, two domains (Wikipedia and Twitter), and nine languages. We implement a simplified version of the frame-semantic parser introduced in Das et al. (2014). Finally, we show how to modify this parser to learn any-language frame-semantic parsing models using inter-lingual word embeddings (Søgaard et al., 2015a).

2 Data annotation

Figure 1 depicts a FRAMENET 1.5 frame-semantic analysis of a German sentence from Wikipedia. The annotator marked two words, *Idee* and *kam*, as targets. In frame-semantic parsing, target identification is the task of deciding which words (i.e. targets) trigger FRAMENET frames. Frame identification is the problem of disambiguating targets by labeling them with frames, e.g., COGITATION or COMING_UP_WITH. Argument identification is the problem of identifying the arguments of frames, e.g., *Idee* for COMING_UP_WITH.

We had linguistically trained students anno-

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 2062–2066,
Lisbon, Portugal, 17-21 September 2015.

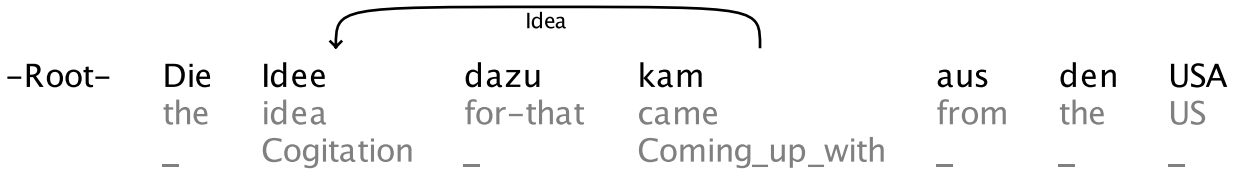


Figure 1: Frame semantic annotation from the German Wikipedia data (Women's Rights)

tate about 200 sentences from Wikipedia and 200 tweets each in their native language. The data was pre-annotated by obtaining all English translation equivalents of the source language words through BABELNET[2], finding associated frames in the FRAMENET 1.5 training data. We presented annotators with *all* frames that could be triggered by any of the target word's translations. Both data from Wikipedia and Twitter cover the same five topics: Google, Angelina Jolie, Harry Potter, Women's Rights, and Christiano Ronaldo. The topics were chosen to guarantee coverage for all nine languages, both in Wikipedia and Twitter. Our corpus, which covers nine languages, is publicly available at `https://github.com/andersjo/any-language-frames` The languages we cover are Bulgarian (BG), Danish (DA), German (DE), Greek (EL), English (EN), Spanish (ES), French (FR), Italian (IT) and Swedish (SV). English is included as a sanity check of our cross-lingual annotation setup.

The English, Danish, and Spanish datasets were doubly-annotated in order to compute inter-annotator agreement (IAA). The overall target identification IAA was 82.4% F_1 for English, 81.6% for Danish, and 80.0% for Spanish. This is lower than a similar monolingual annotation experiment recently reporting target identification IAA at 95.3% (Søgaard et al., 2015b). The frame identification IAA scores were also higher in that study, at 84.5% and 78.1% F_1. The drop in agreement seems mostly due to pre-tagging errors caused by erroneous or irrelevant word-to-word translations. The Spanish data has the lowest agreement score.

We compute test-retest reliability of our annotations as the correlation coefficient (Pearson's ρ) between the two annotations. In Cronbach's α internal consistency table, the cut-off for acceptable reliability is 0.7. While there is certainly noise in our annotations, these are still consistently above the Cronbach cut-off. Also, we evaluate our models across 18 datasets, covering nine different languages with two domains each; although for readability, we combine the Wiktionary and Twitter datasets for each language below.

[2] `http://babelnet.org/`

	Language		
	EN	DA	ES
Twitter and Wikipedia			
TARGET	82.4	81.6	80.0
FRAME	73.5	72.3	60.8
ARGUMENT	70.7	55.0	83.5
Test-retest reliability	74.4	78.6	71.8
Twitter			
TARGET	79.1	80.7	80.5
FRAME	68.8	72.3	58.6
ARGUMENT	70.0	86.2	57.5
Test-retest reliability	71.0	78.7	73.1

Table 1: Inter-annotator agreement (F_1 in %)

The relatively low reliability compared to previous annotation efforts is due to the cross-lingual pre-annotation step, which was necessary to make annotation feasible. All languages, including English, have been pre-annotated using BABELNET. We expect annotators to only assign frames when meaningful frames can be assigned, so the main source of error is that the pre-annotation may exclude valid frames. Hence, we will not only report F_1-scores in our evaluations, but also precision, since recall may be misleading, penalizing for frames that could not be chosen by the annotators.

3 Frame semantic parsing

3.1 Target identification

Following Das et al. (2014), we use part-of-speech heuristics to identify the words that evoke frames (target words). Frame-evoking words typically belong to a narrow range of part of speech. Therefore, we only consider words as target candidates

when tagged with one of the top k part-of-speech tags most commonly seen as targets in the training set. The k parameter is optimized to maximize F_1 on our development language, Spanish, where we found $k = 7$.[3] Surviving candidates are then translated into English by mapping the words into multi-lingual BABELNET synsets, which represent sets of words with similar meaning across languages. All English words in the BABELNET synsets are considered possible translations. If any of the translations are potential targets in FRAMENET 1.5, the current word is identified as a frame-evoking word.

3.2 Frame identification

A target word is, on average, ambiguous between three frames. We use a multinomial log-linear classifier[4] (with default parameters) to decide which of the possible frames evoked by the target word that fits the context best. Our feature representation replicates that of Das et al. (2014) as far as possible, considering the multilingual setting where lexical features cannot be directly used. To compensate for the lack of lexical features, we introduce two groups of language-independent features that rely on multilingual word embeddings. One feature group uses the embedding of the target word directly, while the other is based on distance measures between the target word and the set of English words used as targets for a possible frame. We measure the minimum and mean distance (in embedding space) from the target word to the set of English target words, as well as the distances to each word individually.

Several of the features in the original representation are built on top of automatic POS annotation and syntactic parses. We use the Universal Dependencies v1.1 treebanks for the languages in our data to train part-of-speech taggers (TREETAGGER[5]) and a dependency parser (TURBOPARSER[6]) to generate the syntactic features. In contrast to Das et al. (2014), we use dependency subtrees instead of spans.

[3]The white-listed POS are nouns, verbs, adjectives, proper nouns, adverbs, and determiners.

[4]http://hunch.net/~vw/

[5]http://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/

[6]http://www.cs.cmu.edu/~ark/TurboParser/

3.3 Argument identification

A frame contains a number of named arguments that may or may not be expressed in a given sentence. Argument identification is concerned with assigning frame arguments to spans of words in the sentence. While this task can benefit from information on the joint assignment of arguments, Das et al. (2014) report only an improvement of less than 1% in F_1 using beam search to approximate a global optimal configuration for argument identification. To simplify our system, we take all argument-identification decisions independently. We use a single classifier for argument identification, computing the most probable argument for each frame element. Each word index is associated with a span by the transitive closure of its syntactic dependencies (i.e. subtree). Our greedy approach to argument identification thus amounts to scoring the $n + 1$ possible realisations of an argument for an n-length sentence (i.e. subtrees plus the empty argument), selecting the highest scoring subtree for each argument type allowed by the frame.

As the training data contains very few examples of each frame or role (e.g., *Buyer* in the frame COMMERCE_SCENARIO), we enable sharing of features for frame arguments that have the same name. The assumption is that arguments with identical names have similar semantic properties across frames; that is the argument *Perpetrator*, for example, is similar for the frames ARSON and THEFT.

The scores are the confidences of a binary classifier trained on <*frame*, *argument*, *subtree*> tuples. Positive examples are the observed arguments. We use the remaining n incorrect subtrees for a given <*frame*, *argument*> pair to generate negative training examples . A single binary classification model is trained for the whole data set.

As with frame identification, our features are similar to those of Das et al. (2014), with a few exceptions and additions. We use dependency subtrees instead of spans and replace all lexical features (which do not transfer cross-lingually) with features based on the interlingual word embeddings from Søgaard et al. (2015a). We use the embeddings to find the 20 most similar words in the training data and use these words to generate lexical features that matched the source-language training data. Each feature is weighted by its cosine similarity with the target-language word.

Target identification		BG	DA	DE	EL	EN	ES	FR	IT	SV	Avg.
F_1	SYSTEM	**85.5**	**73.6**	**58.4**	**52.9**	**80.2**	**89.1**	**66.1**	**69.0**	**72.8**	**72.0**
	BASELINE	44.0	56.8	27.2	46.1	78.8	45.9	42.8	47.7	41.4	47.9
Precision	SYSTEM	**89.2**	**70.9**	**66.2**	36.4	**96.3**	**84.9**	51.8	53.4	**63.4**	**67.0**
	BASELINE	56.8	65.0	48.7	**43.2**	88.0	75.2	**55.0**	**55.3**	47.3	59.4

Frame identification		BG	DA	DE	EL	EN	ES	FR	IT	SV	Avg.
F_1	SYSTEM	**66.6**	**59.0**	49.0	**58.3**	37.0	**36.9**	**27.4**	**40.2**	49.5	**47.1**
	BASELINE	19.3	14.1	08.5	12.6	**48.8**	08.2	10.4	15.0	10.1	16.3
	MFS	65.3	54.3	**53.0**	56.2	38.0	34.4	25.5	33.0	**55.3**	46.1
Precision	SYSTEM	**72.8**	**64.7**	**57.9**	**67.1**	49.3	45.6	**36.9**	**47.1**	**65.5**	**56.3**
	BASELINE	37.0	26.4	19.0	27.9	**62.4**	15.7	22.0	25.5	28.3	29.7
	MFS	67.7	59.4	57.4	60.1	46.1	42.3	33.4	41.5	61.5	52.2

Argument identification		BG	DA	DE	EL	EN	ES	FR	IT	SV	Avg.
F_1	SYSTEM	**40.8**	**36.0**	**28.5**	**39.3**	25.3	**19.8**	**18.0**	**26.3**	**28.7**	**29.2**
	BASELINE	26.5	10.5	06.2	09.7	**69.6**	04.6	08.6	14.6	08.6	17.7
Precision	SYSTEM	**39.6**	**33.3**	**26.3**	**36.7**	24.0	**18.1**	**16.8**	**24.8**	**26.4**	**27.3**
	BASELINE	16.2	09.5	05.7	08.8	**66.8**	04.1	08.1	13.8	08.0	16.8

Table 2: Frame semantic parsing results (precision and F_1 in %)

Baseline Our approach to multi-lingual frame semantics parsing extends Das et al. (2014) to cross-lingual learning using the interlingual embeddings from Søgaard et al. (2015a). Our baseline is a more direct application of the SEMAFOR system[7] (Das et al., 2014), translating target language text to English using word-to-word translations and projecting annotation back. For word-to-word translation we use Wiktionary bilingual dictionaries (Ács, 2014), and we use frequency counts from UKWAC[8] to disambiguate words with multiple translations, preferring the most common one. The baseline and our system both use the training data supplied with FRAMENET for learning.

4 Results

Consider first the target identification results in Table 2. We observe that using BABELNET and our re-implementation of Das et al. (2014) performs considerably better than running SEMAFOR on Wiktionary word-by-word translations.

Our frame identification results are also presented in Table 2. Our system is better in six out of nine cases, whereas the most frequent sense baseline is best in two. It is unsurprising that English fares best in this setup, because it does not undergo the word-to-word translation of the other data sets.

Argument identification is a harder task, and scores are generally lower; see the lower part of Table 2. Also, note that errors percolate: If we do not identify a target, or mislabel a frame, we can no longer retrieve the correct arguments. Nevertheless, we observe that we are better than running SEMAFOR on word-by-word translations in eight out of nine languages—all, except English.

Generally, we obtain error reductions over our baseline of 46% for target identification, 37% for frame identification, and 14% for argument identification. For English, we are only 2% (absolute) below IAA for target identification, but about 40% below IAA for frame and argument identification. For Danish, the gap is smaller.

If we compare performance on Wikipedia and Twitter datasets, we see that target identification and frame identification scores are generally higher for Wikipedia, while argument identification scores are higher for Twitter. While Wikipedia is generally more similar to the

[7] http://www.ark.cs.cmu.edu/SEMAFOR/
[8] http://wacky.sslmit.unibo.it/

newswire/balanced corpus in FRAMENET 1.5, the sentence length is shorter in tweets, making it easier to identify the correct arguments.

5 Conclusions

We presented a multi-lingual frame-annotated corpus covering nine languages in two domains. With this corpus we performed experiments to predict target, frame and argument identification, outperforming a word-to-word translated baseline running on SEMAFOR. Our approach is a delexicalized version of Das et al. (2014) with a simpler decoding strategy and, crucially, using multilingual word embeddings to achieve any-language frame-semantic parsing. Over a baseline of using SEMAFOR with word-to-word translations, we obtain error reductions of 46% for target identification, 37% for frame identification, and 14% for argument identification.

References

Judit Ács. 2014. Pivot-based multilingual dictionary building using wiktionary. In *LREC*.

Dipanjan Das, Desai Chen, Andre Martins, Nathan Schneider, and Noah Smith. 2014. Frame-semantic parsing. *Computational linguistics*, 40(1):9–56.

Karl Moritz Hermann, Dipanjan Das, Jason Weston, and Kuzman Ganchev. 2014. Semantic frame identification with distributed word representations. In *ACL*.

Richard Johansson and Pierre Nugues. 2007. Extended constituent-to-dependency conversion for English. In *NODALIDA*.

Dan Shen and Mirella Lapata. 2007. Using semantic roles to improve question answering. In *EMNLP*.

Anders Søgaard, Željko Agić, Héctor Martínez Alonso, Barbara Plank, Bernd Bohnet, and Anders Johannsen. 2015a. Inverted indexing for cross-lingual nlp. In *ACL*.

Anders Søgaard, Barbara Plank, and Hector Martinez Alonso. 2015b. Using frame semantics for knowledge extraction from twitter. In *AAAI*.

Mihai Surdeanu, Sanda Harabagiu, John Williams, and Paul Aarseth. 2003. Using predicate-argument structures for information extraction. In *ACL*.

Oscar Täckström, Kuzman Ganchev, and Dipanjan Das. 2015. Efficient inference and structured learning for semantic role labeling. *TACL*.

Ivan Titov and Alexandre Klementiev. 2012. Crosslingual induction of semantic roles. In *ACL*.

What's in an Embedding? Analyzing Word Embeddings through Multilingual Evaluation

Arne Köhn
Department of Informatics
Universität Hamburg
koehn@informatik.uni-hamburg.de

Abstract

In the last two years, there has been a surge of word embedding algorithms and research on them. However, evaluation has mostly been carried out on a narrow set of tasks, mainly word similarity/relatedness and word relation similarity and on a single language, namely English.

We propose an approach to evaluate embeddings on a variety of languages that also yields insights into the structure of the embedding space by investigating how well word embeddings cluster along different syntactic features.

We show that all embedding approaches behave similarly in this task, with dependency-based embeddings performing best. This effect is even more pronounced when generating low dimensional embeddings.

1 Introduction

Word embeddings map words into a vector space, allowing to reason about words in this space. They have been shown to be beneficial for several tasks such as machine translation (Botha and Blunsom, 2014), parsing (Lei et al., 2014), and named entity recognition (Passos et al., 2014). Recently, word embedding techniques have been studied for their mathematical properties (Levy and Goldberg, 2014b; Stratos et al., 2015), yielding a better understanding of the underlying optimization criteria. However, word embeddings have mostly been studied and evaluated on a single language (English). Therefore, validation on languages other than English is lacking and the question whether word embeddings work the same way across languages has not been empirically evaluated. Evaluations of complex systems – such as parsers – employing word embeddings generally give only little insight into the type of contribution to the result and the structure of word embeddings.

We aim to fill these gaps by evaluating several word embedding algorithms on a set of different languages using tasks that enable additional insight into the learned structures using easily obtainable data. At the same time, we provide baseline results for using word embeddings in several syntax-based classification tasks.

We focus on syntax-related measures because data is available for several languages and we expect a correlation with usefulness of word embeddings for syntax-related tasks such as named entity recognition, parsing, and morphological analysis.

2 Related Work

Previous approaches to word embedding evaluation have either used relatively basic word finding and classification tasks (as this paper also proposes) or application-oriented end-to-end evaluations as part of a larger system. Word finding tasks are of the form "given a pair of words (x, y), find a y' for a given x'", e.g. given (Rome, Italy), find a word for Oslo. These tasks have been introduced by Mikolov et al. (2013a). The downside of this kind of task is that the data is not readily available and has to be constructed for each language. This type of evaluation primarily describes the similarity between vector differences and not similarity between vectors. In addition, Levy et al. (2015) showed for this task that word embedding-based classifiers actually mostly learn whether a word is a *general hypernym* and not, as would be expected, the relation between two words.

Another approach to evaluate embeddings, used by Pennington et al. (2014) amongst others, is to rank a fixed set of words relative to a reference word. The results are then compared to human judgments, e.g. from the WS353 corpus (Finkelstein et al., 2002). This approach has a limited coverage and additional data is expensive to obtain.

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 2067–2073,
Lisbon, Portugal, 17-21 September 2015.

Botha and Blunsom (2014) propose to factorize word vectors into morpheme vectors to better capture similarities between morphologically related words and evaluate their word representations using log-bilinear language models based on their word vectors.[1] They measure model perplexity reduction relative to n-gram language models and include their model into a machine translation system, gaining between 0 (English → German) and 1.2 (English → Russian) BLEU points.

Lei et al. (2014) introduce a syntactic dependency parser using (amongst others) a low-rank tensor component for scoring dependency edges. This scoring can employ word embeddings. Doing so yields an improvement of 0.2 to 0.5 percentage points. If no Part-of-Speech (PoS) tags are available, this difference rises to up to four percentage points. Köhn et al. (2014) show that this gain from using word embeddings is even more pronounced in complete absence of morphological information (including PoS tags), reporting a difference of five to seven percentage points,depending on the language, using the same parser. With these findings, it can be assumed that word embeddings encode some kind of morphological information. Neither, however, investigated what kind of information the word embeddings actually contain.

3 The Embedding Algorithms

To assess the differences between embedding algorithms, we will evaluate six different approaches. The continuous bag-of-words (**cbow**) approach described by Mikolov et al. (2013a) is learned by predicting the word vector based on the context vectors. The skip-gram approach (**skip**) from the same authors is doing the reverse: it predicts the context word vectors based on the embedding of the current word. We use the version of **cbow** and **skip** as described in (Mikolov et al., 2013b) which use negative sampling, i.e. they train by distinguishing the correct word in its context against words not occurring in that context.

Levy and Goldberg (2014a) alter the skipgram approach by not using the neighboring words wrt. the sentence's word sequence but wrt. the dependency tree of the sentence. Therefore, the context of w is defined as all words that are either the head or dependents of w. We will call this approach **dep**.

[1] Their approach has not been evaluated in this paper as the corresponding code is not available as of now.

GloVe, introduced by Pennington et al. (2014), optimizes the ratio of co-occurrence probabilities instead of the co-occurrence probabilities themselves, getting rid of the negative sampling used for the approaches previously mentioned.

Stratos et al. (2015) describe a method to derive word embeddings using canonical correlation analysis. We will call this approach **cca**.

brown clusters (Brown et al., 1992) are constructed by clustering words hierarchically into a binary search tree in a way that maximizes mutual information for a language model. To construct an embedding for a cluster c, we use the following procedure: For each edge on the path from the root to c, add either 1 or -1, depending on the direction of descent. Because not every path has the same depth, we pad missing dimensions with 0. This way, we obtain an embedding interpretation of the clusters. Note that, in contrast to clustering embeddings, no information is lost.

4 Our Evaluation by Classification Tasks

We classify words separately according to several tasks with an L2-regularized linear classifier. All classification tasks are based on the word embedding of a single word alone, without any other information about the word or its context; in particular, the word's lexicalization is not used as a feature. By using the continuous features directly instead of clustering them (as e.g. done by Bansal et al. (2014)), we ensure that no information is lost during preprocessing.

All tasks can be carried out on dependency treebanks with morphological annotation. From each word in the treebank, we extract a data point *(word embedding, class)* for training/testing, where *class* is of one of the following, depending on the task:

pos The Part-of-Speech of the word
headpos The PoS of the word's head
label The label of the word's dependency edge
gender* The gender of the word
case* The case of the word
number* The number of the word
tense* The tense of the word

Tasks marked with an asterisk are only carried out on words with a corresponding feature. Some of these features are absent in some languages, e.g. Basque is mostly genderless and the corpus of English we used is not annotated with morphological information. These combinations of language and feature have been omitted.

We use a one-versus-all linear classifier for two reasons: First, the feature dimensionality is relatively high. Second, and more importantly, training a linear classifier yields insights into the structure of the vector space because the classifier also serves as a tool to obtain a supervised clustering of the vector space.

Let $\mathbf{C}$ be set set of classes. A one-versus-all linear classifier learns a linear function $f_c \in \mathbb{R}^n \rightarrow \mathbb{R}$ for each class $c \in \mathbf{C}$. The classifier assigns to a vector X the best matching class based on these functions:

$$\text{class}(X) = \arg\max_{c \in \mathbf{C}} f_c(X)$$

Due to the linearity of the functions f_c, the vector space is partitioned into convex polytopes, which each represent exactly one class (see Appendix A). Therefore, the classification accuracies can also be interpreted as supervised clustering accuracies. This means that if the classifier yields a high accuracy, the members of each class are clustered in a single convex region of the vector space. We think that this is a fairly strong statement about the structure of the vector space.

To better gauge how well the embeddings are actually clustered, we use a majority baseline which classifies all elements as the one class that occurred most often during training. This is the accuracy a classifier would yield without any information and therefore the information gain obtainable by using word embeddings as features is the difference between the achieved accuracy and the baseline accuracy.

In addition to the lower bound described above, we also provide an approximate upper bound for the accuracy. Because no context information is used during classification, the word vector corresponding to a word will always be classified the same, even though the correct classification might depend on the context, e. g. the word *put* can belong to different tense classes depending on the context. Therefore, an upper bound for the classification task is to assign each word the most probable class for that word (computed on the training set). Assuming that no sparsity issues exist, embedding-based classification can yield at most accuracies as high as this approach. Note that because in reality data sparsity unfortunately does exist, this is only an approximation of the upper bound. We call this approximation *up-approx* and compute it omitting words not seen during training.

5 Experimental Setup

Evaluation was carried out on Basque, English, French, German, Hungarian, Polish, and Swedish datasets. For English, automatically labeled data was obtained by tagging and parsing a subset of the English Wikipedia dump provided by Al-Rfou et al. (2013) using TurboTagger and TurboParser (Martins et al., 2013). The Penn Treebank (Marcus et al., 1994), converted using the LTH converter (Johansson and Nugues, 2007), was used as the corresponding manually annotated resource.

For all other languages, datasets including both automatically and manually annotated data provided as part of the Shared Task on parsing morphologically rich languages (Seddah et al., 2014) were used.[2]

For all languages, we trained embeddings on the automatically labeled data using the approaches described in Section 3, with different window sizes (5 and 11, where applicable) and dimensions (10, 100, 200). The rare word limit was set to five words occurrences. **brown** was only trained with 1024 clusters equaling about 10 dimensions, as the number of clusters can not be increased to generate higher-dimensional embeddings. **dep** was not evaluated on French because the French automatically labeled dataset lacks dependency information.

6 Results

Figure 1 a) shows the accuracies for the evaluated word embeddings on all tasks for the different languages. The results were obtained using the best-performing hyperparameters (200 dimensions for all, window size = 5 for **cca**, **cbow** and **skip**, window size = 11 for **GloVe**, compare Table 1).

All embeddings capture the PoS well. To a lesser degree, the dependency label and head PoS can also be recovered. The better-performing embeddings achieve results near the approximate upper bound for all tasks.

The embeddings also mostly cluster well with respect to tense, number, gender, and case, with tense showing the best correlations. For some of these tasks, the baseline is however fairly high because the number of classes is lower.

[2]Basque: (Aduriz et al., 2003; Aldezabal et al., 2008), French: (Abeillé et al., 2003; Candito et al., 2010), German: (Brants et al., 2002; Seeker and Kuhn, 2012), Hungarian: (Csendes et al., 2005; Vincze et al., 2010), Polish: (Woliński et al., 2011; Świdziński and Woliński, 2010; Wróblewska, 2012), Swedish: (Nivre et al., 2006)

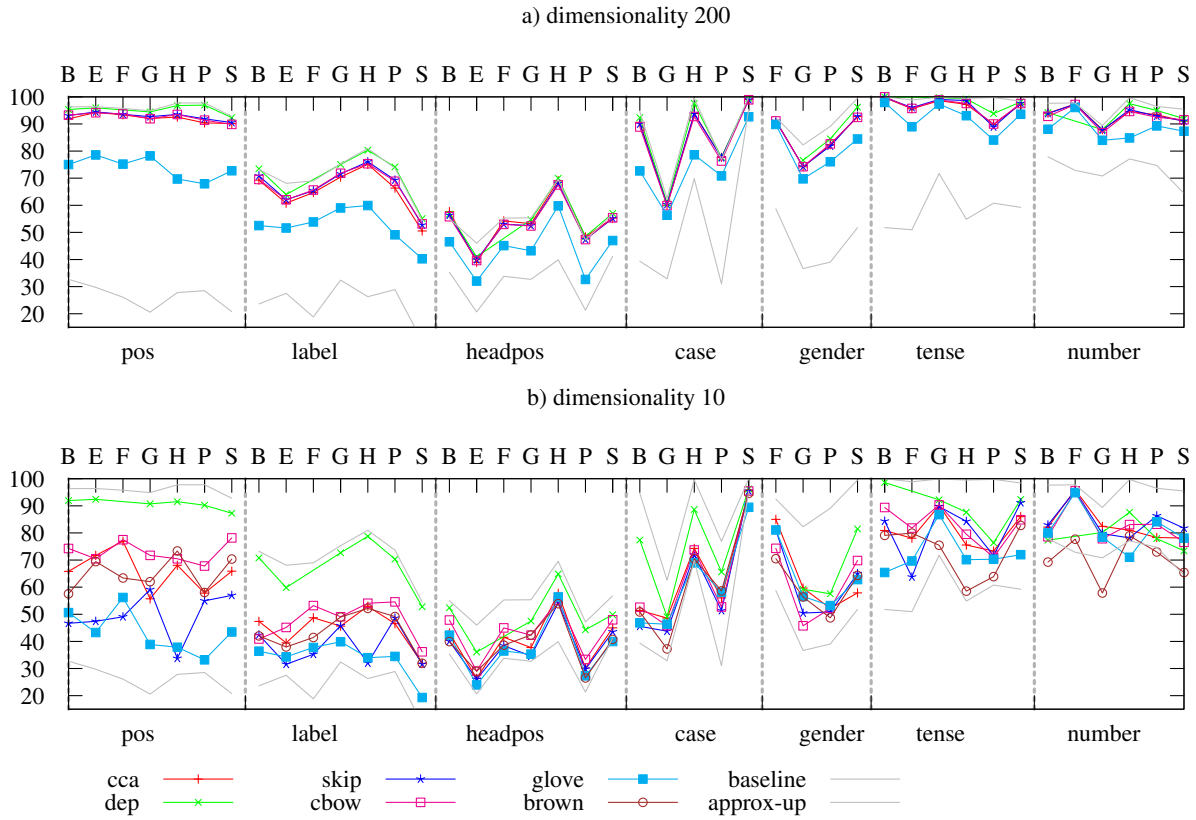


Figure 1: Results with window = 5 (for cbow, cca & skip) / 11 (for GloVe) for **B**asque, **E**nglish, **F**rench, **G**erman, **H**ungarian, **P**olish, **S**wedish. Note: brown is only present in b).

w	d	cca	skip	cbow	dep	GloVe
5	200	80.41	80.69	80.42	82.35	70.05
	100	−1.38	−1.16	−3.31	−0.39	−2.24
	10	−18.06	−22.92	−16.18	−8.38	−16.12
11	200	−1.31	−0.04	−0.05	n/a	+0.57
	100	−3.56	−1.16	−1.17	n/a	−1.73
	10	−23.51	−22.94	−16.34	n/a	−15.64

Table 1: Mean accuracy across tasks for **d**imension=200 and **w**indow=5, and change in mean accuracy when deviating, measured in percentage points. **dep** has no window parameter.

cbow, **cca** and **skip** perform nearly identical, while **dep** performs slightly better. Interestingly, **GloVe** performs consistently worse than all other embeddings, contrary to the findings published in Pennington et al. (2014), but in line with Stratos et al. (2015). **dep** performs best on nearly all tasks, which may indicate that dependency-based context is not only beneficial for preserving dependency-related information, but also for morphology.

This finding is even more pronounced in the evaluation using only ten dimensions (Figure 1 b)): While **dep** can capture the different aspects tested for nearly as well as with 200 dimensions, the other embeddings suffer larger degradations, especially for PoS and label prediction. **cbow** seems to be able to cope better with low dimensionality than **skip**, although they perform nearly identical on the high dimensionality tasks. **brown** behaves similar to the other approaches despite being quite different algorithmically and only producing low-granular data (with values for each dimension being either 1, 0, or −1). Note that results near the baseline signify that the embeddings yield only minimal benefit since the baseline does not use any features at all.

Table 1 gives an overview of the average change in accuracy when changing hyperparameters. Using 200 dimensions instead of 100 is beneficial for all word embeddings. The difference is however not nearly as pronounced as between ten and 100 dimensions. **skip** and **cbow** yield slightly better results with a window of five, whereas for **GloVe** a larger window is advantageous. **dep** achieves both the highest average score and has the lowest degradation when lowering the dimensionality.

Bansal et al. (2014) evaluate word embeddings

wrt. how they cluster along PoS tags. They first divide the embeddings into 1000 clusters using k-means and then associate each cluster with a PoS tag. They report a clustering accuracy of 81.1% for $w = 11$ and 85.8% for $w = 5$ using **skip**. Our results, however, show an accuracy of 94.4% and 94.4%, respectively, i.e. no such difference. That means that the PoS are still mostly linearly separable with larger window sizes. The differences observed by them could result from information getting lost during clustering.

7 Conclusions

Word embeddings are able to capture a range of syntactic and morphological information. They align especially well with the word's part of speech. With a high dimensionality, most embeddings perform similarly, with **GloVe** performing on average ten percentage points worse. With a low dimensionality, the differences become more pronounced and **dep** is the clear choice for applications where using high-dimensional vectors is not feasible and a correlation to the features tested in this paper is wanted.

We have shown that the different word embedding algorithms behave similar over a variety of languages and perform well relative to the task's upper bounds.

The evaluation approach proposed yields insights into the usefulness of embeddings for syntax-related tasks, works on a wide variety of languages and avoids inaccuracies introduced when employing unsupervised clustering for evaluation. We hope that this evaluation approach will be useful for evaluating future embedding techniques.

The software to replicate the experiments for this paper is available on `http://arne.chark.eu/emnlp2015`.

A Proof: Convexity of regions

To show that a one-versus-all classifier generates exactly one convex polytope for each class, we have to show that for any two points belonging to a class, each point between them belongs to the same class.

Let $c \in \mathbf{C}$ be a class and $\mathbf{r}_c \subseteq \mathbb{R}^n$ be the region(s) of c in the vector space[3] , i. e. where the following holds true:

$$f_c > f_o \; \forall o \in \mathbf{C} \setminus c$$

[3]the vector space is assumed to have one dimension for the bias.

Let $x, y \in \mathbf{r}_c$ be two points classified into c. Then the following statement needs to be true:

$$z \in \mathbf{r}_c, \;\; z := (1 - \lambda)x + \lambda y \quad \forall \lambda \in [0, 1]$$

Assume that $z \notin \mathbf{r}_c$. Then, by definition, $f_o(z) > f_c(z)$ for some $o \in \mathbf{C} \setminus c$. We can substitute z with its definition:

$$f_o((1 - \lambda)x + \lambda y) > f_c((1 - \lambda)x + \lambda y)$$

And therefore due to the linearity of f_o and f_c:

$$(1 - \lambda)f_o(x) + \lambda f_o(y) > (1 - \lambda)f_c(x) + \lambda f_c(y)$$

But this cannot be, as by definition, $f_o(x) < f_c(x)$ and $f_o(y) < f_c(y)$. Therefore, there is only one region for c and that region is convex. □

References

Anne Abeillé, Lionel Clément, and François Toussenel. 2003. Building a treebank for french. In Anne Abeillé, editor, *Treebanks*. Kluwer, Dordrecht.

I. Aduriz, M. J. Aranzabe, J. M. Arriola, A. Atutxa, A. Díaz de Ilarraza, A. Garmendia, and M. Oronoz. 2003. Construction of a Basque dependency treebank. In *TLT-03*, pages 201–204.

Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. 2013. Polyglot: Distributed word representations for multilingual nlp. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 183–192, Sofia, Bulgaria, August. Association for Computational Linguistics.

I. Aldezabal, M.J. Aranzabe, A. Diaz de Ilarraza, and K. Fernández. 2008. From dependencies to constituents in the reference corpus for the processing of Basque. In *Procesamiento del Lenguaje Natural, nž 41 (2008)*, pages 147–154. XXIV edición del Congreso Anual de la Sociedad Española para el Procesamiento del Lenguaje Natural (SEPLN).

Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2014. Tailoring continuous word representations for dependency parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 809–815, Baltimore, Maryland, June. Association for Computational Linguistics.

Jan Botha and Phil Blunsom. 2014. Compositional morphology for word representations and language modelling. In Eric P. Xing and Tony Jebara, editors, *Proceedings of The 31st International Conference on Machine Learning*, JMLR Workshop and Conference Proceedings, pages 1899–1907.

Sabine Brants, Stefanie Dipper, Silvia Hansen, Wolfgang Lezius, and George Smith. 2002. The TIGER treebank. In Erhard Hinrichs and Kiril Simov, editors, *Proceedings of the First Workshop on Treebanks and Linguistic Theories (TLT 2002)*, pages 24–41, Sozopol, Bulgaria.

Peter F. Brown, Vincent J. Della Pietra, Peter V. deSouza, Jenifer C. Lai, and Robert L. Mercer. 1992. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479, December.

Marie Candito, Benoit Crabbé, and Pascal Denis. 2010. Statistical French dependency parsing: Treebank conversion and first results. In *Proceedings of LREC*, Valletta, Malta.

Dóra Csendes, János Csirik, Tibor Gyimóthy, and András Kocsor. 2005. The Szeged treebank. In *Proceedings of the 8th International Conference on Text, Speech and Dialogue (TSD)*, Lecture Notes in Computer Science, pages 123–132, Berlin / Heidelberg. Springer.

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

Richard Johansson and Pierre Nugues. 2007. Extended constituent-to-dependency conversion for English. In Kadri Muischnek Joakim Nivre, Heiki-Jaan Kaalep and Mare Koit, editors, *Proceedings of the 16th Nordic Conference of Computational Linguistics NODALIDA-2007*, pages 105–112, University of Tartu, Tartu.

Arne Köhn, U Chun Lao, AmirAli B Zadeh, and Kenji Sagae. 2014. Parsing morphologically rich languages with (mostly) off-the-shelf software and word vectors. In *Proceedings of the 2014 Shared Task of the COLING Workshop on Statistical Parsing of Morphologically Rich Languages*.

Tao Lei, Yu Xin, Yuan Zhang, Regina Barzilay, and Tommi Jaakkola. 2014. Low-rank tensors for scoring dependency structures. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1381–1391, Baltimore, Maryland, June. Association for Computational Linguistics.

Omer Levy and Yoav Goldberg. 2014a. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 302–308, Baltimore, Maryland, June. Association for Computational Linguistics.

Omer Levy and Yoav Goldberg. 2014b. Neural word embedding as implicit matrix factorization. In Z. Ghahramani, M. Welling, C. Cortes, N.D. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2177–2185. Curran Associates, Inc.

Omer Levy, Steffen Remus, Chris Biemann, and Ido Dagan. 2015. Do supervised distributional methods really learn lexical inference relations? In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 970–976, Denver, Colorado, May–June. Association for Computational Linguistics.

Mitchell Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. 1994. The Penn Treebank: Annotating predicate argument structure. In *Proceedings of the Workshop on Human Language Technology*, HLT '94, pages 114–119, Stroudsburg, PA, USA. Association for Computational Linguistics.

Andre Martins, Miguel Almeida, and Noah A. Smith. 2013. Turning on the turbo: Fast third-order non-projective turbo parsers. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 617–622, Sofia, Bulgaria, August.

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

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In C.J.C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

Joakim Nivre, Jens Nilsson, and Johan Hall. 2006. Talbanken05: A Swedish treebank with phrase structure and dependency annotation. In *Proceedings of LREC*, pages 1392–1395, Genoa, Italy.

Alexandre Passos, Vineet Kumar, and Andrew McCallum. 2014. Lexicon infused phrase embeddings for named entity resolution. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*, pages 78–86, Ann Arbor, Michigan, June. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October. Association for Computational Linguistics.

Djamé Seddah, Reut Tsarfaty, Sandra Kübler, Marie Candito, Jinho Choi, Matthieu Constant, Richárd Farkas, Iakes Goenaga, Koldo Gojenola, Yoav Goldberg, Spence Green, Nizar Habash, Marco

Kuhlmann, Wolfgang Maier, Joakim Nivre, Adam Przepiorkowski, Ryan Roth, Wolfgang Seeker, Yannick Versley, Veronika Vincze, Marcin Woliński, Alina Wróblewska, and Eric Villemonte de la Clérgerie. 2014. Overview of the SPMRL 2014 shared task on parsing morphologically rich languages. In *Notes of the SPMRL 2014 Shared Task on Parsing Morphologically-Rich Languages*, Dublin, Ireland.

Wolfgang Seeker and Jonas Kuhn. 2012. Making ellipses explicit in dependency conversion for a german treebank. In *Proceedings of the 8th International Conference on Language Resources and Evaluation*, pages 3132–3139, Istanbul, Turkey. European Language Resources Association (ELRA).

Karl Stratos, Michael Collins, and Daniel Hsu. 2015. Model-based word embeddings from decompositions of count matrices. In *Proceedings of ACL*.

Marek Świdziński and Marcin Woliński. 2010. Towards a bank of constituent parse trees for Polish. In *Proceedings of Text, Speech and Dialogue*, pages 197–204, Brno, Czech Republic.

Veronika Vincze, Dóra Szauter, Attila Almási, György Móra, Zoltán Alexin, and János Csirik. 2010. Hungarian dependency treebank. In *Proceedings of LREC*, Valletta, Malta.

Marcin Woliński, Katarzyna Głowińska, and Marek Świdziński. 2011. A preliminary version of Składnica—a treebank of Polish. In *Proceedings of the 5th Language & Technology Conference*, pages 299–303, Poznań, Poland.

Alina Wróblewska. 2012. Polish Dependency Bank. *Linguistic Issues in Language Technology*, 7(1):1–15.

Joint Event Trigger Identification and Event Coreference Resolution with Structured Perceptron

Jun Araki and **Teruko Mitamura**
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA 15213, USA
{junaraki,teruko}@cs.cmu.edu

Abstract

Events and their coreference offer useful semantic and discourse resources. We show that the semantic and discourse aspects of events interact with each other. However, traditional approaches addressed event extraction and event coreference resolution either separately or sequentially, which limits their interactions. This paper proposes a document-level structured learning model that simultaneously identifies event triggers and resolves event coreference. We demonstrate that the joint model outperforms a pipelined model by 6.9 BLANC F1 and 1.8 CoNLL F1 points in event coreference resolution using a corpus in the biology domain.

1 Introduction

Events convey semantic information such as who did what to whom where and when. They also corefer to each other, playing a role of discourse connection points to form a coherent story. These aspects of events have been already utilized in a wide variety of natural language processing (NLP) applications, such as automated population of knowledge bases (Ji and Grishman, 2011), topic detection and tracking (Allan, 2002), question answering (Bikel and Castelli, 2008), text summarization (Li et al., 2006), and contradiction detection (de Marneffe et al., 2008). This fact illustrates the importance of event extraction and event coreference resolution.

Those semantic and discourse aspects of events are not independent from each other, and in fact often work in interactive manners. We give two examples of the interactions:

(1) British bank Barclays had agreed to **buy**(E1) Spanish rival Banco Zaragozano for 1.14 billion euros. The **combination**(E2) of the banking operations of Barclays Spain and Zaragozano will bring together two complementary businesses.

(2) The Palestinian Authority condemned the **attack**(E3), saying **it**(E4) would divert international sympathy away from the far higher Palestinian civilian death toll.

E1 corefers to E2, and E3 does to E4. E2 is more abstract than E1, and has less evidence of being an event. E4 is a pronoun, and thus may seem to refer to an entity rather than an event. Thus, E2 and E4 are relatively difficult to be recognized as events by themselves. However, event coreference E1-E2, which is supported primarily by E2's participants *Barclays* and *Zaragozano* shared with E1, helps determine that E2 is an event. The same logic applies to E3 and E4. On the other hand, previous works typically rely on a pipelined model that extracts events (e.g., E1 and E3) at the first stage, and then resolves event coreference at the second stage. Although this modularity is preferable from development perspectives, the pipelined model limits the interactions. That is, the first stage alone is unlikely to detect E2 and E4 as events due to the difficulties described above. These missing events make it impossible for the second stage to resolve event coreference E1-E2 and E3-E4.

In this work, we address the problem using the ProcessBank corpus (Berant et al., 2014). Following the terminology defined in the corpus, we introduce several terms:

- **Event**: an abstract representation of a change of state, independent from particular texts.
- **Event trigger**: main word(s) in text, typically a verb or a noun that most clearly expresses an event.
- **Event arguments**: participants or attributes in text, typically nouns, that are involved in an event.
- **Event mention**: a clause in text that describes an event, and includes both a trigger and arguments.
- **Event coreference**: a linguistic phenomenon that two event mentions refer to the same event.

We aim to explore the interactions between event mentions and event coreference. As a first step toward the goal, we focus on the task of identifying event triggers and resolving event coreference, and

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 2074–2080,
Lisbon, Portugal, 17-21 September 2015.

propose a document-level joint learning model using structured perceptron (Collins, 2002) that simultaneously predicts them. Our assumption is that the joint model is able to capture the interactions between event triggers and event coreference adequately, and such comprehensive decision improves the system performance. For instance, the joint model is likely to extract E2 as well as E1 successfully via their event coreference by simultaneously looking at coreference features.

Our contributions are as follows:

1. This is the first work that simultaneously predicts event triggers and event coreference using a single joint model. At the core of the model is a document-level structured perceptron algorithm that learns event triggers and event coreference jointly.
2. The incremental token-based prediction in joint decoding poses a challenge of synchronizing the assignments of event triggers and coreference. To avoid this problem, we propose an incremental decoding algorithm that combines the segment-based decoding and best-first clustering algorithm.
3. Our experiments indicate that the joint model achieves a substantial performance gain in event coreference resolution with a corpus in the biology domain, as compared to a pipelined model.

2 Related Work

No previous work deals with event extraction and event coreference resolution simultaneously. We thus describe how these two tasks have been addressed separately, and how joint structured learning has been studied in other NLP tasks.

Event extraction has been studied mainly in the newswire domain and the biomedical domain as the task of detecting event triggers and determining their event types and arguments. In the former domain, most work took a pipelined approach where local classifiers identify triggers first, and then detect arguments (Ji and Grishman, 2008; Liao and Grishman, 2010; Hong et al., 2011). Li et al. (2013) presented a structured perceptron model to detect triggers and arguments jointly. Similarly, joint dependencies in events were also addressed in the latter domain (Poon and Vanderwende, 2010; McClosky et al., 2011; Riedel and McCallum, 2011; Venugopal et al., 2014). However, none of them incorporated event coreference into their model.

Event coreference resolution is more challenging and less explored. To set up event triggers as a starting point of the task, some works use human annotation in a corpus (Bejan and Harabagiu, 2014; Liu et al., 2014), and others use the output of a separate event extraction system (Lee et al., 2012). Berant et al. (2014) presented a model that jointly predicts event arguments and event coreference (as well as other relations between event triggers). However, none of them tries to predict event triggers and event coreference jointly.

Joint structured learning has been applied to several NLP tasks, such as word segmentation and part-of-speech (POS) tagging (Zhang and Clark, 2008a), POS tagging and dependency parsing (Bohnet and Nivre, 2012), dependency parsing and semantic role labeling (Johansson and Nugues, 2008), the extraction of event triggers and arguments (Li et al., 2013), and the extraction of entity mentions and relations (Li and Ji, 2014). Their underlying ideas are similar to ours. That is, one can train a structured learning model to globally capture the interactions between two relevant tasks via a certain kind of structure, while making predictions specifically for these respective tasks. However, no prior work has studied the interactions between event trigger identification and event coreference resolution.

3 Approach

We formalize the extraction of event triggers and event coreference as a problem of structured prediction. The output structure is a document-level event graph where each node represents an event trigger, and each edge represents an event coreference link between two event triggers.

3.1 Corpus

The ProcessBank corpus consists of 200 paragraphs from the textbook *Biology* (Campbell and Reece, 2005). Table 1 shows statistics of our data splits. The original corpus provides 150 paragraphs as training data, and we split them into 120 and 30 for our training and development, respectively. We chose ProcessBank instead of a larger corpus such as the Automatic Content Extraction (ACE) 2005 corpus for the following two reasons. First, the human annotation of event coreference links in ProcessBank enables us to apply the best-first clustering directly; on the other hand, this is

not readily feasible in ACE 2005 since it annotates event coreference as clusters, and gold standard event coreference links required for the best-first clustering are not available. Second, event coreference resolution using ProcessBank is novel since almost no previous work on the task used that corpus. The only exception could be (Berant et al., 2014), where they extracted several types of relations between event triggers, including event coreference. However, they did not report any performance scores of their system specifically on event coreference, and thus their work is not comparable to ours.

	Train	Dev	Test	Total
# of paragraphs	120	30	50	200
# of event triggers	823	224	356	1403
# of event coreferences	73	28	30	131

Table 1: Statistics of our dataset.

Unlike previous work (Berant et al., 2014; Li et al., 2013), we explicitly allow an event trigger to have multiple tokens, such as verb phrase 'look into' and compound proper noun 'World War II'. This is a more realistic setting for event trigger identification since in general there are a considerable number of multi-token event triggers[1].

3.2 Event Graph Learning

Let x denote an input document with n tokens where x_i is the i-th token in the document. For event graph learning, we use structured perceptron (Collins, 2002), and average weights to reduce overfitting as suggested in (Collins, 2002). The algorithm involves decoding to generate the best event graph for each input document. We elaborate on our decoding algorithm in Section 3.3. Since an event graph has an exponentially large search space, we use beam search to approximate exact inference. We extract a range of features by using Stanford CoreNLP (Manning et al., 2014), MATE (Björkelund et al., 2009), OpenNLP[2], Nomlex (Macleod et al., 1998), and Levin verb classes (Levin, 1993). For brevity, we provide details of the structured perceptron algorithm and features in the supplementary material.

We use the standard-update strategy in our structured perceptron model. As variants of structured perceptron, one could employ the early update (Collins and Roark, 2004) and max-violation update (Huang et al., 2012) to our model. Our initial experiments indicated that early updates happen too early to gain sufficient feedback on weights from entire documents in training examples, ending up with a poorer performance than the standard update. This contrasts with the fact that the early-update strategy was successfully applied to other NLP tasks such as constituent parsing (Collins and Roark, 2004) and dependency parsing (Zhang and Clark, 2008b). The main reason why the early update fell short of the standard update in our setting is that joint event trigger identification and event coreference resolution is a much more difficult task since they require more complex knowledge and argument structures. Due to the difficultly of the task, it is also very difficult to develop such an effective feature set that beam search can explore the search space of an entire document thoroughly with early updates. This observation follows (Björkelund and Kuhn, 2014) on entity coreference resolution. In contrast, the max-violation update showed almost the same performance as the standard update on the development data. From these results, we chose the standard-update strategy for simplicity.

3.3 Joint Decoding

Given that an event trigger has one or more tokens, event trigger identification could be solved as a token-level sequential labeling problem with `BIO` or `BILOU` scheme in the same way as named entity recognition (Ratinov and Roth, 2009). If one uses this approach, a beam state may represent a partial assignment of an event trigger. However, event coreference can be explored only from complete assignments of an event trigger. Thus, one would need to synchronize the search process of event coreference by comparing event coreferences from the complete assignment at a certain position with those from complete assignments at following positions. This makes it complicated to implement the formalization of token-level sequential labeling for joint decoding in our task. One possible way to avoid this problem is to extract event trigger candidates with a preference on high recall first, and then search event coreference from those candidates, regarding them as complete assignments of an event trigger. This recall-oriented pre-filtering is often used in entity coreference resolution (Lee et al., 2013; Björkelund

[1]For example, around 13.4% of the 1403 event triggers in ProcessBank have multiple tokens.

[2]http://opennlp.apache.org/

Algorithm 1 Joint decoding for event triggers and coreference with beam search.

Input: input document $x = (x_1, x_2, \ldots, x_n)$
Input: beam width k, max length of event trigger l_{max}
Output: best event graph $\hat{y}$ for x

```
 1: initialize empty beam history B[1..n]
 2: for i ← 1..n do
 3:   for l ← 1..l_max do
 4:     for y ∈ B[i − l] do
 5:       e ← CreateEventTrigger(l, i).
 6:       AppendEventTrigger(y, e)
 7:       B[i] ← k-Best(B[i] ∪ y)
 8:       for j ← 1..i − 1 do
 9:         c ← CreateEventCoref(j, e).
10:         AddEventCoref(y, c)
11:         B[i] ← k-Best(B[i] ∪ y)
12: return B[n][0]
```

and Farkas, 2012). In our initial experiments, we observed that our rule-based filter gained around 97% recall, but extracted around 12,400 false positives against 823 true positives in the training data. This made it difficult for our structured perceptron to learn event triggers, which underperformed on event coreference resolution.

We, therefore, employ segment-based decoding with multiple-beam search (Zhang and Clark, 2008a; Li and Ji, 2014) for event trigger identification, and combine it with the best-first clustering (Ng and Cardie, 2002) for event coreference resolution in document-level joint decoding. The key idea of segment-based decoding with multiple-beam search is to keep previous beam states available, and use them to form segments from previous positions to the current position. Let l_{max} denote the upper bound on the number of tokens in one event trigger. The k-best partial structures (event subgraphs) in beam B at the j-th token is computed as follows:

$$B[j] = \underset{y \in \{y_{[1:j-l]} \in B[j-l],\ y_{[j-l+1,j]}=s\}}{k\text{-BEST}} \Phi(x, y) \cdot \mathbf{w}$$

where $1 \leq l \leq l_{max}$, $y_{[1:j]}$ is an event subgraph ending at the j-th token, and $y_{[j-l+1,j]} = s$ means that partial structure $y_{[j-l+1,j]}$ is a segment, i.e., an event trigger candidate with a subsequence of tokens $x_{[j-l+1,j]}$. This approximates Viterbi decoding with beam search.

The best-first clustering incrementally makes coreference decisions by selecting the most likely antecedent for each trigger. Our joint decoding algorithm makes use of the incremental process to combine the segment-based decoding and best-first clustering. Algorithm 1 shows the summary of the joint decoding algorithm. Line 3 - 7 implements the segment-based decoding, and line 8 - 11 implements the best-first clustering. Once a new event trigger is appended to an event subgraph at line 6, the decoder uses it as a referring mention regardless of whether the event subgraph is in the beam, and seeks the best antecedent for it. This enables the joint model to make a more global decision on event trigger identification and event coreference decision, as described in Section 1.

4 Experimental Settings

When training our model, we observed that 20-iteration training almost reached convergence, and thus we set the number of iterations to 20. We set l_{max} to 6 because we observed that the longest event trigger in the entire ProcessBank corpus has six tokens. When tuning beam width k on the development set, large beam width did not give us a significant performance difference. We attribute this result to the small size of the development data. In particular, the development data has only 28 event coreferences, which makes it difficult to reveal the effect of beam width. We thus set k to 1 in our experiments.

4.1 Baseline Systems

Our baseline is a pipelined model that divides the event trigger decoding and event coreference decoding in Algorithm 1 into two separate stages. It uses the same structured perceptron with the same hyperparameters and feature templates. We choose this baseline because it clearly reveals the effectiveness of the joint model by focusing only on the architectural difference. One could develop other baseline systems. One of them is a deterministic sieve-based approach by Lee et al. (2013). A natural extension to the approach for performing event trigger identification as well as event coreference resolution would be to develop additional sieves to classify singletons into real event triggers or spurious ones. We leave it for future work.

4.2 Evaluation

We evaluate our system using a reference implementation of coreference scoring algorithms (Pradhan et al., 2014; Luo et al., 2014). As for event trigger identification, this scorer computes precision (P), recall (R), and the F1 score. With respect to event coreference resolution, the scorer computes MUC (Vilain et al., 1995), B^3 (Bagga and Baldwin, 1998), two CEAF metrics CEAF_m and CEAF_e (Luo, 2005), and

	MUC			B^3			$CEAF_m$			$CEAF_e$			BLANC			CoNLL
System	R	P	F1	R	P	F1	R	P	F1	R	P	F1	R	P	F1	F1
Baseline	26.66	19.51	22.53	55.47	58.64	57.01	53.08	60.38	56.50	52.68	63.14	57.44	30.13	25.10	25.05	45.66
Joint	20.00	37.50	26.08	53.37	63.36	57.93	53.93	62.95	58.09	55.06	62.11	58.38	27.51	38.43	31.91	47.45

Table 2: Results of event coreference resolution. 'Baseline' refers to the second stage of our baseline.

BLANC (Recasens and Hovy, 2011) extended by Luo et al. (2014). We also report the CoNLL average (Denis and Baldridge, 2009), which is the average of MUC F1, B^3 F1, and $CEAF_e$ F1.

5 Results and Discussions

We first show the result of event coreference resolution on the test data in Table 2. The joint model outperforms the baseline by 6.9 BLANC F1 and 1.8 CoNLL F1 points. We observed that this overall performance gain comes largely from a precision gain, more specifically, substantially reduced false positives. We explain the superiority of the joint model as follows. In the baseline, the second stage uses the output of the first stage. Since event triggers are fixed at this point, the baseline explores coreference links only between these event triggers. In contrast, the joint model seeks event triggers and event coreference simultaneously, and thus it explores a larger number of false positives in the search process, thereby learning to penalize false positives more adequately than the baseline.

System	Recall	Precision	F1
Baseline	57.02	64.85	60.68
Joint	55.89	65.24	60.21

Table 3: Results of event trigger identification. 'Baseline' refers to the first stage of our baseline.

Table 3 shows the results of event trigger identification on the test data. We observed that the joint model also reduced false positives, similarly in event coreference resolution. However, its improvement on precision is small, ending up with almost the same F1 point as the baseline. We speculate that this is due to the small size of the corpus, and the joint model was unable to show its advantages in event trigger identification.

Below are two error cases in event coreference resolution, where our model fails to resolve E5-E6 and E7-E8. The model was unable to adequately extract features for both event triggers and event coreference, particularly because their surface strings are not present in training data, they are lexically and syntactically different, and they do not share key semantic roles (e.g., agents and patients) in a clear argument structure.

(3) When the cell is stimulated, gated channels open that facilitate Na+ **diffusion**(E5). Sodium ions then **"fall"**(E6) down their electrochemical gradient, ...

(4) The next seven steps **decompose**(E7) the citrate back to oxaloacetate. It is this **regeneration**(E8) of oxaloacetate that makes this process a cycle.

6 Conclusion and Future Work

We present a joint structured prediction model for event trigger identification and event coreference resolution. To our knowledge, this is the first work that solves these two tasks simultaneously. Our experiment shows that the proposed method effectively penalizes false positives in joint search, thereby outperforming a pipelined model substantially in event coreference resolution.

There are a number of avenues for future work. One can further ensure the advantage of the joint model using a larger corpus. Our preliminary experiment on the ACE 2005 corpus shows that due to its larger document size and event types, one will need to reduce training time by a distributed learning algorithm such as mini-batches (Zhao and Huang, 2013). Another future work is to incorporate other components of events into the model. These include event types, event arguments, and other relations such as subevents. One could leverage them as other learning targets or constraints, and investigate further benefits of joint modeling.

Acknowledgments

This research was supported in part by DARPA grant FA8750-12-2-0342 funded under the Deep Exploration and Filtering of Text (DEFT) Program, and by U.S. Army Research Office (ARO) grant W911NF-14-1-0436 under the Reading, Extraction, and Assembly of Pathways for Evidentiary Reading (REAPER) Program. Any opinion, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the view of DARPA, ARO, or the U.S. government. Jun Araki is partly supported by a Funai Overseas Scholarship.

References

James Allan. 2002. *Topic Detection and Tracking: Event-based Information Organization*. Kluwer Academic Publishers.

Amit Bagga and Breck Baldwin. 1998. Algorithms for scoring coreference chains. In *Proceedings of LREC 1998 Workshop on Linguistics Coreference*, pages 563–566.

Cosmin Adrian Bejan and Sanda M. Harabagiu. 2014. Unsupervised event coreference resolution. *Computational Linguistics*, 40(2):311–347.

Jonathan Berant, Vivek Srikumar, Pei-Chun Chen, Abby Vander Linden, Brittany Harding, Brad Huang, Peter Clark, and Christopher D. Manning. 2014. Modeling biological processes for reading comprehension. In *Proceedings of EMNLP 2014*, pages 1499–1510.

Daniel M. Bikel and Vittorio Castelli. 2008. Event matching using the transitive closure of dependency relations. In *Proceedings of ACL 2008*, pages 145–148.

Anders Björkelund and Richárd Farkas. 2012. Data-driven multilingual coreference resolution using resolver stacking. In *Proceedings of EMNLP/CoNLL 2012*, pages 49–55.

Anders Björkelund and Jonas Kuhn. 2014. Learning structured perceptrons for coreference resolution with latent antecedents and non-local features. In *Proceedings of ACL 2014*, pages 47–57.

Anders Björkelund, Love Hafdell, and Pierre Nugues. 2009. Multilingual semantic role labeling. In *Proceedings of CoNLL 2009*, pages 43–48.

Bernd Bohnet and Joakim Nivre. 2012. A transition-based system for joint part-of-speech tagging and labeled non-projective dependency parsing. In *Proceedings of EMNLP/CoNLL 2012*, pages 1455–1465.

Neil Campbell and Jane Reece. 2005. *Biology*. Benjamin Cummings.

Michael Collins and Brian Roark. 2004. Incremental parsing with the perceptron algorithm. In *Proceedings of ACL 2004*, pages 111–118.

Michael Collins. 2002. Discriminative training methods for Hidden Markov Models: Theory and experiments with perceptron algorithms. In *Proceedings of EMNLP 2002*, pages 1–8.

Marie-Catherine de Marneffe, Anna N. Rafferty, and Christopher D. Manning. 2008. Finding contradictions in text. In *Proceedings of ACL-HLT 2008*, pages 1039–1047.

Pascal Denis and Jason Baldridge. 2009. Global joint models for coreference resolution and named entity classification. *Procesamiento del Lenguaje Natural*, 42:87–96.

Yu Hong, Jianfeng Zhang, Bin Ma, Jianmin Yao, Guodong Zhou, and Qiaoming Zhu. 2011. Using cross-entity inference to improve event extraction. In *Proceedings of ACL-HLT 2011*, pages 1127–1136.

Liang Huang, Suphan Fayong, and Yang Guo. 2012. Structured perceptron with inexact search. In *Proceedings of NAACL-HLT 2012*, pages 142–151.

Heng Ji and Ralph Grishman. 2008. Refining event extraction through cross-document inference. In *Proceedings of ACL-HLT 2008*, pages 254–262.

Heng Ji and Ralph Grishman. 2011. Knowledge base population: Successful approaches and challenges. In *Proceedings of ACL-HLT 2011*, pages 1148–1158.

Richard Johansson and Pierre Nugues. 2008. Dependency-based semantic role labeling of PropBank. In *Proceedings of EMNLP 2008*, pages 69–78.

Heeyoung Lee, Marta Recasens, Angel Chang, Mihai Surdeanu, and Dan Jurafsky. 2012. Joint entity and event coreference resolution across documents. In *Proceedings of EMNLP/CoNLL 2012*, pages 489–500.

Heeyoung Lee, Angel Chang, Yves Peirsman, Nathanael Chambers, Mihai Surdeanu, and Dan Jurafsky. 2013. Deterministic coreference resolution based on entity-centric, precision-ranked rules. *Computational Linguistics*, 39(4):885–916.

Beth Levin. 1993. *English Verb Classes and Alternation: A Preliminary Investigation*. The University of Chicago Press.

Qi Li and Heng Ji. 2014. Incremental joint extraction of entity mentions and relations. In *Proceedings of ACL 2014*, pages 402–412.

Wenjie Li, Mingli Wu, Qin Lu, Wei Xu, and Chunfa Yuan. 2006. Extractive summarization using inter- and intra- event relevance. In *Proceedings of ACL/COLING 2006*, pages 369–376.

Qi Li, Heng Ji, and Liang Huang. 2013. Joint event extraction via structured prediction with global features. In *Proceedings of ACL 2013*, pages 73–82.

Shasha Liao and Ralph Grishman. 2010. Using document level cross-event inference to improve event extraction. In *Proceedings of ACL 2010*, pages 789–797.

Zhengzhong Liu, Jun Araki, Eduard Hovy, and Teruko Mitamura. 2014. Supervised within-document event coreference using information propagation. In *Proceedings of LREC 2014*.

Xiaoqiang Luo, Sameer Pradhan, Marta Recasens, and Eduard Hovy. 2014. An extension of BLANC to system mentions. In *Proceedings of ACL 2014*, pages 24–29.

Xiaoqiang Luo. 2005. On coreference resolution performance metrics. In *Proceedings of HLT/EMNLP 2005*, pages 25–32.

Catherine Macleod, Ralph Grishman, Adam Meyers, Leslie Barrett, and Ruth Reeves. 1998. Nomlex: A lexicon of nominalizations. In *Proceedings of EURALEX 1998*, pages 187–193.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings ACL 2014: System Demonstrations*, pages 55–60.

David McClosky, Mihai Surdeanu, and Christopher Manning. 2011. Event extraction as dependency parsing. In *Proceedings of ACL-HLT 2011*, pages 1626–1635.

Vincent Ng and Claire Cardie. 2002. Improving machine learning approaches to coreference resolution. In *Proceedings of ACL 2002*, pages 104–111.

Hoifung Poon and Lucy Vanderwende. 2010. Joint inference for knowledge extraction from biomedical literature. In *Proceedings of NAACL-HLT 2010*, pages 813–821.

Sameer Pradhan, Xiaoqiang Luo, Marta Recasens, Eduard Hovy, Vincent Ng, and Michael Strube. 2014. Scoring coreference partitions of predicted mentions: A reference implementation. In *Proceedings of ACL 2014*, pages 30–35.

Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proceedings of CoNLL 2009*, pages 147–155.

Marta Recasens and Eduard Hovy. 2011. BLANC: Implementing the Rand index for coreference evaluation. *Natural Language Engineering*, 17(4):485–510.

Sebastian Riedel and Andrew McCallum. 2011. Fast and robust joint models for biomedical event extraction. In *Proceedings of EMNLP 2011*, pages 1–12.

Deepak Venugopal, Chen Chen, Vibhav Gogate, and Vincent Ng. 2014. Relieving the computational bottleneck: Joint inference for event extraction with high-dimensional features. In *Proceedings of EMNLP 2014*, pages 831–843.

Marc Vilain, John Burger, John Aberdeen, Dennis Connolly, and Lynette Hirschman. 1995. A model-theoretic coreference scoring scheme. In *Proceedings of MUC-6*, pages 45–52.

Yue Zhang and Stephen Clark. 2008a. Joint word segmentation and POS tagging using a single perceptron. In *Proceedings of ACL-HLT 2008*, pages 888–896.

Yue Zhang and Stephen Clark. 2008b. A tale of two parsers: Investigating and combining graph-based and transition-based dependency parsing. In *Proceedings of EMNLP 2008*, pages 562–571.

Kai Zhao and Liang Huang. 2013. Minibatch and parallelization for online large margin structured learning. In *Proceedings of NAACL-HLT 2013*, pages 370–379.

A Joint Dependency Model of Morphological and Syntactic Structure for Statistical Machine Translation

Rico Sennrich and **Barry Haddow**
School of Informatics, University of Edinburgh
rico.sennrich@ed.ac.uk, bhaddow@inf.ed.ac.uk

Abstract

When translating between two languages that differ in their degree of morphological synthesis, syntactic structures in one language may be realized as morphological structures in the other, and SMT models need a mechanism to learn such translations. Prior work has used morpheme splitting with flat representations that do not encode the hierarchical structure between morphemes, but this structure is relevant for learning morphosyntactic constraints and selectional preferences. We propose to model syntactic and morphological structure jointly in a dependency translation model, allowing the system to generalize to the level of morphemes. We present a dependency representation of German compounds and particle verbs that results in improvements in translation quality of 1.4–1.8 BLEU in the WMT English–German translation task.

1 Introduction

When translating between two languages that differ in their degree of morphological synthesis, syntactic structures in one language may be realized as morphological structures in the other. Machine Translation models that treat words as atomic units have poor learning capabilities for such translation units, and morphological segmentations are commonly used (Koehn and Knight, 2003). Like words in a sentence, the morphemes of a word have a hierarchical structure that is relevant in translation. For instance, compounds in Germanic languages are head-final, and the head is the segment that determines agreement within the noun phrase, and is relevant for selectional preferences of verbs.

1. sie erheben eine Hand|gepäck|gebühr.
   they charge a carry-on bag fee.

In example 1, agreement in case, number and gender is enforced between *eine* 'a' and *Gebühr* 'fee', and selectional preference between *erheben* 'charge' and *Gebühr* 'fee'. A flat representation, as is common in phrase-based SMT, does not encode these relationships, but a dependency representation does so through dependency links.

function/postion	English/German example
finite (main)	he **walks away** quickly er **geht** schnell **weg**
finite (sub.)	[...] because he **walks away** quickly [...] weil er schnell **weggeht**
bare infinitive	he can **walk away** quickly er kann schnell **weggehen**
to/zu-infinitive	he promises **to walk away** quickly er verspricht, schnell **wegzugehen**

Table 1: Surface realizations of particle verb *weggehen* 'walk away'.

In this paper, we investigate a dependency representation of morphologically segmented words for SMT. Our representation encodes syntactic and morphological structure jointly, allowing a single model to learn the translation of both. Specifically, we work with a string-to-tree model with GHKM-style rules (Galley et al., 2006), and a relational dependency language model (Sennrich, 2015). We focus on the representation of German syntax and morphology in an English-to-German system, and two morphologically complex word classes in German that are challenging for translation, compounds and particle verbs.

German makes heavy use of compounding, and compounds such as *Abwasserbehandlungsanlage* 'waste water treatment plant' are translated into complex noun phrases in other languages, such as French *station d'épuration des eaux résiduaires*.

German particle verbs are difficult to model because their surface realization differs depending on the finiteness of the verb and the type of clause. Verb particles are separated from the finite verb in

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 2081–2087,
Lisbon, Portugal, 17-21 September 2015.

main clauses, but prefixed to the verb in subordinated clauses, or when the verb is non-finite. The infinitive marker *zu* 'to', which is normally a premodifying particle, appears as an infix in particle verbs. Table 1 shows an illustrating example.

2 A Dependency Representation of Compounds and Particle Verbs

The main focus of research on compound splitting has been on the splitting algorithm (Popovic et al., 2006; Nießen and Ney, 2000; Weller et al., 2014; Macherey et al., 2011). Our focus is not the splitting algorithm, but the representation of compounds. For splitting, we use an approach similar to (Fritzinger and Fraser, 2010), with segmentation candidates identified by a finite-state morphology (Schmid et al., 2004; Sennrich and Kunz, 2014), and statistical evidence from the training corpus to select a split (Koehn and Knight, 2003).

German compounds are head-final, and premodifiers can be added recursively. Compounds are structurally ambiguous if there is more than one modifier. Consider the distinction between *(Stadtteil)projekt* (literally: '(city part) project)') and *Stadt(teilprojekt)* 'city sub-project'. We opt for a left-branching representation by default.[1] We also split linking elements, and represent them as a postmodifier of each non-final segment, including the empty string ("ϵ"). We use the same representation for noun compounds and adjective compounds.

An example of the original[2] and the proposed compound representation is shown in Figure 1. Importantly, the head of the compound is also the parent of the determiners and attributes in the noun phrase, which makes a bigram dependency language model sufficient to enforce agreement. Since we model morphosyntactic agreement within the main translation step, and not in a separate step as in (Fraser et al., 2012), we deem it useful that inflection is marked at the head of the compound. Consequently, we do not split off inflectional or derivational morphemes.

For German particle verbs, we define a common representation that abstracts away from the various surface realizations (see Table 1). Separated

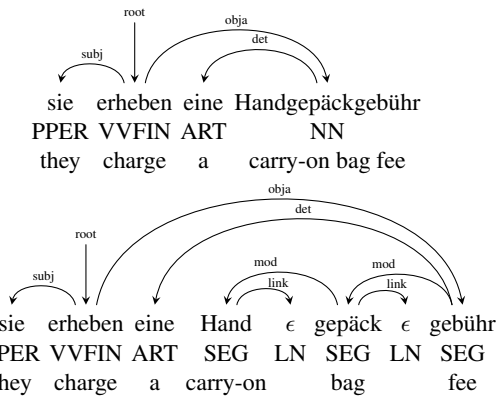


Figure 1: Original and proposed representation of German compound.

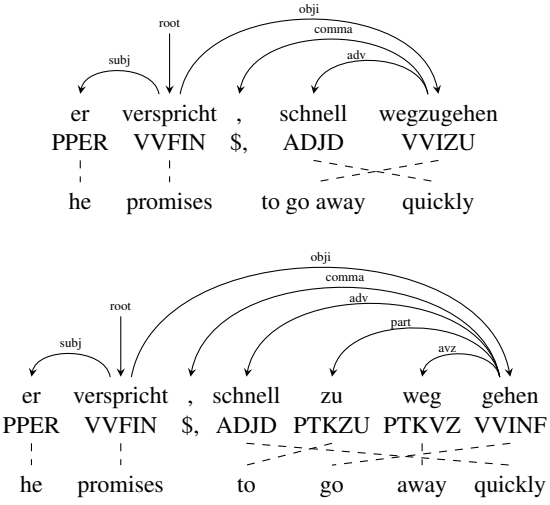


Figure 2: Original and proposed representation of German particle verb with infixed *zu*-marker.

verb particles are reordered to be the closest premodifier of the verb. Prefixed particles and the *zu*-infix are identified by the finite-state-morphology, and split from the verb so that the particle is the closest, the *zu* marker the next-closest premodifier of the verb, as shown in Figure 2. Agreement, selectional preferences, and other phenomena involve the verb and its dependents, and the proposed representation retains these dependency links, but reduces data sparsity from affixation and avoids discontinuity of the verb and its particle.

3 Tree Binarization

We follow Williams et al. (2014) and map dependency trees into a constituency representation, which allows for the extraction of GHKM-style translation rules (Galley et al., 2006). This conversion is lossless, and we can still apply a de-

[1] We follow prior work in leaving frequent words or subwords unsplit, which has a disambiguating effect. With more aggressive splitting, frequency information could be used for the structural disambiguation of internal structure.

[2] The original dependency trees follow the annotation guidelines by Foth (2005).

pendency language model (RDLM). Figure 3 (a) shows the constituency representation of the example in Figure 1.

Our model should not only be able to produce new words productively, but also to memorize words it has observed during training. Looking at the compound *Handgepäckgebühr* in Figure 3 (a), we can see that it does not form a constituent, and cannot be extracted with GHKM extraction heuristics. To address this, we binarize the trees in our training data (Wang et al., 2007).

A complicating factor is that the binarization should not impair the RDLM. During decoding, we map the internal tree structure of each hypothesis back to the unbinarized form, which is then scored by the RDLM. Virtual nodes introduced by the binarization must also be scorable by RDLM if they form the root of a translation hypothesis. A simple right or left binarization would produce virtual nodes without head and without meaningful dependency representation. We ensure that each virtual node dominates the head of the full constituent through a mixed binarization.[3] Specifically, we perform right binarization of the head and all pre-modifiers, then left binarization of all post-modifiers. This head-binarized representation is illustrated in Figure 3 (b).[4]

Head binarization ensures that even hypotheses whose root is a virtual node can be scored by the RDLM. This score is only relevant for pruning, and discarded when the full constituent is scored. Still, these hypotheses require special treatment in the RDLM to mitigate search errors. The virtual node labels (such as $\overline{\text{OBJA}}$) are unknown symbols to the RDLM, and we simply replace them with the original label (OBJA). The RDLM uses sibling context, and this is normally padded with special start and stop symbols, analogous to BOS/EOS symbols in n-gram models. These start and stop symbols let the RDLM compute the probability that a node is the first or last child of its ancestor node. However, computing these probabilities for virtual nodes would unfairly bias the search, since the first/last child of a virtual node is not necessarily the first/last child of the full constituent. We adapt the representation of virtual nodes in

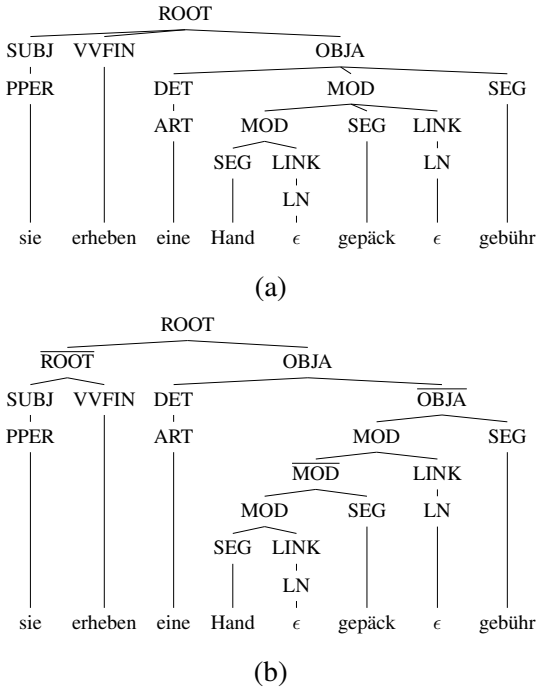


Figure 3: Unbinarized (a) and head-binarized (b) constituency representation of Figure 1.

RDLM to take this into account. We distinguish between virtual nodes based on whether their span is a string prefix, suffix, or infix of the full constituent. For prefixes and infixes, we do not add a stop symbol at the end, and use null symbols, which denote unavailable context, for padding to the right. For suffixes and infixes, we do the same at the start.

4 Post-Processing

For SMT, all German training and development data is converted into the representation described in sections 2–3. To restore the original representation, we start from the tree output of the string-to-tree decoder. Merging compounds is trivial: all segments and linking elements can be identified by the tree structure, and are concatenated.

For verbs that dominate a verb particle, the original order is restored through three rules:

1. non-finite verbs are concatenated with the particle, and *zu*-markers are infixed.

2. finite verbs that head a subordinated clause (identified by its dependency label) are concatenated with the particle.

3. finite verbs that head a main clause have the

[3] In other words, every node is a fixed well-formed dependency structure (Shen et al., 2010) with our binarization.

[4] Note that our definition of head binarization is different from that of Wang et al. (2007), who left-binarize a node if the head is the first child, and right-binarize otherwise. Our algorithm also covers cases where the head has both pre- and post-modifiers, as *erheben* and *gepäck* do in Figure 3.

particle moved to the right clause bracket.[5]

Previous work on particle verb translation into German proposed to predict the position of particles with an n-gram language model (Nießen and Ney, 2001). Our rules have the advantage that they are informed by the syntax of the sentence and consider the finiteness of the verb.

Our rules only produce projective trees. Verb particles may also appear in positions that violate projectivity, and we leave it to future research to determine if our limitation to projective trees affects translation quality, and how to produce non-projective trees.

5 SMT experiments

5.1 Data and Models

We train English–German string-to-tree SMT systems on the training data of the shared translation task of the Workshop on Statistical Machine Translation (WMT) 2015. The data set consists of 4.2 million sentence pairs of parallel data, and 160 million sentences of monolingual German data.

We base our systems on that of Williams et al. (2014). It is a string-to-tree GHKM translation system implemented in Moses (Koehn et al., 2007), and using the dependency annotation by ParZu (Sennrich et al., 2013). Additionally, our baseline system contains a dependency language model (RDLM) (Sennrich, 2015), trained on the target-side of the parallel training data.

We report case-sensitive BLEU scores on the newstest2014/5 test sets from WMT, averaged over 3 optimization runs of k-batch MIRA (Cherry and Foster, 2012) on a subset of newstest2008-12.[6]

We split all particle verbs and hyphenated compounds, but other compounds are only split if they are rare (frequency in parallel text < 5).

For comparison with the state-of-the-art, we train a *full system* on our restructured representation, which incorporates all models and settings of our WMT 2015 submission system (Williams et al., 2015).[7] Note that our WMT 2015 submission uses the dependency representation of compounds and tree binarization introduced in this paper; we achieve additional gains over the submission system through particle verb restructuring.

system	newstest2014	newstest2015
baseline	20.7	22.0
+split compounds	21.3	22.4
+particle verbs	21.4	22.8
head binarization	20.9	22.7
+split compounds	22.0	23.4
+particle verbs	22.1	23.8
full system	22.6	24.4

Table 2: English–German translation results (BLEU). Average of three optimization runs.

system	compound	particle verb		
		sep.	pref.	*zu*-infix
reference	2841	553	1195	176
baseline	845	96	847	71
+head binarization	798	157	858	106
+split compounds	1850	160	877	94
+particle verbs	1992	333	953	169

Table 3: Number of compounds [that would be split by compound splitter] and particle verbs (separated, prefixed and with *zu*-infix) in newstest2014/5. Average of three optimization runs.

5.2 SMT Results

Table 2 shows translation quality (BLEU) with different representations of German compounds and particle verbs. Head binarization not only yields improvements over the baseline, but also allows for larger gains from morphological segmentation. We attribute this to the fact that full compounds, and prefixed particle verbs, are not always a constituent in the segmented representation, and that binarization compensates this theoretical drawback.

With head binarization, we find substantial improvements from compound splitting of 0.7–1.1 BLEU. On newstest2014, the improvement is almost twice of that reported in related work (Williams et al., 2014), which also uses a hierarchical representation of compounds, albeit one that does not allow for dependency modelling. Examples of correct, unseen compounds generated include *Staubsauger|roboter* 'vacuum cleaner robot', *Gravitation|s|wellen* 'gravitational waves', and *NPD|-|verbot|s|verfahren* 'NPD banning process'.[8]

[5] We use the last position in the clause as default location, but put the particle before any subordinated and coordinated clauses, which occur in the *Nachfeld* (the 'final field' in topological field theory).

[6] We use *mteval-v13a.pl* for comparability to official WMT results; all significance values reported are obtained with MultEval (Clark et al., 2011).

[7] In contrast to our other systems in this paper, RDLM is trained on all monolingual data for the full system, and two models are added: a 5-gram Neural Network language model (Vaswani et al., 2013), and soft source-syntactic constraints (Huck et al., 2014).

[8] Note that *Staubsauger*, despite being a compound, is not

Particle verb restructuring yields additional gains of 0.1–0.4 BLEU. One reason for the smaller effect of particle verb restructuring is that the difficult cases – separated particle verbs and those with infixation – are rarer than compounds, with 2841 rare compounds [that would be split by our compound splitter] in the reference texts, in contrast to 553 separated particle verbs, and 176 particle verbs with infixation, as Table 3 illustrates. If we only evaluate the sentences containing a particle verb with *zu*-infix in the reference, 165 in total for newstest2014/5, we observe an improvement of 0.8 BLEU on this subset (22.1→22.9), significant with $p < 0.05$.

The positive effect of restructuring is also apparent in frequency statistics. Table 3 shows that the baseline system severely undergenerates compounds and separated/infixed particle verbs. Binarization, compound splitting, and particle verb restructuring all contribute to bringing the distribution of compounds and particle verbs closer to the reference.

In total, the restructured representation yields improvements of 1.4–1.8 BLEU over our baseline. The *full system* is competitive with official submissions to the WMT 2015 shared translation tasks. It outperforms our submission (Williams et al., 2015) by 0.4 BLEU, and outperforms other phrase-based and syntax-based submissions by 0.8 BLEU or more. The best reported result according to BLEU is an ensemble of Neural MT systems (Jean et al., 2015), which achieves 24.9 BLEU. In the human evaluation, both our submission and the Neural MT system were ranked 1–2 (out of 16), with no significant difference between them.

5.3 Synthetic LM Experiment

We perform a synthetic experiment to test our claim that a dependency representation allows for the modelling of agreement between morphemes. For 200 rare compounds [that would be split by our compound splitter] in the newstest2014/5 references, we artificially introduce agreement errors by changing the gender of the determiner. For instance, we create the erroneous sentence *sie erheben ein Handgepäckgebühr* as a complement to Example 1. We measure the ability of language models to prefer (give a higher probability to) the original reference sentence over the erroneous one. In the original representation, both a Kneser-Ney 5-gram LM and RDLM perform poorly due to data sparseness, with 70% and 57.5% accuracy, respectively. In the split representation, the RDLM reliably prefers the correct agreement (96.5% accuracy), whilst the performance of the 5-gram model even deteriorates (to 60% accuracy). This is because the gender of the first segment(s) is irrelevant, or even misleading, for agreement. For instance, *Handgepäck* is neuter, which could lead a morpheme-level n-gram model to prefer the determiner *ein*, but *Handgepäckgebühr* is feminine and requires *eine*.

segmented due to its frequency.

6 Conclusion

Our main contribution is that we exploit the hierarchical structure of morphemes to model them jointly with syntax in a dependency-based string-to-tree SMT model. We describe the dependency annotation of two morphologically complex word classes in German, compounds and particle verbs, and show that our tree representation yields improvements in translation quality of 1.4–1.8 BLEU in the WMT English–German translation task.[9]

The principle of jointly representing syntactic and morphological structure in dependency trees can be applied to other language pairs, and we expect this to be helpful for languages with a high degree of morphological synthesis. However, the annotation needs to be adapted to the respective languages. For example, French compounds such as *arc-en-ciel* 'rainbow' are head-initial, in contrast to head-final Germanic compounds.

Acknowledgments

This project received funding from the European Union's Horizon 2020 research and innovation programme under grant agreements 645452 (QT21), 644402 (HimL), 644333 (TraMOOC), and from the Swiss National Science Foundation under grant P2ZHP1_148717.

References

Colin Cherry and George Foster. 2012. Batch Tuning Strategies for Statistical Machine Translation. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL HLT '12, pages 427–436, Montreal, Canada. Association for Computational Linguistics.

[9] We released source code and configuration files at `https://github.com/rsennrich/wmt2014-scripts`.

Jonathan H. Clark, Chris Dyer, Alon Lavie, and Noah A. Smith. 2011. Better Hypothesis Testing for Statistical Machine Translation: Controlling for Optimizer Instability. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, pages 176–181, Portland, Oregon. Association for Computational Linguistics.

Killian A. Foth. 2005. *Eine umfassende Constraint-Dependenz-Grammatik des Deutschen*. University of Hamburg, Hamburg.

Alexander Fraser, Marion Weller, Aoife Cahill, and Fabienne Cap. 2012. Modeling Inflection and Word-Formation in SMT. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 664–674, Avignon, France. Association for Computational Linguistics.

Fabienne Fritzinger and Alexander Fraser. 2010. How to Avoid Burning Ducks: Combining Linguistic Analysis and Corpus Statistics for German Compound Processing. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, WMT '10, pages 224–234, Uppsala, Sweden. Association for Computational Linguistics.

Michel Galley, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang, and Ignacio Thayer. 2006. Scalable inference and training of context-rich syntactic translation models. In *ACL-44: Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 961–968, Sydney, Australia. Association for Computational Linguistics.

Matthias Huck, Hieu Hoang, and Philipp Koehn. 2014. Preference Grammars and Soft Syntactic Constraints for GHKM Syntax-based Statistical Machine Translation. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 148–156, Doha, Qatar. Association for Computational Linguistics.

Sébastien Jean, Orhan Firat, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. Montreal Neural Machine Translation Systems for WMT'15 . In *Proceedings of the Tenth Workshop on Statistical Machine Translation*.

Philipp Koehn and Kevin Knight. 2003. Empirical Methods for Compound Splitting. In *EACL '03: Proceedings of the Tenth Conference on European Chapter of the Association for Computational Linguistics*, pages 187–193, Budapest, Hungary. Association for Computational Linguistics.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open Source Toolkit for Statistical Machine Translation. In *Proceedings of the ACL-2007 Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic. Association for Computational Linguistics.

Klaus Macherey, Andrew Dai, David Talbot, Ashok Popat, and Franz Och. 2011. Language-independent compound splitting with morphological operations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1395–1404, Portland, Oregon, USA. Association for Computational Linguistics.

Sonja Nießen and Hermann Ney. 2000. Improving SMT quality with morpho-syntactic analysis. In *18th Int. Conf. on Computational Linguistics*, pages 1081–1085.

Sonja Nießen and Hermann Ney. 2001. Morpho-syntactic analysis for Reordering in Statistical Machine Translation. In *Machine Translation Summit*, pages 247–252, Santiago de Compostela, Spain.

Maja Popovic, Daniel Stein, and Hermann Ney. 2006. Statistical Machine Translation of German Compound Words. In *Advances in Natural Language Processing, 5th International Conference on NLP, FinTAL 2006*, pages 616–624, Turku, Finland.

Helmut Schmid, Arne Fitschen, and Ulrich Heid. 2004. A German Computational Morphology Covering Derivation, Composition, and Inflection. In *Proceedings of the IVth International Conference on Language Resources and Evaluation (LREC 2004)*, pages 1263–1266.

Rico Sennrich and Beat Kunz. 2014. Zmorge: A German Morphological Lexicon Extracted from Wiktionary. In *Proceedings of the 9th International Conference on Language Resources and Evaluation (LREC 2014)*, Reykjavik, Iceland, May.

Rico Sennrich, Martin Volk, and Gerold Schneider. 2013. Exploiting Synergies Between Open Resources for German Dependency Parsing, POS-tagging, and Morphological Analysis. In *Proceedings of the International Conference Recent Advances in Natural Language Processing 2013*, pages 601–609, Hissar, Bulgaria.

Rico Sennrich. 2015. Modelling and Optimizing on Syntactic N-Grams for Statistical Machine Translation. *Transactions of the Association for Computational Linguistics*, 3:169–182.

Libin Shen, Jinxi Xu, and Ralph Weischedel. 2010. String-to-dependency Statistical Machine Translation. *Comput. Linguist.*, 36(4):649–671.

Ashish Vaswani, Yinggong Zhao, Victoria Fossum, and David Chiang. 2013. Decoding with Large-Scale Neural Language Models Improves Translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013*, pages 1387–1392, Seattle, Washington, USA.

Wei Wang, Kevin Knight, and Daniel Marcu. 2007. Binarizing Syntax Trees to Improve Syntax-Based Machine Translation Accuracy. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*.

Marion Weller, Fabienne Cap, Stefan Müller, Sabine Schulte im Walde, and Alexander Fraser. 2014. Distinguishing Degrees of Compositionality in Compound Splitting for Statistical Machine Translation. In *Proceedings of the First Workshop on Computational Approaches to Compound Analysis (ComAComA 2014)*, pages 81–90, Dublin, Ireland.

Philip Williams, Rico Sennrich, Maria Nadejde, Matthias Huck, Eva Hasler, and Philipp Koehn. 2014. Edinburgh's Syntax-Based Systems at WMT 2014. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 207–214, Baltimore, Maryland, USA. Association for Computational Linguistics.

Philip Williams, Rico Sennrich, Maria Nadejde, Matthias Huck, and Philipp Koehn. 2015. Edinburgh's Syntax-Based Systems at WMT 2015. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, Lisbon, Portugal. Association for Computational Linguistics.

Variable-Length Word Encodings for Neural Translation Models

Rohan Chitnis and **John DeNero**
Computer Science Division
University of California, Berkeley
{ronuchit,denero}@berkeley.edu

Abstract

Recent work in neural machine translation has shown promising performance, but the most effective architectures do not scale naturally to large vocabulary sizes. We propose and compare three variable-length encoding schemes that represent a large vocabulary corpus using a much smaller vocabulary with no loss in information. Common words are unaffected by our encoding, but rare words are encoded using a sequence of two pseudo-words. Our method is simple and effective: it requires no complete dictionaries, learning procedures, increased training time, changes to the model, or new parameters. Compared to a baseline that replaces all rare words with an *unknown word* symbol, our best variable-length encoding strategy improves WMT English-French translation performance by up to 1.7 BLEU.

1 Introduction

Bahdanau et al. (2014) propose a neural translation model that learns vector representations for individual words as well as word sequences. Their approach jointly predicts a translation and a latent word-level alignment for a sequence of source words. However, the architecture of the network does not scale naturally to large vocabularies (Jean et al., 2014).

In this paper, we propose a novel approach to circumvent the large-vocabulary challenge by preprocessing the source and target word sequences, encoding them as a longer token sequence drawn from a small vocabulary that does not discard any information. Common words are unaffected, but rare words are encoded as a sequence of two pseudo-words. The exact same learning and inference machinery applied to these transformed data yields improved translations.

We evaluate a family of 3 different encoding schemes based on Huffman codes. All of them eliminate the need to replace rare words with the *unknown word* symbol. Our approach is simpler than other methods recently proposed to address the same issue. It does not introduce new parameters into the model, change the model structure, affect inference, require access to a complete dictionary, or require any additional learning procedures. Nonetheless, compared to a baseline system that replaces all rare words with an *unknown word* symbol, our encoding approach improves English-French news translation by up to 1.7 BLEU.

2 Background

2.1 Neural Machine Translation

Neural machine translation describes approaches to machine translation that learn from corpora in a single integrated model that embeds words and sentences into a vector space (Kalchbrenner and Blunsom, 2013; Cho et al., 2014; Sutskever et al., 2014). We focus on one recent approach to neural machine translation, proposed by Bahdanau et al. (2014), that predicts both a translation and its alignment to the source sentence, though our technique is relevant to related approaches as well.

The architecture consists of an encoder and a decoder. The encoder receives a source sentence $\mathbf{x}$ and encodes each prefix using a recurrent neural network that recursively combines embeddings x_j for each word position j:

$$\overrightarrow{h}_j = f(x_j, \overrightarrow{h}_{j-1}) \quad (1)$$

where f is a non-linear function. Reverse encodings $\overleftarrow{h}_j$ are computed similarly to represent suffixes of the sentence. These vector representations are stacked to form h_j, a representation of the

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 2088–2093,
Lisbon, Portugal, 17-21 September 2015.

whole sentence focused on position j.

The decoder predicts each target word y_i sequentially according to the distribution

$$P(y_i|y_{i-1}, ..., y_1, \mathbf{x}) = g(y_{i-1}, s_i, c_i) \quad (2)$$

where s_i is a hidden decoder state summarizing the prefix of the translation generated so far, c_i is a summary of the entire input sequence, and g is another non-linear function. Encoder and decoder parameters are jointly optimized to maximize the log-likelihood of a training corpus.

Depending on the approach to neural translation, c can take multiple forms. Bahdanau et al. (2014) propose integrating an attention mechanism in the decoder, which is trained to determine on which portions of the source sentence to focus. The decoder computes c_i, the summarizing context vector, as a convex combination of the h_j. The coefficients of this combination are proportional (softmax) to an alignment model prediction $\exp a(h_j, s_i)$, where a is a non-linear function.

The speed of prediction scales with the output vocabulary size, due to the denominator of Equation 2 (Jean et al., 2014). The input vocabulary size is also a challenge for storage and learning. As a result, neural machine translation systems only consider the top 30K to 100K most frequent words in a training corpus, replacing the other words with an *unknown word* symbol.

2.2 Related Work

There has been much recent work in improving translation quality by addressing these vocabulary size challenges. Luong et al. (2014) describe an approach that, similar to ours, treats the translation system as a black box. They eliminate unknown symbols by training the system to recognize from where in the source text each unknown word in the target text came, so that in a postprocessing phase, the unknown word can be replaced by a dictionary lookup of the corresponding source word. In contrast, our method does not rely on access to a complete dictionary, and instead transforms the data to allow the system itself to learn translations for even the rare words.

Some approaches have altered the model to circumvent the expensive normalization computation, rather than applying preprocessing and postprocessing on the text. Jean et al. (2014) develop an importance sampling strategy for approximating the softmax computation. Mnih and Kavukcuoglu (2013) present a technique for approximation of the target word probability using noise-contrastive estimation.

Sequential or hierarchical encodings of large vocabularies have played an important role in recurrent neural network language models, primarily to address the inference time issue of large vocabularies. Mikolov et al. (2011b) describe an architecture in which output word types are grouped into classes by frequency: the network first predicts a class, then a word in that class. Mikolov et al. (2013) describe an encoding of the output vocabulary as a binary tree. To our knowledge, hierarchical encodings have not been applied to the input vocabulary of a machine translation system.

Other methods have also been developed to work around large-vocabulary issues in language modeling. Morin and Bengio (2005), Mnih and Hinton (2009), and Mikolov et al. (2011a) develop hierarchical versions of the softmax computation; Huang et al. (2012) and Collobert and Weston (2008) remove the need for normalization, thus avoiding computation of the summation term over the entire vocabulary.

2.3 Huffman Codes

An encoding can be used to represent a sequence of tokens from a large vocabulary $\mathcal{V}$ using a small vocabulary $\mathcal{W}$. In the case of translation, let $\mathcal{V}$ be the original corpus vocabulary, which can number in the millions of word types in a typical corpus. Let $\mathcal{W}$ be the vocabulary size of a neural translation model, typically set to a much smaller number such as 30,000.

A deterministically invertible, variable-length encoding maps each $v \in \mathcal{V}$ to a sequence $w \in \mathcal{W}+$ such that no other $v' \in \mathcal{V}$ is mapped to a prefix of w. Encoding simply replaces each element of $\mathcal{V}$ according to the map, and decoding is unambiguous because of this prefix restriction. An encoding can be represented as a tree in which each leaf corresponds to an element of $\mathcal{V}$, each node contains a symbol from $\mathcal{W}$, and the encoding of any leaf is its path from the root.

A Huffman code is an optimal encoding that uses as few symbols from $\mathcal{W}$ as possible to encode an original sequence of symbols from $\mathcal{V}$. Although binary codes are typical, $\mathcal{W}$ can have any size. An optimal encoding can be found using a greedy algorithm (Huffman, 1952).

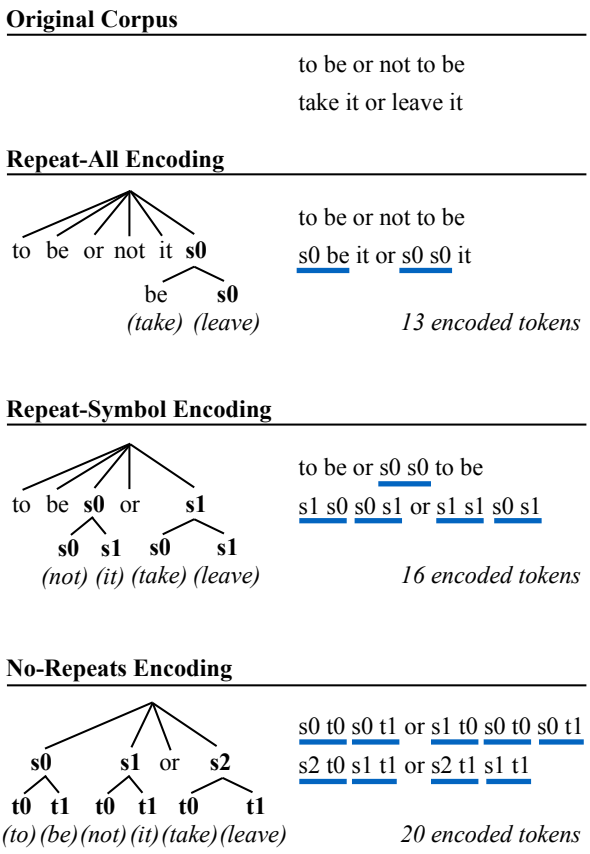


Figure 1: Our three encoding schemes are applied to a two-sentence toy corpus for which each word type appears one or two times, and the total vocabulary size V is 7. An optimal encoding tree under each scheme is shown for an encoded vocabulary size W of 6. As stricter constraints are imposed on the encoding, the encoded corpus length increases and the number of elements of V that can be represented using a single symbol decreases. Two-symbol encodings of rare words are underlined.

3 Variable-Length Encoding Methods

We consider three different encoding schemes that are based on Huffman codes. The encoding for a toy corpus under each scheme is depicted in Figure 1. While a Huffman code achieves the shortest possible encoded length using a fixed vocabulary size W, symbols are often shared between both common words and rare words. The variants we consider are designed to prevent specific forms of symbol sharing across encodings.

3.1 Encoding Schemes

Repeat-All. The first scheme is a standard Huffman code. In our experiments with $V \approx 2 \cdot 10^6$, $W = 3 \cdot 10^4$, and frequencies drawn from the WMT corpus, all words in $\mathcal{V}$ are encoded as either a single symbol or two symbols of $\mathcal{W}$. We denote the single-symbol words (which have the highest frequency) as *common*, and we call the other words *rare*. The *Repeat-All* encoding scheme has the highest number of common words. In Figure 1, common words are represented as themselves. Rare words are represented by two words, and the first is always a pseudo-word symbol introduced into $\mathcal{W}$ of the form **sX** for an integer X.

Repeat-Symbol. The *Repeat-Symbol* encoding scheme does not allow common-word symbols to appear in the encoding of rare words. Instead, each rare word is encoded as a two-symbol sequence of the form "**sX sY**," where X and Y are integers that may be the same or different. This scheme decreases the number of common words in order to encode all rare words using a restricted set of symbols. In this scheme, a common word in the encoded vocabulary always corresponds to a common word in the original vocabulary, reducing ambiguity of common word symbols at the expense of increasing ambiguity of pseudo-word symbols.

No-Repeats. Our final encoding scheme, *No-Repeats*, uses a different vocabulary for the first and second symbols in each rare word. That is, rare words are represented as "**sX tY**," where X and Y are integers that may be the same or different. In this scheme, common words and rare words do not share symbols, and each symbol can immediately be identified as common, the first of a rare encoding pair, or the second of a rare encoding pair.

3.2 Symbol Counts

To maximize performance, it is critical to set the number of common words (which transform to themselves) as high as possible while satisfying the desired total vocabulary size, counting all the newly introduced symbols. In this section, we algebraically derive this optimal number of common words for each encoding scheme. We define the following:

V: Size of the original vocabulary.

W: Size of the encoded vocabulary.

C: Number of common words.

S: Number of pseudo-words of the form **sX**.

T: Number of pseudo-words of the form **tX**.

We are interested in maximizing C so that total encoding length is minimized.

Repeat-All. We would like to encode the $V - C$ rare words, using only $W - C$ new symbols. To do so, for each new symbol (non-terminal node in our encoding tree), we have all W symbols under it in that branch. Therefore, we maximize C satisfying the constraint that

$$V - C \leq (W - C) \cdot W$$

Repeat-Symbol. Out of the $V - C$ rare words, we would like to pack them into a complete tree so that they may be encoded using our remaining $W - C$ symbols. Therefore, we maximize C satisfying the constraint that

$$V - C \leq (W - C)^2$$

No-Repeats. Again, we desire to pack $V - C$ rare words into a complete tree where we may use $W - C$ symbols. To maximize C, we let $S = T$. Because $S + T + C = W$, we have that $2S + C = W$. Therefore, we maximize C satisfying the constraint that

$$V - C \leq \left(\frac{W - C}{2}\right)^2$$

4 Experimental Results

We trained a public implementation[1] of the system described in Bahdanau et al. (2014) on the English-French parallel corpus from ACL WMT 2014, which contains 348M tokens. We evaluated on news-test-2014, also from WMT 2014, which contains 3003 sentences. All experiments used the same learning parameters and vocabulary size of 30,000.

We constructed each encoding by the following method. First, we used the formulas derived in the previous section to calculate the optimal number of common words C for each encoding scheme, using V to be the true vocabulary size of the training corpus and $W = 30,000$. We then found the C most common words in the text and encoded them as themselves. For the remaining rare words, we encoded them using a distinct symbol whose form matched the one prescribed for each encoding scheme. The encoding was then applied separately to both the source text and the target text. Our encoding schemes all increased the total number of tokens in the training corpus by approximately 4%.

[1] github.com/lisa-groundhog/GroundHog

Encoding	BLEU	# Common Words
None	25.77	30,000
Repeat-All	27.45	29,940
Repeat-Symbol	26.52	28,860
No-Repeats	25.79	27,320

Table 1: BLEU scores (%) on detokenized test set for each encoding scheme after training for 5 days.

To construct the mapping from rare words to their 2-word encodings, we binned rare words by frequency into branches. Thus, rare words of similar frequency in the training corpus tended to have encodings with the same first symbol. Similarly, the standard Huffman construction algorithm groups together rare words with similar frequencies within subtrees. More intelligent heuristics for constructing trees, such as using translation statistics instead of training corpus frequency, would be an interesting area of future work.

4.1 Results

We used the RNNsearch-50 architecture from Bahdanau et al. (2014) as our machine translation system. We report results for this system alone, as well as for each of our three encoding schemes, using the BLEU metric (Papineni et al., 2002). Table 1 summarizes our results after training each variant for 5 days, corresponding to roughly 2 passes through the 180K-sentence training corpus.

Alternative techniques that leverage bilingual resources have been shown to provide larger improvements. Jean et al. (2014) demonstrate an improvement of 3.1 BLEU by using bilingual word co-occurrence statistics in an aligned corpus to replace *unknown word* tokens. Luong et al. (2014) demonstrate an improvement of up to 2.8 BLEU over a series of stronger baselines using an unknown word model that also makes predictions using a bilingual dictionary.

4.2 Analysis

Our results indicate that the encoding scheme that keeps the highest number of common words, *Repeat-All*, performs best. Table 2 shows the unigram precision of each output. The common word translation accuracy is higher for all encoding schemes than for the baseline, although all preci-

Encoding	Common	Rare	1st Symbol
None	62.0	0.0	-
Repeat-All	65.8	28.0	64.8
Repeat-Symbol	65.5	16.5	24.8
No-Repeats	63.6	15.8	25.7

Table 2: Test set precision (%) on common words and rare words for each encoding strategy. *1st Symbol* denotes the precision of the first pseudo-word symbol in an encoded rare word.

sions are similar. Larger differences appear in the precision of rare words. The scheme that encodes rare words using both pseudo-words and common words gives substantially higher rare word accuracy than any other approach.

The final column of Table 2 shows the unigram precision of the first pseudo-word in an encoded rare word. The *Repeat-All* scheme uses only 60 different first symbols to encode all rare words. The other schemes require over 1,000. The fact that *Repeat-All* has a constrained set of rare word first symbols may account for its higher rare word precision.

It is possible for the model to predict an invalid encoded sequence that does not correspond to any word in the original vocabulary. However, in our experiments, we did not observe any such sequences in the decoding of the test set. A reasonable way to deal with invalid sequences would be to drop them from the output during decoding.

5 Conclusion and Future Work

We described a novel approach for encoding the source and target text based on Huffman coding schemes, eliminating the use of the *unknown word* symbol. An important continuation of our work would be to develop heuristics for effectively grouping "similar" words in the source and target text, so that they tend to have encodings that share a symbol. Even with our naive grouping by corpus frequency, our approach offers a simple way to predict both common and rare words in a neural translation model. As a result, performance improves by up to 1.7 BLEU. We expect that the simplicity of our technique will allow for straightforward combination with other enhancements and neural models.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the International Conference on Machine Learning*.

Eric H. Huang, Richard Socher, Christopher D. Manning, and Andrew Y. Ng. 2012. Improving word representations via global context and multiple word prototypes. In *Proceedings of the Association for Computational Linguistics*.

David A. Huffman. 1952. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*.

Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2014. On using very large target vocabulary for neural machine translation. *CoRR*, abs/1412.2007.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models.

Thang Luong, Ilya Sutskever, Quoc V. Le, Oriol Vinyals, and Wojciech Zaremba. 2014. Addressing the rare word problem in neural machine translation. *CoRR*, abs/1410.8206.

Tomáš Mikolov, Anoop Deoras, Daniel Povey, Lukáš Burget, and Jan Černocký. 2011a. Strategies for training large scale neural network language models. In *Proceedings of ASRU*.

Tomáš Mikolov, S. Kombrink, L. Burget, J.H. Cernocky, and Sanjeev Khudanpur. 2011b. Extensions of recurrent neural network language model. In *Acoustics, Speech and Signal Processing*.

Tomáš Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

Andriy Mnih and Geoffrey E. Hinton. 2009. A scalable hierarchical distributed language model. In *Advances in Neural Information Processing Systems*.

Andriy Mnih and Koray Kavukcuoglu. 2013. Learning word embeddings efficiently with noise-contrastive estimation. In *Advances in Neural Information Processing Systems*.

Frederic Morin and Yoshua Bengio. 2005. Hierarchical probabilistic neural network language model. In *AI Stats*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the Association for Computational Linguistics*.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. *CoRR*, abs/1409.3215.

A Binarized Neural Network Joint Model for Machine Translation

Jingyi Zhang[1,2], **Masao Utiyama**[1], **Eiichro Sumita**[1]
Graham Neubig[2], **Satoshi Nakamura**[2]
[1]National Institute of Information and Communications Technology,
3-5Hikaridai, Keihanna Science City, Kyoto 619-0289, Japan
[2]Graduate School of Information Science, Nara Institute of Science and Technology,
Takayama, Ikoma, Nara 630-0192, Japan
jingyizhang/mutiyama/eiichiro.sumita@nict.go.jp
neubig/s-nakamura@is.naist.jp

Abstract

The neural network joint model (NNJM), which augments the neural network language model (NNLM) with an m-word source context window, has achieved large gains in machine translation accuracy, but also has problems with high normalization cost when using large vocabularies. Training the NNJM with noise-contrastive estimation (NCE), instead of standard maximum likelihood estimation (MLE), can reduce computation cost. In this paper, we propose an alternative to NCE, the binarized NNJM (BNNJM), which learns a binary classifier that takes both the context and target words as input, and can be efficiently trained using MLE. We compare the BNNJM and NNJM trained by NCE on various translation tasks.

1 Introduction

Neural network translation models, which learn mappings over real-valued vector representations in high-dimensional space, have recently achieved large gains in translation accuracy (Hu et al., 2014; Devlin et al., 2014; Sundermeyer et al., 2014; Auli et al., 2013; Schwenk, 2012; Sutskever et al., 2014; Bahdanau et al., 2015).

Notably, Devlin et al. (2014) proposed a neural network joint model (NNJM), which augments the n-gram neural network language model (NNLM) with an m-word source context window, as shown in Figure 1a. While this model is effective, the computation cost of using it in a large-vocabulary SMT task is quite expensive, as probabilities need to be normalized over the entire vocabulary. To solve this problem, Devlin et al. (2014) presented a technique to train the NNJM to be self-normalized and avoided the expensive normalization cost during decoding. However, they also

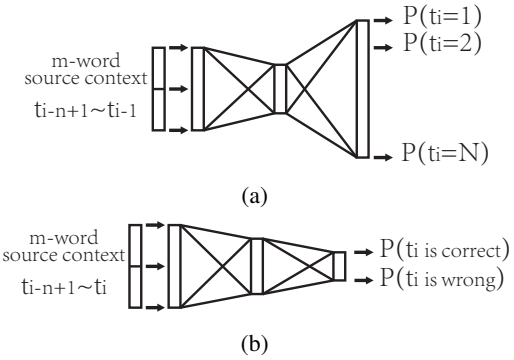


Figure 1: (a) the traditional NNJM and (b) the proposed BNNJM

note that this self-normalization technique sacrifices neural network accuracy, and the training process for the self-normalized neural network is very slow, as with standard maximum likelihood estimation (MLE).

To remedy the problem of long training times in the context of NNLMs, Vaswani et al. (2013) used a method called noise contrastive estimation (NCE). Compared with MLE, NCE does not require repeated summations over the whole vocabulary and performs nonlinear logistic regression to discriminate between the observed data and artificially generated noise.

This paper proposes an alternative framework of binarized NNJMs (BNNJM), which are similar to the NNJM, but use the current target word not as the output, but as the input of the neural network, estimating whether the target word under examination is correct or not, as shown in Figure 1b. Because the BNNJM uses the current target word as input, the information about the current target word can be combined with the context word information and processed in the hidden layers.

The BNNJM learns a simple binary classifier, given the context and target words, therefore it can be trained by MLE very efficiently. "Incorrect" target words for the BNNJM can be generated in the same way as NCE generates noise

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 2094–2099,
Lisbon, Portugal, 17-21 September 2015.

for the NNJM. We present a novel noise distribution based on translation probabilities to train the NNJM and the BNNJM efficiently.

2 Neural Network Joint Model

Let $T = t_1^{|T|}$ be a translation of $S = s_1^{|S|}$. The NNJM (Devlin et al., 2014) defines the following probability,

$$P(T|S) = \prod_{i=1}^{|T|} P\left(t_i | s_{a_i-(m-1)/2}^{a_i+(m-1)/2}, t_{i-n+1}^{i-1}\right) \quad (1)$$

where target word t_i is affiliated with source word s_{a_i}. Affiliation a_i is derived from the word alignments using heuristics[1]. To estimate these probabilities, the NNJM uses m source context words and $n-1$ target history words as input to a neural network and performs estimation of unnormalized probabilities $p(t_i|C)$ before normalizing over all words in the target vocabulary V,

$$\begin{aligned} P(t_i|C) &= \frac{p(t_i|C)}{Z(C)} \\ Z(C) &= \sum_{t_i' \in V} p(t_i'|C) \end{aligned} \quad (2)$$

where C stands for source and target context words as in Equation 1.

The NNJM can be trained on a word-aligned parallel corpus using standard MLE, but the cost of normalizing over the entire vocabulary to calculate the denominator in Equation 2 is quite large. Devlin et al. (2014)'s self-normalization technique can avoid normalization cost during decoding, but not during training.

NCE can be used to train NNLM-style models (Vaswani et al., 2013) to reduce training times. NCE creates a noise distribution $q(t_i)$, selects k noise samples $t_{i1}, ..., t_{ik}$ for each t_i and introduces a random variable v which is 1 for training examples and 0 for noise samples,

$$\begin{aligned} P(v=1, t_i|C) &= \frac{1}{1+k} \cdot \frac{p(t_i|C)}{Z(C)} \\ P(v=0, t_i|C) &= \frac{k}{1+k} \cdot q(t_i). \end{aligned}$$

NCE trains the model to distinguish training data from noise by maximize the conditional likelihood,

$$L = \log P(v=1|C, t_i) + \sum_{j=1}^{k} \log P(v=0|C, t_{ik}).$$

The normalization cost can be avoided by using $p(t_i|C)$ as an approximation of $P(t_i|C)$.[2]

[1] If t_i aligns to exactly one source word, a_i is the index of this source word; If t_i aligns to multiple source words, a_i is the index of the aligned word in the middle; If t_i is unaligned, they inherit its affiliation from the closest aligned word.

[2] The theoretical properties of self-normalization techniques, including NCE and Devlin et al. (2014)'s method, are investigated by Andreas and Klein (2015).

3 Binarized NNJM

In this paper, we propose a new framework of the binarized NNJM (BNNJM), which is similar to the NNJM but learns not to predict the next word given the context, but solves a binary classification problem by adding a variable $v \in \{0, 1\}$ that stands for whether the current target word t_i is correctly/wrongly produced in terms of source context words $s_{a_i-(m-1)/2}^{a_i+(m-1)/2}$ and target history words t_{i-n+1}^{i-1},

$$P\left(v | s_{a_i-(m-1)/2}^{a_i+(m-1)/2}, t_{i-n+1}^{i-1}, t_i\right).$$

The BNNJM is learned by a feed-forward neural network with $m+n$ inputs $\left\{s_{a_i-(m-1)/2}^{a_i+(m-1)/2}, t_{i-n+1}^{i-1}, t_i\right\}$ and two outputs for $v = 1/0$.

Because the BNNJM uses the current target word as input, the information about the current target word can be combined with the context word information and processed in the hidden layers. Thus, the hidden layers can be used to learn the difference between correct target words and noise in the BNNJM, while in the NNJM the hidden layers just contain information about context words and only the output layer can be used to discriminate between the training data and noise, giving the BNNJM more power to learn this classification problem.

We can use the BNNJM probability in translation as an approximation for the NNJM as below,

$$\begin{aligned} &P\left(t_i | s_{a_i-(m-1)/2}^{a_i+(m-1)/2}, t_{i-n+1}^{i-1}\right) \\ &\approx P\left(v=1 | s_{a_i-(m-1)/2}^{a_i+(m-1)/2}, t_{i-n+1}^{i-1}, t_i\right). \end{aligned}$$

As a binary classifier, the gradient for a single example in the BNNJM can be calculated efficiently by MLE without it being necessary to calculate the softmax over the full vocabulary. On the other hand, we need to create "positive" and "negative" examples for the classifier. Positive examples can be extracted directly from the word-aligned parallel corpus as $\left\langle s_{a_i-(m-1)/2}^{a_i+(m-1)/2}, t_{i-n+1}^{i-1}, t_i \right\rangle$; Negative examples can be generated for each positive example in the same way that NCE generates noise data as $\left\langle s_{a_i-(m-1)/2}^{a_i+(m-1)/2}, t_{i-n+1}^{i-1}, t_i' \right\rangle$, where $t_i' \in V \setminus \{t_i\}$.

4 Noise Sampling

4.1 Unigram Noise

Vaswani et al. (2013) adopted the unigram probability distribution (UPD) to sample noise for train-

Figure 2: A parallel sentence pair.

ing NNLMs with NCE,

$$q\left(t_i'\right) = \frac{occur(t_i')}{\sum\limits_{t_i'' \in V} occur(t_i'')}$$

where $occur\left(t_i'\right)$ stands for how many times t_i' occurs in the training corpus.

4.2 Translation Model Noise

In this paper, we propose a noise distribution specialized for translation models, such as the NNJM or BNNJM.

Figure 2 gives a Chinese-to-English parallel sentence pair with word alignments to demonstrate the intuition behind our method. Focusing on s_{a_i}="安排", this is translated into t_i ="*arrange*". For this positive example, UPD is allowed to sample any arbitrary noise, such as t_i' = "*banana*". However, in this case, noise t_i' = "*banana*" is not useful for model training, as constraints on possible translations given by the phrase table ensure that "安排" will never be translated into "*banana*". On the other hand, noise t_i' = "*arranges*" and "*arrangement*" are both possible translations of "安排" and therefore useful training data, that we would like our model to penalize.

Based on this intuition, we propose the use of another noise distribution that only uses t_i' that are possible translations of s_{a_i}, i.e., $t_i' \in U\left(s_{a_i}\right) \setminus \{t_i\}$, where $U\left(s_{a_i}\right)$ contains all target words aligned to s_{a_i} in the parallel corpus.

Because $U\left(s_{a_i}\right)$ may be quite large and contain many wrong translations caused by wrong alignments, "*banana*" may actually be included in U("安排"). To mitigate the effect of uncommon examples, we use a translation probability distribution (TPD) to sample noise t_i' from $U\left(s_{a_i}\right) \setminus \{t_i\}$ as follows,

$$q\left(t_i'|s_{a_i}\right) = \frac{align\left(s_{a_i}, t_i'\right)}{\sum_{t_i'' \in U\left(s_{a_i}\right)} align\left(s_{a_i}, t_i''\right)}$$

where $align\left(s_{a_i}, t_i'\right)$ is how many times t_i' is aligned to s_{a_i} in the parallel corpus.

Note that t_i could be unaligned, in which case we assume that it is aligned to a special $null$ word. Noise for unaligned words is sampled according to the TPD of the $null$ word. If several target/source words are aligned to one source/target word, we choose to combine these target/source words as a new target/source word.[3]

5 Experiments

5.1 Setting

We evaluated the effectiveness of the proposed approach for Chinese-to-English (CE), Japanese-to-English (JE) and French-to-English (FE) translation tasks. The datasets officially provided for the patent machine translation task at NTCIR-9 (Goto et al., 2011) were used for the CE and JE tasks. The development and test sets were both provided for the CE task while only the test set was provided for the JE task. Therefore, we used the sentences from the NTCIR-8 JE test set as the development set. Word segmentation was done by BaseSeg (Zhao et al., 2006) for Chinese and Mecab[4] for Japanese. For the FE language pair, we used standard data for the WMT 2014 translation task. The training sets for CE, JE and FE tasks contain 1M, 3M and 2M sentence pairs, respectively.

For each translation task, a recent version of Moses HPB decoder (Koehn et al., 2007) with the training scripts was used as the baseline (Base). We used the default parameters for Moses, and a 5-gram language model was trained on the target side of the training corpus using the IRSTLM Toolkit[5] with improved Kneser-Ney smoothing. Feature weights were tuned by MERT (Och, 2003).

The word-aligned training set was used to learn the NNJM and the BNNJM.[6] For both NNJM and BNNJM, we set $m = 7$ and $n = 5$. The NNJM was trained by NCE using UPD and TPD as noise distributions. The BNNJM was trained by standard MLE using UPD and TPD to generate negative examples.

The number of noise samples for NCE was set to be 100. For the BNNJM, we used only one negative example for each positive example in each training epoch, as the BNNJM needs to calculate

[3]The processing for multiple alignments helps sample more useful negative examples for TPD, and had little effect on the translation performance when UPD was used as the noise distribution for the NNJM and the BNNJM in our preliminary experiments.

[4]http://sourceforge.net/projects/mecab/files/

[5]http://hlt.fbk.eu/en/irstlm

[6]Both the NNJM and the BNNJM had one hidden layer, 100 hidden nodes, input embedding dimension 50, output embedding dimension 50. A small set of training data was used as validation data. The training process was stopped when validation likelihood stopped increasing.

		CE		JE		FE	
		E	T	E	T	E	T
NNJM	UPD	204	22	196	49	204	28
	TPD						
BNNJM	UPD	1411	16	129	34	119	22
	TPD						

Table 1: Epochs (E) and time (T) in minutes per epoch for each task.

		CE	JE	FE
Base		32.95	30.13	24.56
NNJM	UPD	34.36+	**31.30+**	24.68
	TPD	34.60+	**31.50+**	24.80
BNNJM	UPD	32.89	30.04	24.50
	TPD	**35.05+***	**31.42+**	**25.84+***

Table 2: Translation results. The symbol + and * represent significant differences at the $p < 0.01$ level against Base and NNJM+UPD, respectively. Significance tests were conducted using bootstrap resampling (Koehn, 2004).

the whole neural network (not just the output layer like the NNJM) for each noise sample and thus noise computation is more expensive. However, for different epochs, we resampled the negative example for each positive example, so the BNNJM can make use of different negative examples.

5.2 Results and Discussion

Table 1 shows how many epochs these two models needed and the training time for each epoch on a 10-core 3.47GHz Xeon X5690 machine.[7] Translation results are shown in Table 2.

We can see that using TPD instead of UPD as a noise distribution for the NNJM trained by NCE can speed up the training process significantly, with a small improvement in performance. But for the BNNJM, using different noise distributions affects translation performance significantly. The BNNJM with UPD does not improve over the baseline system, likely due to the small number of noise samples used in training the BNNJM, while the BNNJM with TPD achieves good performance, even better than the NNJM with TPD on the Chinese-to-English and French-to-English translation tasks.

From Table 2, the NNJM does not improve translation performance significantly on the FE task. Note that the baseline BLEU for the FE task is lower than CE and JE tasks, indicating that learning is harder for the FE task than CE and JE tasks. The validation perplexities of the NNJM with UPD for CE, JE and FE tasks are 4.03, 3.49 and 8.37. Despite these difficult learning circumstances and lack of large gains for the NNJM, the BNNJM improves translations significantly for the FE task, suggesting that the BNNJM is more robust to difficult translation tasks that are hard for the NNJM.

[7]The decoding time for the NNJM and the BNNJM were similar, since the NNJM trained by NCE uses $p(t_i|C)$ as an approximation of $P(t_i|C)$ without normalization and the BNNJM only needs to be normalized over two output neurons.

S: 该(this) 移动(movement) 持续(continued) 到(until) 寄生虫(parasite) 由(by) 两(two) 个 舌(tongues) 部 21 彼此(each other) 接触(contact) 时(where) 的 点(point) 接触(touched) 。
R: *this movement is continued until the parasite is touched by the point where the two tongues 21 contact each other .*
T_1**:** *the mobile continues to the parasite from the two tongue 21 contacts the points of contact with each other .*
T_2**:** *this movement is continued until the parasite by two tongue 21 contact points of contact with each other .*

Table 3: Translation examples. Here, S: source; R: reference; T_1 uses NNJM; T_2 uses BNNJM.

	NNJM	BNNJM
该− >the	1.681	-0.126
移动− >mobile	-4.506	-3.758
持续− >continues	-1.550	-0.130
到− >to	2.510	-0.220
SUM	-1.865	-4.236
该− >this	-2.414	-0.649
移动− >movement	-1.527	-0.200
$null$− >is	0.006	-0.055
持续− >continued	-0.292	-0.249
到− >until	-6.846	-0.186
SUM	-11.075	-1.341

Table 4: Scores for different translations.

Table 3 gives Chinese-to-English translation examples to demonstrate how the BNNJM (with TPD) helps to improve translations over the NNJM (with TPD). In this case, the BNNJM helps to translate the phrase "该 移动 持续 到" better. Table 4 gives translation scores for these two translations calculated by the NNJM and the BNNJM. Context words are used for predictions but not shown in the table.

As can be seen, the BNNJM prefers T_2 while the NNJM prefers T_1. Among these predictions, the NNJM and the BNNJM predict the translation for "到" most differently. The NNJM clearly predicts that in this case "到" should be translated into "to" more than "until", likely because this example rarely occurs in the training corpus. However, the BNNJM prefers "until" more than "to", which

demonstrates the BNNJM's robustness to less frequent examples.

5.3 Analysis for JE Translation Results

Finally, we examine the translation results to explore why the BNNJM with TPD did not outperform the NNJM with TPD for the JE translation task, as it did for the other translation tasks. We found that using the BNNJM instead of the NNJM on the JE task did improve translation quality significantly for infrequent words, but not for frequent words.

First, we describe how we estimate translation quality for infrequent words. Suppose we have a test set S, a reference set R and a translation set T with I sentences,

$$S_i\,(1 \leq i \leq I)\,, R_i\,(1 \leq i \leq I)\,, T_i\,(1 \leq i \leq I)$$

T_i contains J individual words,

$$W_{ij} \in Words\,(T_i)$$

$T_o\,(W_{ij})$ is how many times W_{ij} occurs in T_i and $R_o\,(W_{ij})$ is how many times W_{ij} occurs in R_i.

The general 1-gram translation accuracy (Papineni et al., 2002) is calculated as,

$$P_g = \frac{\sum_{i=1}^{I}\sum_{j=1}^{J} \min\left(T_o(W_{ij}), R_o(W_{ij})\right)}{\sum_{i=1}^{I}\sum_{j=1}^{J} T_o(W_{ij})}$$

This general 1-gram translation accuracy does not distinguish word frequency.

We use a modified 1-gram translation accuracy that weights infrequent words more heavily,

$$P_c = \frac{\sum_{i=1}^{I}\sum_{j=1}^{J} \min(T_o(W_{ij}), R_o(W_{ij})) \cdot \frac{1}{Occur\left(W_{ij}\right)}}{\sum_{i=1}^{I}\sum_{j=1}^{J} T_o(W_{ij})}$$

where $Occur\,(W_{ij})$ is how many times W_{ij} occurs in the whole reference set. Note P_c will not be 1 even in the case of completely accurate translations, but it can approximately reflect infrequent word translation accuracy, since correct frequent word translations contribute less to P_c.

Table 5 shows P_g and P_c for different translation tasks. It can be seen that the BNNJM improves infrequent word translation quality similarly for all translation tasks, but improves general translation quality less for the JE task than the other translation tasks. We conjecture that the reason why the BNNJM is less useful for frequent word translations on the JE task is the fact that the JE parallel corpus has less accurate function word alignments than other language pairs, as the grammatical features of Japanese and English are quite different.[8] Wrong function word alignments will make noise sampling less effective and therefore lower the BNNJM performance for function word translations. Although wrong word alignments will also make noise sampling less effective for the NNJM, the BNNJM only uses one noise sample for each positive example, so wrong word alignments affect the BNNJM more than the NNJM.

	CE		JE		FE	
	P_g	P_c	P_g	P_c	P_g	P_c
NNJM	70.3	5.79	68.2	4.15	61.2	6.70
BNNJM	70.9	5.97	68.4	4.30	61.7	6.86
Imp. (%)	0.85	3.1	0.29	3.6	0.81	2.4

Table 5: 1-gram precisions and improvements.

6 Related Work

Xu et al. (2011) proposed a method to use binary classifiers to learn NNLMs. But they also used the current target word in the output, similarly to NCE. The BNNJM uses the current target word as input, so the information about the current target word can be combined with the context word information and processed in hidden layers.

Mauser et al. (2009) presented discriminative lexicon models to predict target words. They train a separate classifier for each target word, as these lexicon models use discrete representations of words and different classifiers do not share features. In contrast, the BNNJM uses real-valued vector representations of words and shares features, so we train one classifier and can use the similarity information between words.

7 Conclusion

This paper proposes an alternative to the NNJM, the BNNJM, which learns a binary classifier that takes both the context and target words as input and combines all useful information in the hidden layers. We also present a novel noise distribution based on translation probabilities to train the BNNJM efficiently. With the improved noise sampling method, the BNNJM can achieve comparable performance with the NNJM and even improve the translation results over the NNJM on Chinese-to-English and French-to-English translations.

[8] Infrequent words are usually content words and frequent words are usually function words.

References

Jacob Andreas and Dan Klein. 2015. When and why are log-linear models self-normalizing? In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 244–249.

Michael Auli, Michel Galley, Chris Quirk, and Geoffrey Zweig. 2013. Joint language and translation modeling with recurrent neural networks. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1044–1054.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations*.

Jacob Devlin, Rabih Zbib, Zhongqiang Huang, Thomas Lamar, Richard Schwartz, and John Makhoul. 2014. Fast and robust neural network joint models for statistical machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1370–1380.

Isao Goto, Bin Lu, Ka Po Chow, Eiichiro Sumita, and Benjamin K Tsou. 2011. Overview of the patent machine translation task at the NTCIR-9 workshop. In *Proceedings of The 9th NII Test Collection for IR Systems Workshop Meeting*, pages 559–578.

Yuening Hu, Michael Auli, Qin Gao, and Jianfeng Gao. 2014. Minimum translation modeling with recurrent neural networks. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 20–29.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 388–395.

Arne Mauser, Saša Hasan, and Hermann Ney. 2009. Extending statistical machine translation with discriminative and trigger-based lexicon models. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 210–218.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 160–167.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318.

Holger Schwenk. 2012. Continuous space translation models for phrase-based statistical machine translation. In *Proceedings of International Conference on Computational Linguistics : Posters*, pages 1071–1080.

Martin Sundermeyer, Tamer Alkhouli, Joern Wuebker, and Hermann Ney. 2014. Translation modeling with bidirectional recurrent neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 14–25.

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

Ashish Vaswani, Yinggong Zhao, Victoria Fossum, and David Chiang. 2013. Decoding with large-scale neural language models improves translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1387–1392.

Puyang Xu, Asela Gunawardana, and Sanjeev Khudanpur. 2011. Efficient subsampling for training complex language models. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1128–1136.

Hai Zhao, Chang-Ning Huang, and Mu Li. 2006. An improved Chinese word segmentation system with conditional random field. In *Proceedings of the Fifth SIGHAN Workshop on Chinese Language Processing*, pages 162–165.

Bayesian Optimization of Text Representations

Dani Yogatama **Lingpeng Kong**
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213, USA
{dyogatama,lingpenk}@cs.cmu.edu

Noah A. Smith
Computer Science & Engineering
University of Washington
Seattle, WA 98195, USA
nasmith@cs.washington.edu

Abstract

When applying machine learning to problems in NLP, there are many choices to make about how to represent input texts. They can have a big effect on performance, but they are often uninteresting to researchers or practitioners who simply need a module that performs well. We apply sequential model-based optimization over this space of choices and show that it makes standard linear models competitive with more sophisticated, expensive state-of-the-art methods based on latent variables or neural networks on various topic classification and sentiment analysis problems. Our approach is a first step towards black-box NLP systems that work with raw text and do not require manual tuning.

1 Introduction

NLP researchers and practitioners spend a considerable amount of time comparing machine-learned models of text that differ in relatively uninteresting ways. For example, in categorizing texts, should the "bag of words" include bigrams, and is tf-idf weighting a good idea? In learning word embeddings, distributional similarity approaches have been shown to perform competitively with neural network models when the hyperparameters (e.g., context window, subsampling rate, smoothing constant) are carefully tuned (Levy et al., 2015). These choices matter experimentally, often leading to big differences in performance, with little consistency across tasks and datasets in which combination of choices works best. Unfortunately, these differences tell us little about language or the problems that machine learners are supposed to solve.

We propose that these decisions can be automated in a similar way to hyperparameter selection (e.g., choosing the strength of a ridge or lasso regularizer). Given a particular text dataset and classification task, we show a technique for optimizing over the space of representational choices, along with other "nuisances" that interact with these decisions, like hyperparameter selection. For example, using higher-order n-grams means more features and a need for stronger regularization and more training iterations. Generally, these decisions about instance representation are made by humans, heuristically; our work seeks to automate them, not unlike Daelemans et al. (2003), who proposed to use genetic algorithms to optimize representational choices.

Our technique instantiates sequential model-based optimization (SMBO; Hutter et al., 2011). SMBO and other Bayesian optimization approaches have been shown to work well for hyperparameter tuning (Bergstra et al., 2011; Hoffman et al., 2011; Snoek et al., 2012). Though popular in computer vision (Bergstra et al., 2013), these techniques have received little attention in NLP.

We apply it to logistic regression on a range of topic and sentiment classification tasks. Consistently, our method finds representational choices that perform better than linear baselines previously reported in the literature, and that, in some cases, are competitive with more sophisticated non-linear models trained using neural networks.

2 Problem Formulation and Notation

Let the training data consist of a collection of pairs $\boldsymbol{d}_{train} = \langle\langle d.i_1, d.o_1\rangle, \ldots, \langle d.i_n, d.o_n\rangle\rangle$, where each input $d.i \in \mathcal{I}$ is a text document and each output $d.o \in \mathcal{O}$, the output space. The overall training goal is to maximize a performance function f (e.g., classification accuracy, log-likelihood, F_1 score, etc.) of a machine-learned model, on a held-out dataset, $\boldsymbol{d}_{dev} \in (\mathcal{I} \times \mathcal{O})^{n'}$.

Classification proceeds in three steps: first, $\mathbf{x} : \mathcal{I} \rightarrow \mathbb{R}^N$ maps each input to a vector representation. Second, a predictive model (typically, its parameters) is learned from the inputs (now transformed into vectors) and outputs: $L : (\mathbb{R}^N \times \mathcal{O})^n \rightarrow (\mathbb{R}^N \rightarrow \mathcal{O})$. Finally, the resulting classifier $c : \mathcal{I} \rightarrow \mathcal{O}$ is fixed as $L(\boldsymbol{d}_{train}) \circ \mathbf{x}$ (i.e., the composition of the representation function with

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 2100–2105,
Lisbon, Portugal, 17-21 September 2015.

the learned mapping).

Here we consider linear classifiers of the form $c(d.i) = \arg\max_{o \in \mathcal{O}} \mathbf{w}_o^\top \mathbf{x}(d.i)$, where the parameters $\mathbf{w}_o \in \mathbb{R}^N$, for each output o, are learned using logistic regression on the training data. We let $\mathbf{w}$ denote the concatenation of all $\mathbf{w}_o$. Hence the parameters can be understood as a function of the training data and the representation function $\mathbf{x}$. The performance function f, in turn, is a function of the held-out data $\boldsymbol{d}_{dev}$ and $\mathbf{x}$—also $\mathbf{w}$ and $\boldsymbol{d}_{train}$, through $\mathbf{x}$. For simplicity, we will write "$f(\mathbf{x})$" when the rest are clear from context.

Typically, $\mathbf{x}$ is fixed by the model designer, perhaps after some experimentation, and learning focuses on selecting the parameters $\mathbf{w}$. For logistic regression and many other linear models, this training step reduces to convex optimization in $N|\mathcal{O}|$ dimensions—a solvable problem that is costly for large datasets and/or large output spaces. In seeking to maximize f with respect to $\mathbf{x}$, we do not wish to carry out training any more times than necessary.

Choosing $\mathbf{x}$ can be understood as a problem of selecting *hyperparameter values*. We therefore turn to Bayesian optimization, a family of techniques that can be used to select hyperparameter values intelligently when solving for parameters ($\mathbf{w}$) is costly.

3 Bayesian Optimization

Our approach is based on sequential model-based optimization (SMBO; Hutter et al., 2011). It iteratively chooses representation functions $\mathbf{x}$. On each round, it makes this choice through a probabilistic model of f, then evaluates f—we call this a "trial." As in any iterative search algorithm, the goal is to balance exploration of options for $\mathbf{x}$ with exploitation of previously-explored options, so that a good choice is found in a small number of trials.

More concretely, in the tth trial, $\mathbf{x}_t$ is selected using an acquisition function $\mathcal{A}$ and a "surrogate" probabilistic model p_t. Second, f is evaluated given $\mathbf{x}_t$—an expensive operation which involves training to learn parameters $\mathbf{w}$ and assessing performance on the held-out data. Third, the surrogate model is updated. See Algorithm 1; details on $\mathcal{A}$ and p_t follow.

Acquisition Function. A good acquisition function returns high values for $\mathbf{x}$ when either the value $f(\mathbf{x})$ is predicted to be high, or the uncertainty about $f(\mathbf{x})$'s value is high; balancing between these is the classic tradeoff between exploitation and exploration. We use a criterion called Expected Improvement (EI; Jones, 2001),[1] which is the expectation (under the current surrogate model p_t) that $f(\mathbf{x}) = y$ will exceed $f(\mathbf{x}^*) = y^*$:

$$\mathcal{A}(\mathbf{x}; p_t, y^*) = \int_{-\infty}^{\infty} \max(y - y^*, 0) p_t(y \mid \mathbf{x}) dy$$

where $\mathbf{x}^*$ is chosen depending on the surrogate model, discussed below. (For now, think of it as a strongly-performing "benchmark" discovered in earlier iterations.) Other options for the acquisition function include maximum probability of improvement (Jones, 2001), minimum conditional entropy (Villemonteix et al., 2009), Gaussian process upper confidence bound (Srinivas et al., 2010), or a combination of them (Hoffman et al., 2011).

Algorithm 1 SMBO algorithm

Input: number of trials T, target function f
$p_1 =$ initial surrogate model
Initialize y^*
for $t = 1$ **to** T **do**
  $\mathbf{x}_t \leftarrow \arg\max_{\mathbf{x}} \mathcal{A}(\mathbf{x}; p_t, y^*)$
  $y_t \leftarrow$ evaluate $f(\mathbf{x}_t)$
  Update y^*
  Estimate p_t given $\mathbf{x}_{1:t}$ and $y_{1:t}$
end for

Surrogate Model. As a surrogate model, we use a tree-structured Parzen estimator (TPE; Bergstra et al., 2011).[2] This is a nonparametric approach to density estimation. We seek to estimate $p_t(y \mid \mathbf{x})$ where $y = f(\mathbf{x})$, the performance function that is expensive to compute exactly. The TPE approach seeks $p_t(y \mid \mathbf{x}) \propto p_t(y) \cdot \begin{cases} p_t^<(\mathbf{x}), & \text{if } y < y^* \\ p_t^\geq(\mathbf{x}), & \text{if } y \geq y^* \end{cases}$, where $p_t^<$ and $p_t^\geq$ are densities estimated using observations from previous trials that are less than and greater than y^*, respectively. In TPE, y^* is defined as some quantile of the observed y from previous trials; we use 15-quantiles.

As shown by Bergstra et al. (2011), the Expected Improvement in TPE can be written as:

[1] EI is the most widely used acquisition function that has been shown to work well on a range of tasks.

[2] Another common approach to the surrogate is the Gaussian process (Rasmussen and Williams, 2006; Hoffman et al., 2011; Snoek et al., 2012). Like Bergstra et al. (2011), our preliminary experiments found the TPE to perform favorably. Further TPE's tree-structured configuration space is advantageous, because it allows nested definitions of hyperparameters, which we exploit in our experiments (e.g., only allows bigrams to be chosen if unigrams are also chosen).

Hyperparameter	Values
n_{min}	$\{1, 2, 3\}$
n_{max}	$\{n_{min}, \ldots, 3\}$
weighting scheme	{tf, tf-idf, binary}
remove stop words?	{True, False}
regularization	$\{\ell_1, \ell_2\}$
regularization strength	$[10^{-5}, 10^{5}]$
convergence tolerance	$[10^{-5}, 10^{-3}]$

Table 1: The set of hyperparameters considered in our experiments. The top half are hyperparameters related to text representation, while the bottom half are logistic regression hyperparameters, which also interact with the chosen representation.

$\mathcal{A}(\mathbf{x}; p_t, y^*) \propto \left(\gamma + \frac{p_t^<(\mathbf{x})}{p_t^>(\mathbf{x})}(1-\gamma)\right)^{-1}$, where $\gamma = p_t(y < y^*)$, fixed at 0.15 by definition of y^* (above). Here, we prefer $\mathbf{x}$ with high probability under $p_t^>(\mathbf{x})$ and low probability under $p_t^<(\mathbf{x})$. To maximize this quantity, we draw many candidates according to $p_t^>(\mathbf{x})$ and evaluate them according to $p_t^<(\mathbf{x})/p_t^>(\mathbf{x})$. Note that $p(y)$ does not need to be given an explicit form. To compute $p_t^<(\mathbf{x})$ and $p_t^>(\mathbf{x})$, we associate each hyperparameter with a node in the graphical model and multiply individual probabilities at every node—see Bergstra et al. (2011) for details.

4 Experiments

We fix L to logistic regression. We optimize text representation based on the types of n-grams used, the type of weighting scheme, and the removal of stopwords; we also optimize the regularizer and training convergence criterion, which interact with the representation. See Table 1 for a complete list.

Note that even with this limited number of options, the number of possible combinations is huge,[3] so exhaustive search is computationally expensive. In all our experiments for all datasets, we limit ourselves to 30 trials per dataset. The only preprocessing we applied was downcasing.

We always use a development set to evaluate $f(\mathbf{x})$ during learning and report the final result on an unseen test set. We summarize the hyperparameters selected by our method, and the accuracies achieved (on test data) in Table 5. We discuss comparisons to baselines for each dataset in turn. For each of our datasets, we select supervised, non-ensemble classification methods from previous literature as baselines. In each case, we emphasize comparisons with the best-published linear method (often an SVM with a linear kernel with representation selected by experts) and the best-published method overall. In the following, "SVM" always means "linear SVM." All methods were trained and evaluated on the same training/testing splits as baselines; in cases where standard development sets were not available, we used a random 20% of the training data as a development set.

Stanford sentiment treebank (Socher et al., 2013)—Table 2. A sentence-level sentiment analysis dataset of rottentomatoes.com movie reviews: http://nlp.stanford.edu/sentiment. We use the binary classification task where the goal is to predict whether a review is positive or negative (no neutral). Our logistic regression model outperforms the baseline SVM reported by Socher et al. (2013), who used only unigrams but did not specify the weighting scheme for their SVM baseline. While our result is still below the state-of-the-art based on the the recursive neural tensor networks (Socher et al., 2013) and the paragraph vector (Le and Mikolov, 2014), we show that logistic regression is comparable with recursive and matrix-vector neural networks (Socher et al., 2011; Socher et al., 2012).

Method	Acc.
Naïve Bayes	81.8
SVM	79.4
Vector average	80.1
Recursive neural networks	82.4
LR (this work)	82.4
Matrix-vector RNN	82.9
Recursive neural tensor networks	85.4
Paragraph vector	87.8

Table 2: Comparisons on the Stanford sentiment treebank dataset. Scores are as reported by Socher et al. (2013) and Le and Mikolov (2014). Test size $= 6,920$.

Amazon electronics (McAuley and Leskovec, 2013)—Table 3. A binary sentiment analysis dataset of Amazon electronics product reviews: http://riejohnson.com/cnn_data.html. The best-performing methods on this dataset are based on convolutional neural networks (Johnson and Zhang, 2015).[4] Our method is on par with the second-best of these, outperforming all of the reported feed-forward neural networks and SVM variants Johnson and Zhang used as baselines. They varied

[3] It is actually infinite since the reg. strength and conv. tolerance are continuous values, but we could discretize them.

[4] These are convolutional neural networks with a rectifier activation function, trained under ℓ_2 regularization with stochastic gradient descent. The authors also consider an extension based on parallel CNN that we do not include here.

the representations, and used log term frequency and normalization to unit vectors as the weighting scheme, after finding that this outperformed term frequency. Our method achieved the best performance with binary weighting, which they did not consider.

IMDB movie reviews (Maas et al., 2011)—Table 3. A binary sentiment analysis dataset of highly polar IMDB movie reviews: http://ai.stanford.edu/~amaas/data/sentiment. The results parallel those for Amazon electronics; our method comes close to convolutional neural networks (Johnson and Zhang, 2015), which are state-of-the-art.[5] It outperforms SVMs and feed-forward neural networks, the restricted Boltzmann machine approach presented by Dahl et al. (2012), and compressive feature learning (Paskov et al., 2013).[6]

Method	Accuracy	
	Amazon	IMDB
SVM-unigrams	88.29	88.64
RBM		89.23
SVM-$\{1,2\}$-grams	90.95	90.26
Compressive feature learning		90.40
SVM-$\{1,2,3\}$-grams	91.29	90.58
LR-$\{1,2,3,4,5\}$-grams		90.60
NN-$\{1,2,3\}$-grams	91.52	90.83
LR (this work)	91.56	90.85
Bag of words CNN	91.61	91.34
Sequential CNN	92.52	91.61

Table 3: Comparisons on the Amazon electronics and IMDB reviews datasets. SVM results are from Wang and Manning (2012), the RBM (restricted Bolzmann machine) result is from Dahl et al. (2012), NN and CNN results are from Johnson and Zhang (2015), and LR-$\{1,2,3,4,5\}$-grams and compressive feature learning results are from Paskov et al. (2013). Test size $= 20,000$ for both datasets.

Congressional vote (Thomas et al., 2006)—Table 4. A dataset of transcripts from the U.S. Congressional debates: http://www.cs.cornell.edu/~ainur/sle-data.html. Similar to previous work (Thomas et al., 2006; Bansal et al., 2008; Yessenalina et al., 2010), we consider the task to predict the vote ("yea" or "nay") for the speaker of each speech segment (speaker-based speech-segment classification). Our method outperforms the best results of Yessenalina et al. (2010), which use a multi-level structured model based on a latent-variable SVM. We show comparisons to two weaker baselines as well.

Method	Acc.
SVM-link	71.28
Min-cut	75.00
SVM-SLE	77.67
LR (this work)	78.59

Table 4: Comparisons on the congress vote dataset. SVM-link exploits link structures (Thomas et al., 2006); the min-cut result is from Bansal et al. (2008); and SVM-SLE result is reported by Yessenalina et al. (2010). Test size $= 1,175$.

20 Newsgroups (Lang, 1995) all topics—Table 6. 20 Newsgroups is a benchmark topic classification dataset: http://qwone.com/~jason/20Newsgroups. There are 20 topics in this dataset. Our method outperforms state-of-the-art methods including the distributed structured output model (Srikumar and Manning, 2014).[7] The strong logistic regression baseline from Paskov et al. (2013) uses all 5-grams, heuristic normalization, and elastic net regularization; our method found that unigrams and bigrams, with binary weighting and ℓ_2 penalty, achieved far better results.

Method	Acc.
Discriminative RBM	76.20
LR-$\{1,2,3,4,5\}$-grams	82.80
Compressive feature learning	83.00
Distributed structured output	84.00
LR (this work)	87.84

Table 6: Comparisons on the 20 Newsgroups dataset for classifying documents into all topics. The disriminative RBM result is from Larochelle and Bengio (2008); compressive feature learning and LR-5-grams results are from Paskov et al. (2013), and the distributed structured output result is from Srikumar and Manning (2014). Test size $= 9,052$.

20 Newsgroups: talk.religion.misc vs. alt.atheism and comp.graphics vs. comp.windows.x. We derived three additional topic classification tasks from the 20N dataset. The first and second tasks are talk.religion.misc vs. alt.atheism (test size $= 686$) and comp.graphics vs. comp.windows.x (test size $= 942$). Wang and Manning (2012) report a bigram naïve Bayes model achieving 85.1% and 91.2% on these tasks, respectively (best single model results).[8] Our

[5] As noted, semi-supervised and ensemble methods are excluded for a fair comparison.

[6] This approach is based on minimum description length, using unlabeled data to select a set of higher-order n-grams to use as features.

[7] This method was designed for structured prediction, but Srikumar and Manning (2014) also applied it to classification. It attempts to learn a distributed representation for features and for labels. The authors used unigrams and did not discuss the weighting scheme.

[8] They also report a naïve Bayes/SVM ensemble achieving 87.9% and 91.2%.

Dataset	Acc.	n_{min}	n_{max}	Weighting	Stopword removal?	Reg.	Strength	Conv.
Stanford sentiment	82.43	1	2	tf-idf	F	ℓ_2	10	0.098
Amazon electronics	91.56	1	3	binary	F	ℓ_2	120	0.022
IMDB reviews	90.85	1	2	binary	F	ℓ_2	147	0.019
Congress vote	78.59	2	2	binary	F	ℓ_2	121	0.012
20N all topics	87.84	1	2	binary	F	ℓ_2	16	0.008
20N all science	95.82	1	2	binary	F	ℓ_2	142	0.007
20N atheist/religion	86.32	1	2	binary	T	ℓ_1	41	0.011
20N x/graphics	92.09	1	1	binary	T	ℓ_2	91	0.014

Table 5: Classification accuracies and the best hyperparameters for each of the datasets in our experiments. "Acc" shows accuracies for our logistic regression model. "Min" and "Max" correspond to the min n-grams and max n-grams respectively. "Reg." is the regularization type, "Strength" is the regularization strength, and "Conv." is the convergence tolerance. For regularization strength, we round it to the nearest integer for readability.

method achieves 86.3% and 92.1% using slightly different representations (see Table 5). The last task is to classify related science documents into four science topics (sci.crypt, sci.electronics, sci.space, sci.med; test size $= 1,899$). We were not able to find previous results that are comparable to ours on this task; we include our result (95.82%) to enable further comparisons in the future.

5 Discussion

Optimized representations. For each task, the chosen representation is different. Out of all possible choices in our experiments (Table 1), each of them is used by at least one of the datsets (Table 5). For example, on the Congress vote dataset, we only need to use bigrams, whereas on the Amazon electronics dataset we need to use $\{1, 2, 3\}$-grams. The binary weighting scheme works well for most of the datasets, except the sentence-level sentiment analysis task, where the tf-idf weighting scheme was selected. ℓ_2 regularization was best in all cases but one. We do not believe that an NLP expert would be likely to make these particular choices, except through the same kind of trial-and-error process our method automates efficiently.

Number of trials. We ran 30 trials for each dataset in our experiments. Figure 1 shows each trial accuracy and the best accuracy on development data as we increase the number of trials for two datasets. We can see that 30 trials are generally enough for the model to obtain good results, although the search space is large.

Transfer learning and multitask setting. We treat each dataset independently and create a separate model for each of them. It is also possible to learn from previous datasets (i.e., transfer learning) or to learn from all datasets simultaneously (i.e., multitask learning) to improve performance. This has the potential to reduce the number of trials required even further. See Bardenet et al. (2013), Swersky et al. (2013), and Yogatama and Mann (2014) for more about how to perform Bayesian optimization in these settings.

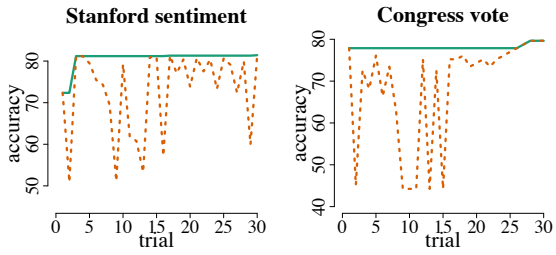


Figure 1: Classification accuracies on development data for Stanford sentiment treebank (left) and congressional vote (right) datasets. In each plot, the green solid line indicates the best accuracy found so far, while the dotted orange line shows accuracy at each trial. We can see that in general the model is able to obtain reasonably good representation in 30 trials.

Beyond supervised learning. Our framework could also be extended to unsupervised and semi-supervised models. For example, in document clustering (e.g., k-means), we also need to construct representations for documents. Log-likelihood might serve as a performance function. A range of random initializations might be considered. Investigation of this approach for nonconvex problems is an exciting area for future work.

6 Conclusion

We used Bayesian optimization to optimize choices about text representations for various categorization problems. Our technique identifies settings for a standard linear model (logistic regression) that are competitive with far more sophisticated methods on topic classification and sentiment analysis.

Acknowledgments

We thank several reviewers for their helpful feedback. This work was supported by the Defense Advanced Research Projects Agency through grant FA87501420244 and computing resources provided by Amazon. This research was completed while NAS was at CMU.

References

Mohit Bansal, Clair Cardie, and Lillian Lee. 2008. The power of negative thinking: Exploiting label disagreement in the min-cut classification framework. In *Proc. of COLING*.

Remi Bardenet, Matyas Brendel, Balazs Kegl, and Michele Sebag. 2013. Collaborative hyperparameter tuning. In *Proc. of ICML*.

James Bergstra, Remi Bardenet, Yoshua Bengio, and Balazs Kegl. 2011. Algorithms for hyper-parameter optimization. In *NIPS*.

James Bergstra, Daniel Yamins, and David Cox. 2013. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In *Proc. of ICML*.

Walter Daelemans, Veronique Hoste, Fien De Meulder, and Bart Naudts. 2003. Combined optimization of feature selection and algorithm parameters in machine learning of language. In *Proc. of ECML*.

George E. Dahl, Ryan P. Adams, and Hugo Larochelle. 2012. Training restricted Boltzmann machines on word observations. In *Proc. of ICML*.

Matthew Hoffman, Eric Brochu, and Nando de Freitas. 2011. Portfolio allocation for Bayesian optimization. In *Proc. of UAI*.

Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. 2011. Sequential model-based optimization for general algorithm configuration. In *Proc. of LION*.

Rie Johnson and Tong Zhang. 2015. Effective use of word order for text categorization with convolutional neural networks. In *Proc. of NAACL*.

Donald R. Jones. 2001. A taxonomy of global optimization methods based on response surfaces. *Journal of Global Optimization*, 21:345–385.

Ken Lang. 1995. Newsweeder: Learning to filter netnews. In *Proc. of ICML*.

Hugo Larochelle and Yoshua Bengio. 2008. Classification using discriminative restricted Boltzmann machines. In *Proc. of ICML*.

Quoc V. Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proc. of ICML*.

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

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

Julian McAuley and Jure Leskovec. 2013. Hidden factors and hidden topics: understanding rating dimensions with review text. In *Proc. of RecSys*.

Hristo S. Paskov, Robert West, John C. Mitchell, and Trevor J. Hastie. 2013. Compressive feature learning. In *Proc of NIPS*.

Carl Edward Rasmussen and Christopher K. I. Williams. 2006. *Gaussian Processes for Machine Learning*. The MIT Press.

Jasper Snoek, Hugo Larrochelle, and Ryan P. Adams. 2012. Practical Bayesian optimization of machine learning algorithms. In *NIPS*.

Richard Socher, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher D. Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proc. of EMNLP*.

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

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

Vivek Srikumar and Christopher D. Manning. 2014. Learning distributed representations for structured output prediction. In *NIPS*.

Niranjan Srinivas, Andreas Krause, Sham Kakade, and Matthias Seeger. 2010. Gaussian process optimization in the bandit setting: No regret and experimental design. In *Proc. of ICML*.

Kevin Swersky, Jasper Snoek, and Ryan P. Adams. 2013. Multi-task Bayesian optimization. In *NIPS*.

Matt Thomas, Bo Pang, and Lilian Lee. 2006. Get out the vote: Determining support or opposition from congressional floor-debate transcripts. In *Proc. of EMNLP*.

Julien Villemonteix, Emmanuel Vazquez, and Eric Walter. 2009. An informational approach to the global optimization of expensive-to-evaluate functions. *Journal of Global Optimization*, 44(4):509–534.

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

Ainur Yessenalina, Yisong Yue, and Claire Cardie. 2010. Multi-level structured models for document sentiment classification. In *Proc. of EMNLP*.

Dani Yogatama and Gideon Mann. 2014. Efficient transfer learning method for automatic hyperparameter tuning. In *Proc. of AISTATS*.

A Comparative Study on Regularization Strategies for Embedding-based Neural Networks

Hao Peng,[*1] **Lili Mou,**[*1] **Ge Li,**[†1] **Yunchuan Chen,**[2] **Yangyang Lu,**[1] **Zhi Jin**[1]
[1]Software Institute, Peking University, 100871, P. R. China
{penghao.pku, doublepower.mou}@gmail.com,{lige, luyy11, zhijin}@sei.pku.edu.cn
[2]University of Chinese Academy of Sciences, chenyunchuan11@mails.ucas.ac.cn

Abstract

This paper aims to compare different regularization strategies to address a common phenomenon, severe overfitting, in embedding-based neural networks for NLP. We chose two widely studied neural models and tasks as our testbed. We tried several frequently applied or newly proposed regularization strategies, including penalizing weights (embeddings excluded), penalizing embeddings, re-embedding words, and dropout. We also emphasized on incremental hyperparameter tuning, and combining different regularizations. The results provide a picture on tuning hyperparameters for neural NLP models.

1 Introduction

Neural networks have exhibited considerable potential in various fields (Krizhevsky et al., 2012; Graves et al., 2013). In early years on neural NLP research, neural networks were used in language modeling (Bengio et al., 2003; Morin and Bengio, 2005; Mnih and Hinton, 2009); recently, they have been applied to various supervised tasks, such as named entity recognition (Collobert and Weston, 2008), sentiment analysis (Socher et al., 2011; Mou et al., 2015), relation classification (Zeng et al., 2014; Xu et al., 2015), etc. In the field of NLP, neural networks are typically combined with word embeddings, which are usually first pre-trained by unsupervised algorithms like Mikolov et al. (2013); then they are fed forward to standard neural models, fine-tuned during supervised learning. However, embedding-based neural networks usually suffer from severe overfitting because of the high dimensionality of parameters.

A curious question is whether we can regularize embedding-based NLP neural models to improve generalization. Although existing and newly proposed regularization methods might alleviate the problem, their inherent performance in neural NLP models is not clear: the use of embeddings is sparse; the behaviors may be different from those in other scenarios like image recognition. Further, selecting hyperparameters to pursue the best performance by validation is extremely time-consuming, as suggested in Collobert et al. (2011). Therefore, new studies are needed to provide a more complete picture regarding regularization for neural natural language processing. Specifically, we focus on the following research questions in this paper.

RQ 1: How do different regularization strategies typically behave in embedding-based neural networks?

RQ 2: Can regularization coefficients be tuned incrementally during training so as to ease the burden of hyperparameter tuning?

RQ 3: What is the effect of combining different regularization strategies?

In this paper, we systematically and quantitatively compared four different regularization strategies, namely penalizing weights, penalizing embeddings, newly proposed word re-embedding (Labutov and Lipson, 2013), and dropout (Srivastava et al., 2014). We analyzed these regularization methods by two widely studied models and tasks. We also emphasized on incremental hyperparameter tuning and the combination of different regularization methods.

Our experiments provide some interesting results: (1) Regularizations do help generalization, but their effect depends largely on the datasets' size. (2) Penalizing ℓ_2-norm of embeddings helps optimization as well, improving training accuracy unexpectedly. (3) Incremental hyperparameter tuning achieves similar performance, indicat-

[*]Equal contribution. [†]Corresponding author.

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 2106–2111,
Lisbon, Portugal, 17-21 September 2015.

ing that regularizations mainly serve as a "local" effect. (4) Dropout performs slightly worse than ℓ_2 penalty in our experiments; however, provided very small ℓ_2 penalty, dropping out hidden units and penalizing ℓ_2-norm are generally complementary. (5) The newly proposed re-embedding words method is not effective in our experiments.

2 Tasks, Models, and Setup

Experiment I: Relation extraction. The dataset in this experiment comes from SemEval-2010 Task 8.[1] The goal is to classify the relationship between two marked entities in each sentence. We refer interested readers to recent advances, e.g., Hashimoto et al. (2013), Zeng et al. (2014), and Xu et al. (2015). To make our task and model general, however, we do not consider entity tagging information; we do not distinguish the order of two entities either. In total, there are 10 labels, i.e., 9 different relations plus a default `other`.

Regarding the neural model, we applied Collobert's convolutional neural network (CNN) (Collobert and Weston, 2008) with minor modifications. The model comprises a fixed-window convolutional layer with size equal to 5, **0** padded at the end of each sentence; a max pooling layer; a $\tanh$ hidden layer; and a $\mathrm{softmax}$ output layer.

Experiment II: Sentiment analysis. This is another testbed for neural NLP, aiming to predict the sentiment of a sentence. The dataset is the Stanford sentiment treebank (Socher et al., 2011)[2]; target labels are `strongly/weakly positive/negative`, or `neutral`.

We used the recursive neural network (RNN), which is proposed in Socher et al. (2011), and further developed in Socher et al. (2012); Irsoy and Cardie (2014). RNNs make use of binarized constituency trees, and recursively encode children's information to their parent's; the root vector is finally used for sentiment classification.

Experimental Setup. To setup a fair comparison, we set all layers to be 50-dimensional in advance (rather than by validation). Such setting has been used in previous work like Zhao et al. (2015). Our embeddings are pretrained on the Wikipedia corpus using Collobert and Weston (2008). The learning rate is 0.1 and fixed in Experiment I; for RNN, however, we found learning rate decay helps to prevent parameter blowup (probably due to the recursive, and thus chaotic nature). Therefore, we applied power decay (Senior et al., 2013) with power equal to -1. For each strategy, we tried a large range of regularization coefficients, $10^{-9}, \cdots, 10^{-2}$, extensively from underfitting to no effect with granularity 10x. We ran the model 5 times with different initializations. We used mini-batch stochastic gradient descent; gradients are computed by standard backpropagation. For source code, please refer to our project website.[3]

It needs to be noticed that, the goal of this paper is not to outperform or reproduce state-of-the-art results. Instead, we would like to have a fair comparison. The testbed of our work is two widely studied models and tasks, which were not chosen on purpose. During the experiments, we tried to make the comparison as fair as possible. Therefore, we think that the results of this work can be generalized to similar scenarios.

3 Regularization Strategies

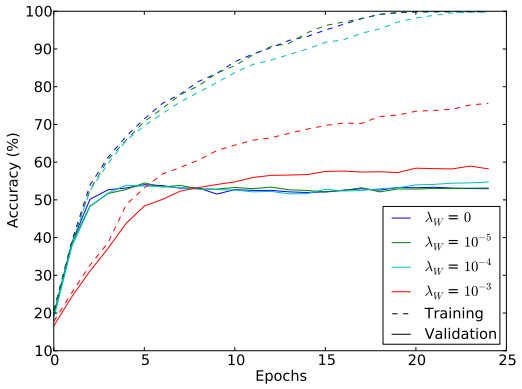
In this section, we describe four regularization strategies used in our experiment.

- Penalizing ℓ_2-norm of weights. Let E be the cross-entropy error for classification, and R be a regularization term. The overall cost function is $J = E + \lambda R$, where λ is the coefficient. In this case, $R = \|W\|^2$, and the coefficient is denoted as λ_W.
- Penalizing ℓ_2-norm of embeddings. Some studies do not distinguish embeddings or connectional weights for regularization (Tai et al., 2015). However, we would like to analyze their effect separately, for embeddings are sparse in use. Let Φ denote embeddings; then we have $R = \|\Phi\|^2$.
- Re-embedding words (Labutov and Lipson, 2013). Suppose Φ_0 denotes the original embeddings trained on a large corpus, and Φ denotes the embeddings fine-tuned during supervised training. We would like to penalize the norm of the difference between Φ_0 and Φ, i.e., $R = \|\Phi_0 - \Phi\|^2$. In the limit of penalty to infinity, the model is mathematically equivalent to "frozen embeddings," where word vectors are used as surface features.
- Dropout (Srivastava et al., 2014). In this strategy, each neural node is set to 0 with a predefined dropout probability p during training; when testing, all nodes are used, with activation multiplied by $1 - p$.

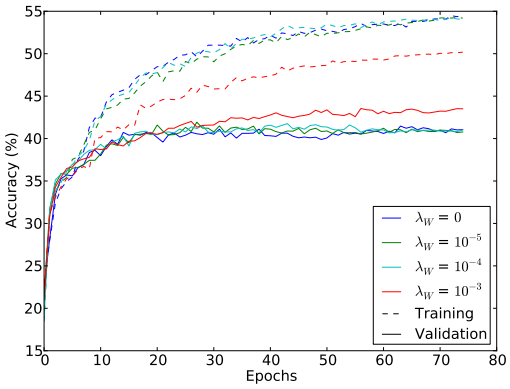
[1] http://www.aclweb.org/anthology/S10-1006

[2] http://nlp.stanford.edu/sentiment/

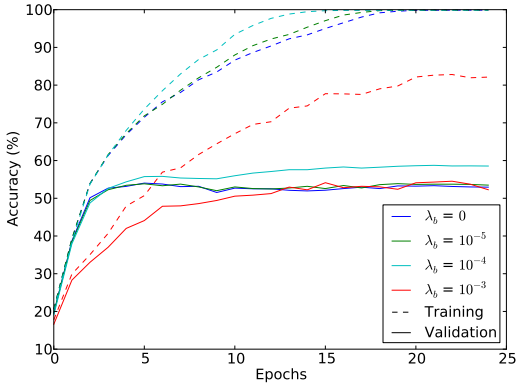
[3] https://sites.google.com/site/regembeddingnn/



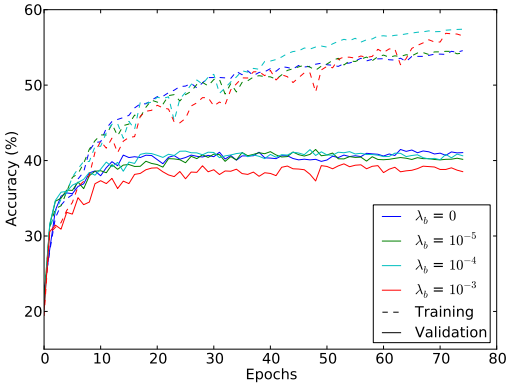
(a) Penalizing weights in Experiment I.



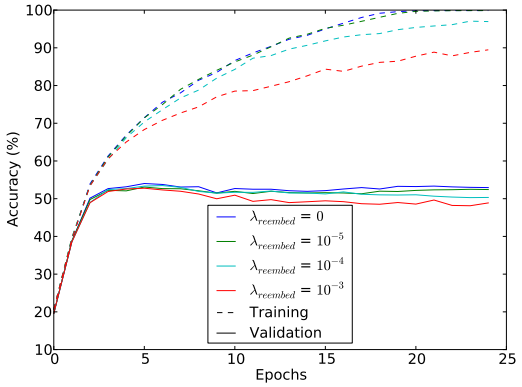
(b) Penalizing weights in Experiment II.



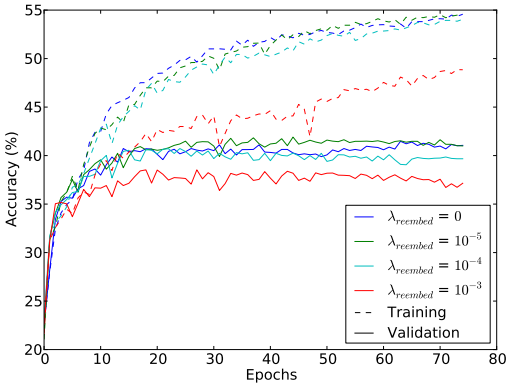
(c) Penalizing embeddings in Experiment I.



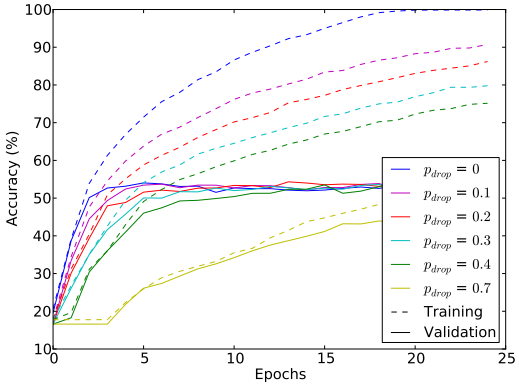
(d) Penalizing embeddings in Experiment II.



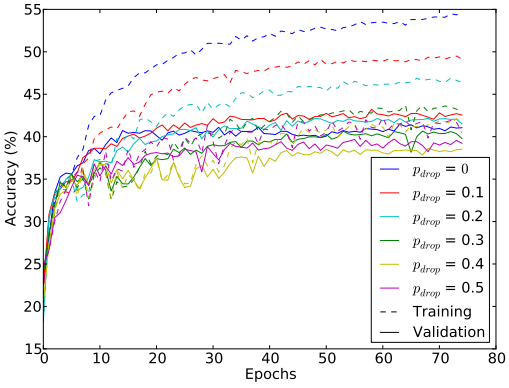
(e) Re-embedding words in Experiment I.



(f) Re-embedding words in Experiment II.



(g) Applying dropout in Experiment I. $p = 0.5, 0.6$ are omitted because they are similar to small values.



(h) Applying dropout in Experiment II.

Figure 1: Averaged learning curves. Left: Experiment I, relation extraction with CNN. Right: Experiment II, sentiment analysis with RNN. From top to bottom, we penalize weights, penalize embeddings, re-embed words, and drop out. Dashed lines refer to training accuracies; solid lines are validation accuracies.

4 Individual Regularization Behaviors

This section compares the behavior of each strategy. We first conducted both experiments without regularization, achieving accuracies of $54.02 \pm 0.84\%$, $41.47 \pm 2.85\%$, respectively. Then we plot in Figure 1 learning curves when each regularization strategy is applied individually. We report training and validation accuracies through out this paper. The main findings are as follows.

- Penalizing ℓ_2-norm of weights helps generalization; the effect depends largely on the size of training set. Experiment I contains 7,000 training samples and the improvement is 6.98%; Experiment II contains more than 150k samples, and the improvement is only 2.07%. Such results are consistent with other machine learning models.

- Penalizing ℓ_2-norm of embeddings unexpectedly helps optimization (improves training accuracy). One plausible explanation is that since embeddings are trained on a large corpus by unsupervised methods, they tend to settle down to large values and may not perfectly agree with the tasks of interest. ℓ_2 penalty pushes the embeddings towards small values and thus helps optimization. Regarding validation accuracy, Experiment I is improved by 6.89%, whereas Experiment II has no significant difference.

- Re-embedding words does not improve generalization. Particularly, in Experiment II, the ultimate accuracy is improved by 0.44, which is not large. Further, too much penalty hurts the models in both experiments. In the limit λ_{reembed} to infinity, re-embedding words is mathematically equivalent to using embeddings as surface features, that is, freezing embeddings. Such strategy is sometimes applied in the literature like Hu et al. (2014), but is not favorable as suggested by the experiment.

- Dropout helps generalization. Under the best settings, the eventual accuracy is improved by 3.12% and 1.76%, respectively. In our experiments, dropout alone is not as useful as ℓ_2 penalty. However, other studies report that dropout is very effective (Irsoy and Cardie, 2014). Our results are not consistent; different dimensionality may contribute to this disagreement, but more experiments are needed to confirm the hypothesis.

5 Incremental Hyperparameter Tuning

The above experiments show that regularization generally helps prevent overfitting. To pursue the best performance, we need to try out different hyperparameters through validation. Unfortunately, training deep neural networks is time-consuming, preventing full grid search from being a practical technique. Things will get easier if we can incrementally tune hyperparameters, that is, to train the model without regularization first, and then add penalty.

In this section, we study whether ℓ_2 penalty of weights and embeddings can be tuned incrementally. We exclude the dropout strategy because its does not make much sense to incrementally drop out hidden units. Besides, from this section, we only focus on Experiment I due to time and space limit.

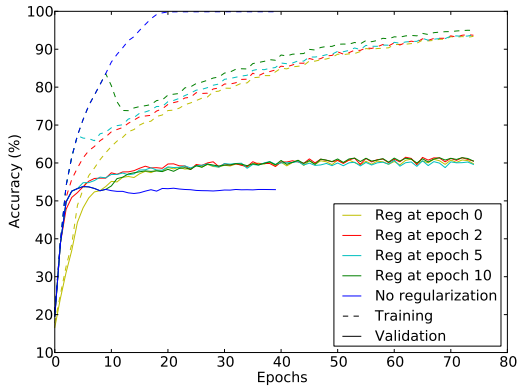
Before continuing, we may envision several possibilities on how regularization works.

- (On initial effects) As ℓ_2-norm prevents parameters from growing large, adding it at early stages may cause parameters settling down to local optima. If this is the case, delayed penalty would help parameters get over local optima, leading to better performance.
- (On eventual effects) ℓ_2 penalty lifts error surface of large weights. Adding such penalty may cause parameters settling down to (a) almost the same catchment basin, or (b) different basins. In case (a), when the penalty is added does not matter much. In case (b), however, it makes difference, because parameters would have already gravitated to catchment basins of larger values before regularization is added, which means incremental hyperparameter tuning would be ineffective.

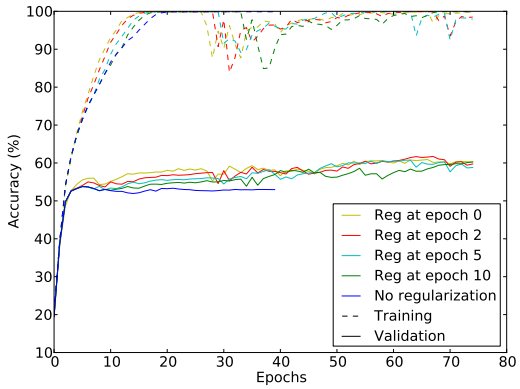
To verify the above conjectures, we design four settings: adding penalty (1) at the beginning, (2) before overfitting at epoch 2, (3) at peak performance (epoch 5), and (4) after overfitting (validation accuracy drops) at epoch 10.

Figure 2 plots the learning curves regarding penalizing weights and embeddings, respectively; baseline (without regularization) is also included.

For both weights and embeddings, all settings yield similar ultimate validation accuracies. This shows ℓ_2 regularization mainly serves as a "local" effect—it changes the error surface, but parameters tend to settle down to a same catchment basin. We notice a recent report also shows local optima



(a) Incrementally penalizing ℓ_2-norm of weights.



(b) Incrementally penalizing ℓ_2-norm of biases.

Figure 2: Tuning hyperparameters incrementally in Experiment I. Penalty is added at epochs 0, 2, 5, 10, respectively. We chose the coefficients yielding the best performance in Figure 1. The controlled trial (no regularization) is early stopped because the accuracy has already decreased.

	λ_{W}			
λ_{embed}	0	10^{-4}	$3{\cdot}10^{-4}$	10^{-3}
0	54.02	57.88	59.96	61.00
10^{-5}	54.94	57.82	60.68	62.05
$3{\cdot}10^{-5}$	55.68	61.02	**64.00**	**63.15**
10^{-4}	60.91	**64.00**	**63.07**	60.56
$3{\cdot}10^{-4}$	58.92	61.33	59.85	42.93
10^{-3}	54.77	56.43	54.05	16.50

Table 1: Accuracy in percentage when we combine ℓ_2-norm of weights and embeddings (Experiment I). Bold numbers are among highest accuracies (greater than peak performance minus 1.5 times standard deviation, i.e., 1.26 in percentage).

	λ_W			λ_{embed}		
p	10^{-4}	$3{\cdot}10^{-4}$	10^{-3}	10^{-5}	$3{\cdot}10^{-5}$	10^{-4}
0	57.88	**59.96**	**61.00**	54.94	55.68	**60.91**
1/6	58.36	59.36	43.42	58.49	59.59	**60.00**
2/6	58.22	**60.00**	16.60	59.34	**60.08**	59.61
3/6	58.63	59.73	16.60	59.59	**59.98**	58.82
4/6	56.43	54.63	16.60	56.76	59.19	56.64
5/6	38.07	16.60	16.60	49.79	53.63	49.75

Table 2: Combining ℓ_2 regularization and dropout. Left: connectional weights. Right: embeddings. (p refers to the dropout rate.)

may not play an important role in training neural networks, if the effect of parameter symmetry is ruled out (Breuel, 2015).

We also observe that regularization helps generalization as soon as it is added (Figure 2a), and that regularizing embeddings helps optimization also right after the penalty is applied (Figure 2b).

6 Combination of Regularizations

We are further curious about the behaviors when different regularization methods are combined.

Table 1 shows that combining ℓ_2-norm of weights and embeddings results in a further accuracy improvement of 3–4 percents from applying either single one of them. In a certain range of coefficients, weights and embeddings are complementary: given one hyperparameter, we can tune the other to achieve a result among highest ones.

Such compensation is also observed in penalizing ℓ_2-norm versus dropout (Table 2)—although the peak performance is obtained by pure ℓ_2 regularization, applying dropout with small ℓ_2 penalty also achieves a similar accuracy. The dropout rate is not very sensitive, provided it is small.

7 Discussion

In this paper, we systematically compared four regularization strategies for embedding-based neural networks in NLP. Based on the experimental results, we answer our research questions as follows. (1) Regularization methods (except re-embedding words) basically help generalization. Penalizing ℓ_2-norm of embeddings unexpectedly helps optimization as well. Regularization performance depends largely on the dataset's size. (2) ℓ_2 penalty mainly acts as a local effect; hyperparameters can be tuned incrementally. (3) Combining ℓ_2-norm of weights and biases (dropout and ℓ_2 penalty) further improves generalization; their coefficients are mostly complementary within a certain range. These empirical results of regularization strategies shed some light on tuning neural models for NLP.

Acknowledgments

This research is supported by the National Basic Research Program of China (the 973 Program) under Grant No. 2015CB352201 and the National Natural Science Foundation of China under Grant No. 61232015. We would also like to thank Hao Jia and Ran Jia.

References

Yoshua Bengio, Réjean Ducharme, P. Vincent, and C. Jauvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155.

Thomas M. Breuel. 2015. The effects of hyperparameters on sgd training of neural networks. *arXiv preprint arXiv:1508.02788*.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine learning*.

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

Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *Proceedings of 2013 IEEE International Conference on Acoustics, Speech and Signal Processing*.

Kazuma Hashimoto, Makoto Miwa, Yoshimasa Tsuruoka, and Takashi Chikayama. 2013. Simple customization of recursive neural networks for semantic relation classification. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*.

Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. In *Advances in Neural Information Processing Systems*.

Ozan Irsoy and Claire Cardie. 2014. Deep recursive neural networks for compositionality in language. In *Advances in Neural Information Processing Systems*.

Alex Krizhevsky, Ilyz Sutskever, and Geoffrey Hinton. 2012. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*.

Igor Labutov and Hod Lipson. 2013. Re-embedding words. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2)*.

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

Andriy Mnih and Geoffrey Hinton. 2009. A scalable hierarchical distributed language model. In *Advances in Neural Information Processing Systems*.

Frederic Morin and Yoshua Bengio. 2005. Hierarchical probabilistic neural network language model. In *Proceedings of International Conference on Artificial Intelligence and Statistics*.

Lili Mou, Hao Peng, Ge Li, Yan Xu, Lu Zhang, and Zhi Jin. 2015. Tree-based convolution: A new neural architecture for sentence modeling. In *Proceedings of Conference on Empirical Methods in Natural Language Processing (to appear)*.

Andrew Senior, Georg Heigold, Marc'aurelio Ranzato, and Ke Yang. 2013. An empirical study of learning rates in deep neural networks for speech recognition. In *Proceedings of 2013 IEEE International Conference on Acoustics, Speech and Signal Processing*.

Richard Socher, Jeffrey Pennington, Eric Huang, Andrew Ng, and Christopher D. Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*.

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

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*.

Yan Xu, Lili Mou, Ge Li, Yunchuan Chen, Hao Peng, and Zhi Jin. 2015. Classifying relations via long short term memory networks along shortest dependency paths. In *Proceedings of Conference on Empirical Methods in Natural Language Processing (to appear)*.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation classification via convolutional deep neural network. In *Proceedings of Computational Linguistics*.

Han Zhao, Zhengdong Lu, and Pascal Poupart. 2015. Self-adaptive hierarchical sentence model. In *Proceedings of Intenational Joint Conference in Artificial Intelligence*.

Efficient Hyper-parameter Optimization for NLP Applications

Lidan Wang[1], **Minwei Feng**[1], **Bowen Zhou**[1], **Bing Xiang**[1], **Sridhar Mahadevan**[2,1]
[1]IBM Watson, T. J. Watson Research Center, NY
[2]College of Information and Computer Sciences, U. of Massachusetts Amherst, MA
{wangli,mfeng,zhou,bingxia}@us.ibm.com
mahadeva@cs.umass.edu

Abstract

Hyper-parameter optimization is an important problem in natural language processing (NLP) and machine learning. Recently, a group of studies has focused on using sequential Bayesian Optimization to solve this problem, which aims to reduce the number of iterations and trials required during the optimization process. In this paper, we explore this problem from a different angle, and propose a multi-stage hyper-parameter optimization that breaks the problem into multiple stages with increasingly amounts of data. Early stage provides fast estimates of good candidates which are used to initialize later stages for better performance and speed. We demonstrate the utility of this new algorithm by evaluating its speed and accuracy against state-of-the-art Bayesian Optimization algorithms on classification and prediction tasks.

1 Introduction

Hyper-parameter optimization has been receiving an increasingly amount of attention in the NLP and machine learning communities (Thornton et al., 2013; Komer et al., 2014; Bergstra et al., 2011; Bardenet et al., 2013; Zheng et al., 2013). The performance of learning algorithms depend on the correct instantiations of their hyper-parameters, ranging from algorithms such as logistic regression and support vector machines, to more complex model families such as boosted regression trees and neural networks. While hyper-parameter settings often make the difference between mediocre and state-of-the-art performance (Hutter et al., 2014), it is typically very time-consuming to find an optimal setting due to the complexity of model classes, *and* the amount of training data available for tuning. The issue is particularly important in large-scale problems where the size of the data can be so large that even a quadratic running time is prohibitively large.

Recently several sequential Bayesian Optimization methods have been proposed for hyper-parameter search (Snoek et al., 2012; Eggensperger et al., 2015; Brochu et al., 2010; Hutter et al., 2011; Eggensperger et al., 2014). The common theme is to perform a set of iterative hyper-parameter optimizations, where in each round, these methods fit a hyper-parameter response surface using a probabilistic regression function such as Gaussian Process (Snoek et al., 2012) or tree-based models (Hutter et al., 2011), where the response surface maps each hyper-parameter setting to an approximated accuracy. The learned regression model is then used as a cheap surrogate of the response surface to quickly explore the search space and identify promising hyper-parameter candidates to evaluate next in order to enhance validation accuracy.

While these methods have enjoyed great success compared to conventional random search (Bergstra et al., 2012; Bengio et al., 2013) and grid search algorithms by significantly reducing the number of iterations and trials required during the process, the focus and starting point of these work have largely been on dealing with many dimensions of hyper-parameters, *rather than* scaling to large amount of data as typical in many NLP tasks, where the efficiency bottleneck stems from the size of the training data in addition to hyper-parameter dimensions. For example, as dataset size grows, even simple models (with few hyper-parameters) such as logistic regression can require more training time per iteration in these algorithms, leading to increased overall time complexity.

In this work, we introduce a multi-stage Bayesian Optimization framework for efficient

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 2112–2117,
Lisbon, Portugal, 17-21 September 2015.

hyper-parameter optimization, and empirically study the impact of the multi-stage algorithm on hyper-parameter tuning. Unlike the previous approaches, the multi-stage approach considers hyper-parameter optimization in successive stages with increasingly amounts of training data. The first stage uses a small subset of training data, applies sequential optimization to quickly identify an initial set of promising hyper-parameter settings, and these promising candidates are then used to initialize Bayesian Optimization on later stages with full training dataset to enable the expensive stages operate with better prior knowledge and converge to optimal solution faster.

The key intuition behind the proposed approach is that both dataset size and search space of hyper-parameter can be large, and applying the Bayesian Optimization algorithm on the data can be both expensive and unnecessary, since many evaluated candidates may not even be within range of best final settings. We note our approach is orthogonal and complementary to parallel Bayesian Optimization (Snoek et al., 2012) and multi-task learning (Yogatama et al., 2014; Swersky et al., 2012), because the improved efficiency per iteration, as achieved by our algorithm, is a basic building block of the other algorithms, thus can directly help the efficiency of multiple parallel runs (Snoek et al., 2012), as well as runs across different datasets (Yogatama et al., 2014; Swersky et al., 2012).

2 Methodology

The new multi-stage Bayesian Optimization is a generalization of the standard Bayesian Optimization for hyper-parameter learning (Snoek et al., 2012; Feurer et al., 2015). It is designed to scale standard Bayesian Optimization to large amounts of training data. Before delving into the details, we first describe hyper-parameter optimization and give a quick overview on the standard Bayesian Optimization solution for it.

2.1 Hyper-parameter Optimization

Let $\boldsymbol{\lambda} = \{\lambda_1, \ldots, \lambda_m\}$ denote the hyper-parameters of a machine learning algorithm, and let $\{\Lambda_1, \ldots, \Lambda_m\}$ denote their respective domains. When trained with $\boldsymbol{\lambda}$ on training data T_{train}, the validation accuracy on T_{valid} is denoted as $L(\boldsymbol{\lambda}, \mathrm{T}_{train}, \mathrm{T}_{valid})$. The goal of hyper-parameter optimization is to find a hyper-parameter setting $\boldsymbol{\lambda}^*$ such that the validation accuracy L is maximized. Current state-of-the-art methods have focused on using model-based Bayesian Optimization (Snoek et al., 2012; Hutter et al., 2011) to solve this problem due to its ability to identify good solutions within a small number of iterations as compared to conventional methods such as grid search.

2.2 Bayesian Optimization for Hyper-parameter Learning

Model-based Bayesian Optimization (Brochu et al., 2010) starts with an initial set of hyper-parameter settings $\boldsymbol{\lambda}_1, \ldots \boldsymbol{\lambda}_n$, where each setting denotes a set of assignments to all hyper-parameters. These initial settings are then evaluated on the validation data and their accuracies are recorded. The algorithm then proceeds in rounds to iteratively fit a probabilistic regression model V to the recorded accuracies. A new hyper-parameter configuration is then suggested by the regression model V with the help of acquisition function (Brochu et al., 2010). Then the accuracy of the new setting is evaluated on validation data, which leads to the next iteration. A common acquisition function is the expected improvement, EI (Brochu et al., 2010), over best validation accuracy seen so far L^*:

$$a(\boldsymbol{\lambda}, V) = \int_{-\infty}^{\infty} max(L - L^*, 0) p_V(L|\boldsymbol{\lambda}) d_L$$

where $p_V(L|\boldsymbol{\lambda})$ denotes the probability of accuracy L given configuration $\boldsymbol{\lambda}$, which is encoded by the probabilistic regression model V. The acquisition function is used to identify the next candidate (the one with the highest expected improvement over current best L^*). More details of acquisition functions can be found in (Brochu et al., 2010).

The most common probabilistic regression model V is the Gaussian Process prior (Snoek et al., 2012), which is a convenient and powerful prior distribution on functions. For the purpose of our experiments, we also use Gaussian Process prior as the regression model. However, we would like to note the fact that the proposed multi-stage Bayesian Optimization is agnostic of the regression model used, and can easily handle other instantiations of the regression model.

Algorithm 1: Multi-stage Bayesian Optimization for Hyper-parameter Tuning

Input: Loss function L, number of stages S, iterations per stage $\boldsymbol{Y} = \langle Y_1, \ldots, Y_S \rangle$, training data per stage $T_{train} = \langle T^1_{train}, \ldots, T^S_{train} \rangle$, validation data T_{valid}, initialization $\boldsymbol{\lambda}_{1:k}$

Output: hyper-parameter $\boldsymbol{\lambda}^*$

for *stage s=1 to S* **do**
  for *i=1 to k* **do**
    L_i = Evaluate L($\boldsymbol{\lambda}_i, T^s_{train}, T_{valid}$)
  end
  for *j=k+1 to* Y_s **do**
    V: regression model on $\langle \boldsymbol{\lambda}_i, L_i \rangle_{i=1}^{j-1}$
    $\boldsymbol{\lambda}_j = \arg\max_{\boldsymbol{\lambda} \in \boldsymbol{\Lambda}} a(\boldsymbol{\lambda}, V)$
    L_j = Evaluate L($\boldsymbol{\lambda}_j, T^s_{train}, T_{valid}$)
  end
  reset $\boldsymbol{\lambda}_{1:k}$=best k configs $\in \langle \boldsymbol{\lambda}_1, \ldots \boldsymbol{\lambda}_{Y_s} \rangle$ based on validation accuracy $\boldsymbol{L}$
end
return $\boldsymbol{\lambda}^* = \arg\max_{\boldsymbol{\lambda}_j \in \{\boldsymbol{\lambda}^{Y_1}, \ldots, \boldsymbol{\lambda}^{Y_S}\}} L_j$

2.3 Multi-stage Bayesian Optimization for Hyper-parameter Tuning

The multi-stage algorithm as shown in Algorithm 1 is an extension of the standard Bayesian Optimization (Section 2.2) to enable speed on large-scale datasets. It proceeds in multiple stages of Bayesian Optimization with increasingly amounts of training data $|T^1_{train}| \leq \ldots, \leq |T^S_{train}|$. During each stage s, the k best configurations (based on validation accuracy) passed from the previous stage[1] are first evaluated on the current stage's training data T^s_{train}, and then the standard Bayesian Optimization algorithm are initialized with these k settings and applied for $Y_s - k$ iterations on T^s_{train} (discounting the k evaluations done earlier in the stage), where Y_s is the total number of iterations for stage s. Then the top k configurations based on validation accuracy are used to initialize the next stage's run.

We note after the initial stage, rather than only considering candidates passed from the previous stage, the algorithm expands from these points on larger data. Continued exploration using larger data allows the algorithm to eliminate any potential sensitivity the hyper-parameters may have with respect to dataset size. After running all S stages the algorithm terminates, and outputs the configuration with the highest validation accuracy from all hyper-parameters explored by all stages (including the initialization points explored by the first stage).

[1] A special case is the initial stage. We adopt the convention that a Sobol sequence is used to initialize the first stage (Snoek et al., 2012). The value k for the first stage is the number of points in the Sobol sequence.

	Hyper-parameters
SVM	bias, cost parameter, and regularization parameter
Boosted regression trees	feature sampling rate, data sampling rate, learning rate, # trees, # leaves, and minimum # instance per leaf

Table 1: Hyper-parameters used in SVM and boosted regression trees.

This multi-stage algorithm subsumes the standard Bayesian optimization algorithm as a special case when the total number of stages $S = 1$. In our case, for datasets used at stages $1, \ldots, S-1$, we use random sampling of full training data to get subsets of data required at these initial stages, while stage S has full data. For the number of top configurations k used to initialize each following stage, we know the larger k is, the better results in the next stage since Bayesian Optimization relies on good initial knowledge to fit good regression models (Feurer et al., 2015). However, larger k value also leads to high computation cost at the next stage, since these initial settings will have to be evaluated first. In practice, the number of stages S and the value of k depend on the quantity of the data and the quality of stage-wise model. In our experiments, we empirically choose their values to be $S = 2$ and $k = 3$ which result in a good balance between accuracy and speed on the given datasets.

3 Experiment

We empirically evaluate the algorithm on two tasks: classification and question answering. For classification we use the Yelp dataset (Yelp, 2014) which is a customer review dataset. Each review contains a star/rating (1-5) for a business, and the task is to predict the rating based on the textual information in the review. The training data contains half-million feature vectors, and unique unigrams are used as features (after standard stop-word re-

moval and stemming (Manning et al., 2008)). For question answering (QA), the task is to identify correct answers for a given question. We use a commercial QA dataset containing about 3800 unique training questions and a total of 900,000 feature vectors. Each feature vector corresponds to an answer candidate for a given question, the vector consists of a binary label (1=correct, 0=incorrect) and values from standard unigram/bigram, syntactic, and linguistic features used in typical QA applications (Voorhees et al., 2011). Both QA and Yelp datasets contain independent training, validation, and test data, from which the machine learning models are built, accuracies are evaluated, and test results are reported, respectively.

We evaluate our multi-stage method against two methods: 1) state-of-the-art Bayesian Optimization for hyper-parameter learning (Snoek et al., 2012)), and 2) the same Bayesian Optimization but only applied on a small subset of data for speed. For experiments, we consider learning hyper-parameters for two machine learning algorithms: SVM implementation for classification (Fan et al., 2008) and boosted regression trees for question answering (Ganjisaffar et al., 2011) as shown in Table 1.

3.1 Accuracy vs time

Figures 1 and 2 compare the test accuracy of our proposed multi-stage Bayesian optimization as a function of tuning time for QA and Yelp, respectively. The state-of-the-art Bayesian optimization (Snoek et al., 2012) is applied on full training data, and the fast variant of Bayesian Optimization is applied with 30% of training data (randomly sampled from full dataset). The top-1 and classification accuracies on test data are reported on the y-axis for QA and Yelp, respectively, and the tuning time is reported on the x-axis. For fairness of comparison, the multi-stage method uses the same 30% training data at the initial stage, and full training data at the subsequent stage.

From these figures, while in general both of the comparison methods produce more effective results when given more time, the multi-stage method consistently achieves higher test accuracy than the other two methods across all optimization time values. For example, best test accuracy is achieved by the multi-stage algorithm at time (45 min) for the QA task, while both the full Bayesian Optimization and the subset variant

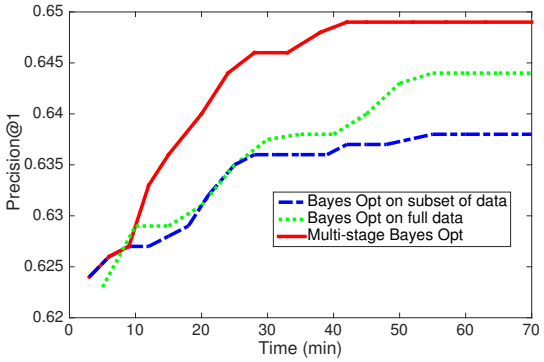


Figure 1: QA task: test accuracy vs tuning time.

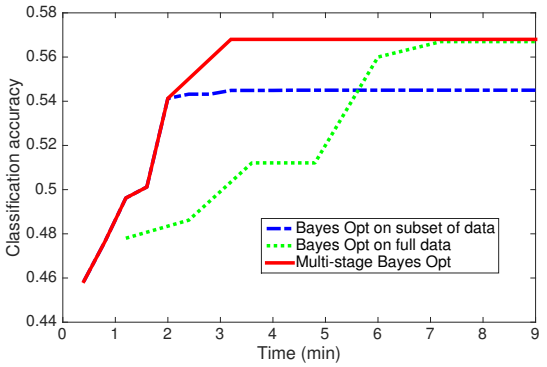


Figure 2: Yelp classification: test accuracy vs tuning time.

can only achieve a fraction of the best value at the same time value. We also note in general the multi-stage algorithm approaches the upper bound more rapidly as more time is given. This shows that the new algorithm is superior across a wide range of time values.

3.2 Expected accuracy and cost per iteration

To investigate the average accuracy and cost per iteration achieved by different methods across different time points, we compare their mean expected accuracy (according to precision@1 for QA and classification accuracy for Yelp) in Table 2, and their average speed in Table 3. In terms of average accuracy, we see that the state-of-the-art Bayesian optimization on full training data and the multi-stage algorithm achieve similar test accuracy, and they both outperform the sub-

	QA	Yelp
Bayes opt on small subset	0.633	0.530
Bayes opt on full data	0.639	0.543
Multi-stage algorithm	0.641	0.542

Table 2: Average test accuracy for QA (precision@1) and Yelp dataset (classif. accuracy).

	QA	Yelp
Bayes opt on small subset	3.2 min	0.4 min
Bayes opt on full data	7.8 min	1.2 min
Multi-stage algorithm	6 min	0.6 min

Table 3: Average time (min) per iteration.

set variant of Bayesian Optimization. However, in terms of time per iteration, the full Bayesian Optimization is the most expensive, taking more than twice amount of time over subset variant algorithm, while the multi-stage is 23% and 50% faster than standard Bayesian Optimization on QA and Yelp (Table 3), respectively, while maintaining the same accuracy as full Bayesian Optimization. This demonstrates the multi-stage approach achieves a good balance between the two baselines and can simultaneously delivers good speedup and accuracy.

4 Conclusion

We introduced a multi-stage optimization algorithm for hyper-parameter optimization. The proposed algorithm breaks the problem into multiple stages with increasingly amounts of data for efficient optimization. We demonstrated its improved performance as compared to the state-of-the-art Bayesian optimization algorithm and fast variants of Bayesian optimization on sentiment classification and QA tasks.

References

Remi Bardenet, Matyas Brendel, Balazs Kegls, and Michele Sebag. 2013. Collaborative hyperparameter tuning. In *ICML 2013: Proceedings of the 30th International Conference on Machine Learning.*

Yoshua Bengio, Aaron Courville, and Pascal Vincent. 2013. Representation Learning: A Review and New Perspectives. In *Pattern Analysis and Machine Intelligence,* Volume:35 , Issue: 8, 2013.

James Bergstra, Remi Bardenet, Yoshua Bengio, and Balazs Kegl. 2011. Algorithms for Hyper-Parameter Optimization. In *NIPS 2011: Advances in Neural Information Processing Systems.*

James Bergstra, and Yoshua Bengio. 2012. Random search for hyper-parameter optimization. In *The Journal of Machine Learning Research,* Volume 13 Issue 1, January 2012.

Eric Brochu, Vlad M. Cora, and Nando de Freitas. 2010. A Tutorial on Bayesian Optimization of Expensive Cost Functions, with Application to Active User Modeling and Hierarchical Reinforcement Learning. In *arXiv:1012.2599v1, Dec 12, 2010.*

Katharina Eggensperger, Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown 2014. Surrogate Benchmarks for Hyperparameter Optimization. In *Meta-Learning and Algorithm Selection Workshop, 2014.*

Katharina Eggensperger, Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. 2015. Efficient Benchmarking of Hyperparameter Optimizers via Surrogates. In *AAAI 2015: Twenty-ninth AAAI Conference.*

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A Library for Large Linear Classification. In *Journal of Machine Learning Research: 2008, 1871-1874.*

Matthias Feurer, Jost Tobias Springenberg, and Frank Hutter. 2015. Initializing Bayesian Hyperparameter Optimization via Meta-Learning. In *AAAI 2015: Twenty-ninth AAAI Conference.*

Yasser Ganjisaffar, Rich Caruana, and Cristina Lopes. 2011. Bagging Gradient-Boosted Trees for High Precision, Low Variance Ranking Models. In *SIGIR 2011: Proceedings of the 34th international ACM SIGIR conference on Research and development in Information.*

Frank Hutter, Holger H. Hoos and Kevin Leyton-Brown. 2011. Sequential Model-Based Optimization for General Algorithm Configuration. In *LION4, 2011.*

Frank Hutter, Holger Hoos and Kevin Leyton-brown. 2014. An Efficient Approach for Assessing Hyperparameter Importance. In *ICML 2014: Proceedings of the 31st International Conference on Machine Learning*

Brent Komer, James Bergstra, and Chris Eliasmith. 2014. Hyperopt-Sklearn: Automatic Hyperparameter Configuration for Scikit-Learn. In *SCIPY 2014: Proceedings of the 13th Python In Science Conference.*

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schutze. 2008. Introduction to Information Retrieval. In *Cambridge University Press, 2008.*

Jasper Snoek, Hugo Larochelle, and Ryan Adams. 2012. Practical Bayesian Optimization of Machine Learning Algorithms. In *NIPS 2012: Advances in Neural Information Processing Systems.*

Kevin Swersky, Jasper Snoek, and Ryan Adams. 2013. Multi-Task Bayesian Optimization. In *NIPS 2013: Advances in Neural Information Processing Systems.*

Chris Thornton, Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. 2013. Auto-WEKA: combined selection and hyperparameter optimization of classification algorithms. In *KDD 2013: Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining.*

Ellen M. Voorhees. 2011. The TREC question answering track. In *Natural Language Engineering,* Volume 7 Issue 4, December 2001

Yelp Academic Challenge Dataset. `http://www.yelp.com/dataset_challenge`.

Dani Yogatama, and Gideon Mann. 2014. Efficient Transfer Learning Method for Automatic Hyperparameter Tuning. In *AISTATS 2014: International Conference on Artificial Intelligence and Statistics* .

Alice X. Zheng, and Mikhail Bilenkoh. 2013. Lazy Paired Hyper-Parameter Tuning. In *IJCAI 2013: Twenty-third International Joint Conference on Artificial Intelligence* .

Improved Arabic Dialect Classification with Social Media Data

Fei Huang
Facebook Inc.
Menlo Park, CA
feihuang@fb.com

Abstract

Arabic dialect classification has been an important and challenging problem for Arabic language processing, especially for social media text analysis and machine translation. In this paper we propose an approach to improving Arabic dialect classification with semi-supervised learning: multiple classifiers are trained with weakly supervised, strongly supervised, and unsupervised data. Their combination yields significant and consistent improvement on two different test sets. The dialect classification accuracy is improved by 5% over the strongly supervised classifier and 20% over the weakly supervised classifier. Furthermore, when applying the improved dialect classifier to build a Modern Standard Arabic (MSA) language model (LM), the new model size is reduced by 70% while the English-Arabic translation quality is improved by 0.6 BLEU point.

1 Introduction

As more and more users share increasing amount of information on various social media platforms (Facebook, Twitter, etc.), text analysis for social media language is getting more important and challenging. When people share their stories, opinions, post comments or tweets on social media platforms, they frequently use colloquial languages, which are more similar to spoken languages. In addition to typical natural language processing problems, the informal nature of social media languages presents additional challenges, such as frequent spelling errors, improper casing, internet slang, spontaneity, dis-fluency and ungrammatical utterances (Eisenstein, 2014). Dialect classification and dialect-specific processing are extra challenges for languages such as Arabic and Chinese.

Considering Arabic as an example: there are big differences between MSA and various dialectal Arabic: MSA is the standardized and literary variety of Arabic used in writing and in most formal speech.[1] It is widely used in government proceedings, newspapers and product manuals. Many research and linguistic resources for Arabic natural language processing are based on MSA. For example, most existing Arabic-English bilingual data are MSA-English parallel sentences. The dialect Arabic has more varieties: 5 major dialects are spoken in different regions of the Arab world: *Egyptian, Gulf, Iraqi, Levantine* and *Maghrebi* (Zaidan and Callison-Burch, 2011). These dialects differ in morphologies, grammatical cases, vocabularies and verb conjugations. These differences call for dialect-specific processing and modeling when building Arabic automatic speech recognition (ASR) systems or machine translation (MT) systems. Therefore, identification and classification of Arabic text is fundamental for building social media Arabic speech and language processing systems.

In order to build better MT systems between Arabic and English, we first analyze the distribution of different Arabic dialects appearing on a very large scale social media platform, as well as their effect on Arabic-English machine translation. We propose several methods to improve the dialect classification accuracy by training models with distant supervision: a weakly supervised model is trained with data whose labels are automati-

[1] http://en.wikipedia.org/wiki/Modern_Standard_Arabic

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 2118–2126,
Lisbon, Portugal, 17-21 September 2015.

cally assigned based on authors' geographical information. A strongly supervised model is trained with manually annotated data. More importantly, semi-supervised learning on large amount of unlabeled data effectively increases the classification accuracy. We also combine different classifiers to achieve even bigger improvement. When evaluated on two test sets, the widely adopted Arabic Online Commentary (AOC) corpus and a test set created from the social media domain (Facebook), our methods demonstrate an absolute 20% improvement over the weakly supervised classifier, and 5% over the strongly supervised classifier. Furthermore, the improved classifier is applied on large amount of Arabic social media text to filter out non-MSA data. An LM trained with the cleaned data is used for English-Arabic (MSA) translation. Compared with the baseline model trained with the unfiltered data, the MSA LM reduces the training data by 85%, model size by 70%, and it brings 0.6 BLEU point (Papineni et al., 2002) gain in MT.

The rest of the paper is organized as follows: in section 2 we review previous research on this topic. In section 3 we analyze the dialect distribution and its impact on social media data translation. We present the problem formulation in section 4. In section 5 we introduce two supervised classifiers trained with weakly and strongly labeled data. We describe different semi-supervised learning methods in section 6, followed by the combination of multiple classifiers in section 7. In section 8 we show the experimental results on dialect classification as well as machine translation. The paper finishes with discussions and conclusion in section 9.

2 Related Work

Previous research on Arabic dialect identification focused on two problems: spoken dialect classification for speech recognition ((Novotney et al., 2011) and (Lei and Hansen, 2011)), and written text dialect classification mostly for machine translation. (Habash and Rambow, 2006), (Habash et al., 2008), (Diab et al., 2010) and (Elfardy and Diab, 2012) developed annotation guidelines and morphology analyzer for Arabic dialect.

(Zaidan and Callison-Burch, 2011) created the AOC data set by extracting reader commentary from online Arabic newspaper forums. The selected Arabic sentences are manually labeled with one of 4 dialect labels with the help of crowd sourcing: *Egyptian*, *Gulf*, *Iraqi* and *Levantine*. A dialect classifier using unigram features is trained from the labeled data. In the BOLT (Broad Operational Language Translation) project, translation from dialectal Arabic (especially Egyptian Arabic) to English is a main problem. (Elfardy and Diab, 2013) uses the same labeled AOC data to generate token-based features and perplexity-based features for sentence level dialect identification between MSA and Egyptian Arabic. (Tillmann et al., 2014) trained feature-rich linear classifier based on linear SVM then evaluated the classification between MSA and Egyptian Arabic, reporting 1.4% improvement. All these experiments are based on the AOC corpus. The characteristics and distribution of the Arabic dialects could be different for online social media data. (Darwish et al., 2014) selected Twitter data and developed models taking consideration of lexical, morphological, and phonological information from different dialects, then classified Egyptian and MSA Arabic tweets. (Cotterell and Callison-Burch, 2014) collected dialect data covering Iraqi and Maghrebi Arabic from Twitter as well.

When translating Arabic dialect into English, (Sawaf, 2010) and (Salloum and Habash, 2011) normalized dialect words into MSA equivalents considering character- and morpheme-level features, then translated the normalized input with MSA Arabic-English MT system. (Zbib et al., 2012) used crowd sourcing to build Levantine-English and Egyptian-English parallel data. Even with small amount of parallel corpora for each dialect, they obtained significant gains (6-7 BLEU pts) over a baseline MSA-English MT system.

3 Social Media Arabic Dialect Distribution and Translation

The population speaking a dialect does not necessarily reflect its popularity on internet and social media. Many factors, such as a country's social-economic development status, internet access and government policy, play important roles. To understand the distribution of Arabic dialects on social media, we select data from the largest social media platform, Facebook. There are around one billion users sharing content in 60+ languages every day. The Arabic content comes from different regions of the Arabic world, representative enough for our analysis. We randomly select 2700

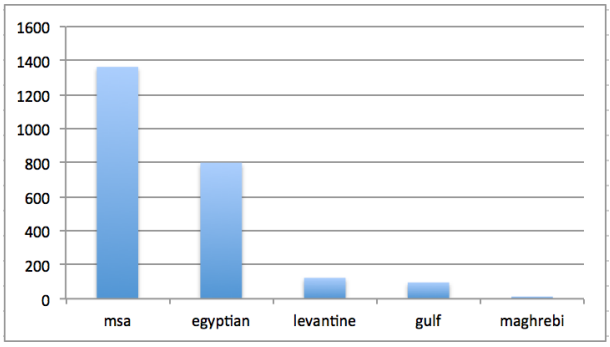


Figure 1: Distribution of various Arabic dialect on the social media platform

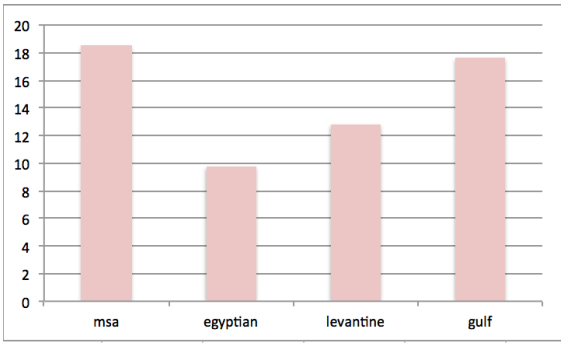


Figure 2: BLEU scores of different Arabic dialects in Arabic-English translation. The MT model is trained with mostly MSA-English parallel data.

sentences from public posts, then ask human annotators to label their dialect types[2]. The result is shown in Figure 1. Not surprisingly, MSA is the most widely used, accounting for 58% of sentences. Besides that, Egyptian Arabic is the most frequent dialect (34%), followed by Levantine and Gulf. Maghrebi is the least frequent. There are other sentences which are not labeled as Arabic dialect, such as classical Arabic, verses from the Quran, foreign words and their transliterations, etc.

We also investigate the effect of different dialects on Arabic-English translation. We ask humans to translate the Arabic sentences into English to create reference translations. We build a phrase-based Arabic-English MT system with 1M sentence pairs selected from MSA Arabic-English parallel corpora (UN corpus, Arabic news corpus, etc.).[3] The training and decoding procedures are similar to those described in (Koehn et al., 2007). More details about the MT system are given in section 8. We group the source Arabic sentences into different subsets based on their dialect labels, then translate them with the MT system. We measure the BLEU score for each subset, as shown in Figure 2. As expected the MSA subset has the highest BLEU scores (18), followed by the Gulf dialect, which is somewhat similar to MSA. The translation of the Egyptian and Levantine dialects is more challenging, with BLEU scores around 10-12, even though they are 40% of the total Arabic data. To improve Arabic-English MT quality, increasing the bilingual data coverage for these two dialects should be most effective, as seen in (Zbib et al., 2012). Because the Maghrebi dialect sample size is too small, we do not report its BLEU score. From these experiments, we further appreciate the importance of accurately identifying Arabic dialect and building dialect-specific translation models.

[2]The data was annotated by a translation service provider under confidentiality agreement.

[3]Because existing Arabic-English bilingual corpora do not include parallel data from social media domain, increasing training data size does not increase the translation quality.

4 Problem Formulation

In this section we present the general framework of dialect classification. Given a sentence $S = \{w_1, w_2, .., w_l\}$ generated by user u, its dialect class label d^* is determined based on the following functions:

$$d^* = \arg\max_i P(d_i|S, u),$$

where the probability function is defined according to the following exponential model:

$$P(d_i|S, u) = \frac{\exp \sum_k \lambda_k f_k(d_i, \cdot)}{\sum_j \exp \sum_k \lambda_k f_k(d_j, \cdot)}$$

$$d^* = \arg\max_i \sum_k \lambda_k f_k(d_i, \cdot).$$

Here $f_k(d_i, \cdot)$ is the k-th feature function. For example. $f(d_i, u)$ models the likelihood of writing dialect d_i by user u given the user's profile information. $f(d_i, S)$ models the likelihood of generating sentence S with d_i's n-gram language model:

$$f(d_i, S) = \log p(S|d_i)$$
$$= \sum_{k=1}^{l} \log p_{d_i}(w_k|w_{k-1}, ..., w_{k-n+1}).$$

This framework allows the incorporation of rich feature functions such as geographical, lexical,

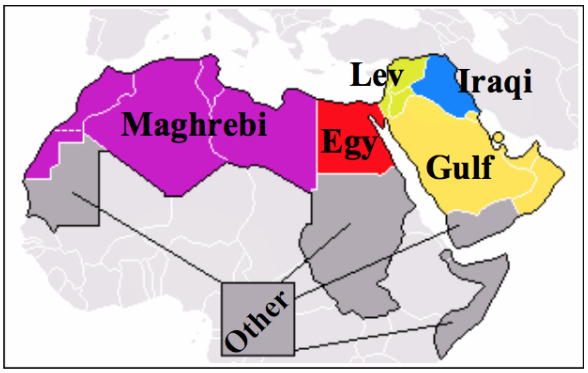


Figure 3: Arabic dialect map, from (Zaidan and Callison-Burch, 2011).

morphological and n-gram information, as seen in previous work ((Zaidan and Callison-Burch, 2011) , (Darwish et al., 2014), (Tillmann et al., 2014) and (Elfardy and Diab, 2013)). However, in this paper we focus on training classifiers with weakly and strongly labeled data, as well as semi-supervised learning methods. So we only choose the geographical and text-based features. Exploration of other features will be reported in another paper.

Previous research (Zaidan and Callison-Burch, 2014) indicated that the unigram model obtains the best accuracy in dialect classification. However, (Tillmann et al., 2014) and (Darwish et al., 2014) exploited more sophisticated text features that lead to better accuracy on selected test set. In our experiments, we find that the unigram model does outperform bigram and trigram models, so we stick to the unigram features.

5 Supervised Learning

5.1 Learning with Weakly Labeled Data

In the chosen social media platform, each user is associated with a unique profile, which includes user-specific information such as the age and gender of the user, the country where s/he is from, etc.. As different Arabic dialects are spoken in different countries, one approach is to classify a post's dialect type based on the author's country, assuming that there is at least a major dialect spoken in each country. This approach is not highly accurate, because the user's country information may be missing or inaccurate; one dialect may be spoken in multiple countries (for example, Egyptian is very popular in different regions of the Arabic world) and multiple dialects may be spoken in the same country; the user can post in MSA instead of dialect Arabic or a mixture of both. However, using data from certain countries as the "approximate" dialect training data, we can train a baseline classifier. As the training data labels are inferred from user profiles instead of manually annotated, such data is called weakly labeled data.

According to the dialect map shown in Fig.3, we group the social media posts into the following 5 dialects according to the author's country:

1. *Egyptian*: Egypt
2. *Gulf*: Saudi Arabia, United Arab Emirate, Qatar, Bahrain, Oman, Yemen
3. *Levantine*: Syrian, Jordan, Palestinian, Lebanese
4. *Iraqi*: Iraq
5. *Maghrebi*: Algeria, Libya, Tunisia, Morocco

Table 1 shows the number of words for each dialect group. Considering the dialect distribution in the social media platform (shown in Figure 1), we focus on the classification of *MSA* (msa) and 3 Arabic dialects: *Egyptian* (egy), *Gulf* (gul) and *Levantine* (lev).

We train an n-gram model for each dialect from the collected data. To train the MSA model, we select sentences from Arabic UN corpus and news collections. All the dialect and MSA models share the same vocabulary, thus perplexity can be compared properly. At classification time, given an input sentence, the classifier computes the perplexity for each dialect type and choose the one with minimum perplexity as the label.

Dialect	Weakly Labeled	Strongly Labeled
egy	22M	0.45M
gul	6M	0.34M
lev	8M	0.45M
msa	27M	1.34M
iraqi	3M	0.01M

Table 1: Corpus size (word count) of weakly and strongly labeled data for supervised learning. The weakly labeled dialect data is from Facebook based on users' country information. The strongly labeled data is manually annotated from the AOC corpus.

5.2 Learning with Strongly Labeled Data

In the AOC corpus, every sentence's dialect type is labeled by human annotators. As these labels

are gold labels, the AOC corpus is strongly labeled data. Because of the high cost of manual annotation, the strongly labeled data is much less than the weakly labeled data, but the higher quality makes it possible to train a better classifier. Table 1 shows the corpus size. Although over 50% data is MSA. Egyptian, Gulf and Levantine dialects still have significant presence while the Iraqi dialect has the least labeled data. Such distribution is consistent with what we observed from the social media data. Using these strongly labeled data, we can train a classifier that significant outperforms the weakly supervised classifier.

6 Semi-supervised Learning

6.1 Self-training

Given the small amount of gold labeled data from the AOC corpus and large amount of unlabeled data from the social media platform, a natural combination is semi-supervised learning. In other words, by applying the strongly supervised classifier on the unlabeled data, we can obtain "automatically labeled" dialect data that could further improve the classification accuracy. From the social media platform we select additional Arabic posts with a total of 646M words. The sizes of the newly created dialect corpora are shown in Table 2. Notice that the MSA data accounts for more than 75% of all the labeled data. We train a new classifier with these additional data. As the new labels are only from the original strong classifier, this is self-training.

6.2 Co-Training

Another approach for automatic labeling is co-training (Blum and Mitchell, 1998). With two classifier C_1 and C_2 classifying the same input sentence S with labels l_1 and l_2, S is labeled as l only if $l_1 = l_2 = l$. In other words, a sentence is labeled and used to train a model only when the two classifiers agree. In our experiment we use both the weakly and strongly supervised classifiers to classify the same unlabeled data. Table 2 lists the sizes of the dialect corpora from co-training. Compared with the self-training approach, the co-training method filters out 25% data.

6.3 Data Filtering

Because of domain mismatch, even the strongly supervised classifier does not achieve very high accuracy on the social media test set, thus there is lots of noise in the automatically labeled data. To filter this noise, we only keep the sentences whose minimum perplexity score (corresponding to the winning dialect label) is smaller than any other perplexity score by a margin. Lower perplexity means higher probability of generating the sentence from the dialect model. In other words, sentence S is assigned with label l and used in model re-training if and only if $perp_l(S) < perp_k(S) \times threshold$, for $k \neq l$. The threshold is selected to optimize the classification accuracy on a tuning set. Table 2 also shows the corpora size after filtering. We can see that the filtered dialect is only a quarter of the self-training data. We will compare the three semi-supervised learning methods and evaluate the gains to dialect classification.

Dialect	Self-training	Co-training	Filter
egy	73M	54M	21M
gul	46M	5.1M	2.7M
lev	34M	11.7M	2.5M
msa	493M	406M	139M
All	646M	476M	165M

Table 2: The size of dialect corpora from semi-supervised learning.

7 Classifier Combination

Now we have 3 types of classifiers:

1. The weakly supervised classifier trained with data whose labels are automatically assigned according to author's country;
2. The strongly supervised classifier trained with human labeled data;
3. The semi-supervised classifier trained with automatically classified data, with different data selection methods.

How should we combine them to further improve the classification accuracy?

One approach is data combination: simply adding all the training data together to train a unified n-gram model for each dialect. This experiment is straightforward but the performance is suboptimal because the classifier will be dominated by the model with the most training data, even though its accuracy may not be the best.

The second approach is model combination: we compute the model scores of the weakly supervised (w), strongly supervised (s) and semi-supervised (e) classifiers, then combines them

with linear interpolation:

$$p(S|d_i) = \sum_{m=\{w,m,e\}} w_m p_m(S|d_i)$$

As the dialect n-gram perplexity is computed separately, the model weights w_m can be tuned. In our experiments we optimize them with a tuning set from all the dialects.

8 Experiment Results

8.1 Dialect Classification

We already described the training data for supervised and semi-supervised classifiers in previous sections. In this section we will compare their dialect classification accuracies. We select two test sets: 9.5K sentences from the AOC corpus as the AOC test set and 2.3K sentences from the Facebook data set as the FB test set[4]. Both test sets have the dialect of each sentence labeled by human. The accuracy is computed as the percentage of sentences whose classified label is the same as the human label. 90% of the AOC labeled data are used for training the strongly supervised classifier, and the remaining 10% data containing 9.5K sentences is for evaluation. We also keep 200 sentences from the AOC corpus as the development set to tune the model combination parameters.

Model	AOC	FB
weakly supervised	68.4%	48.5%
strongly supervised	83.4%	63.1%
semi-supervised	86.2%	67.7%
combination	87.8%	68.2%

Table 3: Arabic dialect classification accuracies with the weakly and strongly supervised classifiers, as well as the semi-supervised model.

In Table 3 we show the overall classification accuracies of different models on both test sets. Notice that the weakly supervised classifier trained with 68M words obtains 68% accuracy on the AOC test set and 48% on the FB test set (row 1), which is not much higher. However, considering this classifier is trained without any human labeled dialect data, the performance is expected and can be improved with better training data and models. The strongly supervised classifier (row 2), which is trained with much less human labeled data (only 2.6M words), outperforms the weak classifier by 15%. Such a difference is consistently observed in both test tests. This confirms the significant benefits from the gold labeled data.

We apply the strong classifier to large amount of unlabeled data, and train several semi-supervised classifiers with these automatically labeled data. The best result is obtained with the co-training strategy, which brings significant improvement over the strongly supervised model: 2.8-4.6% (row 3), as the label noise is effectively reduced among the agreed labels from two supervised classifiers. Finally, combining all three classifiers (row 1, 2 and 3) with model combination achieves the best result: about 5% improvement of the strong baseline and 20% over the weak baseline. These results demonstrate the effectiveness of combining labeled and unlabeled data obtained from social media platform.

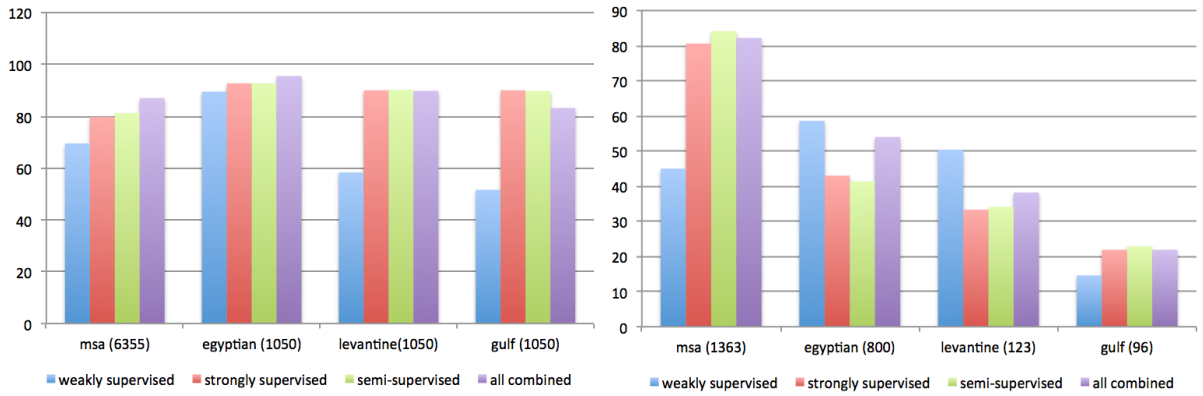
Model	AOC	FB
strongly supervised baseline	83.4%	63.1%
self-training	84.4%	65.5%
co-training	86.2%	67.7%
data filtering	85.2%	64.8%
model interpolation	87.8%	68.2%
data concatenation	82.1%	67.4%

Table 4: Comparison of semi-supervised learning and combination methods.

With semi-supervised learning, we evaluate three data selection methods: self-training, co-training and data filtering. The results are shown in Table 4. Compared with the strong classifier baseline, the self-training method improves by 1% - 2.4%, the co-training method improves by 2.8-4.6%, and the data filtering method improves by 1.7-1.8%. The co-training method is the most effective for both test sets because the information are from two independent classifiers. Data filtering is more effective for the AOC test set (which has the same domain as the baseline model) but less so for the FB test set because valuable in-domain data are filtered out.

In the same table we also compare the results from model and data combination: one from the semi-supervised co-training and the other from the strongly supervised learning. On the AOC test set, the data concatenation method is significantly

[4]The FB test set is available for download at https://www.facebook.com/groups/2419174607/10153205046974608/.



(a) Result on the AOC test set.

(b) Result on the Facebook test set.

Figure 4: Classification precisions by dialect. The number in parenthesis is the number of sentences from each dialect.

worse than the model interpolation method. Its accuracy is even lower than that of the supervised classifier (82.1% vs. 83.4%). However the gap is much smaller on the FB test set. The automatically labeled data is much more than the human labeled data, thus it dominates the combined training data set, which is not a good match for the AOC test data, but is more relevant to the FB test data. In both cases, the model combination obtains better classification accuracies, where the supervised model is assigned higher weights (0.9) and the semi-supervised model is used for smoothing, therefore the combined model is able to improve over the strong classifier.

We further analyze the classification precision for each type of dialect on both test sets in Figure 4. Figure 4a shows the result on the AOC test set. The number after the dialect type (in the parenthesis) is the number of sentences from that dialect. Precisions increase from the weakly supervised to the strongly supervised to the semi-supervised classifier, and the combined classifier generally outperforms all three classifiers, except for the Gulf dialect. However, considering the smaller percentage of the Gulf dialect, we still observe significant improvement overall. Figure 4b shows the result on the FB test set, where the MSA and Egyptian dialects are much more frequent than the Levantine and Gulf dialects. Improving classification on the MSA and Egyptian dialect (especially MSA) will be very helpful. We notice that the supervised classifier improves over the unsupervised classifier by a large margin on the MSA and Gulf dialects, but performs worse on the Egyptian and Levantine dialects. This is different from the result in the AOC test set, where the supervised classifier consistently improves over the unsupervised classifier. One reason is that in the AOC test set, the training and test data are from the same corpus, thus the supervised training from in-domain data is very effective. For the FB test set, the strongly labeled data and the test data mismatch in genre and topics. The automatically labeled data is less similar to the dialect test set, thus it is less effective for the Egyptian and Levantine dialects. This further confirms the necessity of combining information from multiple sources. The combined classifier performs significantly better for the MSA and Gulf dialect, but slightly worse for the Egyptian and Levantine dialects. The overall result is still positive.

We also compare our approach with other dialect classification methods on the AOC corpus, which is commonly used so the results are comparable. Most previous work focus on the classification of MSA vs. EGY dialect, and report the accuracies from 85.3% (Elfardy and Diab, 2013), 87.9% (Zaidan and Callison-Burch, 2014) to 89.1% (Tillmann et al., 2014), adding morphological features, using word-based unigram-model and linear SVM models. Our MSA vs. EGY dialect classification accuracy is 92.0%, the best known result on this test set. We do not use more sophisticated features; the improvement is just from the mined unlabeled data and the combination of different classifiers. On the FB test set, our strongly supervised classifier is the same as (Zaidan and Callison-Burch, 2014), both using word-based unigram model. We see 5% gain with the combined classifiers.

8.2 Machine Translation

The motivation of this research is to handle challenges from Arabic dialects to improve machine translation quality. For example, using the dialect classifier output one can build dialect-specific Arabic-English MT systems. Given an Arabic sentence, the system first identifies its dialect type, then translates with the corresponding MT system. When building English-to-Arabic (MSA) translation systems for social media translation, the target LM trained from in-domain data is very helpful to improve the translation quality. Considering that the Arabic in-domain data contains lots of dialects, an effective dialect classifier helps filter out dialect Arabic and only keep the MSA to train a cleaner LM.

Because of the limited bilingual resources of dialect Arabic-English, we will focus on English-Arabic MT system first. In this experiment, the training data for the English-Arabic MT system is 1M parallel sentences selected from publicly available Arabic-English bilingual data (LDC, OPUS). Because none of the parallel corpora is for social media translation, we select a subset closer to the social media domain by maximizing the n-gram coverage on the test domain. The development and test sets contain 700 and 892 English sentences, respectively. These sentences are translated into MSA by human translators. We apply the standard SMT system building procedures: pre-processing, automatic word alignment, phrase extraction, parameter tuning with MERT, and decoding with a typical phrase-based decoder similar to (Koehn et al., 2007). The LM is trained with the target side of the parallel data, plus 200M in-domain Arabic sentences.

Using the above combined dialect classifier, we label the dialect type of each sentence in the in-domain data, filter out any non-MSA sentences and re-train the target LM. Again to keep the in-domain data clean, we also apply the threshold-based data filtering. As shown in Table 5, the dialect filtering reduces the LM training data by 85%, which corresponds to 70% less memory footprint. Thanks to the cleaner LM, the translation quality is also improved by 0.6 BLEU point.[5]

[5]Due to the challenging nature of social media data, and the lack of in-domain training data, the BLEU score is much lower than the one in news translation.

	All Arabic Data	Filtered MSA data
number of sentences	200M	30M
memory footprint	23G	6.6G
BLEU score (1-reference)	12.52	13.14

Table 5: Cleaned MSA LM after dialect filtering for English-Arabic(MSA) translation.

9 Discussion and Conclusion

Existing Arabic dialect classification methods solely rely on textural features, be they n-gram language model or morphology/POS-based features. This paper utilizes authors' geographical information to train a weakly supervised dialect classifier. Using the weakly and strongly supervised classifiers to classify and filter unlabeled data leads to several improved semi-supervised classifiers. The combination of all three significantly improves the Arabic dialect classification accuracy on both in-domain and out-of-domain test sets: 20% absolute improvement over the weak baseline and 5% absolute over the strong baseline. After applying the proposed classifier to filter out Arabic dialect data, and building a cleaned MSA LM, we observe 70% model size reduction with 0.6 BLEU point gain in English-Arabic translation quality.

In future work, we would like to explore more user-specific information for dialect classification, apply the classifier for Arabic-to-English MT systems, and extend the approach to a larger family of languages and dialects.

References

Avrim Blum and Tom Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100. ACM.

Ryan Cotterell and Chris Callison-Burch. 2014. A multi-dialect, multi-genre corpus of informal written arabic. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, Reykjavik, Iceland, may.

Kareem Darwish, Hassan Sajjad, and Hamdy Mubarak. 2014. Verifiably effective arabic dialect identification. In *Proceedings of the 2014 Conference on*

Empirical Methods in Natural Language Processing (EMNLP), pages 1465–1468. Association for Computational Linguistics.

Mona Diab, Nizar Habash, Owen Rambow, Mohamed Altantawy, and Yassine Benajiba. 2010. Colaba: Arabic dialect annotation and processing. In *LREC Workshop on Semitic Language Processing*, pages 66–74.

Jacob Eisenstein. 2014. Identifying regional dialects in online social media.

Heba Elfardy and Mona T Diab. 2012. Simplified guidelines for the creation of large scale dialectal arabic annotations. In *LREC*, pages 371–378.

Heba Elfardy and Mona T Diab. 2013. Sentence level dialect identification in arabic. In *ACL (2)*, pages 456–461.

Nizar Habash and Owen Rambow. 2006. Magead: a morphological analyzer and generator for the arabic dialects. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 681–688. Association for Computational Linguistics.

Nizar Habash, Owen Rambow, Mona Diab, and Reem Kanjawi-Faraj. 2008. Guidelines for annotation of arabic dialectness. In *Proceedings of the LREC Workshop on HLT & NLP within the Arabic world*, pages 49–53.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, ACL '07, pages 177–180, Stroudsburg, PA, USA. Association for Computational Linguistics.

Yun Lei and John HL Hansen. 2011. Dialect classification via text-independent training and testing for arabic, spanish, and chinese. *Audio, Speech, and Language Processing, IEEE Transactions on*, 19(1):85–96.

Scott Novotney, Richard M Schwartz, and Sanjeev Khudanpur. 2011. Unsupervised arabic dialect adaptation with self-training. In *INTERSPEECH*, pages 541–544. Citeseer.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Wael Salloum and Nizar Habash. 2011. Dialectal to standard arabic paraphrasing to improve arabic-english statistical machine translation. In *Proceedings of the First Workshop on Algorithms and Resources for Modelling of Dialects and Language Varieties*, pages 10–21. Association for Computational Linguistics.

Hassan Sawaf. 2010. Arabic dialect handling in hybrid machine translation. In *Proceedings of the 9th Conference of the Association for Machine Translation in the Americas*.

Christoph Tillmann, Saab Mansour, and Yaser Al-Onaizan, 2014. *Proceedings of the First Workshop on Applying NLP Tools to Similar Languages, Varieties and Dialects*, chapter Improved Sentence-Level Arabic Dialect Classification, pages 110–119. Association for Computational Linguistics and Dublin City University.

Omar F. Zaidan and Chris Callison-Burch. 2011. The arabic online commentary dataset: an annotated dataset of informal arabic with high dialectal content. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 37–41, Portland, Oregon, USA, June. Association for Computational Linguistics.

Omar F Zaidan and Chris Callison-Burch. 2014. Arabic dialect identification. *Computational Linguistics*, 40(1):171–202.

Rabih Zbib, Erika Malchiodi, Jacob Devlin, David Stallard, Spyros Matsoukas, Richard Schwartz, John Makhoul, Omar F Zaidan, and Chris Callison-Burch. 2012. Machine translation of arabic dialects. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 49–59. Association for Computational Linguistics.

Exploiting Debate Portals for Semi-Supervised Argumentation Mining in User-Generated Web Discourse

Ivan Habernal[†] and **Iryna Gurevych**[†‡]

[†]Ubiquitous Knowledge Processing Lab (UKP-TUDA)
Department of Computer Science, Technische Universität Darmstadt
[‡]Ubiquitous Knowledge Processing Lab (UKP-DIPF)
German Institute for Educational Research
www.ukp.tu-darmstadt.de

Abstract

Analyzing arguments in user-generated Web discourse has recently gained attention in argumentation mining, an evolving field of NLP. Current approaches, which employ fully-supervised machine learning, are usually domain dependent and suffer from the lack of large and diverse annotated corpora. However, annotating arguments in discourse is costly, error-prone, and highly context-dependent. We asked whether leveraging unlabeled data in a semi-supervised manner can boost the performance of argument component identification and to which extent is the approach independent of domain and register. We propose novel features that exploit clustering of unlabeled data from debate portals based on a word embeddings representation. Using these features, we significantly outperform several baselines in the cross-validation, cross-domain, and cross-register evaluation scenarios.

1 Introduction

Argumentation mining, an evolving sub-field of NLP, deals with analyzing argumentation[1] in various genres, such as legal cases (Mochales and Moens, 2011), student essays (Stab and Gurevych, 2014a), and medical and scientific articles (Green, 2014; Teufel and Moens, 2002). Recently, the focus of argumentation mining has also shifted to the Web registers (such as comments to articles, forum posts, or blogs) which is motivated by the need of retrieving and understanding ordinary people's arguments to various contentious topics on the large scale. Applications include passenger rights and protection (Park and Cardie, 2014), hotel reviews (Wachsmuth et al., 2014), and controversies in education (Habernal et al., 2014).

Despite the plethora of existing argumentation theories (van Eemeren et al., 2014), the prevalent view in argumentation mining treats arguments as discourse structures consisting of several argument components, such as *claims* and *premises* (Peldszus and Stede, 2013). Current approaches to automatic analysis of argumentation usually follow the fully supervised machine-learning paradigm (Biran and Rambow, 2011; Stab and Gurevych, 2014b; Park and Cardie, 2014) and rely on manually annotated datasets. Only few publicly available argumentation corpora exist, as annotations are costly, error-prone, and require skilled human annotators (Stab and Gurevych, 2014a; Habernal et al., 2014).

To overcome the limited scope and size of the existing annotated corpora, semi-supervised methods can be adopted, as they gain performance by exploiting large unlabeled datasets (Settles, 2012). However, unlike in other NLP tasks where data can be cheaply labeled using for example distant supervision, employing such methods in argumentation mining is questionable. First, argumentation is an act of persuasion (Nettel and Roque, 2011; Mercier and Sperber, 2011) but not all user-generated texts can be treated as persuasive (Park and Cardie, 2014; Habernal et al., 2014), thus the selection of an appropriate unlabeled dataset represents a problem on its own. Second, argument components (e.g., *claims* or *premises*) are highly context-dependent and cannot be easily labeled in distant data using predefined patterns. So far, semi-supervised methods for argumentation mining remain unexplored.

In this article, we tackle argumentation min-

[1]Argumentation is *a verbal activity for which the goal consists of convincing the listener or reader of the acceptability of a standpoint by means of a constellation of propositions justifying or refuting the proposition expressed in the standpoint* (van Eemeren et al., 2002) or *the art of persuading others to think or act in a definite way, including all writing and speaking which is persuasive in form* (Ketcham, 1917).

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 2127–2137,
Lisbon, Portugal, 17-21 September 2015.

ing of user-generated Web data by exploiting *debate portals*—semi-structured discussion websites where members pose contentious questions to the community and allow others to pick a side and provide their opinions and arguments in order to 'win' the debate.[2] Our first research question is whether debate portals (which contain noisy user-generated data) can be utilized in a semi-supervised manner for fine-grained identification of argument components. As a second research question, we investigate to what extent our methods are domain independent and evaluate their adaptation across several domains and registers.

Our contribution is three-fold. First, to the best of our knowledge, we present the first successful attempt to semi-supervised argumentation mining in Web data based on exploiting unlabeled external resources. We leverage these resources and derive features in an unsupervised manner by projecting data from debate portals into a latent argument space using unsupervised word embeddings and clustering. Second, our novel features significantly outperform state-of-the-art features in all scenarios, namely in cross-validation, cross-domain evaluation, and cross-register evaluation. Third, to ensure full reproducibility of our experiments, we provide all data and source codes under free licenses.[3]

2 Related work

Analysis of argumentation has been an active topic in numerous research areas, such as philosophy (van Eemeren et al., 2014), communication studies (Mercier and Sperber, 2011), and informal logic (Blair, 2004), among others. In this section, we will focus on the most related works on argumentation mining techniques in NLP in the first part, with an emphasis on Web data in the second part.

Mochales and Moens (2011) based their work on argumentation schemes (Walton et al., 2008) and experimented with Araucaria and ECHR datasets using supervised models to classify argumentative and non-argumentative sentences ($\approx 0.7F_1$) and their structure. Feng and Hirst (2011) classified argument schemes on the Araucaria dataset, reaching 0.6-0.9 accuracy. Experiments on this dataset were also conducted by Rooney et al. (2012), who classified sentences to four categories (*conclusion*, *premise*, *conclusion-premise*, and *none*) and achieved 0.65 accuracy. These approaches assume the text is already segmented into argument components. Stab and Gurevych (2014b) examined argumentation in persuasive essays and classified argument components into four categories (*premise*, *claim*, *major claim*, *non-argumentative*) using SVM and achieved 0.73 macro F_1 score. They further classified argument relations (support and attack) and reached 0.72 macro F_1 score. The best-performing features were structural features (such as the location or length ratios), as persuasive essays usually comply with a certain structure which can be seen as a potential drawback of this approach.

Regarding user-generated Web data, Biran and Rambow (2011) used naive Bayes for classifying justification of subjective claims from blogs and Wikipedia talk pages, relying on features from RST Treebank and manually-processed n-grams. In similar Web registers, Rosenthal and McKeown (2012) automatically determined whether a sentence is a *claim* using logistic regression and various lexical and sentiment-related features and achieved accuracy about 0.66-0.71. Park and Cardie (2014) classified propositions in user comments into three classes (*verifiable experiential*, *verifiable non-experiential*, and *unverifiable*) using SVM and reached 0.69 macro F_1 score. Goudas et al. (2014) identified *premises* in Greek social media texts using BIO encoding and achieved 0.42 F_1 score with Conditional Random Fields. The research gaps in the above-mentioned approaches are the following. First, the argumentation models are simplified to either *claims* or a few types of *premises/propositions*. Second, the segmentation of discourse into argument components is ignored (except the work of Goudas et al. (2014)). Recently, Boltužić and Šnajder (2015) employed hierarchical clustering to cluster arguments in online debates using embeddings projection, but in contrast to our work they performed only intrinsic evaluation of the clusters.

Debate portals have been used in a related body of research, such as classifying support and attack between posts by Cabrio and Villata (2012), or stance detection by Hasan and Ng (2013) or Gottipati et al. (2013). These approaches consider the complete documents (posts) but do not analyze the micro-level argumentation (e.g., *claims* or *premises*).

[2] For instance `createdebate.com` or `debate.org`

[3] `https://github.com/habernal/emnlp2015`

Doc #2823 (article comment, public-private-schools): [*claim:* I agree - Kids can do great in the public school system and parents DO need to be involved.] The more people leave, the worse its going to become. [*premise:* The public school system lets them deal with real life too, unfortunate that it may be but that is what's out there in college and the work force too.] [*premise:* There are still great teachers in the public schools - lets stand behind them.]

Doc #2224 (forumpost, single-sex-education): [*backing:* I went to an all boys school –] [*claim:* Can't say I particularly liked it, I would of much preferred gone to a co-ed.] [*premise:* It is closer to the 'real world' that way. Kids should grow up in the company of both sexes... They will be more at ease around the opposite sex when they are older and it just makes sense.] If it is purely education you are concerned about (and not so much behaviour), our year (at a private school) went shockingly bad in OP scores. We were the worst in 12 years and were beaten by LOTS of co-ed and public schools... So you can never tell. In saying that my sister really enjoyed going to an all girls school. Her year went really well too. Ask your daughters what they would prefer... [*backing:* Btw, I work at a co-ed school at the moment and the kids there get on just fine.]

Figure 1: Two examples of argument annotation of an article comment and a forum post.

3 Data

As data for training and evaluation of our methods, we use a corpus consisting of 340 English documents (approx. 90k tokens) annotated[4] with argumentation by Habernal et al. (2014). Compared to other corpora mentioned in the related work, this corpus is the largest one to date that covers different domains and spans several registers of user-generated Web content. In particular, the corpus comprises four registers (comments to articles, forum posts, blogs, and argumentative newswire articles) and covers six domains related to educational controversies (homeschooling, private vs. public schools, mainstreaming, single-sex education, prayer in schools, and redshirting).

The argumentation model used in this corpus is based on extended Toulmin's model (Toulmin, 1958). Each document contains usually one argument, where each argument consists of several argument components. There are five different components in this model, namely, the *claim* (the statement about to be established in the argument which conveys author's stance towards the topic), the *premise(s)* (propositions that are intended to give reasons of some kind for the claim), the *backing* (additional information used to back-up the argument), the *rebuttal* (attacks the claim), and the *refutation* (which attacks the rebuttal). Relations between the argument components are encoded implicitly in the function of the particular component type, for instance, premises are always attached to the claim. We made two observations in the data: the *claim* is often implicit (must be inferred by the reader), and some sentences have no argumentative function (thus are not labeled by any argument component).[5]

Figure 1 depicts two example annotations from the corpus. Argument components were annotated on the token level as non-overlapping annotation spans. We therefore represent the argument annotations using BIO encoding. Each token is labeled with one of the 11 categories (5 argument component types $\times$ B or I tag $+$ one O category for non-argumentative text).

4 Method

We cast the task of identifying argument components as a sequence tagging problem and employ SVM^{hmm} (Joachims et al., 2009).[6] For linguistic annotations and feature engineering, we rely on two UIMA-based frameworks – DKProCore (Eckart de Castilho and Gurevych, 2014) and DKProTC (Daxenberger et al., 2014).

Although the argument component annotations in the corpus are aligned to the token boundaries (token-level annotations), the minimal classification unit in our sequence tagging approach is set to the sentence level. First, this allows us to capture rich features that are available for entire sentences as opposed to the token level. Second, by modeling sequences on the token level we would lose the advantage of SVM^{hmm} to estimate dependencies between labels, as the label context is limited due to computational feasibility. On the token level, the label sequences are rather static (long sequences with the same label), as opposed to the sentence level. Before the classification step, we adjust all annotation boundaries (note that we use 11 BIO labels) so that they are aligned to the sentence boundaries and each sentence is then treated as a single classification unit with one label (for example, the first sentence from Figure 1 with token labels *Claim-B, Claim-I, Claim-I, ...* be-

[4] Available at `www.ukp.tu-darmstadt.de/data/argumentation-mining/`

[5] A publication containing a thorough analysis of the dataset is pending.

[6] Keerthi and Sundararajan (2007) conclude that performance of SVM^{hmm} is comparable to another widely used method, Conditional Random Fields (Lafferty et al., 2001)

comes *Claim-B*). After classification, the labels are mapped back to tokens (so that, for example, *Claim-B* sentence label is transformed to *Claim-B, Claim-I, ...* token labels). However, all evaluations are performed on the token level and the performance is always measured against the original token labels. Using this approximation, we lose only about 10% of F_1 performance.[7]

4.1 Baseline features

Lexical baseline (FS0) We encode the presence of unigrams, bigrams, and trigrams in the sentence as 'one-hot' (binary) features.

Structural and syntactic features (FS1) Since the presence of discourse markers has been shown to be helpful in argument component analysis (e.g, "therefore" and "since" for *premises* or "think" and "believe" for *claims*), we encode the first and last three words as binary features. Furthermore, we capture the relative position of the sentence in the paragraph and the document, the number of part of speech 1-3 grams, maximum dependency tree depth, constituency tree production rules, and number of sub-clauses (Stab and Gurevych, 2014b). We used Stanford POS Tagger (Toutanova et al., 2003), Berkeley parser (Petrov et al., 2006), and Malt parser (Nivre, 2009).

Sentiment and topic features (FS2) We assume that claims express sentiment, thus we compute five sentiment categories (from very negative to very positive) using Stanford sentiment analyzer (Socher et al., 2013) and use these values directly as features. Furthermore, in order to help detecting off-topic and non-argument sentences, we employ topic model features. In particular, we use features taken from a vector representation of the sentence obtained by using Gibbs sampling on LDA model (Blei et al., 2003; McCallum, 2002) with topics trained on unlabeled data provided as a part of the corpus.[8]

Semantic and discourse features (FS3) Features based on semantic frames has been introduced in relevant works on stance recognition (Hasan and Ng, 2013). Our features, based on PropBank semantic role labels and obtained from NLP Semantic Role Labeler (Choi, 2012), extract various semantic information (agent, predicate + agent, predicate + agent + patient + (optional) negation, argument type + argument value) and discourse markers. Discourse relations also play an important role in argumentation analysis (Cabrio et al., 2013). We thus employ binary features (such as the presence of the sentence in a chain, the transition type, the distance to previous/next sentences in the chain, or the number of inter-sentence coreference links) obtained from Stanford Coreference Chain Resolver (Lee et al., 2013). Furthermore, we include features resulting from a PTDB-style discourse parser (Li et al., 2012), such as the type of discourse relation (explicit, implicit), the presence of discourse connectives, and attributions.

4.2 Unsupervised features

We enrich the above-mentioned features by utilizing external large unlabeled resources – *debate portals*. They fulfill several criteria, namely (a) they are 'argumentative' (meant as opposed to, for example, prose or encyclopedic genres), (b) they are comprised of user-generated content and (c) and there is at least some overlap with topics from our experimental corpus. On the other hand, they contain noisy texts of questionable quality and they do not provide any specific argumentative structure (in fact, these debates are simple discussions to a topic, where each post is only labeled with a *pro* or *contra* stance). Nevertheless, we assume that the posts from (unlabeled) debate portals contain valuable information that will help us with classifying argument components in labeled data. In order to do so, we employ clustering based on latent semantics, which we now formalize as *argument space* features.

We assume that phrases (sentences or documents) can be projected into a latent vector space, using, typically, a sum or a weighted average of all the word embeddings vectors in the phrase; see for example (Le and Mikolov, 2014). Neighboring vectors in the latent vector space exhibit some interesting properties, such as semantic similarity (thoroughly studied within the distributional semantics area). If the latent vector space is clustered, each n-dimensional vector gets reduced to a single cluster number; such clusters have been used directly as features in many tasks, such as NER (Turian et al., 2010), POS tagging (Owoputi

[7] In only 1% of the sentences there are two or more argument components in it; we arbitrarily choose the largest one.

[8] The number of topics was empirically set to 30, therefore for each sentence the topic distribution results into 30 real-valued features.

et al., 2013), or sentiment analysis (Habernal and Brychcín, 2013).

We build upon the above-mentioned approach (described by Søgaard (2013) as 'clusters-as-features' semi-supervised paradigm) and extend it further. We take both sentences and posts from the unlabeled debate portals, project them into a latent space using word embeddings and cluster them. The motivation is that these clusters will contain similar phrases or (similar 'arguments'). Centroids of these clusters would then represent a 'prototypical argument' (note that the centroids exist only in the latent vector space and thus do not correspond to any existing sentence or post). Then we project each sentence (classification unit) in the labeled data to the latent vector space, compute its distance vector to all the cluster centroids, and encode this distance vector directly as real-valued features. By contrast to the above-mentioned works using a single cluster label as a feature, the distance vector to cluster centroids resembles a soft labeling where each sentence belongs to several clusters with a certain 'weight'. We also use the latent vector space representation of the sentence directly as a feature vector.

As unlabeled data, we use data from two largest debate portals.[9] As a pre-processing step we removed all posts with less than one 'point' earned.[10] The data were then indexed using the Lucene framework and the top 100 debates for each of the 6 domains were retrieved which resulted into 5,759 posts ($\approx$ 35k sentences) in the unlabeled data in total. Our approach is formalized in the following paragraph.

Argument space features (FS4) Let $\vec{e}(w)$ be the embedding vector of word w and $\mathrm{tfidf}(w)$ be the TD-IDF value of w. Sentence $\vec{s} = (w_1, \ldots, w_n)$ is then projected into the embedding space $\mathbb{E}$ as $\vec{s}_e = \sum_{i=1}^{n} \mathrm{tfidf}(w_i)\vec{e}(w_i)n^{-1}$ so $\dim(\vec{s}_e) = \dim(\mathbb{E})$. Analogically to $\vec{s}$, we project the entire post $\vec{a} = (w_1, \ldots, w_m)$ to the same embedding space $\mathbb{E}$ such that $\vec{a}_e = \sum_{i=1}^{m} \mathrm{tfidf}(w_i)\vec{e}(w_i)m^{-1}$.

Let K be the number of *sentence* clusters in $\mathbb{E}$ and $\vec{c}_k$ a centroid vector of cluster $k \in K$. Then $\vec{s}_c$ denotes the distance of sentence $\vec{s}_e$ to the sentence cluster centroids such that $\vec{s}_c = (\cos(\vec{s}_e, \vec{c}_1), \ldots, \cos(\vec{s}_e, \vec{c}_k))$ where $\dim(\vec{s}_c) = K$ and $\cos(\bullet, \bullet)$ denotes cosine similarity. Analogically, let L be the number of *post* clusters in $\mathbb{E}$ and $\vec{a}_l$ a centroid vector of cluster $l \in L$. Then $\vec{s}_a$ denotes the distance of sentence $\vec{s}_e$ to the post cluster centroids such that $\vec{s}_a = (\cos(\vec{s}_e, \vec{a}_1), \ldots, \cos(\vec{s}_e, \vec{a}_l))$. We construct the feature vector by concatenating $\vec{s}_e$, $\vec{s}_c$ and $\vec{s}_a$.

For word embeddings, we use pre-trained skip-gram word vectors[11] produced by Mikolov et al. (2013) ($\dim(\mathbb{E}) = 300$). To create clusters for the argument space features, we used CLUTO software package[12] with Repeated Bisection clustering method (Zhao and Karypis, 2002). We clustered the data using different hyper-parameters K and L (we experimented with $K = \{50, 100, 500, 1000\}$ and $L = \{50, 100, 500, 1000\}$).

5 Results

We investigate three evaluation scenarios. First, we report 10-fold cross validation over all 340 documents, where the data are randomly distributed across the folds regardless of the domain or register. In this scenario, the model can benefit from domain-dependent features for the testing data, such as lexical knowledge (FS0) or domain-relevant argument space features (FS4). Second, we evaluate the cross-domain performance; the model is always trained on five domains and tested on the sixth one. In this settings, we also remove all features that exploit distant data relevant to the test set. For instance, if the test domain is *mainstreaming*, we exclude all debates relevant to this domain before constructing the argument space features (FS4). This evaluates the model's cross-domain performance without any target domain data available. Finally, we test cross-register performance in two set-ups: we train the models using comments and forum posts and test on blogs and newswire articles, and then the other way round. We divided the data into these two parts based on similar properties of blogs/articles and comments/forums, such as the length, or the distribution of argumentative and non-argumentative text.

In the evaluation, we focus on F_1 scores achieved on *claims*, *premises*, *backing*, and non-argumentative text (the 'O' class). Although the

[9] `createdebate.com` and `convinceme.net`, licensed under Creative Commons (CC-BY and CC0, resp.)

[10] 'Points' is the sum of up-votes/down-votes by other users to the particular post. Zero-point posts were usually noisy and spam-like.

[11] `https://code.google.com/p/word2vec/`

[12] `http://www.cs.umn.edu/~karypis/cluto`

FS	B-B	B-I	C-B	C-I	O	P-B	P-I	Avg
Human	.664	.579	.739	.728	.833	.673	.736	.707
0	.154	.211	.118	.159	**.718**	.202	.272	.262
01⋆	.237	.254	.167	.129	.671	.280	.356	.299
4⋆	.194	.283	.225	.197	.715	.230	.292	.305
012⋆	.258	.282	.189	.172	.685	.276	.359	.317
1234†	.235	.315	.181	.145	.690	.290	.394	.321
0123⋆	.313	.333	.152	.140	.691	.287	.372	.327
01234†	**.265**	.332	.183	.167	.690	**.314**	**.405**	.337
34⋆	.232	**.344**	**.256**	**.235**	.704	.269	.372	.345
234†	.238	.339	.253	.227	.703	.291	.388	**.348**

Table 1: F_1 results for the 10-fold cross-validation scenario. Feature set combination (the **FS** column) naming is explained in Section 4.1. Class labels: **B-B/I** = *Backing-B/I*, **C-B/I** = *Claim-B/I*, **O** = *non-argumentative*, **P-B/I** = *Premise-B/I*. Star ($\star$) denotes that the row is significantly better than the previous row; dagger (†) means the row is not significantly better than the previous row, but is significantly better than the previous row minus one; $p < 0.001$ using exact Liddell's test (Liddell, 1983).

classifier is trained and tested on all 11 classes including *rebuttal* and *refutation*, we do not report performance of these two argument components—the results are very poor regardless of the parameters for two reasons. First, these classes are underrepresented in the data (*Rebuttal-B*, *Rebuttal-I*, *Refutation-B* and *Refutation-I* are present in only about 4% of sentences). Second, the inter-annotator agreement reached on these classes were reported to be very low (Habernal et al., 2014).

Cross validation results Table 1 shows results for the cross-validation scenario. The human baseline in the first row is an average score between three original annotators of the dataset. The baseline features (FS0) perform poorly, yet they beat the random assignment and majority vote (< 0.12 F_1). The argument space features (FS4) increase the performance in every combination. The best results for *claims* are achieved when only discourse, sentiment, and argument space features are involved (FS3 and FS4), whereas *premises* and *backing* benefit from the presence of lexical, syntactic, and semantic features (the richest feature set). The overall average best results are obtained from a feature combination with higher level of abstraction, in particular without low-level lexical features from FS0.

After the cross validation experiments, we also fixed the hyperparameters (using grid search) to $K = 1000, L = 100$ for the cluster sizes and $t = 1$ and $e = 0$ for the hyperparameters of SVM^{hmm} .

Cross-domain results For each domain, the cross-domain results are shown in Table 2. On average, the best results are about 0.10 F_1 points worse than in the cross-validation settings (Table 1). In all domains, the best average performance was achieved using only the argument space features (FS4); in four cases this system significantly outperforms all other systems ($p < 0.001$). Moreover, more high-level feature set combinations that also contain argument space features (such as FS2+FS3+F4 or FS3+FS4) yield usually better results for particular argument components in contrast to features based on lexical or syntactic information (FS0 and FS1). For identifying non-argumentative texts, there is no clear winner with respect to feature set abstraction (in three domains the best results are achieved using FS4 but in other three domains the baseline FS0 performs best).

Cross-register results The argument space features (FS4) performs best in average also in the cross-register evaluation (see Table 3). In recognizing *premises*, better results were achieved by a system trained on blogs and articles and tested on comments and forum posts. Recognizing *claims* exhibits similar behavior. On the other hand, recognizing non-argumentative text performs better in the opposite direction. On average, the cross-register results are much worse than cross-validation and slightly worse than cross-domain results.

5.1 Error analysis

First, we quantitatively investigate errors in the cross-validation scenario. The confusion matrix in Table 4 shows that about 50-60% of errors for each argument component were caused by misclassifying it as non-argumentative (the 'O' class). The system tends to prefer the 'O' predictions because of the high presence of non-argumentative sentences in the corpus (about 57%). *Backing* is often confused with *premises*; in particular, *Backing-B* with *Premise-B* in 14%, *Backing-I* with *Premise-I* in 17%. These two argument components have a similar function–to support the claim–so the differences in the discourse (which are sometimes very subtle) confuse the system. Note that despite the confusion between these classes, the *-I* and *-B* tags mostly remain the same (the system correctly predicts whether the argument component begins or not).[13]

We also analyzed the errors of the best-

[13]To provide the complete picture, we also show the previously unreported classes (*rebuttal* and *refutation*). *Rebuttal* is usually misclassified as non-argumentative or *premise*, *refutation* as either non-argumentative, *backing*, or *premise*.

FS	B-B	B-I	C-B	C-I	O	P-B	P-I	Avg
Target domain: Homeschooling								
01234	.039	.249	.000	.000	.145	.000	.000	.062
34	.000	.000	.000	.027	.005	.184	.386	.086
1234	.063	**.263**	.000	.000	.289	.000	.000	.088
234	.026	.030	.000	.000	.000	**.197**	**.387**	.091
01	.000	.000	.000	.000	.689	.000	.000	.098
012	.000	.000	.000	.000	.690	.000	.017	.101
0123	.000	.020	.000	.000	.689	.000	.018	.104
0	.000	.000	.063	.032	.683	.079	.098	.136
4◇	**.182**	.258	**.069**	**.069**	**.700**	.143	.224	**.235**
Target domain: Mainstreaming								
01234	.065	.262	.000	.000	.086	.000	.054	.067
34	.000	.000	.000	.000	.000	.184	**.352**	.077
234	.000	.287	.000	.000	.000	**.222**	.241	.107
0	.000	.000	.000	.000	**.689**	.054	.046	.113
1234	.126	.279	.000	.000	.060	.158	.221	.121
01	.000	.000	.000	.000	.666	.103	.079	.121
012	.000	.000	.000	.000	.663	.054	.141	.123
0123	.000	.000	.000	.000	.630	.261	.307	.171
4⋆	**.222**	**.448**	.000	.000	.674	.145	.247	**.248**
Target domain: Prayer in schools								
1234	.040	**.150**	.000	.000	.163	.000	.014	.052
0123	.000	.000	.000	.000	.080	.061	.292	.062
01234	.000	.115	.000	.000	.080	.149	.175	.074
234	**.058**	.042	.000	.000	.012	**.215**	**.303**	.090
34	.000	.000	**.098**	.105	.034	.203	.297	.105
0	.000	.111	.000	.000	.745	.000	.000	.122
01	.000	.115	.000	.000	**.810**	.000	.000	.132
012	.000	.000	.027	.045	.689	.120	.187	.153
4	.000	.146	.083	.048	.695	.168	.156	**.185**

FS	B-B	B-I	C-B	C-I	O	P-B	P-I	Avg
Target domain: Public vs. private schools								
01	.000	.000	.026	.004	.645	.000	.000	.096
012	.000	.000	.026	.005	.647	.000	.000	.097
01234	.000	.000	.000	.000	.591	.069	.093	.108
0	.026	.038	.000	.000	.638	.019	.054	.111
0123	.058	.064	.019	.023	.622	.019	.025	.119
234	.000	.089	.026	.042	.013	**.240**	.424	.119
1234	.000	.022	.000	.000	.496	.133	.239	.127
34	.051	.166	.037	.045	.011	.203	**.386**	.128
4◇	**.228**	**.251**	**.275**	**.270**	**.653**	.232	.220	**.304**
Target domain: Redshirting								
34	.073	.047	.000	.000	.144	.132	**.265**	.094
01234	.000	.000	**.076**	**.070**	.251	.024	.264	.098
234	.079	.162	.000	.000	.195	.101	.179	.102
0	.000	.000	.000	.000	**.740**	.000	.000	.106
01	.000	.000	.000	.000	.733	.000	.029	.109
012	.000	.000	.000	.000	.738	.049	.045	.119
1234	.102	.321	.000	.000	.118	.022	.277	.120
0123	**.304**	.356	.000	.000	.603	.082	.108	.208
4◇	.226	**.390**	.000	.000	.736	**.161**	.227	**.249**
Target domain: Single-sex education								
0123	**.137**	.178	.000	.000	.107	.000	.000	.060
012	.000	.033	.000	.000	.712	.000	.000	.106
01234	.138	.194	.024	.036	.247	.056	.148	.120
34	.065	.124	.000	.000	.073	**.208**	.379	.121
01	.000	.000	.000	.000	.708	.092	.209	.144
0	.000	.000	.000	.000	.728	.154	.130	.145
234	.061	.125	.000	.000	.395	.180	.269	.147
1234	.067	**.187**	**.078**	**.073**	.522	.067	.117	.159
4◇	.104	.185	.000	.000	**.689**	.204	**.397**	**.226**

Table 2: F_1 results for the cross-domain evaluation scenario ranked by performance. Feature set combination naming (the **FS** column) is explained in Section 4.1. Class labels: **B-B/I** = *Backing-B/I*, **C-B/I** = *Claim-B/I*, **O** = *non-argumentative*, **P-B/I** = *Premise-B/I*. Diamond (◇) in the last (winning) row signals a significant difference between this row and all other rows while star (⋆) denotes that the row is significantly better than the previous row; $p < 0.001$ using exact Liddell's test (Liddell, 1983).

FS	B-B	B-I	C-B	C-I	O	P-B	P-I	Avg
Train: blogs, articles; Test: comments, forums								
01234	.063	**.259**	.027	.051	.147	.000	.064	.087
012	.000	.000	.000	.000	.643	.000	.000	.092
0	.010	.237	.000	.000	.352	.014	.036	.093
01	.000	.000	.000	.000	.643	.010	.013	.095
0123	.021	.032	.000	.000	**.645**	.005	.002	.101
1234	**.097**	.215	.052	.068	.369	.000	.013	.116
234	.042	.068	.065	.068	.534	.093	.168	.148
34	.030	.061	.098	.099	.221	**.211**	**.385**	.158
4◇	.076	.206	**.167**	**.158**	.611	.151	.209	**.225**

FS	B-B	B-I	C-B	C-I	O	P-B	P-I	Avg
Train: comments, forums; Test: blogs, articles								
34	.052	.130	.036	.037	.057	.000	.000	.045
01234	.000	.008	.000	.000	.003	.080	.301	.056
234	.055	**.182**	.033	.036	.121	.025	.015	.067
1234	.071	.176	.014	.021	.050	.061	.290	.098
0	.000	.000	.000	.000	**.773**	.012	.019	.115
01	.000	.000	.051	**.058**	.720	.025	.043	.128
012	.000	.000	.039	.037	.746	.063	.046	.133
0123	.000	.000	.000	.000	.679	.099	.227	.144
4◇	**.142**	.162	**.061**	.032	.693	**.161**	**.353**	**.229**

Table 3: F_1 results for the cross-register evaluation scenario ranked by performance. Feature set combination naming (the **FS** column) is explained in Section 4.1. Class labels: **B-B/I** = *Backing-B/I*, **C-B/I** = *Claim-B/I*, **O** = *non-argumentative*, **P-B/I** = *Premise-B/I*. Diamond (◇) in the last (winning) row signals a significant difference between this row and all other rows; $p < 0.001$ using exact Liddell's test (Liddell, 1983).

performing cross-domain system in detail.[14] We randomly sampled 40 documents and manually compared the predicted arguments with the gold data. We found that 11 predicted documents were simply wrong or no argument components were predicted at all (e.g., document #1640, #1658, #1021, #5258). Most of these errors occur in blogs, which seem to convey rather complex argumentation structure (#1666, #1197, #4586, #5258). In 8 documents, we identified that only some premises were (correctly) spotted by the system. This happened mostly in long comments (#452) and blogs (#400, #697, #4583). In 7 investigated documents, we identified errors caused by slightly different boundaries of recognized argument components (#4517, #2447, #2252, #4840) or when multiple segments were merged/split (#1604, #2180, #2310).

By analyzing the predicted output, we also found that in 12 documents the recognized argument components seemed to be valid to some extent, although this was our subjective judge. For instance, in #4285 (see Figure 2), the first *premise* was misclassified as a *claim*. The gold-data argument was annotated as an *enthymeme* (with implicit *claim* that advocates private schools), while in the prediction, the same proposition was identified as the an explicit *claim* supporting private

[14] Available also as PDF at `https://github.com/habernal/emnlp2015`; we use #ID to point to the particular documents.

↓ gold \ pred. →	Bac-B	Bac-I	Cla-B	Cla-I	O	Pre-B	Pre-I	Reb-B	Reb-I	Ref-B	Ref-I
Backing-B	54	12	12	1	106	31	8	0	0	0	0
Backing-I	12	3,238	1	353	5,089	17	1,777	1	18	0	45
Claim-B	7	3	41	5	107	19	9	1	1	0	2
Claim-I	0	160	0	713	2,095	1	456	0	25	0	25
O	97	3,170	53	1,135	36,061	156	5,459	4	178	1	38
Premise-B	35	17	17	2	290	142	28	6	0	0	1
Premise-I	18	1,680	2	544	10,779	51	7,015	2	234	2	41
Rebuttal-B	3	4	3	2	40	9	7	0	0	0	0
Rebuttal-I	1	199	0	47	1,063	10	859	0	0	0	0
Refutation-B	2	2	0	1	16	1	3	0	1	0	0
Refutation-I	0	86	0	7	592	2	148	0	6	0	0

Table 4: Confusion matrix for the best performing configuration in the cross-validation scenario.

schools with one *premise* why the education was not satisfying, which might be also another valid interpretation. The second example #2180 in Figure 2 shows that the boundaries of the predicted *premises* are mixed up (two recognized instead of three), but the longer *backing* is also meaningful. These examples demonstrate that argument analysis is in some cases ambiguous and allows for different valid interpretations.

6 Conclusion

In this article, we proposed a semi-supervised model for argumentation mining of user-generated Web content. We developed new unsupervised features for argument component identification that exploit clustering of unlabeled argumentative data from debate portals based on word embeddings representation. With the help of these features we significantly improved performance of the argumentation mining system and outperformed several baselines. While the improvement was decent in cross-validation scenario, we gained almost 100% improvement in cross-domain and cross-register settings.

We evaluated the methods on a publicly available corpus annotated with argumentation that origins from user-generated Web data. By a detailed analysis of the errors, we pointed out the strengths (such as domain adaptability) and weaknesses (such as unsatisfying results for *rebuttal* and *refutation* components), as well as the challenges for the argumentation mining task (such as boundary identification issues or ambiguous arguments). If we put our results into the context of existing works, the most relevant one by (Goudas et al., 2014) achieved 0.42 F_1 score on identifying only premises. We get comparable results in the cross-validation settings (F_1 0.31-0.40) yet with more complex argumentation model (five different components).

Although argumentation mining in user-generated Web discourse has a long way to go (our methods currently achieve only about 50% of human performance), we see a huge potential for various future tasks, such as information seeking for better-informed personal decision making or support for argument quality assessment. To foster the research within the community, we provide all source codes and data required for the experiments under free licenses.

Acknowledgements

This work has been supported by the Volkswagen Foundation as part of the Lichtenberg-Professorship Program under grant Nº I/82806 and by the German Institute for Educational Research (DIPF). Access to the CERIT-SC computing and storage facilities provided under the programme Center CERIT Scientific Cloud, part of the Operational Program Research and Development for Innovations, reg. no. CZ. 1.05/3.2.00/08.0144, is greatly appreciated. Lastly, we would like to thank the anonymous reviewers for their valuable feedback.

References

Or Biran and Owen Rambow. 2011. Identifying justifications in written dialogs by classifying text as argumentative. *International Journal of Semantic Computing*, 5(4):363–381.

J. Anthony Blair. 2004. Argument and its uses. *Informal Logic*, 24:137151.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, March.

Filip Boltužić and Jan Šnajder. 2015. Identifying Prominent Arguments in Online Debates Using Semantic Textual Similarity. In *Proceedings of the 2nd*

Gold

[*premise:* I sent my kid to private school so that she could get a better education.] [*backing:* She was at a public school that was 90% hispanic.] [*premise:* The problem was not their race but the fact that they were way behind in reading langauge and math. This situation was holding my kid and preventing her from excelling in her studies.] Do you think I should of just left my kid in this class? Give me a break!

Predicted

[*claim:* I sent my kid to private school so that she could get a better education.] She was at a public school that was 90% hispanic. [*premise:* The problem was not their race but the fact that they were way behind in reading langauge and math.] This situation was holding my kid and preventing her from excelling in her studies. Do you think I should of just left my kid in this class? Give me a break!

(a) Doc #4285 (article comment, public vs. private schools)

Gold

[*claim:* Personally i'd go co-ed.]
[*backing:* As someone who went to a same sex school for 8 years, I found it lacked the diversity you get in a co-ed environment.] [*premise:* I found the attitude and behaviour of students in the co ed school to be better, and i attribute that to the influence of the opposite sex.] [*premise:* There's no doubt boys behave a little different when girls are watching, and i also found boys were quite good at limiting the bitchyness girls are renowned for. So both kept one another in line, and made for a more positive and dynamic environment.]
[*premise:* I also think there's a few extra life lessons and skills children can learn at co ed schools. Dating, relationships, interacting with the opposite sex, i think children at co ed schools tend to have a far better grasp of these skills then students who've only attended same sex schools.]

Predicted

[*backing:* Personally i'd go co-ed.
As someone who went to a same sex school for 8 years, I found it lacked the diversity you get in a co-ed environment. I found the attitude and behaviour of students in the co ed school to be better, and i attribute that to the influence of the opposite sex.] [*premise:* There's no doubt boys behave a little different when girls are watching, and i also found boys were quite good at limiting the bitchyness girls are renowned for.] [*premise:* So both kept one another in line, and made for a more positive and dynamic environment.
I also think there's a few extra life lessons and skills children can learn at co ed schools. Dating, relationships, interacting with the opposite sex, i think children at co ed schools tend to have a far better grasp of these skills then students who've only attended same sex schools.]

(b) Doc #2180 (forum post, single-sex education)

Figure 2: Examples of gold data annotations (on the left-hand side) and system predictions in the best-performing cross-domain evaluation scenario (on the right-hand side).

Workshop on Argumentation Mining, pages 110–115, Denver, Colorado. Association for Computational Linguistics.

Elena Cabrio and Serena Villata. 2012. Combining textual entailment and argumentation theory for supporting online debates interactions. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers - Volume 2*, ACL '12, pages 208–212, Jeju Island, Korea. Association for Computational Linguistics.

Elena Cabrio, Sara Tonelli, and Serena Villata. 2013. From discourse analysis to argumentation schemes and back: Relations and differences. In João Leite, Tran Cao Son, Paolo Torroni, Leon Torre, and Stefan Woltran, editors, *Proceedings of 14th International Workshop on Computational Logic in Multi-Agent Systems*, volume 8143 of *Lecture Notes in Computer Science*, pages 1–17. Springer Berlin Heidelberg.

Jinho D. Choi. 2012. *Optimization of Natural Language Processing Components for Robustness and Scalability*. Ph.D. Thesis, University of Colorado Boulder, Computer Science and Cognitive Science.

Johannes Daxenberger, Oliver Ferschke, Iryna Gurevych, and Torsten Zesch. 2014. DKPro TC: a Java-based framework for supervised learning experiments on textual data. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 61–66, Baltimore, Maryland, June. Association for Computational Linguistics.

Richard Eckart de Castilho and Iryna Gurevych. 2014. A broad-coverage collection of portable NLP components for building shareable analysis pipelines. In Nancy Ide and Jens Grivolla, editors, *Proceedings of the Workshop on Open Infrastructures and Analysis Frameworks for HLT (OIAF4HLT) at COLING 2014*, pages 1–11, Dublin, Ireland. Association for Computational Linguistics and Dublin City University.

Vanessa Wei Feng and Graeme Hirst. 2011. Classifying arguments by scheme. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 987–996, Portland, Oregon. Association for Computational Linguistics.

Swapna Gottipati, Minghui Qiu, Yanchuan Sim, Jing Jiang, and Noah A. Smith. 2013. Learning topics and positions from Debatepedia. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1858–1868, Seattle, Washington, USA, October. Association for Computational Linguistics.

Theodosis Goudas, Christos Louizos, Georgios Petasis, and Vangelis Karkaletsis. 2014. Argument extraction from news, blogs, and social media. In Aristidis Likas, Konstantinos Blekas, and Dimitris Kalles, editors, *Artificial Intelligence: Methods and Applications*, pages 287–299. Springer International Publishing.

Nancy L Green. 2014. Argumentation for scientific claims in a biomedical research article. In Elena Cabrio, Serena Villata, and Adam Wyner, editors, *Proceedings of the Workshop on Frontiers and Connections between Argumentation Theory and Natural Language Processing*, pages 5–10, Bertinoro, Italy, July. CEUR-WS.

Ivan Habernal and Tomáš Brychcín. 2013. Semantic spaces for sentiment analysis. In *Text, Speech and Dialogue*, volume 8082 of *Lecture Notes in Computer Science*, pages 482–489, Berlin Heidelberg. Springer.

Ivan Habernal, Judith Eckle-Kohler, and Iryna Gurevych. 2014. Argumentation mining on the web from information seeking perspective. In Elena Cabrio, Serena Villata, and Adam Wyner, editors, *Proceedings of the Workshop on Frontiers and Connections between Argumentation Theory and Natural Language Processing*, pages 26–39, Bertinoro, Italy, July. CEUR-WS.

Kazi Saidul Hasan and Vincent Ng. 2013. Stance classification of ideological debates: Data, models, features, and constraints. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 1348–1356, Nagoya, Japan, October. Asian Federation of Natural Language Processing.

Thorsten Joachims, Thomas Finley, and Chun-Nam John Yu. 2009. Cutting-plane training of structural SVMs. *Machine Learning*, 77(1):27–59.

Sathiya Keerthi and Sellamanickam Sundararajan. 2007. CRF versus SVM-struct for sequence labeling. Technical report, Yahoo! Research.

Victor Alvin Ketcham. 1917. *The theory and practice of argumentation and debate*. Macmillan, New York.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA. Morgan Kaufmann Publishers Inc.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In Tony Jebara and Eric P. Xing, editors, *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, volume 32, pages 1188–1196, Beijing, China. JMLR Workshop and Conference Proceedings.

Heeyoung Lee, Angel Chang, Yves Peirsman, Nathanael Chambers, Mihai Surdeanu, and Dan Jurafsky. 2013. Deterministic Coreference Resolution Based on Entity-centric, Precision-ranked Rules. *Computational Linguistics*, 39(4):885–916.

Lianghao Li, Xiaoming Jin, Sinno Jialin Pan, and Jian-Tao Sun. 2012. Multi-domain active learning for text classification. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '12, pages 1086–1094, Beijing, China. ACM.

Douglas Liddell. 1983. Simplified exact analysis of case-referent studies: matched pairs; dichotomous exposure. *Journal of Epidemiology & Community Health*, 37(1):82–84.

Andrew Kachites McCallum. 2002. MALLET: A Machine Learning for Language Toolkit. http://mallet.cs.umass.edu.

Hugo Mercier and Dan Sperber. 2011. Why do humans reason? Arguments for an argumentative theory. *The Behavioral and Brain Sciences*, 34(2):57–74; discussion 74–111.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

Raquel Mochales and Marie-Francine Moens. 2011. Argumentation mining. *Artificial Intelligence and Law*, 19(1):1–22, April.

Ana Laura Nettel and Georges Roque. 2011. Persuasive argumentation versus manipulation. *Argumentation*, 26(1):55–69.

Joakim Nivre. 2009. Non-projective dependency parsing in expected linear time. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1*, ACL '09, pages 351–359, Suntec, Singapore. Association for Computational Linguistics.

Olutobi Owoputi, Brendan O'Connor, Chris Dyer, Kevin Gimpel, Nathan Schneider, and Noah A Smith. 2013. Improved Part-of-Speech Tagging for Online Conversational Text with Word Clusters. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 380–390, Atlanta, Georgia. Association for Computational Linguistics.

Joonsuk Park and Claire Cardie. 2014. Identifying appropriate support for propositions in online user comments. In *Proceedings of the First Workshop on Argumentation Mining*, pages 29–38, Baltimore, Maryland, June. Association for Computational Linguistics.

Andreas Peldszus and Manfred Stede. 2013. Ranking the annotators: An agreement study on argumentation structure. In *Proceedings of the 7th Linguistic*

Annotation Workshop and Interoperability with Discourse, pages 196–204, Sofia, Bulgaria, August. Association for Computational Linguistics.

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

Niall Rooney, Hui Wang, and Fiona Browne. 2012. Applying kernel methods to argumentation mining. In *Proceedings of the Twenty-Fifth International Florida Artificial Intelligence Research Society Conference*, pages 272–275. Association for the Advancement of Artificial Intelligence.

Sara Rosenthal and Kathleen McKeown. 2012. Detecting opinionated claims in online discussions. In *2012 IEEE Sixth International Conference on Semantic Computing*, pages 30–37, Palermo, Italy. IEEE.

Burr Settles. 2012. *Active Learning*. Morgan & Claypool Publishers.

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

Anders Søgaard. 2013. *Semi-Supervised Learning and Domain Adaptation in Natural Language Processing*. Morgan & Claypool Publishers.

Christian Stab and Iryna Gurevych. 2014a. Annotating argument components and relations in persuasive essays. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 1501–1510, Dublin, Ireland, August. Dublin City University and Association for Computational Linguistics.

Christian Stab and Iryna Gurevych. 2014b. Identifying argumentative discourse structures in persuasive essays. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 46–56, Doha, Qatar, October. Association for Computational Linguistics.

Simone Teufel and Marc Moens. 2002. Summarizing scientific articles: Experiments with relevance and rhetorical status. *Computational Linguistics*, 28:409–445.

Stephen E. Toulmin. 1958. *The Uses of Argument*. Cambridge University Press.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, pages 173–180, Edmonton, Canada. Association for Computational Linguistics.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*, number July, pages 384–394, Uppsala, Sweden. Association for Computational Linguistics.

Frans H. van Eemeren, R. Grootendorst, and A. F. Snoeck Henkemans. 2002. *Argumentation: Analysis, evaluation, presentation*. Lawrence Erlbaum, Mahwah, NJ, USA.

Frans H. van Eemeren, Bart Garssen, Erik C. W. Krabbe, A. Francisca Snoeck Henkemans, Bart Verheij, and Jean H. M. Wagemans. 2014. *Handbook of Argumentation Theory*. Springer, Berlin/Heidelberg.

Henning Wachsmuth, Martin Trenkmann, Benno Stein, Gregor Engels, and Tsvetomira Palakarska. 2014. A review corpus for argumentation analysis. In Alexander Gelbukh, editor, *15th International Conference on Intelligent Text Processing and Computational Linguistics (CICLing 14)*, pages 115–127, Kathmandu, Nepal. Springer.

Douglas Walton, Christopher Reed, and Fabrizio Macagno. 2008. *Argumentation Schemes*. Cambridge University Press.

Y. Zhao and G. Karypis. 2002. Criterion functions for document clustering: Experiments and analysis. Technical report, Department of Computer Science, University of Minnesota, Minneapolis.

Confounds and Consequences in Geotagged Twitter Data

Umashanthi Pavalanathan and **Jacob Eisenstein**
School of Interactive Computing
Georgia Institute of Technology
Atlanta, GA 30308
{umashanthi + jacobe}@gatech.edu

Abstract

Twitter is often used in quantitative studies that identify geographically-preferred topics, writing styles, and entities. These studies rely on either GPS coordinates attached to individual messages, or on the user-supplied location field in each profile. In this paper, we compare these data acquisition techniques and quantify the biases that they introduce; we also measure their effects on linguistic analysis and text-based geolocation. GPS-tagging and self-reported locations yield measurably different corpora, and these linguistic differences are partially attributable to differences in dataset composition by age and gender. Using a latent variable model to induce age and gender, we show how these demographic variables interact with geography to affect language use. We also show that the accuracy of text-based geolocation varies with population demographics, giving the best results for men above the age of 40.

1 Introduction

Social media data such as Twitter is frequently used to identify the unique characteristics of geographical regions, including topics of interest (Hong et al., 2012), linguistic styles and dialects (Eisenstein et al., 2010; Gonçalves and Sánchez, 2014), political opinions (Caldarelli et al., 2014), and public health (Broniatowski et al., 2013). Social media permits the aggregation of datasets that are orders of magnitude larger than could be assembled via traditional survey techniques, enabling analysis that is simultaneously fine-grained and global in scale. Yet social media is not a representative sample of any "real world" population, aside from social media itself. Using social media as a sample therefore risks introducing both geographic and demographic biases (Mislove et al., 2011; Hecht and Stephens, 2014; Longley et al., 2015; Malik et al., 2015).

This paper examines the effects of these biases on the geo-linguistic inferences that can be drawn from Twitter. We focus on the ten largest metropolitan areas in the United States, and consider three sampling techniques: drawing an equal number of GPS-tagged tweets from each area; drawing a *county-balanced* sample of GPS-tagged messages to correct Twitter's urban skew (Hecht and Stephens, 2014); and drawing a sample of *location-annotated* messages, using the location field in the user profile. Leveraging self-reported first names and census statistics, we show that the age and gender composition of these datasets differ significantly.

Next, we apply standard methods from the literature to identify geo-linguistic differences, and test how the outcomes of these methods depend on the sampling technique and on the underlying demographics. We also test the accuracy of text-based geolocation (Cheng et al., 2010; Eisenstein et al., 2010) in each dataset, to determine whether the accuracies reported in recent work will generalize to more balanced samples.

The paper reports several new findings about geotagged Twitter data:

- In comparison with tweets with self-reported locations, GPS-tagged tweets are written more often by young people and by women.
- There are corresponding linguistic differences between these datasets, with GPS-tagged tweets including more geographically-specific non-standard words.
- Young people use significantly more geographically-specific non-standard words. Men tend to mention more geographically-specific entities than women, but these

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 2138–2148,
Lisbon, Portugal, 17-21 September 2015.

differences are significant only for individuals at the age of 30 or older.

- Users who GPS-tag their tweets tend to write more, making them easier to geolocate. Evaluating text-based geolocation on GPS-tagged tweets probably overestimates its accuracy.
- Text-based geolocation is significantly more accurate for men and for older people.

These findings should inform future attempts to generalize from geotagged Twitter data, and may suggest investigations into the demographic properties of other social media sites.

We first describe the basic data collection principles that hold throughout the paper (§ 2). The following three sections tackle demographic biases (§ 3), their linguistic consequences (§ 4), and the impact on text-based geolocation (§ 5); each of these sections begins with a discussion of methods, and then presents results. We then summarize related work and conclude.

2 Dataset

This study is performed on a dataset of tweets gathered from Twitter's streaming API from February 2014 to January 2015. During an initial filtering step we removed retweets, repetitions of previously posted messages which contain the "retweeted_status" metadata or "RT" token which is widely used among Twitter users to indicate a retweet. To eliminate spam and automated accounts (Yardi et al., 2009), we removed tweets containing URLs, user accounts with more than 1000 followers or followees, accounts which have tweeted more than 5000 messages at the time of data collection, and the top 10% of accounts based on number of messages in our dataset. We also removed users who have written more than 10% of their tweets in any language other than English, using Twitter's `lang` metadata field. Exploration of code-switching (Solorio and Liu, 2008) and the role of second-language English speakers (Eleta and Golbeck, 2014) is left for future work.

We consider the ten largest Metropolitan Statistical Areas (MSAs) in the United States, listed in Table 1. MSAs are defined by the U.S. Census Bureau as geographical regions of high population with density organized around a single urban core; they are not legal administrative divisions. MSAs include outlying areas that may be substantially less urban than the core itself. For example, the Atlanta MSA is centered on Fulton County (1750 people per square mile), but extends to Haralson County (100 people per square mile), on the border of Alabama. A per-county analysis of this data therefore enables us to assess the degree to which Twitter's skew towards urban areas biases geo-linguistic analysis.

3 Representativeness of geotagged Twitter data

We first assess potential biases in sampling techniques for obtaining geotagged Twitter data. In particular, we compare two possible techniques for obtaining data: the location field in the user profile (Poblete et al., 2011; Dredze et al., 2013), and the GPS coordinates attached to each message (Cheng et al., 2010; Eisenstein et al., 2010).

3.1 Methods

To build a dataset of GPS-tagged messages, we extracted the GPS latitude and longitude coordinates reported in the tweet, and used GIS-TOOLS[1] reverse geocoding to identify the corresponding counties. This set of geotagged messages will be denoted $\mathcal{D}_G$. Only 1.24% of messages contain geo-coordinates, and it is possible that the individuals willing to share their GPS comprise a skewed population. We therefore also considered the user-reported location field in the Twitter profile, focusing on the two most widely-used patterns: (1) city name, (2) city name and two letter state name (e.g. *Chicago* and *Chicago, IL*). Messages that matched any of the ten largest MSAs were grouped into a second set, $\mathcal{D}_L$.

While the inconsistencies of writing style in the Twitter location field are well-known (Hecht et al., 2011), analysis of the intersection between $\mathcal{D}_G$ and $\mathcal{D}_L$ found that the two data sources agreed the overwhelming majority of the time, suggesting that most self-provided locations are accurate. Of course, there may be many false negatives — profiles that we fail to geolocate due to the use of non-standard toponyms like *Pixburgh* and *ATL*. If so, this would introduce a bias in the population sample in $\mathcal{D}_L$. Such a bias might have linguistic consequences, with datasets based on the location field containing less non-standard language overall.

[1] `https://github.com/DrSkippy/Data-Science-45min-Intros/blob/master/gis-tools-101/gis_tools.ipynb`

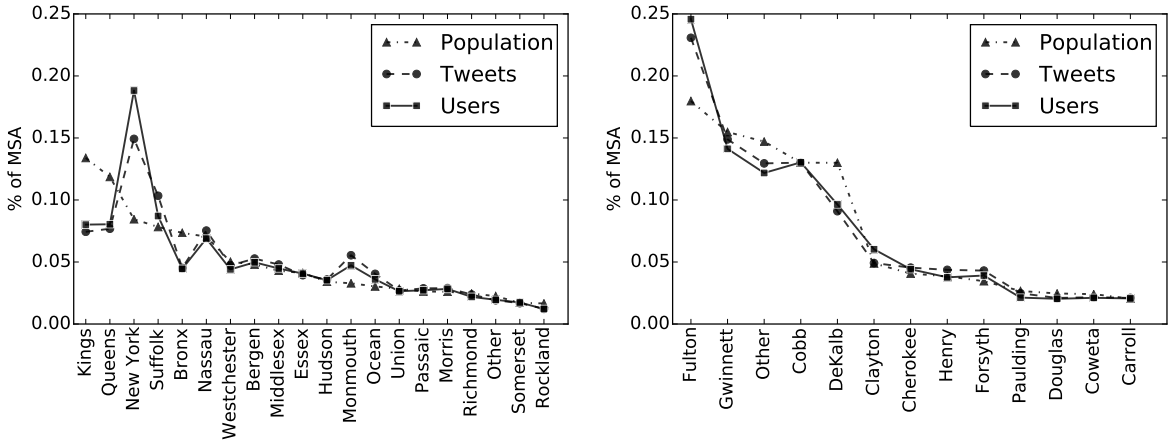


Figure 1: Proportion of census population, Twitter messages, and Twitter user accounts, by county. New York is shown on the left, Atlanta on the right.

3.1.1 Subsampling

The initial samples $\mathcal{D}_G$ and $\mathcal{D}_L$ were then resampled to create the following balanced datasets:

GPS-MSA-BALANCED From $\mathcal{D}_G$, we randomly sampled 25,000 tweets per MSA as the message-balanced sample, and all the tweets from 2,500 users per MSA as the user-balanced sample. Balancing across MSAs ensures that the largest MSAs do not dominate the linguistic analysis.

GPS-COUNTY-BALANCED We resampled $\mathcal{D}_G$ based on county-level population (obtained from the U.S. Census Bureau), and again obtained message-balanced and user-balanced samples. These samples are more geographically representative of the overall population distribution across each MSA.

LOC-MSA-BALANCED From $\mathcal{D}_L$, we randomly sampled 25,000 tweets per MSA as the message-balanced sample, and all the tweets from 2,500 users per MSA as the user-balanced sample. It is not possible to obtain county-level geolocations in $\mathcal{D}_L$, as exact geographical coordinates are unavailable.

3.1.2 Age and gender identification

To estimate the distribution of ages and genders in each sample, we queried statistics from the Social Security Administration, which records the number of individuals born each year with each given name. Using this information, we obtained the probability distribution of age values for each given name. We then matched the names against the first token in the name field of each user's profile, enabling us to induce approximate distributions over ages and genders. Unlike Facebook and Google+, Twitter does not have a "real name" policy, so users are free to give names that are fake, humorous, etc. We eliminate user accounts whose names are not sufficiently common in the social security database (i.e. first names which are at least 100 times more frequent in Twitter than in the social security database), thereby omitting 33% of user accounts, and 34% of tweets. While some individuals will choose names not typically associated with their gender, we assume that this will happen with roughly equal probability in both directions. So, with these caveats in mind, we induce the age distribution for the GPS-MSA-BALANCED sample and the LOC-MSA-BALANCED sample as,

$$p(a \mid \text{name} = n) = \frac{count(\text{name} = n, \text{age} = a)}{\sum_{a'} count(\text{name} = n, \text{age} = a')} \quad (1)$$

$$p_{\mathcal{D}}(a) \propto \sum_{i \in \mathcal{D}} p(a \mid \text{name} = n_i). \quad (2)$$

We induce distributions over author gender in much the same way (Mislove et al., 2011). This method does not incorporate prior information about the ages of Twitter users, and thus assigns too much probability to the extremely young and old, who are unlikely to use the service. While it would be easy to design such a prior — for example, assigning zero prior probability to users under the age of five or above the age of 95 — we see no principled basis for determining these cutoffs. We therefore focus on the *differences* between the estimated $p_{\mathcal{D}}(a)$ for each sample $\mathcal{D}$.

MSA	Num. Counties	L1 Dist. Population vs. Users	L1 Dist. Population vs. Tweets
New York	23	0.2891	0.2825
Los Angeles	2	0.0203	0.0223
Chicago	14	0.0482	0.0535
Dallas	12	0.1437	0.1176
Houston	10	0.0394	0.0472
Philadelphia	11	0.1426	0.1202
Washington DC	22	0.2089	0.2750
Miami	3	0.0428	0.0362
Atlanta	28	0.1448	0.1730
Boston	7	0.1878	0.2303

Table 1: L1 distance between county-level population and Twitter users and messages

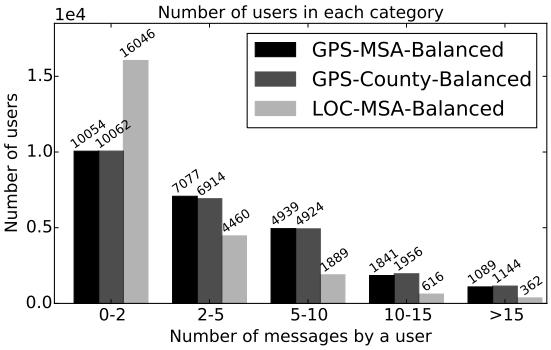


Figure 2: User counts by number of Twitter messages

3.2 Results

Geographical biases in the GPS Sample We first assess the differences between the true population distributions over counties, and the per-tweet and per-user distributions. Because counties vary widely in their degree of urbanization and other demographic characteristics, this measure is a proxy for the representativeness of GPS-based Twitter samples (county information is not available for the LOC-MSA-BALANCED sample). Population distributions for New York and Atlanta are shown in Figure 1. In Atlanta, Fulton County is the most populous and most urban, and is over-represented in both geotagged tweets and user accounts; most of the remaining counties are correspondingly underrepresented. This coheres with the urban bias noted earlier by Hecht and Stephens (2014). In New York, Kings County (Brooklyn) is the most populous, but is underrepresented in both the number of geotagged tweets and user accounts, at the expense of New York County (Manhattan). Manhattan is the commercial and entertainment center of the New York MSA, so residents of outlying counties may be tweeting from their jobs or social activities.

To quantify the representativeness of each sample, we use the L1 distance $||\mathbf{x} - \mathbf{y}||_1 = \sum_c |p_c - t_c|$, where p_c is the proportion of the MSA population residing in county c and t_c is the proportion of tweets (Table 1). County boundaries are determined by states, and their density varies: for example, the Los Angeles MSA covers only two counties, while the smaller Atlanta MSA is spread over 28 counties. The table shows that while New York is the most extreme example, most MSAs feature an asymmetry between county population and Twitter adoption.

Usage Next, we turn to differences between the GPS-based and profile-based techniques for obtaining ground truth data. As shown in Figure 2, the LOC-MSA-BALANCED sample contains more low-volume users than either the GPS-MSA-BALANCED or GPS-COUNTY-BALANCED samples. We can therefore conclude that the county-level geographical bias in the GPS-based data does not impact usage rate, but that the difference between GPS-based and profile-based sampling does; the linguistic consequences of this difference will be explored in the following sections.

Demographics Table 2 shows the expected age and gender for each dataset, with bootstrap confidence intervals. Users in the LOC-MSA-BALANCED dataset are on average two years older than in the GPS-MSA-BALANCED and GPS-COUNTY-BALANCED datasets, which are statistically indistinguishable. Focusing on the difference between GPS-MSA-BALANCED and LOC-MSA-BALANCED, we plot the difference in age probabilities in Figure 3, showing that GPS-MSA-BALANCED includes many more teens and people in their early twenties, while LOC-MSA-BALANCED includes more people at middle age and older. Young people are especially likely to use social media on cellphones (Lenhart, 2015), where location tagging would be more relevant than when Twitter is accessed via a personal computer. Social media users in the age brackets 18-29 and 30-49 are also more likely to tag their locations in social media posts than social media users in the age brackets 50-64 and 65+ (Zickuhr, 2013), with women and men tagging at roughly equal rates. Table 2 shows that the GPS-MSA-BALANCED and GPS-COUNTY-BALANCED samples contain significantly more women than LOC-

Sample	Expected Age	95% CI	% Female	95% CI
GPS-MSA-BALANCED	36.17	[36.07 – 36.27]	51.5	[51.3 – 51.8]
GPS-COUNTY-BALANCED	36.25	[36.16 – 36.30]	51.3	[51.1 – 51.6]
LOC-MSA-BALANCED	38.35	[38.25 – 38.44]	49.3	[49.1 – 49.6]

Table 2: Demographic statistics for each dataset

Figure 3: Difference in age probability distributions between GPS-MSA-BALANCED and LOC-MSA-BALANCED.

MSA-BALANCED, though all three samples are close to 50%.

4 Impact on linguistic generalizations

Many papers use Twitter data to draw conclusions about the relationship between language and geography. What role do the demographic differences identified in the previous section have on the linguistic conclusions that emerge? We measure the differences between the linguistic corpora obtained by each data acquisition approach. Since the GPS-MSA-BALANCED and GPS-COUNTY-BALANCED methods have nearly identical patterns of usage and demographics, we focus on the difference between GPS-MSA-BALANCED and LOC-MSA-BALANCED. These datasets differ in age and gender, so we also directly measure the impact of these demographic factors on the use of geographically-specific linguistic variables.

4.1 Methods

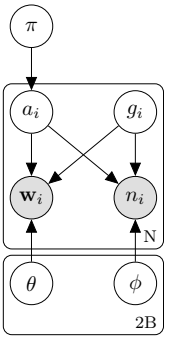
Discovering geographical linguistic variables We focus on lexical variation, which is relatively easy to identify in text corpora. Monroe et al. (2008) survey a range of alternative statistics for finding lexical variables, demonstrating that a regularized log-odds ratio strikes a good balance between distinctiveness and robustness. A similar approach is implemented in SAGE (Eisenstein et al., 2011a)[2], which we use here. For each sample — GPS-MSA-BALANCED and LOC-MSA-BALANCED — we apply SAGE to identify the twenty-five most salient lexical items for each metropolitan area.

Keyword annotation Previous research has identified two main types of geographical lexical variables. The first are non-standard words and spellings, such as *hella* and *yinz*, which have been found to be very frequent in social media (Eisenstein, 2015). Other researchers have focused on the "long tail" of entity names (Roller et al., 2012). A key question is the relative importance of these two variable types, since this would decide whether geo-linguistic differences are primarily topic-based or stylistic. It is therefore important to know whether the frequency of these two variable types depends on properties of the sample. To test this, we take the lexical items identified by SAGE (25 per MSA, for both the GPS-MSA-BALANCED and LOC-MSA-BALANCED samples), and annotate them as NONSTANDARD-WORD, ENTITY-NAME, or OTHER. Annotation for ambiguous cases is based on the majority sense in randomly-selected examples. Overall, we identify 24 NONSTANDARD-WORDS and 185 ENTITY-NAMES.

Inferring author demographics As described in § 3.1.2, we can obtain an approximate distribution over author age and gender by linking self-reported first names with aggregate statistics from the United States Census. To sharpen these estimates, we now consider the text as well, building a simple latent variable model in which both the name and the word counts are drawn from distributions associated with the latent age and gender (Chang et al., 2010). The model is shown in Figure 4, and involves the following generative process:

For each user $i \in \{1...N\}$,
(a) draw the age, $a_i \sim \text{Categorical}(\pi)$
(b) draw the gender, $g_i \sim \text{Categorical}(0.5)$

[2] https://github.com/jacobeisenstein/jos-gender-2014



a_i	Age (bin) for author i
g_i	Gender of author i
$\mathbf{w}_i$	Word counts for author i
n_i	First name of author i
π	Prior distribution over age bins
$\theta_{a,g}$	Word distribution for age a and gender g
$\phi_{a,g}$	First name distribution for age a and gender g

Figure 4: Plate diagram for latent variable model of age and gender

(c) draw the author's given name, $n_i \sim \text{Categorical}(\phi_{a_i,g_i})$

(d) draw the word counts, $w_i \sim \text{Multinomial}(\theta_{a_i,g_i})$,

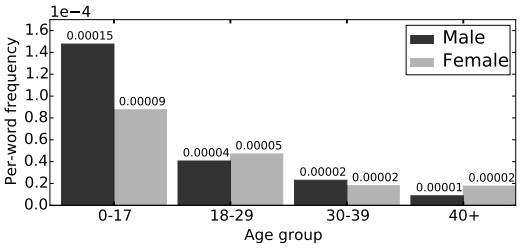
where we elide the second parameter of the multinomial distribution, the total word count. We use expectation-maximization to perform inference in this model, binning the latent age variable into four groups: 0-17, 18-29, 30-39, above 40.[3] Because the distribution of names given demographics is available from the Social Security data, we clamp the value of ϕ throughout the EM procedure. Other work in the domain of demographic prediction often involves more complex methods (Nguyen et al., 2014; Volkova and Durme, 2015), but since it is not the focus of our research, we take a relatively simple approach here, assuming no labeled data for demographic attributes.

4.2 Results

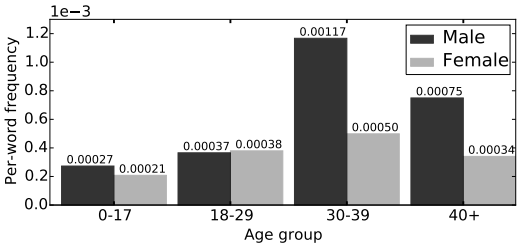
Linguistic differences by dataset We first consider the impact of the data acquisition technique on the lexical features associated with each city. The keywords identified in GPS-MSA-BALANCED dataset feature more geographically-specific non-standard words, which occur at a rate of 3.9×10^{-4} in GPS-MSA-BALANCED, versus 2.6×10^{-4} in LOC-MSA-BALANCED; this difference is statistically significant ($p < .05, t = 3.2$).[4]

[3] Binning is often employed in work on text-based age prediction (Garera and Yarowsky, 2009; Rao et al., 2010; Rosenthal and McKeown, 2011); it enables word and name counts to be shared over multiple ages, and avoids the complexity inherent in regressing a high-dimensional textual predictors against a numerical variable.

[4] We employ a paired t-test, comparing the difference in frequency for each word across the two datasets. Since we cannot test the complete set of entity names or non-standard words, this quantifies whether the observed difference is robust across the subset of the vocabulary that we have selected.



(a) non-standard words



(b) entity names

Figure 5: Aggregate statistics for geographically-specific non-standard words and entity names across imputed demographic groups, from the GPS-MSA-BALANCED sample.

For entity names, the difference between datasets was not significant, with a rate of 4.0×10^{-3} for GPS-MSA-BALANCED, and 3.7×10^{-3} for LOC-MSA-BALANCED. Note that these rates include only the non-standard words and entity names detected by SAGE as among the top 25 most distinctive for one of the ten largest cities in the US; of course there are many other relevant terms that are below this threshold.

In a pilot study of the GPS-COUNTY-BALANCED data, we found few linguistic differences from GPS-MSA-BALANCED, in either the aggregate word-group frequencies or the SAGE word lists — despite the geographical imbalances shown in Table 1 and Figure 1. Informal examination of specific counties shows some expected differences: for example, Clayton County, which hosts Atlanta's Hartsfield-Jackson airport, includes terms related to air travel, and other counties include mentions of local cities and business districts. But the aggregate statistics for under-represented counties are not substantially different from those of overrepresented counties, and are largely unaffected by county-based resampling.

Demographics Aggregate linguistic statistics for demographic groups are shown in Figure 5. Men use significantly more geographically-specific entity names than women ($p \ll .01, t =$

Age	Sex	New York	Dallas
0-17	F	*niall, ilysm, hemmings, stalk, ily*	*fanuary, idol, lmbo, lowkey, jonas*
	M	*ight, technique, kisses, lesbian, dicks*	*homies, daniels, oomf, teenager, brah*
18-29	F	*roses, castle, hmmmm, chem, sinking*	*socially, coma, hubby, bra, swimming*
	M	*drunken, manhattan, spoiler, guardians, gonna*	*harden, watt, astros, rockets, mavs*
30-39	F	*suite, nyc, colleagues, york, portugal*	*astros, sophia, recommendations, houston, prepping*
	M	*mets, effectively, cruz, founder, knicks*	*texans, rockets, embarrassment, tcu, mississippi*
40+	F	*cultural, affected, encouraged, proverb, unhappy*	*determine, islam, rejoice, psalm, responsibility*
	M	*reuters, investors, shares, lawsuit, theaters*	*mph, wazers, houston, tx, harris*

Table 3: Most characteristic words for demographic subsets of each city, as compared with the overall average word distribution

8.0), but gender differences for geographically-specific non-standard words are not significant ($p \approx .2$).[5] Younger people use significantly more geographically-specific non-standard words than older people (ages 0–29 versus 30+, $p \ll .01, t = 7.8$), and older people mention significantly more geographically-specific entity names ($p \ll .01, t = 5.1$). Of particular interest is the intersection of age and gender: the use of geographically-specific non-standard words decreases with age much more profoundly for men than for women; conversely, the frequency of mentioning geographically-specific entity names increases dramatically with age for men, but to a much lesser extent for women. The observation that high-level patterns of geographically-oriented language are more age-dependent for men than for women suggests an intriguing site for future research on the intersectional construction of linguistic identity.

For a more detailed view, we apply SAGE to identify the most salient lexical items for each MSA, subgrouped by age and gender. Table 3 shows word lists for New York (the largest MSA) and Dallas (the 5th-largest MSA), using the GPS-MSA-BALANCED sample. Non-standard words tend to be used by the youngest authors: *ilysm* ('I love you so much'), *ight* ('alright'), *oomf* ('one of my followers'). Older authors write more about local entities (*manhattan, nyc, houston*), with men focusing on sports-related entities (*harden, watt, astros, mets, texans*), and women above the age of 40 emphasizing religiously-oriented terms (*proverb, islam, rejoice, psalm*).

[5]But see Bamman et al. (2014) for a much more detailed discussion of gender and standardness.

5 Impact on text-based geolocation

A major application of geotagged social media is to predict the geolocation of individuals based on their text (Eisenstein et al., 2010; Cheng et al., 2010; Wing and Baldridge, 2011; Hong et al., 2012; Han et al., 2014). Text-based geolocation has obvious commercial implications for location-based marketing and opinion analysis; it is also potentially useful for researchers who want to measure geographical phenomena in social media, and wish to access a larger set of individuals than those who provide their locations explicitly.

Previous research has obtained impressive accuracies for text-based geolocation: for example, Hong et al. (2012) report a median error of 120 km, which is roughly the distance from Los Angeles to San Diego, in a prediction space over the entire continental United States. These accuracies are computed on test sets that were acquired through the same procedures as the training data, so if the acquisition procedures have geographic and demographic biases, then the resulting accuracy estimates will be biased too. Consequently, they may be overly optimistic (or pessimistic!) for some types of authors. In this section, we explore where these text-based geolocation methods are most and least accurate.

5.1 Methods

Our data is drawn from the ten largest metropolitan areas in the United States, and we formulate text-based geolocation as a ten-way classification problem, similar to Han et al. (2014).[6] Using our

[6]Many previous papers have attempted to identify the precise latitude and longitude coordinates of individual authors, but obtaining high accuracy on this task involves much more complex methods, such as latent variable models (Eisenstein et al., 2010; Hong et al., 2012), or multilevel grid structures (Cheng et al., 2010; Roller et al., 2012). Tuning such

user-balanced samples, we apply ten-fold cross validation, and tune the regularization parameter on a development fold, using the vocabulary of the sample as features.

5.2 Results

Many author-attribute prediction tasks become substantially easier as more data is available (Burger et al., 2011), and text-based geolocation is no exception. Since GPS-MSA-BALANCED and LOC-MSA-BALANCED have very different usage rates (Figure 2), perceived differences in accuracy may be purely attributable to the amount of data available per user, rather than to users in one group being inherently harder to classify than another. For this reason, we bin users by the number of messages in our sample of their timeline, and report results separately for each bin. All errorbars represent 95% confidence intervals.

GPS versus location As seen in Figure 6a, there is little difference in accuracy across sampling techniques: the location-based sample is slightly easier to geolocate at each usage bin, but the difference is not statistically significant. However, due to the higher average usage rate in GPS-MSA-BALANCED(see Figure 2), the overall accuracy for a sample of users will appear to be higher on this data.

Demographics Next, we measure classification accuracy by gender and age, using the posterior distribution from the expectation-maximization algorithm to predict the gender of each user (broadly similar results are obtained by using the prior distribution). For this experiment, we focus on the GPS-MSA-BALANCED sample. As shown in Figure 6b, text-based geolocation is consistently more accurate for male authors, across almost the entire spectrum of usage rates. As shown in Figure 6c, older users also tend to be easier to geolocate: at each usage level, the highest accuracy goes to one of the two older groups, and the difference is significant in almost every case. As discussed in § 4, older male users tend to mention many entities, particularly sports-related terms; these terms are apparently more predictive than the non-standard spellings and slang favored by younger authors.

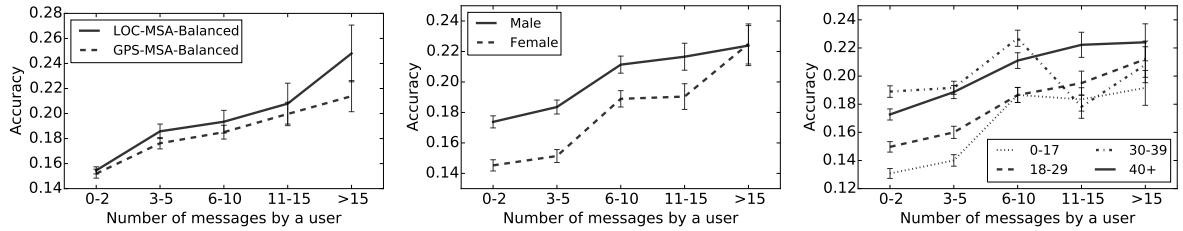
models can be challenging, and the resulting accuracies might be affected by initial conditions or hyperparameters. We therefore focus on classification, employing the familiar and well-understood method of logistic regression.

6 Related Work

Several researchers have studied how adoption of Internet technology varies with factors such as socioeconomic status, age, gender, and living conditions (Zillien and Hargittai, 2009). Hargittai and Litt (2011) use a longitudinal survey methodology to compare the effects of gender, race, and topics of interest on Twitter usage among young adults. Geographic variation in Twitter adoption has been considered both internationally (Kulshrestha et al., 2012) and within the United States, using both the Twitter location field (Mislove et al., 2011) and per-message GPS coordinates (Hecht and Stephens, 2014). Aggregate demographic statistics of Twitter users' geographic census blocks were computed by O'Connor et al. (2010) and Eisenstein et al. (2011b); Malik et al. (2015) use census demographics in spatial error model. These papers draw similar conclusions, showing that the the distribution of geotagged tweets over the US population is not random, and that higher usage is correlated with urban areas, high income, more ethnic minorities, and more young people. However, this prior work did not consider the biases introduced by relying on geotagged messages, nor the consequences for geo-linguistic analysis.

Twitter has often been used to study the geographical distribution of linguistic information, and of particular relevance are Twitter-based studies of regional dialect differences (Eisenstein et al., 2010; Doyle, 2014; Gonçalves and Sánchez, 2014; Eisenstein, 2015) and text-based geolocation (Cheng et al., 2010; Hong et al., 2012; Han et al., 2014). This prior work rarely considers the impact of the demographic confounds, or of the geographical biases mentioned in § 3. Recent research shows that accuracies of core language technology tasks such as part-of-speech tagging are correlated with author demographics such as author age (Hovy and Søgaard, 2015); our results on location prediction are in accord with these findings. Hovy (2015) show that including author demographics can improve text classification, a similar approach might improve text-based geolocation as well.

We address the question about the impact of geographical biases and demographic confounds by measuring differences between three sampling techniques, in both language use and in the accuracy of text-based geolocation. Recent unpublished work proposes reweighting Twitter data to



(a) Classification accuracy by sampling technique

(b) Classification accuracy by user gender

(c) Classification accuracy by imputed age

Figure 6: Classification accuracies

correct biases in political analysis (Choy et al., 2012) and public health (Culotta, 2014). Our results suggest that the linguistic differences between user-supplied profile locations and per-message geotags are more significant, and that accounting for the geographical biases among geotagged messages is not sufficient to offer a representative sample of Twitter users.

7 Discussion

Geotagged Twitter data offers an invaluable resource for studying the interaction of language and geography, and is helping to usher in a new generation of location-aware language technology. This makes critical investigation of the nature of this data source particularly important. This paper uncovers demographic confounds in the linguistic analysis of geo-located Twitter data, but is limited to demographics that can be readily induced from given names. A key task for future work is to quantify the representativeness of geotagged Twitter data with respect to factors such as race and socioeconomic status, while holding geography constant. However, these features may be more difficult to impute from names alone. Another crucial task is to expand this investigation beyond the United States, as the varying patterns of use for social media across countries (Pew Research Center, 2012) implies that the findings here cannot be expected to generalize to every international context.

Acknowledgments Thanks to the anonymous reviewers for their useful and constructive feedback on our submission. The following members of the Georgia Tech Computational Linguistics Laboratory offered feedback throughout the research process: Naman Goyal, Yangfeng Ji, Vinodh Krishan, Ana Smith, Yijie Wang, and Yi Yang. This research was supported by the National Science Foundation under awards IIS-1111142 and RI-1452443, by the National Institutes of Health under award number R01GM112697-01, and by the Air Force Office of Scientific Research. The content is solely the responsibility of the authors and does not necessarily represent the official views of these sponsors.

References

David Bamman, Jacob Eisenstein, and Tyler Schnoebelen. 2014. Gender identity and lexical variation in social media. *Journal of Sociolinguistics*, 18(2):135–160.

David A Broniatowski, Michael J Paul, and Mark Dredze. 2013. National and local influenza surveillance through twitter: An analysis of the 2012-2013 influenza epidemic. *PloS one*, 8(12):e83672.

John D. Burger, John Henderson, George Kim, and Guido Zarrella. 2011. Discriminating gender on twitter. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Guido Caldarelli, Alessandro Chessa, Fabio Pammolli, Gabriele Pompa, Michelangelo Puliga, Massimo Riccaboni, and Gianni Riotta. 2014. A multi-level geographical study of Italian political elections from Twitter Data. *PloS one*, 9(5):e95809.

Jonathan Chang, Itamar Rosenn, Lars Backstrom, and Cameron Marlow. 2010. ePluribus: Ethnicity on social networks. In *Proceedings of the International Conference on Web and Social Media (ICWSM)*, pages 18–25, Menlo Park, California. AAAI Publications.

Zhiyuan Cheng, James Caverlee, and Kyumin Lee. 2010. You are where you tweet: a content-based approach to geo-locating twitter users. In *Proceedings of the International Conference on Information and Knowledge Management (CIKM)*, pages 759–768.

Murphy Choy, Michelle Cheong, Ma Nang Laik, and Koo Ping Shung. 2012. Us presidential election 2012 prediction using census corrected twitter model. *arXiv preprint arXiv:1211.0938*.

Aron Culotta. 2014. Reducing sampling bias in social media data for county health inference. In *Joint Statistical Meetings Proceedings*.

Gabriel Doyle. 2014. Mapping dialectal variation by querying social media. In *Proceedings of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 98–106, Stroudsburg, Pennsylvania. Association for Computational Linguistics.

Mark Dredze, Michael J Paul, Shane Bergsma, and Hieu Tran. 2013. Carmen: A Twitter geolocation system with applications to public health. In *AAAI Workshop on Expanding the Boundaries of Health Informatics Using Artificial Intelligence*, pages 20–24.

Jacob Eisenstein, Brendan O'Connor, Noah A. Smith, and Eric P. Xing. 2010. A latent variable model for geographic lexical variation. In *Proceedings of Empirical Methods for Natural Language Processing (EMNLP)*, pages 1277–1287, Stroudsburg, Pennsylvania. Association for Computational Linguistics.

Jacob Eisenstein, Amr Ahmed, and Eric P. Xing. 2011a. Sparse additive generative models of text. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 1041–1048, Seattle, WA.

Jacob Eisenstein, Noah A. Smith, and Eric P. Xing. 2011b. Discovering sociolinguistic associations with structured sparsity. In *Proceedings of the Association for Computational Linguistics (ACL)*, pages 1365–1374, Portland, OR.

Jacob Eisenstein. 2015. Written dialect variation in online social media. In Charles Boberg, John Nerbonne, and Dom Watt, editors, *Handbook of Dialectology*. Wiley.

Irene Eleta and Jennifer Golbeck. 2014. Multilingual use of twitter: Social networks at the language frontier. *Computers in Human Behavior*, 41:424–432.

Nikesh Garera and David Yarowsky. 2009. Modeling latent biographic attributes in conversational genres. In *Proceedings of the Association for Computational Linguistics (ACL)*, pages 710–718.

Bruno Gonçalves and David Sánchez. 2014. Crowdsourcing dialect characterization through twitter. *PloS one*, 9(11):e112074.

Bo Han, Paul Cook, and Timothy Baldwin. 2014. Text-based twitter user geolocation prediction. *Journal of Artificial Intelligence Research (JAIR)*, 49:451–500.

Eszter Hargittai and Eden Litt. 2011. The tweet smell of celebrity success: Explaining variation in twitter adoption among a diverse group of young adults. *New Media & Society*, 13(5):824–842.

Brent Hecht and Monica Stephens. 2014. A tale of cities: Urban biases in volunteered geographic information. In *Proceedings of the International Conference on Web and Social Media (ICWSM)*, pages 197–205, Menlo Park, California. AAAI Publications.

Brent Hecht, Lichan Hong, Bongwon Suh, and Ed H Chi. 2011. Tweets from Justin Bieber's heart: the dynamics of the location field in user profiles. In *Proceedings of Human Factors in Computing Systems (CHI)*, pages 237–246.

Liangjie Hong, Amr Ahmed, Siva Gurumurthy, Alexander J. Smola, and Kostas Tsioutsiouliklis. 2012. Discovering geographical topics in the twitter stream. In *Proceedings of the Conference on World-Wide Web (WWW)*, pages 769–778, Lyon, France.

Dirk Hovy and Anders Søgaard. 2015. Tagging performance correlates with author age. In *Proceedings of the Association for Computational Linguistics (ACL)*, pages 483–488, Beijing, China.

Dirk Hovy. 2015. Demographic factors improve classification performance. In *Proceedings of the Association for Computational Linguistics (ACL)*, pages 752–762, Beijing, China.

Juhi Kulshrestha, Farshad Kooti, Ashkan Nikravesh, and Krishna P. Gummadi. 2012. Geographic Dissection of the Twitter Network. In *Proceedings of the International Conference on Web and Social Media (ICWSM)*, Menlo Park, California. AAAI Publications.

Amanda Lenhart. 2015. Mobile access shifts social media use and other online activities. Technical report, Pew Research Center, April.

P. A. Longley, M. Adnan, and G. Lansley. 2015. The geotemporal demographics of twitter usage. *Environment and Planning A*, 47(2):465–484.

Momin Malik, Hemank Lamba, Constantine Nakos, and Jürgen Pfeffer. 2015. Population bias in geotagged tweets. In *Papers from the 2015 ICWSM Workshop on Standards and Practices in Large-Scale Social Media Research*, pages 18–27. The AAAI Press.

Alan Mislove, Sune Lehmann, Yong-Yeol Ahn, Jukka-Pekka Onnela, and J. Niels Rosenquist. 2011. Understanding the demographics of twitter users. In *Proceedings of the International Conference on Web and Social Media (ICWSM)*, pages 554–557, Menlo Park, California. AAAI Publications.

Burt L Monroe, Michael P Colaresi, and Kevin M Quinn. 2008. Fightin'words: Lexical feature selection and evaluation for identifying the content of political conflict. *Political Analysis*, 16(4):372–403.

Dong Nguyen, Dolf Trieschnigg, A Seza Dogruöz, Rilana Gravel, Mariët Theune, Theo Meder, and Franciska de Jong. 2014. Why gender and age prediction from tweets is hard: Lessons from a crowdsourcing experiment. In *Proceedings of the International Conference on Computational Linguistics (COLING)*, pages 1950–1961.

Brendan O'Connor, Jacob Eisenstein, Eric P. Xing, and Noah A. Smith. 2010. A mixture model of demographic lexical variation. In *Proceedings of NIPS Workshop on Machine Learning for Social Computing*, Vancouver.

Pew Research Center. 2012. Social networking popular across globe. Technical report, December.

Barbara Poblete, Ruth Garcia, Marcelo Mendoza, and Alejandro Jaimes. 2011. Do all birds tweet the same? characterizing Twitter around the world. In *Proceedings of the International Conference on Information and Knowledge Management (CIKM)*, pages 1025–1030. ACM.

Delip Rao, David Yarowsky, Abhishek Shreevats, and Manaswi Gupta. 2010. Classifying latent user attributes in twitter. In *Proceedings of Workshop on Search and mining user-generated contents*.

Stephen Roller, Michael Speriosu, Sarat Rallapalli, Benjamin Wing, and Jason Baldridge. 2012. Supervised text-based geolocation using language models on an adaptive grid. In *Proceedings of Empirical Methods for Natural Language Processing (EMNLP)*, pages 1500–1510.

Sara Rosenthal and Kathleen McKeown. 2011. Age prediction in blogs: A study of style, content, and online behavior in pre- and Post-Social media generations. In *Proceedings of the Association for Computational Linguistics (ACL)*, pages 763–772, Portland, OR.

Thamar Solorio and Yang Liu. 2008. Learning to predict code-switching points. In *Proceedings of Empirical Methods for Natural Language Processing (EMNLP)*, pages 973–981, Honolulu, HI, October. Association for Computational Linguistics.

Svitlana Volkova and Benjamin Van Durme. 2015. Online bayesian models for personal analytics in social media. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 2325–2331.

Benjamin Wing and Jason Baldridge. 2011. Simple supervised document geolocation with geodesic grids. In *Proceedings of the Association for Computational Linguistics (ACL)*, pages 955–964, Portland, OR.

Sarita Yardi, Daniel Romero, Grant Schoenebeck, et al. 2009. Detecting spam in a twitter network. *First Monday*, 15(1).

Kathryn Zickuhr. 2013. Location-based services. Technical report, Pew Research Center, Septmeber.

Nicole Zillien and Eszter Hargittai. 2009. Digital distinction: Status-specific types of internet usage*. *Social Science Quarterly*, 90(2):274–291.

Modeling Reportable Events as Turning Points in Narrative

Jessica Ouyang
Department of Computer Science
Columbia University
New York, NY 10027
ouyangj@cs.columbia.edu

Kathleen McKeown
Department of Computer Science
Columbia University
New York, NY 10027
kathy@cs.columbia.edu

Abstract

We present novel experiments in modeling the rise and fall of story characteristics within narrative, leading up to the *Most Reportable Event* (MRE), the compelling event that is the nucleus of the story. We construct a corpus of personal narratives from the bulletin board website Reddit, using the organization of Reddit content into topic-specific communities to automatically identify narratives. Leveraging the structure of Reddit comment threads, we automatically label a large dataset of narratives. We present a change-based model of narrative that tracks changes in formality, affect, and other characteristics over the course of a story, and we use this model in distant supervision and self-training experiments that achieve significant improvements over the baselines at the task of identifying MREs.

1 Introduction

What is a narrative? In one of the early linguistic analyses of storytelling, Prince (1973) defines a story as describing an event that causes a change of state. Prince's minimal story has three parts: the starting state, the ending state, and the event that transforms the stating state into the ending state. An example of a minimal story is as follows:

> A man was unhappy, then he fell in love, then as a result, he was happy.

Polanyi (1976) notes that minimal stories are toy examples that would never hold an audience's interest. So what makes a story interesting?

Labov (1967; 1997) defines a well-formed narrative as a series of actions leading to a *Most Reportable Event* (MRE). The MRE is the point of the story – the most unusual event that has the greatest emotional impact on the narrator and the audience. For a story to be interesting, Prince's change-of-state event should be an MRE.

The following is an example of a narrative from the corpus we create in this work, with the sentence containing the MRE emphasized:

> This isn't exactly creepy, but it's one of the scariest things that's ever happened to me. I was driving down the motorway with my boyfriend in the passenger seat, and my dad in the seat behind my own. My dad is an epileptic and his fits are extremely sporadic. Sometimes he goes extremely stiff and other times he will try to get out of places or grab and punch people. *Mid-conversation I felt his hands wrap around my throat as I was driving, pulling my head back and making it increasingly difficult to drive.* My boyfriend managed to help steer the car into the hard shoulder but it was one of the scariest experiences in my life.

The MRE is the shortest possible summary of a story; it is what we would say about the story if we could only say one thing. If we could identify the MRE of a narrative, we could automatically generate summaries or headline-style titles for online stories. Detecting MREs could also allow us to explore how storytellers build emotional impact as they lead up to the climaxes of their stories.

In this work, we present a novel approach to modeling narrative in order to automatically identify the MRE. The MRE is a real world event underlying the story and thus is difficult to infer; instead, we identify sentences that describe or refer to it. We incorporate Prince's change-of-state formalization as well as Labov's definition of the MRE by modeling changes in story characteristics suggested by Prince, Polanyi, and Labov, such as measures of syntactic complexity and emotional

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 2149–2158,
Lisbon, Portugal, 17-21 September 2015.

content. If Prince and Labov are both correct, we should find the MRE at a point of change in the story and in our story characteristics.

We create a corpus of thousands of personal narratives collected from Reddit, a social bulletin board website organized into topic-specific 'subreddit' communities. We automatically label most of this data using heuristics based on the comment-thread structure of Reddit content. Using this corpus, we conduct two experiments in classifying sentences of a story as containing the MRE or not: the first using distant supervision, and the second using self-training.

In Section 2, we discuss prior work on automatically identifying personal narratives, as well as related experiments using Labov's theory of narrative analysis. Section 3 discusses data collection and labeling. Sections 4-5 present our change-based model of narrative and our experiments. Finally, Section 6 discusses our experimental results and proposes directions for future work.

2 Related Work

Prior work using Labov's theory of narrative has focused on classifying clauses by their function.

Rahimtoroghi et al. (2013)worked on 20 of Aesop's fables. The 315 clauses were manually annotated with the three labels of Labov and Waltezky (1967), Orientation (background information), Action (events), and Evaluation (author's perspective), which we discuss in Section 4. Rahimtoroghi et al. used two annotators with high agreement and achieved accuracy and precision around 0.9 on all three labels, as well as recall above 0.9 on all but Orientation. They noted that their data set was very clean: interannotator agreement was nearly perfect, the language was simple, and each clause served a clear narrative purpose.

Ouyang and McKeown (2014) explored identifying the Action chain of the oral narratives in Labov (2013). They used a dataset of 49 narratives (1,277 clauses), transcribed from recordings of speech and annotated by Labov and achieved 0.72 f-score on classifying clauses as Action or not. This task is easier than our proposed task of identifying sentences containing MREs. Actions account for nearly half the clauses in the Labov (2013) dataset, while there are only and average of 2.5 MRE sentences per story. Additionally, identifying Labov's Actions is a problem of detecting causal and temporal relations among events; identifying the MRE is a problem of measuring how impactful and shocking an event is.

Swanson et al. (2014) used 50 stories, which were annotated with an extended label set by three annotators, and each of the 1,602 clauses was assigned the label given by the majority of annotators. The extended label set was then mapped to Labov and Waletzky's three labels. Nearly half of the clauses in this dataset are Evaluations, and Orientations and Actions each make up nearly one quarter of the dataset. Swanson et al. achieved 0.69 overall f-score on three-way classification of clauses. Again, this task is less difficult than our proposed task. The three labels, Orientation, Action, and Evaluation have distinct functions that are reflected in tense, mood, and a clause's position in the narrative. The MRE is not a sentence or clause but an event that may be described or referred to by any sentence in a narrative; it is distinguished from the other events only by its surprisingness and emotional impact, dimensions that are difficult to model computationally without a deep semantic understanding of the story.

The stories that Swanson et al. used were drawn from a corpus drawn from weblog posts (Gordon and Swanson, 2009). Gordon and Swanson used unigram features to classify posts as either stories or not, achieving 75% precision. They note that only about 17% of weblog text consists of stories.

In contrast to the relatively small datasets used by Rahimtoroghi et al., Ouyang and McKeown, and Swanson et al., we use a larger dataset automatically collected from Reddit. Our collection method achieved 94% precision in identifying narratives. A number of researchers have characterized the structure and use of Reddit, currently the 26th most popular website in the world[1]. Weninger et al. (2013) described the structure of Reddit comment threads. Gilbert (2013) measured user participation in the voting process that ranks Reddit content. Singer et al. (2014) conducted a longitudinal study of the Reddit user community, finding a trend favoring original, user-generated content.

3 Data

3.1 Collection

We collected data from the AskReddit subreddit, where users post questions for other members of the community, who reply with comments answering the questions. Table 1 shows some examples

[1] http://www.alexa.com/siteinfo/reddit.com

of these posts, and we can see some of the wide variety of story topics found on AskReddit.

Post Title
Whats your creepiest (REAL LIFE) story?
Your best "Accidentally Racist" story?
What are your stories of petty revenge?

Table 1: Examples of AskReddit posts.

Using PRAW[2], we scraped the top 50 AskReddit posts containing the keyword 'story.' Of these posts, 10 were tagged as NSFW ('not safe for work'), indicating they contained adult content; we did not use these posts in this work, as we felt the language would be too different from that used in posts without the tag. Another 3 posts did not contain personal narratives, and instead were about fictional stories in movies or music.

With the 37 remaining posts, we treated each top-level comment (those that replied directly to the posted question) as a story. The example given in Section 1 is one such story. We collected 6,000 top-level comments and discarded those without comment threads replying to them. As we discuss in Section 3.2, we use comment threads to automatically label our training data. We tokenized the top-level comments by sentence (Bird et al., 2009) and removed all sentences following any variation of the word 'EDIT', as these were usually responses to readers' comments. We discarded texts with fewer than three sentences, based on Prince's definition of a minimal story as consisting of a starting state, an event, and an ending state. We are left with 4,896 stories, with an average length of 16 sentences and a maxiumum of 198.

3.2 Labeling

We partitioned our data into development, seed, and tuning sets of 100 stories each; a testing set of 200 stories; and a training set of 4,178 stories. The development, seed, tuning, and testing sets were manually annotated by a native English speaker (not one of the authors), who was instructed to label all sentences that contained or referred to the MRE. For convenience, from here on, we will use the term 'MRE' to refer to both the Most Reportable Event itself (of which there can only be one per narrative) and to sentences that contain or refer to it (of which there can be more than one).

To measure interannotator agreement, we also had a second annotator (also a native English speaker and not one of the authors) label MREs in the 100 narratives in our development set. We found substantial agreement (Cohen's $\kappa = 0.729$); the two classes, MRE and not-MRE, are highly unbalanced, so percent agreement between the two annotators was extremely high (95%).

In addition to labeling the MREs, our first annotator identified and discarded 31 texts that were not true stories, but rather Reddit-specific inside jokes or comments on how cool the stories in the thread were. From this, we can see that the precision of our story collection method is very high. Gordon et al. (2007) found that stories were 17% of the weblog text that they collected; of the 500 texts given to our annotators, 94% were stories.

Using the development set, we experimented with seven heuristics, defined below, for automatically labeling the training set. Each predicts a sentence index s_h to be the index of an MRE. We measured the performance of each heuristic using root-mean-square error (RMSE), which measures the standard deviation of how far the heuristic's predictions fall from a true MRE.

$$\text{Let } N = \text{number of narratives}$$
$$s_{MRE} = \text{index of a true MRE}$$
$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(s_{MRE_i} - s_{h_i})^2}$$

We used a linear combination of three heuristics with the lowest RMSE to label our training set.

Similarity to comment. The bag-of-words cosine similarity between a sentence and comments replying to the story. We expect comments to refer to the MRE because of its shocking nature and importance to the story. This heuristic achieved RMSE of 5.5 sentences on the development set.

Similarity to tl;dr. The latent semantic similarity between a sentence and the *tl;dr*. The tl;dr (too long; didn't read) is a very short paraphrase of a post given by its author. They are relatively rare – 663 stories, or 14% of our data, had tl;drs. Since the MRE is the central event of the story, we expect it to be included in the tl;dr. We calculated the similarity using the weighted matrix factorization algorithm described by Guo and Diab (2012). This heuristic achieved RMSE of 5.8 sentences.

[2]https://praw.readthedocs.org/en/v2.1.20/, Python Reddit API Wrapper

In contrast, bag-of-words cosine similarity to the tl;dr performed poorly (RMSE of 13.2). This is due to the tl;dr being both short and a paraphrase of its story. There are few words in the tl;dr, and those words are often synonyms of, but not the same as, words in the story. Guo and Diab's latent semantic similarity score addresses this word sparsity problem by modeling words that are not present in the input text. We also experimented with latent semantic similarity for the similarity-to-comment and similarity-to-prompt heuristics, but in these two cases, it did not perform as well as the bag-of-words cosine similarity.

Similarity to prompt. The bag-of-words cosine similarity between a sentence and the AskReddit post that prompted the story. The story should be relevant to the prompt, so we expect the MRE to be similar to the prompt text. This heuristic achieved RMSE of 6.3 sentences.

We used the heuristic with the fourth lowest RMSE as one of the baselines in our experiments:

Last sentence. The last sentence in the story. Since the events of a story build up to the MRE, the MRE should occur near the end of the story. This heuristic achieved RMSE of 6.9 sentences.

Other heuristics. We also tried the following:

- Single-sentence paragraph (RMSE of 8.7). This heuristic was meant to capture emphasis, as an MRE might be placed in its own, separate paragraph to draw attention to it.
- First sentence (RMSE of 13.7). Narratives occasionally open with a brief introductory paragraph that summarizes the events to come. This heuristic was meant to capture a reference to the MRE in this introduction.

The training set was automatically labeled using a linear combination of the three best-performing heuristics: similarity to comment, similarity to tl;dr, and similarity to prompt.

$$h_{\text{label}} = 0.2 * h_{\text{comment}} + 0.5 * h_{\text{tldr}} + 0.3 * h_{\text{prompt}}$$

This outperformed each of the three alone, achieving an RMSE of 5.1 sentences. The weights for each heuristic were tuned on the development set. For stories without a tl;dr, that heuristic was set to 0. The sentence in the story with the highest heuristic score was selected as the MRE.

In 52 of the 99 stories in the development set, we found that multiple, consecutive sentences were labeled by our annotator as MREs. The average number of consecutive MREs was 2.5 sentences. To reflect this, we labeled our training set in three-sentence blocks. The sentence selected by our labeling heuristic, along with the immediately preceding and following sentences, were all labeled as MREs. The result was the weakly-labeled training set in Table 2.

		Number of Sentences	
Data Set	Stories	MRE	Total
dev*	99	169	1528
seed*	82	184	958
tuning*	95	212	1301
testing*	193	444	2771
training	4178	11205	67954

Table 2: Distribution of labels (*manual).

4 Modeling Narrative

Our approach to modeling narrative is based on both Labov (2013) and Prince (1973). We claim that Labov's MRE is Prince's change of state with the added requirement of reportability or interestingness – in fact, all three components of Prince's minimal story have equivalences in Labov.

Labov and Waletzky (1967) proposed three components of narrative: the Orientation, which we equate with Prince's starting state; the Action, the chain of events culminating in the MRE; and the Evaluation, the author's perspective on the story. Labov (2013) adds three more components: the Resolution, equivalent to Prince's ending state, and the Abstract and Coda, where the author introduces and concludes the story.

We focus on Prince's claim that stories are about change. Polanyi (1985) observes that the turning point of a story is marked by a change in style, formality of language, or emphasis in the telling of the story. Labov (2013) likewise observes that a change in verb tense often accompanies MREs. We hypothesize that the MRE should be found at a point of change in the story.

We score each sentence according to three views of narrative: syntax, semantics, and affect.

Syntax. We model Polanyi's claim that a change in formality marks the changing point by including metrics of sentential syntax; we use the syntactic complexity of a sentence as an approximation for formality. The complexity of a sentence also reflects emphasis – short, staccato sentences bear more emphasis than long, complicated ones. We use the length of the sen-

tence, the length of its verb phrase, and the ratio of these two lengths; the depth of the sentence's parse tree (Klein and Manning, 2003), the depth of its verb phrase's subtree, and the ratio of these two depths. We also use the average word length for the sentence and the syntactic complexity formula proposed by Botel and Granowsky (1972), which scores sentences on specific structures, such as passives, appositives, and clausal subjects. Finally, we use the formality and complexity dictionaries described in Pavlick and Nenkova (2015), which provide human formality judgments for 7,794 words and short phrases and complexity judgments for 5,699 words and phrases. We score each sentence by averaging across all words and phrases in the sentence.

Semantics. As the MRE is surprising and shocking, we expect it to be dissimilar from the surrounding sentences; we use semantic similarity to surrounding sentences as a measure of shock. Our semantic scores are the bag-of-words cosine and the latent semantic similarity scores for adjacent sentences (Guo and Diab, 2012).

Affect. A change in affect reflects a change in style, and we expect the MRE to occur at an emotional peak. We use the Dictionary of Affect in Language (DAL) (Whissell, 1989), augmented with WordNet for coverage (Miller, 1995). The DAL represents lexical affect with three scores: evaluation (*ee*, hereafter 'pleasantness' to avoid confusion with Labov's Evaluation), activation (*aa*, activeness), and imagery (*ii*, concreteness). We also use a fourth score, the activation-evaluation (AE) norm, a measure of subjectivity defined by Agarwal et al. (2009):

$$norm = \frac{\sqrt{ee^2 + aa^2}}{ii}$$

For each of these four word-level scores, we calculate a sentence-level score by averaging across the words in the sentence using the finite state machine described by Agarwal et al. We expect the sentences surrounding an MRE to be more subjective and emotional as the impact of the MRE becomes clear. We also expect a build-up in activeness and intensity, peaking at the MRE.

To model change over the course of a narrative, we look for changes in the syntactic, semantic, and affectual scores. To illustrate this, Figure 1 shows the activeness and pleasantness DAL scores for the example narrative given in Section 1. We can see how the MRE is the most exciting sentence in the story – global maximum in activation – as well as the most horrifying – global minimum in pleasantness. The overall shape of the activeness scores reflects Prince's three components of a minimal story: low initial activation (starting state) and low final activation (ending state) with a build up to a peak at the MRE (change in state) between them.

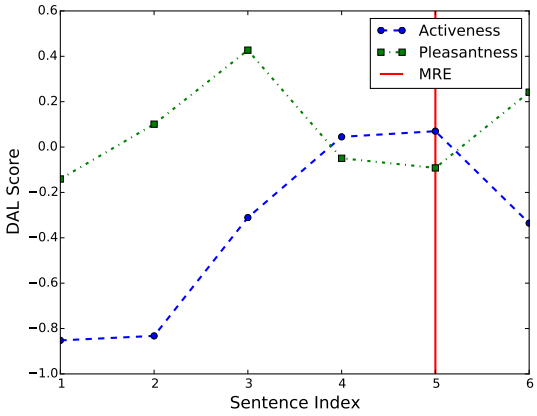


Figure 1: Activeness and pleasantness scores.

5 Experiments

Using our Reddit dataset and change-based model of narrative, we conducted two experiments on automatically identifying MREs. We compare our results with three baselines: random, our labeling heuristic, and the last sentence of the story (best-performing heuristic not used in labeling).

As described in Section 3.2, we labeled our training set in blocks of three consecutive MREs, centered on the sentence from each narrative that was selected by our heuristics. To account for this, in our experiments and baselines, we predicted the presence of an MRE in a three-sentence block. In testing, we considered a predicted block to be correct if it contained at least one gold-label MRE.

5.1 Features

Change-based Features. For each of the fifteen metrics in Section 4, shown in Table 3, we first smooth the scores by applying a Gaussian filter. We also tried weighted and exponential moving averages, as well as a Hamming window, but the Gaussian performed best in experiments on our tuning set. We then generate 11 features for each sentence: the metric score at the sentence; whether or not the sentence is a local maximum or minimum; the sentence's distance from the global

Type	Metric Names
Syntactic	sentlength, vplength, lengthratio, sentdepth, vpdepth, depthratio wordlength, structcomplexity, wordformality, wordcomplexity
Semantic	cossimilarity, lssimilarity
Affectual	pleasantness, activation, imagery, subjectivity

Table 3: The fifteen metrics for change.

maximum and minimum; the difference in score between the sentence and the preceding sentence, the difference between the sentence and the following sentence, and the average of these differences (approximating the incoming, outgoing, and self- slopes for the metric); and the incoming, outgoing, and self- differences of differences (approximating the second derivative).

Other Features.

- The tense of the main verb and whether or not there is a shift from the previous sentence. Labov (2013) suggests a shift between the past and the historical present near the MRE.
- The position of the sentence in the narrative.
- The bag-of-words cosine similarity and latent semantic similarity between the sentence and the first and second sentences in the narrative. The MRE usually appears near the end of a story, but Labov (2013) notes that the Abstract, a short introduction that occurs in some narratives, often refers to the MRE.

5.2 Distant Supervision

Our first experiment used a distant supervision approach with our automatically-labeled training set. Distant supervision has previously been applied to NLP problems such as sentiment analysis (Go et al., 2009; Purver and Battersby, 2012; Suttles and Ide, 2013) and relation extraction (Mintz et al., 2009; Yao et al., 2010; Hoffmann et al., 2011; Nguyen and Moschitti, 2011; Krause et al., 2012; Min et al., 2013; Xu et al., 2013).

We classify blocks of three sentences as containing the MRE or not. The two classes, *MRE* and *not-MRE*, were weighted inversely to their frequencies in the weakly-labeled set, and all features were normalized to the range $[0, 1]$. We trained an SVM with margin $C = 1$ and an RBF kernel with $\gamma = 0.001$, chosen using grid search on our tuning set (Pedregosa et al., 2011).

Trial	Precision	Recall	F-Score
Last sent. baseline	0.208	0.112	0.146
Heuristic baseline	0.107	0.333	0.162
No change*	0.146	0.378	0.211
Random baseline	0.185	0.586	0.281
Change only*	0.351	0.685	0.466
All features*	**0.398**	**0.745**	**0.519**

Table 4: Distant supervision results (*$p < 0.01$).

The results of the distant supervision experiment are shown in Table 4. Our best results use all features, but, notably, using the change-based features alone achieves significant improvement over the three baselines ($p < 0.00005$). The 'no change' trial used the metric scores themselves and the 'other' features but none of the change-based features, such as slopes and proximity to global extremes. This feature set was outperformed by the random baseline ($p < 0.0024$), supporting our hypothesis that it is change in a metric, rather than the score itself, that predicts MREs.

Because we used an non-linear kernel, we were not able to examine feature weights directly. Instead, Table 5 shows the results of a logistic regression model trained on our features. The 10 best features are shown, along with their weights and 95% confidence intervals. From feature 8, we see that the MRE is found in sentences near the narrative's global minimum in imagery (the Evaluation), but feature 1 indicates that sentences containing the MRE show a sharp increase in imagery compared to the previous sentences. The MRE is described in a burst of vivid language, followed by more abstract author opinions .

Features 2 and 9 indicate that the MRE tends to be described using informal language – a textual echo to Labov's observation that the subjects of his sociolinguistic interviews spoke less formally and more colloquially as they relived the climaxes of their stories (Labov, 2013). Feature 3 suggests that sentences containing the MRE are similar to the surrounding sentences. While we expected MRE sentences to be different from their neighbors due to the unusual and shocking nature of the MRE, this feature seems instead to reinforce the idea that MREs tend to described over the course of multiple, consecutive sentences, rather than in a single

	Feature Name	Weight	Confidence Interval
1.	incomingd2_imagery	4.174	(4.062, 4.287)
2.	distancefrommin_wordformality_neg	4.109	(3.952, 4.265)
3.	cossimilarity_adjacent	3.618	(3.425, 3.812)
4.	distancefrommin_activeness	3.377	(2.855, 3.298)
5.	sentdepth	3.364	(3.138, 3.590)
6.	distancefrommin_wordlength_neg	3.321	(3.018, 3.624)
7.	distancefrommin_vpdepth	3.034	(2.823, 3.247)
8.	distancefrommin_imagery_neg	2.790	(2.524, 3.056)
9.	wordformality_neg	2.329	(2.226, 2.432)
10.	incomingd2_vplen	2.128	(1.938, 2.318)

Table 5: Top 10 features.

sentence. From feature 4, we see, as expected, that the MRE is far from the narrative's global minimum in activeness, as it is the end of a chain of events, far away from the stative Orientation.

Finally, features 5 and 10 suggest that MRE sentences are not only long, but much longer than the preceding sentences, and feature 6 indicates that MRE sentences are close to the global minimum in average word length. Shorter average word length is expected, as an indicator of both informal word choice and emphasis. Long sentences, however, suggest a domain difference between our work on text and Labov's work on transcribed speech. Looking over our development set, we find that many authors combine the description of the MRE with evaluative material in a single sentence, resulting in a longer and more syntactically complex MRE sentence than is found in Labov's data.

5.3 Self-Training

Our second experiment used a self-training approach, where a classifier uses a small, labeled seed set to label a larger training set. Self-training has been applied to parsing (McClosky et al., 2006; Reichart and Rappoport, 2007; McClosky and Charniak, 2008; Huang and Harper, 2009; Sagae, 2010) and word sense disambiguation (Mihalcea, 2004). With the same parameters as in the distant supervision experiment, we trained an SVM on our hand-labeled seed set of 958 sentences. We used this initial model to relabel the training set. All sentences where this labeling agreed with our automatically-generated heuristic labels were added to the seed set and used to train a new model, which was in turn used to label the remaining sentences, and so on until none of the current model's labels agreed with any of the remaining heuristic labels. Figure 2 shows the learning curve for the self-training experiment, along with the growth of the self-training set.

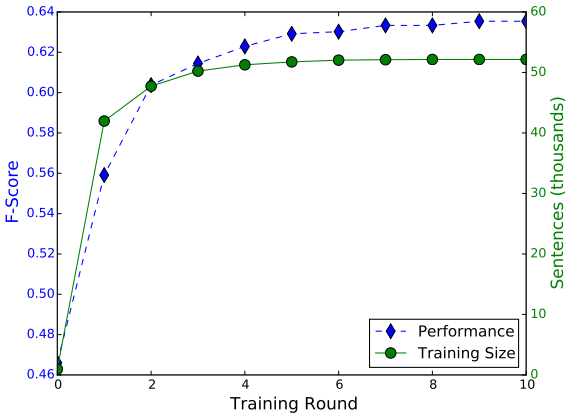


Figure 2: Learning and training set size curves.

The results of the self-training experiment are shown in Table 6. We achieve the best performance, f1 = 0.635, after 9 rounds of self-training. Self-training terminated after 10 rounds, but the 10th round had no effect on performance.

Trial	Precision	Recall	F-Score
Random baseline	0.185	0.586	0.281
Seed only*	0.374	0.617	0.466
Dist. supervision*	0.398	0.745	0.519
Self-training*	**0.478**	**0.946**	**0.635**

Table 6: Best self-training results (*$p < 0.01$).

The initial model, trained only on the seed set, performed nearly as well as our distant supervi-

sion experiment. This illustrates that quantity of data does not overcome the use of accurate manual labels on a small dataset. As described in Section 3.2, the distant supervision labels were based on a linear combination of three heuristics that achieved at best an RMSE of 5.1 sentences. However, with self-training, we can exploit the noisy heuristic labels by using only those labels that agree with the seed-trained model, thus reducing the amount of noise. 52,147 of the 67,954 weakly-labeled sentences were used in self-training.

6 Discussion and Future Work

Identifying MREs is a hard problem. A human annotator can rely on world knowledge to find the most shocking and impactful event in a story, but we do not have access to that knowledge. Additionally, MREs are rare, comprising 15% of the sentences in our hand-annotated datasets. MREs comprise just over 16% of our weakly-labeled training set, but as we discuss below, there is too much noise in the automatically-generated labels.

Despite the difficulty of the task, our experiments show that our change-based model of narrative is effective for identifying MREs, and this model provides evidence supporting the change-in-state view of narrative suggested by Prince (1973), Polanyi (1985), and Labov (1997). We achieve high recall with self-training (95%), but precision is low across the board. This suggests that, while MREs do occur at extremes in syntactic complexity, semantic similarity, and emotional activation, there may be many non-MRE local extremes throughout a narrative.

Examining our results, we find a few common sources of error. False positive sentences tend to have high imagery and activeness. In Table 5, we saw that imagery and activeness alone do not indicate the presence of the MRE. An MRE sentence is not just active; it is separated from the stative introduction by the other events of the story. Nor is it enough for a sentence to have high imagery; the MRE is more vividly described than the preceding events – we see again the importance of change in our model of narrative. False negatives tend to have high scores in syntactic complexity and formality. As low formality was one of our stronger predictors of MRE sentences, we may need to adjust these features in future work.

We also hope to refine our automatically-generated labels in future work. Our self-training experiment showed that 27% of our automatically-generated labels were too noisy to use. We also hope to improve our filters for automatically discarding non-story text. We currently reject texts shorter than three sentences, based on Prince's three-part definition of a story. In spite of this filtering, 7% of our 500 manually-labeled texts were identified as non-stories by our annotator. Extrapolating to our training set, we suspect that over 300 of our training 'narratives' are not narratives at all.

Finally, we hope to explore other theories of narrative analysis that could suggest new ways to quantify change in narrative. Prince, Polanyi, and Labov propose a high-level view of personal narrative: stories are centered around reportable events that cause a change in state for the author. This work tested fifteen surface-level features that reflect this change in state. Are there others? Or is a deeper semantic understanding of the starting and ending states of stories required?

7 Conclusion

We have described a new model of narrative based on Prince (1973), Polanyi (1985), and Labov (1997). Our model tracks story characterstics over the course of a narrative, capturing change in complexity, meaning, and emotion.

We have created a corpus of 4,896 personal narratives, taking advantage of AskReddit, a community where members often prompt each other for stories. Our experiments on this corpus show that our change-based model is able to identify MREs. They also demonstrate that large quantities of hand-labeled data are not required for this task. Our distant supervision and self-training approaches successfully use data weakly labeled using heuristic rules that leverage the comment thread structure of Reddit content. We believe these Reddit stories are representative of the short, personal narratives found online in blogs or discussion forums, and so this work should be useful for finding MREs in a variety of online personal narratives. The one difference between this data and stories from other online sources is the prompt. A personal narrative posted to someone's personal blog is unlikely to have a prompt. We use the prompt for our heuristic labeling, so our automatic labels on non-Reddit data may be noisier, but many blog posts also have titles or tags that may be just as useful.

Identifying MREs is a hard problem that has not

previously been addressed in work on computational narrative. We have shown that the high-level view proposed by linguistic theories of narrative – that stories are about change – holds true. Measuring change over the course of a narrative yields better results than other features and baselines.

Why do we care about MREs? Polanyi (1976) asserts that "one does not produce a narrative text for no reason at all." The Most Reportable Event is that reason. It is the point of the story; the shortest possible summary; the answer to the question, "So what?". It could be used to generate titles or summaries to be used in organizing stories for readers to browse, or it could be used in recommendation systems to help readers find related stories. In future work we hope to be able to generate a text description the full MRE, which would be better suited to summarization or generating headlines, rather than identifying sentences that refer to it. We hope this work will encourage others to further investigate the Most Reportable Event.

Acknowledgments

This work was partially supported by NSF Contract No. IIS-1422863.

References

Apoorv Agarwal, Fadi Biadsy, and Kathleen McKeown. 2009. Contextual phrase-level polarity analysis using lexical affect scoring and syntactic n-grams. *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*.

Steven Bird, Edward Loper, and Ewan Klein. 2009. *Natural Langauge Processing with Python*. O'Reilly Media Inc., Sebastopol, CA.

Morton Botel and Alvin Granowsky. 1972. A formula for measuring syntactic complexity: A directional effort. *Elementary English*.

Eric Gilbert. 2013. Widespread underprovision on reddit. *Proceedings of the 2013 conference on Computer supported cooperative work*.

Alec Go, Richa Bhayani, and Lei Huang. 2009. Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford*.

Andrew S. Gordon, Qun Cao, and Reid Swanson. 2007. Automated Story Capture From Internet Weblogs. *Proceedings of the Fourth International Conference on Knowledge Capture*.

Andrew S. Gordon and Reid Swanson 2009. Identifying Personal Stories in Millions of Weblog Entries. *Third International Conference on Weblogs and Social Media, Data Challenge Workshop*.

Weiwei Guo and Mona Diab. 2012. Modeling sentences in the latent space. *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers – Volume 1*

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S. Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*.

Zhongqiang Huang and Mary Harper. 2009. Self-Training PCFG grammars with latent annotations across languages. *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2*.

Eric Jones, Travis Oliphant, Pearu Peterson, and others. 2001. SciPy: Open source scientific tools for Python. http://www.scipy.org/ Online; accessed 26 Jan. 2015.

Dan Klein and Christopher Manning. 2003. Accurate Unlexicalized Parsing. *Proceedings of the 41st Meeting of the Association for Computational Linguistics*.

Sebastian Krause, Hong Li, Hans Uszkoreit, and Feiyu Xu. 2012. Large-scale learning of relation-extraction rules with distant supervision from the web. *The Semantic WebISWC 2012*, 263-278. Springer Berlin Heidelberg, Berlin.

William Labov. 1997. Some further steps in narrative analysis. *Journal of Narrative and Life History*, 7:395-415.

William Labov. 2013. *The Language of Life and Death*. Cambridge University Press, Cambridge, UK.

William Labov and Joshua Waletzky. 1967. Narrative Analysis: Oral Versions of Personal Experience. *Essays on the Verbal and Visual Arts*, 12-44. June Helm (Ed.). University of Washington Press, Seattle, WA.

Annie Louis and Ani Nenkova. 2013. What Makes Writing Great? First Experiments on Article Quality Prediction in the Science Journalism Domain. *Transactions of ACL*.

David McClosky, Eugene Charniak, and Mark Johnson. 2006. Effective self-training for parsing. *Proceedings of the main conference on human language technology conference of the North American Chapter of the Association of Computational Linguistics*.

David McClosky and Eugene Charniak 2008. Self-training for biomedical parsing. *Proceedings of the*

46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Short Papers.

Neil McIntyre and Mirella Lapata. 2009. Learning to Tell Tales: A Data-driven Approach to Story Generation. *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*.

Rada Mihalcea. 2004. Co-training and self-training for word sense disambiguation. *Proceedings of the Conference on Computational Natural Language Learning (CoNLL-2004)*.

George Miller. 1995. WordNet: A Lexical Database for English. *Communications of the ACM* Vol. 38.

Bonan Min, Ralph Grishman, Li Wan, Chang Wang, and David Gondek. 2013. Distant Supervision for Relation Extraction with an Incomplete Knowledge Base. *Proceedings of NAACL-HLT 2013*.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*.

Truc-Vien T. Nguyen and Alessandro Moschitti. 2011. End-to-end relation extraction using distant supervision from external semantic repositories. *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers*.

Jessica Ouyang and Kathleen McKeown. 2014. Towards automatic detection of narrative structure. *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*.

Ellie Pavlick and Ani Nenkova. 2015. Inducing Lexical Style Properties for Paraphrase and Genre Differentiation. *Proceedings of NAACL-HLT 2015*.

Fabian Pedregosa, Gal Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel et al. 2011. Scikit-learn: Machine learning in Python. *The Journal of Machine Learning Research*, 12: 2825-2830.

Livia Polanyi. 1976. Why the Whats are When: Mutually Contextualizing Realms of Narratives. *Proceedings of the second Annual Meeting of the Berkeley Linguistic Society*.

Livia Polanyi. 1985. *Telling the American story : a structural and cultural analysis of conversational storytelling* Ablex Publishing, Norwood, NJ.

Gerald Prince. 1973. *A Grammar of Stories: An Introduction*. Mouton, The Hague.

Matthew Purver and Stuart Battersby. 2012. Experimenting with distant supervision for emotion classification. *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*.

Elahe Rahimtoroghi, Reid Swanson, Marilyn A. Walker, and Thomas Corcoran. 2013. Evaluation, Orientation, and Action in Interactive StoryTelling. *Proceedings of Intelligent Narrative Technologies 6*.

Roi Reichart and Ari Rappoport. 2007. Self-training for enhancement and domain adaptation of statistical parsers trained on small datasets. *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*.

Kenji Sagae. 2010. Self-training without reranking for parser domain adaptation and its impact on semantic role labeling. *Proceedings of the 2010 Workshop on Domain Adaptation for Natural Language Processing*.

Philipp Singer, Fabian Flck, Clemens Meinhart, Elias Zeitfogel, and Markus Strohmaier. 2014. Evolution of Reddit: From the Front Page of the Internet to a Self-referential Community? *Proceedings of the companion publication of the 23rd international conference on World wide web companion*.

Jared Suttles and Nancy Ide. 2013. Distant supervision for emotion classification with discrete binary values. *Computational Linguistics and Intelligent Text Processing*, 121-136. Springer Berlin Heidelberg, Berlin.

Reid Swanson, Elahe Rahimtoroghi, Thomas Corcoran and Marilyn A. Walker. 2014. Identifying Narrative Clause Types in Personal Stories. *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*.

Tim Weninger, Xihao Avi Zhu, and Jiawei Han. 2013. An Exploration of Discussion Threads in Social News Sites: A Case Study of the Reddit Community. *2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*.

Cynthia Whissell. 1989. The dictionary of affect in language. *Emotion: Theory, research, and experience*, 4:113-131. Academic Press, London.

Wei Xu, Raphael Hoffmann, Le Zhao, and Ralph Grishman. 2013. Filling Knowledge Base Gaps for Distant Supervision of Relation Extraction. *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*

Limin Yao, Sebastian Riedel, and Andrew McCallum. 2010. Collective cross-document relation extraction without labelled data. *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*.

Towards the Extraction of Customer-to-Customer Suggestions from Reviews

Sapna Negi
Insight Centre for Data Analytics
National University of Ireland Galway

Paul Buitelaar
Insight Centre for Data Analytics
National University of Ireland Galway

{firstname.lastname}@insight-centre.org

Abstract

State of the art in opinion mining mainly focuses on positive and negative sentiment summarisation of online customer reviews. We observe that reviewers tend to provide advice, recommendations and tips to the fellow customers on a variety of points of interest. In this work, we target the automatic detection of suggestion expressing sentences in customer reviews. This is a novel problem, and therefore to begin with, requires a well formed problem definition and benchmark dataset. This work provides a 3- fold contribution, namely, problem definition, benchmark dataset, and an approach for detection of suggestions for the customers. The problem is framed as a sentence classification problem, and a set of linguistically motivated features are proposed. Analysis of the nature of suggestions, and classification errors, highlight challenges and research opportunities associated with this problem.

1 Introduction

Opinion mining mainly deals with the summarisation of opinions on the basis of their sentiment polarities (Liu, 2012). However, on a closer observation of such opinionated text, we can discover other facets of opinions. For example, a hotel review sentence like, *Make sure you bring plenty of sun tan lotion-very expensive in the gift shop*, would be labeled as a neutral sentiment in current opinion mining methodologies, since it would only be interested in collecting opinions about the hotel. In the case of aspect based sentiment analysis, the sentence does not comprise of hotel related aspects, and thus will again be labeled as neutral/objective. While such sentences are generally ignored in sentiment based opinion summarisation, these can be very useful information to extract from the reviews. In hotel reviews, such suggestions range from tips and advice on the reviewed entity, to suggestions and recommendations about the neighbourhood, transportation, and things to do. Similarly, in product reviews suggestions can be about how to make a better use the product, accessories which go with them, or availability of better deals. We refer to such sentences as customer-to-customer suggestions (CTC).

Another type of suggestions, which can appear in the reviews, are the ones aiming at manufacturers or service providers, suggesting new features and improvements in products or services. For example, *An electric kettle in the room would have been a useful addition*. Recently, there have been some works on extracting the suggestions for improvements from reviews (section 3), but they did not focus on CTC suggestions. Also, suggestions for improvement discuss only about the reviewed entity and its aspects, unlike suggestions to customers, which might also include other topics of interest.

Suggestion mining and retrieval can be a potential new research area emerging from this kind of research. Industrial importance of suggestions to customers can be validated from the sections like 'Room Tips' (Figure 1) on TripAdvisor[1]. Similarly, Yelp also features 'tips' [2] (Figure 2) related to a business, and defines it as 'key information'. Such tips are often suggestions, or some important information, a user wants to convey to others. These tips are manually entered by the users, in addition to the reviews. We note that using suggestion mining, such information can be automatically extracted from a large number of already ex-

[1] http://www.tripadvisor.com/

[2] http://www.yelp-support.com/article/What-are-tips

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 2159–2167,
Lisbon, Portugal, 17-21 September 2015.

Figure 1: Room Tips on TripAdvisor

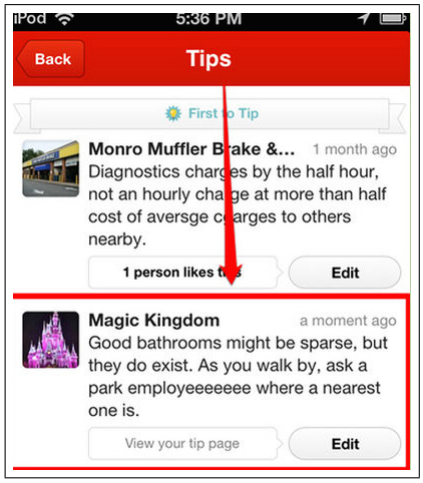


Figure 2: Tips on Yelp

isting reviews. The recommendation type of suggestions are of great importance in the case, when there is no dedicated reviews available for small shops, cafe, restaurants etc. in the vicinity of a hotel/business. Suggestions extracted from a large number of reviews, can also be seen as a kind of summarisation, an alternative or complementary to sentiment summarisation over the reviews.

In order to perform suggestion mining, expressions of suggestions should to be detected in a given text. The presence of a variety of linguistic strategies in the suggestion expressing sentences makes this task interesting from a computational linguistics perspective. Section 2 discusses this in more detail. The detection of suggestions in text goes beyond the scope of sentiment polarity detection, and opens up new problems and challenges in the areas of subjectivity analysis, social media analysis, and extra-propositional semantics.

Our main contributions through this work can be listed as:

- Proposition of the task of detection of CTC suggestions.
- A well formed problem definition and scope.
- Preparation of benchmark dataset for two domains of reviews, hotel and electronics.
- An approach to automatically detect CTC suggestions from review texts.

2 Task Definition

Since suggestion mining is a young problem, there is a need for problem analysis and definition. As indicated in the previous section, the tasks under suggestion mining may vary. Below, we propose three parameters whose value would help define such tasks, and the values for these parameters in the context of the current task of detection of CTC suggestions.

- **Who is the suggestion aimed at?**
  As we explained in section 1, suggestions can be aimed at one of the two kinds of stakeholders, customers and service providers. In this work, we perform the detection of suggestions for customers only.

- **What should be the textual unit of suggestion?**
  The previous works on extraction of suggestions for product improvement considered sentences as the unit of suggestions. In this work, we also consider sentences as units of suggestion. However, we observe that sentences might miss the context, or refer to something mentioned in the previous sentence. Furthermore, punctuation marks are often erroneously used in social media text, so automatic sentence split does not work well with such text. The approach used in this work aims at the detection of expressions of suggestions, regardless of the presence or absence of context in the same sentence. Therefore, we currently ignore the problems associated with using sentences as a unit for suggestions. The datasets used in this work are sentiment analysis review datasets from other works, in which reviews are already split into sentences. For future works, we assume that context can be determined by the neighbouring text once the expression of suggestion is successfully detected.

- **What kind of text should be considered as a suggestion?**
  Oxford dictionary defines suggestion as, *An idea or plan put forward for consideration*. Some of the listed synonyms of suggestions are *proposal, proposition, recommendation, advice, hint, tip, clue*. In a general scenario, this defini-

Sentence	Category	% Annotators	Suggestion Expression
Concierge is not available 24 hours	Inform	70	Implicit
We did not have breakfast at the hotel but opted to grab breads/pastries/coffee from the many food-outlets in the main stations.	Tell	80	Implicit
Room was very quiet despite being close to the elevator.	Remark	50	Implicit
The staff was nice and friendly	Praise	60	Implicit
Double bed quite narrow and not as comfortable as expected.	Disappoint	60	Implicit
The view from the 7th floor was amazing.	Fascinate	50	Implicit
If you do end up here, be sure to specify a room at the back of the hotel.	Suggest	100	Explicit
I recommend going for a Trabi Safari	Recommend	90	Explicit

Table 1: Sentences Which Were Perceived as CTC Suggestions by Laymen

tion of suggestion easily distinguishes suggestions from other kind of text. However, when suggestions are required to be identified in the space of customer reviews, there seems to be a tendency to consider most of the sentences as suggestions to other customers. This observation is based on a preliminary data annotation task, which was meant for data analysis and development of annotation guidelines. Asher et. al (2009) define 20 types of opinion expressions, including suggestions and recommendations, which appear in opinionated text. In order to form a representative sample, we chose 20 sentences from reviews, corresponding to each of these types, and asked 10 people (layman) to decide if these sentences are CTC suggestions or not. Table 1 shows sentences for which 5 or more people agreed of CTC Suggestion label. Asher et. al. (2009) observe that these types are not uniformly distributed in the reviews. According to them, suggestions and recommendations constitute about 10% of the statements, while judgements (blame, praise) and sentiments (love, fascinate, hate, disappoint, sad) constitute about 80%. Later, when we annotate the review datasets for CTC sugggestions, they also shows a smaller percentage of suggestions (see table 2).

Except the *suggest* and *recommend* categories, suggestions in rest of the categories are in an implicit form and need to be inferred. Since human beings can inherently infer suggestions, the layman annotators considered both implicit and explicit form of suggestions as suggestions. However, in a real case scenario, humans cannot go through the large amount of reviews and infer suggestions from all of them. In this work, we aim to automatically detect and extract the explicit expressions of suggestions, rather than inferring them.

For the ease of defining the scope of our work, we propose two form of suggestions:
Explicit: Directly suggests/recommends an entity or action,
Implicit: Only provides the information from which a suggestion can be inferred, but do not authoritatively suggest anything.

Lastly, we frame the problem of CTC suggestion detection problem as a sentence classification problem:
Given a set S of statements $\{s_1, s_2, s_3, ..., s_n\}$*, predict a label* l_i *for each of statement in S, where* $l_i \in \{$*CTC suggestion, non CTC suggestion*$\}$*, where CTC suggestion should be explicitly expressed.*

3 Related Work

Only a few attempts have been made to study suggestion mining, and there is an unavailability of benchmark datasets. Therefore, suggestion mining still remains a young area of study.

- **Suggestion Mining from Customer Reviews**
  As mentioned in section 1, there have been some attempts to extract suggestions for improvements in products from customer reviews. Ramanand et. al. (2010) used manually formulated patterns to extract wishes regarding improvements in products. Brun and Hagege (2013) also used manually formulated rules to extract suggestions for improvements from the product reviews. Negi and Buitelaar (2015) studied linguistic nature of suggestions and wishes for improvements and performed experiments in order to assert that these contain subjunctive mood. These works do not acknowledge the fact that reviews can also contain suggestions for other customers. One major drawback of previous works on customer reviews is the public unavailability of evaluation datasets.

- **Other Domains**
  Two other lines of work extracted suggestions from domains other than reviews. Dong et. al. (2013) performed detection of suggestions for product improvement using tweets. They used a statistical classifier, with features comprising of bag of words, automatically extracted suggestion templates using sequential pattern mining and hashtags. (Wicaksono and Myaeng, 2012; Wicaksono and Myaeng, 2013) extracted advice from discussion threads. They also used a supervised classification approach, where some of the features were domain dependent, like the similarity between original query post and a given sentence. They do not make any distinction between implicit and explicit advice, since there is less ambiguity in the domain of discussion forum.

None of the previous works study the complex and interesting nature of suggestions in opinionated text, and the relationship between suggestions and sentiments. Also, there is an unavailability of benchmark datasets of suggestions customer reviews.

4 Data

Since there is no available dataset of suggestions for customers, we prepare new datasets for this task. We consider two kinds of reviews for this task, hotel reviews and electronic product reviews.
Hotel: Wachsmuth et al. (2014) provide a large dataset of TripAdvisor hotel reviews, where reviews are segmented into statement so that each statement has only one sentiment label. Statements have been manually tagged with positive, negative, conflict and neutral sentiments. We take a smaller subset of these reviews, where each statement is an instance of our dataset. Each statement also bears a unique identity no., which is constituted of hotel identity no., statement number and review identity no. Therefore, the reviews belonging to same hotel, and the statements of the same review can be identified.
Electronics: Hu et. al.(2004) provide a dataset consisting of reviews of electronic products, which is also already split into sentences, and the corresponding sentiment for each sentence is manually tagged.
In the next section, we further annotate the sentences from these two datasets, for the current task.

Agreement	**Hotel(8050 sentences)**	**Electronics(3782 sentences)**
Phase 1		
Confidence	CTC Sugg.	CTC Sugg.
>= 0.6	3220	1488
>= 0.7	1046	604
>= 0.8	1024	562
>= 0.9	1020	558
1	1019	553
Phase 2		
kappa	CTC Sugg.(Explicit)	CTC Sugg.(Explicit)
0.86	407	273

Table 2: Statistics of Phase 1 and Phase 2 Annotations

4.1 Dataset Preparation

We performed a two phase annotation using both crowdsourced and expert annotations. This reduced the number of statements to be annotated by experts.
Phase 1 - Crowdsourced Annotations: The Crowdflower[3] platform was used for crowdsourced annotations. The platform provides a set of management and analytics tools for quality management, as well as for interaction with the annotators. In order to qualify for the annotation task, a Crowdflower worker was required to obtain a score of $7/10$ out of 10 test statements. The annotators were asked to choose one label out of 'Suggestion to Customers' or 'Other Statement' for each sentence. The definition of suggestion was left entirely to the understanding of the annotators. For each sentence, Crowdflower selects the answer with the highest confidence score[4]. We set the system not to seek more than 3 annotations for a statement if one of the labels attained a confidence score of 0.6 or more. At least 3, and at most 5 annotators labeled each statement. Confidence score for each label is the weighted sum of the trust scores of annotators who chose that label[5]. Trust score is determined by annotator's score in the test questions. Table 2 shows the variation in the number of statements tagged as suggestions with the corresponding confidence scores. These suggestions are a mixture of both implicit and explicit types, since the definition of suggestions was not restricted for the annotators. We observed that with the increase of confidence score, the ratio of explicit suggestions among the tagged suggestions increases.

[3] http://www.crowdflower.com/

[4] https://success.crowdflower.com/hc/en-us/articles/201855939-Get-Results-How-to-Calculate-a-Confidence-Score

[5] https://success.crowdflower.com/hc/en-us/articles/202703305-Getting-Started-Glossary-of-Terms

Phase 2 - Expert Annotations: Since we target the extraction of explicitly expressed suggestions, two expert annotators further classified the sentences which were finally labelled as 'Suggestion to Customers' by the Crowdflower platform, as explicit CTC suggestions and implicit CTC suggestions. Therefore, the number of sentences which experts annotated was much smaller than the total number of sentences in the two datasets. The key points of annotation guidelines for identifying the explicit suggestions are:

1. The intention of giving a suggestion and the suggested action or recommended entity should be explicitly stated in the sentence. For example:
   $[[\text{Try}]_{intention}[\text{the cup cakes at the bakery next door}]_{entity}]_{action}$.
   Other explicit forms of this suggestion could be: *I recommend the cup cakes at the bakery next door*, or, *You should definitely taste the cup cakes from the bakery next door*. Implicit form could be, *The cup cakes from the bakery next door were delicious.*

2. The suggestion should have the intention of benefiting the customers, and should not be mere sarcasm or joke. For example, *If the player doesn't work now, you can run it over with your car*.

A kappa value of 0.86 was calculated between the annotations performed by the expert annotators. The datasets from both phase 1 and phase 2 annotations are freely available for research. The final dataset has 3 kinds of labels, Implicit CTC, Explicit CTC, and others. Therefore, this dataset is also usable by the works which intend to extract implicit suggestions as well.

5 Suggestion Detection

We frame the task of CTC suggestion detection as a text classification problem. Our approach performs binary classification in a supervised fashion. Explicit CTC suggestions belong to the positive class, and rest of the sentences to the negative class.

Heuristic Features: A general notion about the task is that suggestions contain some distinctive keywords like, *suggest, recommend* etc, and should be easily detectable using them. Therefore, we use a set of manually selected features in order to test this notion.

- **Suggestion keywords:** These include the verbs: *suggestion, advise, request, ask, warn, recommend, do, do not*; their corresponding nouns: *suggestion, advice, request, warning, tip, recommendation*, and the synonyms of these words obtained from WordNet. Suggestion keywords constitute a single feature, which is binary in nature, and its value is determined by the presence of any of the suggestion keywords in the sentence.
- **POS tag *VB*:** Base form of Verb (VB) appears to be frequently used in the predicates of CTC suggestions.

Generic Features: Standard uni, bi, tri-grams, and uni, bi, tri-grams of Part of Speech tags (Penn Tree bank tagset). We consider the best performing set of Heuristics and generic feature types as the baseline for this task (see Table 3).

5.1 Special Features

We suggest a set of complementary features, which are motivated by the linguistic analysis of explicit CTC suggestions.

1. **Imperative Mood Sequential Patterns:** Imperative mood expressions are often present in explicit suggestions. Wicaksono et. al. (2013) also observed the presence of imperative mood in advice sentences. They used an imperative mood detector, based on a set of handmade rules to determine whether a sentence is imperative.
   In our case, the statements often contain more than one clause, and more than one mood expression. This feature aims to detect if any part of a given statement bears atleast one expression of imperative mood. We aim to identify sequential patterns of linguistic elements (POS tags in our case) which mark the imperative mood, and check if these patterns are present in a given statement. We automatically extract these features (patterns) using sequential pattern mining. We prepare a small dataset of 200 example sentences of imperative mood. These sentences were short sentence of lengths between 3-8 words, and are manually collected from websites related to English grammar, and linguistic research papers on mood and modality.
   Sequential Pattern Mining - Background: State of the art sequential pattern mining algorithms require the dataset to be converted into

Features	Hotel			Electronics		
	Precision	Recall	F score	Precision	Recall	F score
Heuristic	0.260	0.484	0.338	0.216	0.505	0.303
Unigrams	0.492	0.528	0.509	0.527	0.565	0.540
Uni,bi-grams	0.513	**0.577**	0.543	0.571	0.602	0.586
Uni, bi, tri-grams	0.519	0.545	0.532	0.562	**0.605**	0.570
uni, bi-grams + unigrams POS tags	0.539	0.555	**0.547**	0.634	0.565	**0.595**
uni, bi-grams + uni, bi-grams POS tags	0.568	0.491	0.527	**0.662**	0.515	0.575
uni, bi-grams + uni, bi, tri-grams POS tags	**0.593**	0.469	0.524	0.625	0.518	0.556

Table 3: Performance of Heuristic and Generic Features in a 10-Fold Cross Validation

Support	Sequence
0.69	$< VB >< NN >$
0.60	$< VB >< PRP >$
0.54	$< VB >< VB >< NN >$
0.51	$< RB >< VB >$
0.70	$< MD >< VB >$

Table 4: Sequential POS Patterns obtained from Imperative Mood Dataset

an ordered list of *events*. An event is a non-empty unordered collection of *items*, which in turn is the smallest unit of sequences. The output of these algorithms are sequential patterns, which comprise of sequences of items, and is defined to be frequent if its support (a measure of frequency) is equal or more than a user-defined threshold. $(a_1, a_2, ..., a_q)$ is a sequence, where a_i is an event. A sequence $(a_1, a_2, ..., a_n)$, is a subsequence of another sequence $(b_1, b_2, ...b_m)$, if there exist integers $i_1 < i_2 < ... < i_n$ such that $a_1 \subseteq b_{i1}$, $a_2 \subseteq b_{i2}, ..., a_n \subseteq b_{in}$. For example, given a sequence (AB,E,ACD), where B is an item and AB, E and ACD are events. (B,AC) is a subsequence, since B $\subseteq$ AB and AC $\subseteq$ ACD and the order of events is preserved. (AB,E) is not a subsequence of the sequence (ABE). In our case, each POS tag is an event, and all the events are of length 1 item.

We use Clospan (Yan et al., 2003) algorithm to obtain the patterns of POS tags appearing in imperative mood sentences. Closed pattern mining algorithms produce a significantly less number of sequences than the older methods, while preserving the same expressive power. We use 2-3 length sequences with a relative support greater than 0.5, where maximum relative support is 1 (see Table 4).

Sequential patterns and n-grams: Sequential patterns are different from continuous n-grams in the sense that these are similar to the templates with place-holders, or skip n-grams, where the items in a pattern are ordered but should not necessarily be immediate to each other.

The part of speech tags avoid sparse nature of word based patterns, and at the same time captures the language usage model of imperative sentences. However, the pos tags tend to repeat in longer texts, therefore we limit the number of placeholders which could appear between two items of a pattern to 2. Each pattern is treated as a separate feature, whose feature value is binary depending on its presence or absence in a given sentence.

2. **Sentiment Features:** Given that suggestions are being extracted from customer reviews, which are otherwise mostly used for sentiment analysis, a relation between sentiments and suggestions can be suspected. It can be observed From the Figure 3, 4 suggestions do not seem to always carry one particular sentiment, but different sentiments at different instances. We compare three types of sentiment related features:
a) Manually tagged sentiments: These annotations were provided with the used sentiment analysis datasets.
b) Sentiwordnet score summation: SentiWordNet (Esuli and Sebastiani, 2006) sentiment score summation of all the words in a sentence. No sense disambiguation is performed, and all synset scores are summed up for each word.
c) Normalised sentiwordnet score summation: These scores are the sum of all the sentiment scores of the words in a given sentence, normalised over the number of words carrying non-neutral sentiment score.

3. **Information about the subject/s of a statement:** This feature captures the presence of nsubj dependency (Marneffe and Manning,

Features	Hotel				Electronics			
	Precision	Recall	F score	AUC	Precision	Recall	F score	AUC
Baseline (best generic features)	0.539	0.555	0.547	0.763	0.634	0.565	0.595	0.817
+ patterns	0.542	0.511	0.526	0.743	0.607	0.604	0.616	0.790
+ sentiments (manual)	0.529	0.536	0.532	0.754	0.581	**0.630**	0.605	0.797
+ sentiments (score)	0.537	0.541	0.539	0.757	0.563	0.593	0.578	0.779
+ sentiments (normalised score)	0.543	**0.561**	0.550	0.774	0.645	0.593	0.618	0.784
+ nsubj	0.559	0.538	0.548	0.757	0.597	0.586	0.591	0.778
Baseline + special (all)	**0.580**	0.512	**0.567**	**0.781**	**0.645**	0.621	**0.640**	**0.790**

Table 5: Performance of Special Features in a 10 Fold Cross Validation

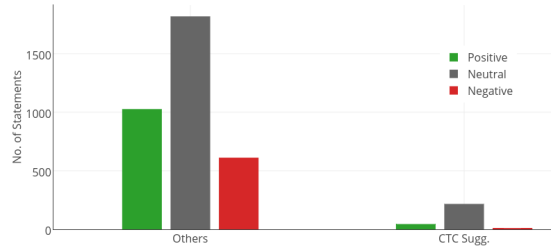


Figure 3: Manually labelled Sentiment Distribution of Electronics Dataset

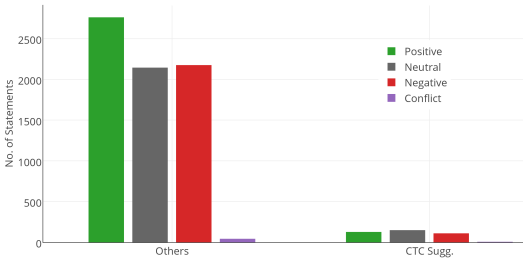


Figure 4: Manually labelled Sentiment Distribution of Hotel Dataset

2008), and if present, the pair of POS tags of the arguments of this dependency. Often a reviewer addresses the reader when giving a suggestion. For the sentence, *If you do end up here, be sure to specify a room at the back of the hotel*, the nsubj dependency is nsubj(do, you). The feature value would be *VBP-PRP* in this case. On the other hand, this suggestion could also have been, *Be sure to specify a room at the back of the hotel.*. In this case the feature value would be *null*. If more than one nsubj dependency is present, the POS pair of each of them will be included in the feature value.

Experimental Setup: We use the Stanford Parser (Toutanova et al., 2003) for obtaining part of speech and dependency information. Stemming did not effect the results. Stopwords were used using a customised stopword list. We employ the LibSVM library (Chang and Lin, 2011) for Support Vector Machine classifiers (SVM), as implemented in Weka machine learning toolkit (Hall et al., 2009). The parameter values of SVM classifiers are: SVM type = C-SVC, Kernel Function = Radial Basis Function. A 10 fold cross validation is performed in order to evaluate the classifier model. Features are ranked using the feature selection algorithm InfoGain (Mitchell, 1997). The other attribute selection algorithms provided with the Weka toolkit were also experimented with, but InfoGain consistently performed best in all the runs. Only the features which posses a positive information gain value are retained. The best performing run has a feature vector of size 300. The problem of having a imbalanced dataset is handled by using a higher class weight of 5:1 (Akbani et al., 2004) for positive class.

6 Results and Discussion

We evaluate the proposed features using 10-fold cross validation. As indicated in section 5, we consider best performing set of generic features as the baseline, which is: uni, bi grams and unigrams of pos tags. Table 5 summarises the classifier performance with the addition of special features, measured in Precision, Recall, F1 score, and ROC Area Under Curve. The results indicate that special features improve the F score in both the domains. However, some of the generic features produced better precision and recall values (Table 3). Imperative patterns improve the baseline results for electronics dataset, but not for the hotel dataset. Also, the special features produce better improvement over baseline for electronics dataset. We attribute this to the smaller size of electronics dataset.

Table 7 shows how the top ranked features change on addition of special features. Heuristic features tend to appear as some of the top ranked generic features. On the addition of special features, some of the special features replace the top general features. This validates the importance of proposed special features. Normalised sentiment

#	Sentence	Classifier	Human
1	Buy this for the storage and price , avoid it if you know nothing about computers.	1	1
2	whichever camera you buy, add upto about 200 dollars for an additional memory card i bought a 256 mb card, usb card reader,camera bag and the warranty.	0	1
3	Looks sort of like picasa software (google it if you dont know) in the interface and is as easy to install and operate as g2	0	1
4	If you cant make an Italian meal don't advertise you can.	1	0
5	It would have been better to have some sort of window on the carrying case , so you could see the display without opening it	1	0
6	If you have a lot of money to waste, make sure you book this hotel	1	0

Table 6: Error Analysis of the Best Performing Feature Set (0 = Negative Class, 1 = Positive Class)

Type	Domain	Top Features
Baseline	Hotel	you, VB, if you, recommend, if, want, you are, highly recommend, I would
	Elec.	you, VB, recommend, if you, if, buy, recommend this, we, I highly recommend, don't, get
Baseline +Special	Hotel	you, VB, $< VB, NN >$, VBD, $< VB, VB, NN >$, recommend, if you, root(Root,VB), norm. sentiment score, suggest
	Elec.	you, VB, $< VB, NN >$, recommend, if you, nsubj, buy, suggest, $< VB, PRP >$, norm. sentiment score

Table 7: Some of the Top Ranked Features in Different Runs

score prove to be better features than the other two types of sentiment features including the manually labelled sentiments. This indicates that this method of sentiment calculation is capturing some universally used phrasing for suggestions, where real sentiment of the sentence fails to capture it.

Observed Challenges: Table 6 shows some instances of Type 1 and Type 2 errors in the best performing feature set. Our error analysis reveals the challenges associated with this task.

1. Non-CTC Suggestions: Example #4 gives a suggestion for the improvement of the hotel restaurant. Similarly, #5 is a suggestion to the product manufacturer instead of fellow customers. These kind of suggestions at times possess features similar to CTC suggestions.
2. Complex sentences: Often, suggestion is only expressed in one part of a very long sentence (#2,#3,). This might generate erroneous rank for features; also part of speech taggers tend to perform poorly for such sentences.
3. Sarcasm: The surface form of #6 is a suggestion, but it is a sarcasm.
4. Biased Datasets: Explicit CTC suggestion expressing sentences occur sparsely, which is unfavourable for supervised learning approaches because of increased cost of data annotation task, and imbalanced class representation.

A general observation is that the text in the form of suggestions may not always be a suggestion, and vice versa. Therefore, syntactic and lexical features seem to be ineffective in many cases.

7 Conclusion

This work serves as an introduction and analysis of the problem of the extraction of customer-to-customer suggestions from reviews. The task is useful for a number of practical applications. We observed that the layman perception of *suggestion* is very wide, especially in the case of reviews. Therefore, we defined and limited the scope of our work to explicit suggestions. Following this, we prepared a well-investigated benchmark dataset, which is freely available for research purposes.

A pilot approach to the problem is presented, which analyses the performance of standard text classification features, and tests a set of complementary/special features. The special features improved the baseline results for both service and product reviews.

Analysis of classification errors highlighted the challenges associated with the task. The classification results have scope for improvement, and therefore the task calls for advanced semantic features and dedicated models, which will be our future direction. Furthermore, the relation between sentiments and suggestions seem to be worth investigating.

Acknowledgement

This work has been funded by the European Unions Horizon 2020 programme under grant agreement No 644632 MixedEmotions, and the Science Foundation Ireland under Grant Number SFI/12/RC/2289 (Insight Center).

References

Rehan Akbani, Stephen Kwek, and Nathalie Japkowicz. 2004. Applying support vector machines to imbalanced datasets. In *Machine Learning: ECML 2004*, Lecture Notes in Computer Science, pages 39–50. Springer Berlin Heidelberg.

Nicholas Asher, Farah Benamara, and Yannick Mathieu. 2009. Appraisal of Opinion Expressions in Discourse. *Lingvistic Investigationes*, 31.2:279–292.

Caroline Brun and Caroline Hagege. 2013. Suggestion mining: Detecting suggestions for improvement in users comments. *Research in Computing Science*.

Chih-Chung Chang and Chih-Jen Lin. 2011. Libsvm: A library for support vector machines. *ACM Trans. Intell. Syst. Technol.*, 2(3):27:1–27:27.

Li Dong, Furu Wei, Yajuan Duan, Xiaohua Liu, Ming Zhou, and Ke Xu. 2013. The automated acquisition of suggestions from tweets. In *AAAI*. AAAI Press.

Andrea Esuli and Fabrizio Sebastiani. 2006. Sentiwordnet: A publicly available lexical resource for opinion mining. In *In Proceedings of the 5th Conference on Language Resources and Evaluation (LREC06*, pages 417–422.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. 2009. The weka data mining software, an update. *SIGKDD Explorations*, 11:10–18.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '04, pages 168–177, New York, NY, USA. ACM.

Bing Liu. 2012. Sentiment analysis and opinion mining. *Synthesis Lectures on Human Language Technologies*, 5(1):1–167.

Marie-Catherine De Marneffe and Christopher D. Manning. 2008. Stanford typed dependencies manual.

Thomas M. Mitchell. 1997. *Machine Learning*. McGraw-Hill, Inc., New York, NY, USA, 1 edition.

Sapna Negi and Paul Buitelaar. 2015. Curse or boon? presence of subjunctive mood in opinionated text. In *Proceedings of the 11th International Conference on Computational Semantics*, pages 101–106, London, UK, April. Association for Computational Linguistics.

Janardhanan Ramanand, Krishna Bhavsar, and Niranjan Pedanekar. 2010. Wishful thinking - finding suggestions and 'buy' wishes from product reviews. In *Proceedings of the NAACL HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text*, pages 54–61, Los Angeles, CA, June. Association for Computational Linguistics.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *In Proceedings of HLT-NAACL 2003*, pages 252–259.

Henning Wachsmuth, Martin Trenkmann, Benno Stein, Gregor Engels, and Tsvetomira Palakarska. 2014. A review corpus for argumentation analysis. In *Proceedings of the 15th International Conference on Computational Linguistics and Intelligent Text Processing*, volume 8404 of *LNCS*, pages 115–127, Kathmandu, Nepal. Springer.

Alfan Farizki Wicaksono and Sung-Hyon Myaeng. 2012. Mining advices from weblogs. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, CIKM '12, pages 2347–2350, New York, NY, USA. ACM.

Alfan Farizki Wicaksono and Sung-Hyon Myaeng. 2013. Automatic extraction of advice-revealing sentences foradvice mining from online forums. In *K-CAP*, pages 97–104. ACM.

Xifeng Yan, Jiawei Han, and Ramin Afshar. 2003. Clospan: Mining closed sequential patterns in large datasets. In *In SDM*, pages 166–177.

Using Content-level Structures for Summarizing Microblog Repost Trees *

Jing Li[1,2], **Wei Gao**[3], **Zhongyu Wei**[4], **Baolin Peng**[1,2] and **Kam-Fai Wong**[1,2]
[1]The Chinese University of Hong Kong, Shatin, N.T., Hong Kong
[2]MoE Key Laboratory of High Confidence Software Technologies, China
[3]Qatar Computing Research Institute, Hamad Bin Khalifa University, Doha, Qatar
[4]The University of Texas at Dallas, Richardson, Texas, USA
{lijing,blpeng,kfwong}@se.cuhk.edu.hk[1,2]
wgao@qf.org.qa[3], zywei@@hlt.utdallas.edu[4]

Abstract

A microblog repost tree provides strong clues on how an event described therein develops. To help social media users capture the main clues of events on microblogging sites, we propose a novel repost tree summarization framework by effectively differentiating two kinds of messages on repost trees called leaders and followers, which are derived from content-level structure information, i.e., contents of messages and the reposting relations. To this end, Conditional Random Fields (CRF) model is used to detect leaders across repost tree paths. We then present a variant of random-walk-based summarization model to rank and select salient messages based on the result of leader detection. To reduce the error propagation cascaded from leader detection, we improve the framework by enhancing the random walk with adjustment steps for sampling from leader probabilities given all the reposting messages. For evaluation, we construct two annotated corpora, one for leader detection, and the other for repost tree summarization. Experimental results confirm the effectiveness of our method.

1 Introduction

Microblogging platforms have become the center for reporting, discussing, and disseminating real-life issues, on which users usually *repost* to share microblog messages with their following users. Also, users can *repost with commentary* for not only further broadcasting but also extending the original microblog post content. Because an individual post is generally too short to cover the main clues of an event, microblogging users cannot easily capture the key information from received posts due to the lack of context. And reposting messages, namely reposts, can provide valuable context information to the previous posts including their background, development, public opinions and so on. However, a popular post usually attracts a large number of reposts. It is impractical for users to read them all and fully understand their contents.

The task of microblog context summarization aims to produce succinct summaries to help users better understand the main clues by extracting salient information among massive reposts of the original posts. An intuitive approach is to directly apply existing extractive summarizers based on the unstructured, plain microblog contents. But such short and informal reposts render the lack of structures in each individual message, and it is difficult for conventional extractive summarizersto identify salient messages. Chang et al. (2013) proposed to summarize Twitter context trees by focusing on modeling user influence. However, the reposts of influential users might not be salient summary candidates necessarily. For instance, celebrities might simply repost with nothing important. Also, modeling user influence accurately needs tremendous historical user interaction data external to the tree being summarized while such kind of information cannot be utilized directly for summarizing the messages on the tree.

In this paper, we propose a novel microblog context summarization framework based on content-level structures, i.e., message contents and reposting relations, rather than user-level influence signals. The reposting relations connect the reposting messages and form a cohesive body as a tree structure named *repost tree*. The root represents the original post and the edges denote re-

*This work is partially supported by General Research Fund of Hong Kong (417112), RGC Direct Grant (417613), and Huawei Noah's Ark Lab, Hong Kong. We would like to thank anonymous reviewers for the useful comments.

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 2168–2178,
Lisbon, Portugal, 17-21 September 2015.

posting relations. Our idea is to exploit the structure of repost tree together with content of messages to help distinguish two different *messages* on repost tree, i.e., leaders and followers. Specifically, **leader** is referred to as a *message* on repost tree covering salient new information, which can lead further comments or discussions in its descendant reposts; **follower** is referred to as a *message* that contains no comment, simply repeats or naively responds to its ancestor leader message, thus providing no important information. The example below illustrates a repost tree path, where we use $[O]$ and $[R_i]$ to indicate the original post and the i-th repost, respectively:

$[O]$ @MAS: Malaysia Airlines has lost contact of MH17 from Amsterdam. The last known position was over Ukrainian airspace.

$[R_1]$ @Hanna: OMG... Poor on MH17... Preying...

$[R_2]$ @Victoria: OMG that's horrible!!! I'm sorry to hear that. God will bless u poor guys. Wish world can be peaceful. And no one will get hurt.

$[R_3]$ @Dr.Dr: Six top HIV scientists are on MH17. They go for AIDS and would NEVER come back!!!

$[R_4]$ @TomyBlack: 6 experts died?! Terrible loss to HIV research :(

$[O]$ reports the news about MH17 missing, which brings about further comments in $[R_1]$ and $[R_2]$. $[R_3]$ does not continue commenting on that but offers some new information and triggers shocking reaction in $[R_4]$. So $[O]$ and $[R_3]$ act as leaders; $[R_1]$, $[R_2]$ and $[R_4]$ are followers.

Intuitively, leaders would be more important than followers from the summarization's perspective since leaders are supposed to capture the main clues or aspects of event evolvement. The first step of our summarization system is to distinguish leaders and followers effectively. Leaders are detected across repost tree paths which provide rich context information owing to the tree structure. We utilize sequence tagging model Conditional Random Fields (CRF) to infer how likely it is each repost being a leader or follower. Then we incorporate leader detection result into an unsupervised summarization model based on random walk. Our model uses content similarities between messages and consider their possibilities of being leaders to rank and select salient reposting messages that form summaries. Furthermore, we improve the framework by enhancing the random walk to reduce the impact of errors cascaded from the leader detection module. Compared to the state-of-the-art baselines, the experimental results confirm the effectiveness of our proposed framework.

Our contributions are given as follows:

- We propose a novel microblog context summarization framework, in which given reposting messages organized as a repost tree (obtaining repost tree is trivial using public microblogging toolkit (Ren et al., 2014)), we summarize the repost trees based on content information and reposting relations of messages.
- We identify a novel problem of leader detection for summarization, which aims to reduce noise on repost trees, and present a CRF-based method for effectively detecting leaders by utilizing the tree structure and message contents.
- We incorporate the leader detection result into an unsupervised summarization model based on random walk and substantially enhance the model to reduce the impact of leader detection errors on summarization.

2 Related Work

The goal of text summarization is to automatically produce a succinct summary for one or more documents that preserves important information (Radev et al., 2002). Generally, text summarization techniques can be categorized into extractive and abstractive methods (Das and Martins, 2007). Extractive approaches focus on how to identify and distill salient contents from original texts whereas abstractive approaches aim at producing grammatical summaries by text generation.

Recently, the development of social media has made microblog summarization a hot topic. Most prior works are on event-level or topic-level summarization. Typically, the first step is to cluster posts into sub-events (Chakrabarti and Punera, 2011; Duan et al., 2012; Shen et al., 2013) or subtopics (Long et al., 2011; Rosa et al., 2011; Meng et al., 2012), and then the second step generates summary for each cluster.

Some works tried to apply conventional extractive summarization models directly, e.g., LexRank (Erkan and Radev, 2004), MEAD (Radev et al., 2004), TF-IDF (Inouye and Kalita, 2011), Integer Linear Programming (Liu et al., 2011; Takamura et al., 2011), etc. Sharif et al. (2010) modeled the problem as optimal path finding on a phrase reinforcement graph. However, these general summarizers were found not suitable for microblog posts, which are informal and noisy (Chang et al., 2013). Researchers

then considered social signals like user following relations and retweet count (Duan et al., 2012; Liu et al., 2012), and reported such features useful to help summarize microblog posts. Our work studies repost tree summarization by leveraging content-level structure to enrich context of messages, which is a different kind of signal.

Chang et al. (2013) proposed a task to summarize Twitter context trees consisting of an original tweet and all its reposts (i.e., replies and retweets). They combined user influence signals into a supervised summarization framework. Our work is different from theirs: 1) They simply treat a context tree as a tweets stream while we consider repost tree structures in summarization; 2) They rely on user interactions to calculate *user influence* for extracting salient messages while we focus on how to utilize contents and repost tree structures to differentiate *leader and follower messages* for summarization; 3) Our summarization module is unsupervised, thus no need of ground-truth summaries.

3 Leader Detection Model

This section deals with how to differentiate leader and follower messages on a repost tree. Intuitively, identifying leaders effectively makes one step closer to obtaining a good summary.

Figure 1 illustrates an example of a repost tree[1]. As shown in the figure, a leader message contains contents that brings essential information increase, such as a new clue about MH17 reported in $[R_6]$, and potentially triggers a new round of information propagation by attracting follower messages to focus on the raised clue, like $[R_7]$, $[R_8]$ and $[R_9]$. As the repost tree grows, it also happens that some new reposts join in, following the clue raised by one of their ancestors, but further extend it by mentioning something new, thus some of these messages may evolve into new leaders, such as $[R_{10}]$.

A simple way to detect leaders on repost tree is to directly apply a binary classifier like SVM on each individual message. However, these models assume reposts are independent without effectively leveraging abundant context along the repost tree paths, such as the reposting relations among different reposts on a path. For instance, $[R_2]$ covering rich content may be misclassified as a leader if not leveraging context information. But

[1]The example in Section 1 actually denotes the left-most path extracted from this tree

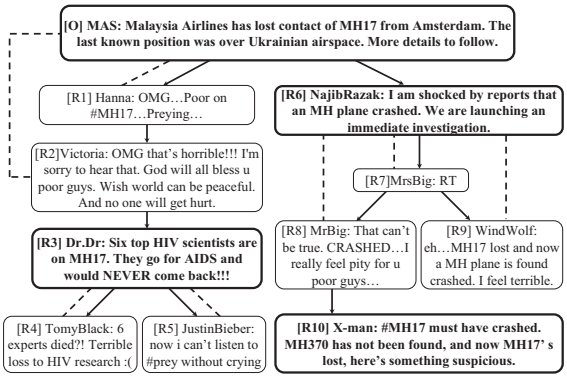


Figure 1: An example of repost tree. $[O]$: the original post; $[R_i]$: the i-th repost; Solid arrow lines: reposting relationship; Dotted lines: hidden leader-follower relationship; Dark boxes: leaders to be detected.

if we look into its context, we can find that $[R_2]$ talks about similar things as $[R_1]$, then $[R_1]$ classified as a follower indicates the higher chance of $[R_2]$ being a follower rather than a leader. Therefore, context information is important for indicating the messages being leaders or followers.

We extract all root-to-leaf paths within a repost tree structure and detect leaders across each path. We formulate leader detection on repost tree paths as a sequence tagging problem by utilizing a state-of-the-art sequence learning model CRF (Lafferty et al., 2001), and taking advantage of its power in maximizing the likelihood of global label sequences. We adopt CRF rather than other competitive context sensitive model like SVM^{hmm} (Altun et al., 2003) due to its probabilistic nature. The probability of prediction by CRF can provide critical chances for the following summarization procedure to reduce the impact of errors made by leader detection model on summarization (see Section 4.2).

We map a repost tree path with n microblogs $(m_1, m_2, \cdots, m_n)$ to a training instance (X, Y). Let $X = (x_1, x_2, \cdots, x_n)$ represents observed sequence, where x_i denotes the observed feature vector extracted from the i-th microblog m_i, and $Y = (y_1, y_2, \cdots, y_n)$ where y_i is the label indicating whether m_i is a leader or not. CRF defines the discriminative function as a joint distribution over Y given X as follows:

$$P(Y|X;\theta) \propto \exp\left(\sum_{i,j}\lambda_j f_j(y_i, y_{i-1}, X) + \sum_{i,k}\mu_k g_k(y_i, X)\right)$$

where f_j and g_k are the fixed feature functions,

Feature category	Feature name	Feature description
Lexical	# of terms	The number of terms in m_i
	POS	The part-of-speech of each term in m_i
	Type of sentence	Whether m_i contains a question mark or an exclamation
Microblog-specific	# of emoticons	The number of emoticons in m_i
	# of hashtags	The number of hashtags in m_i
	# of urls	The number of URLs in m_i
	# of mentions	The number of mentions, or @UserName, in m_i
Path-specific	Similarity to neighbors	Cosine similarity between m_i and m_{i+d} where $d \in \{\pm 1, \pm 2, \pm 3\}$
	Similarity to root	Cosine similarity to the root microblog in repost tree path

Table 1: Features used for leader detection

$\theta = (\lambda_1, \lambda_2, ...; \mu_1, \mu_2, ...)$ are the parameters indicating the weights of features that can be estimated by maximum likelihood procedure in training process. The prediction is done based on dynamic programming. More details can be found in (Lafferty et al., 2001). Table 1 lists the features we use for leader detection.

CRF can utilize both historical and future information for prediction so as to maximize the likelihood of the global label sequences. But we would encounter the problem of label conflict, i.e., the predictions for the same repost in context of different paths might be different. For this reason, we determine a repost as a leader if its average marginal probabilities being a leader in context of different paths exceeds 50%.

4 LeadSum Summarization Model

Let $T = (V, E)$ represent a repost tree to be summarized, where V is a set of nodes corresponding to microblog messages, and $E = \{(u, v)|v \text{ is the repost of } u\}$ is the edge set denoting reposting relations. This section describes how to rank nodes in V to produce repost tree summaries. Enlightened by the general random-walk-based ranking algorithm DivRank (Mei et al., 2010), we propose an unsupervised summarization model called LeadSum that aims to select true and salient leaders into summaries utilizing a variant of random walk based on content similarities and reposting relations of messages. We first present a basic LeadSum model, which assumes leader detection is perfect. Then, we enhance it to become a soft LeadSum model that reduces the impact of leader detection errors on the summarization.

4.1 Basic-LeadSum Model

Due to the nature of leaders, they generally cover more important contents than follows do. Thus our first summarizer selects contents only from detected leaders. For the leaders detected in a repost tree T, we build a similarity graph among leaders denoted as $G_L = (V_L, E_L)$, where $V_L = \{v \in V|v \text{ is a detected leader}\}$ is the vertex set and $E_L = \{(u, v)|u \in V_L, v \in V_L, \text{ and } u \neq v\}$ is the edge set. The weight for any edge (u, v) represents the content similarity between u and v, for which we use cosine similarity.

DivRank (Mei et al., 2010) is a generic graph ranking model that aims to balance high information coverage and low redundancy in top ranking vertices, which are also two key requirements for choosing salient summarization sentences (Li et al., 2009; Liu et al., 2015). Based on that, we present a model to rank and select salient messages from leader set V_L to form a summary. Since this model simply assumes perfect leader detection, it is therefore named Basic-LeadSum.

Similar as DivRank (Mei et al., 2010), the transition probability at the t-th iteration of random walk is given as follows:

$$p_t(u \to v) = (1-\mu) \cdot p_0(v) + \mu \cdot \frac{p_0(u \to v) N_{t-1}(v)}{Z(u)} \quad (1)$$

and $Z(u)$ is the normalizing factor:

$$Z(u) = \sum_{w \in V_L} p_0(u \to w) N_{t-1}(w) \quad (2)$$

where $p_0(u \to v)$ is the organic transition probability which represents the content similarity between u and v; $N_{t-1}(v)$ denotes the times vertex v is visited up to the $(t-1)$-th iteration; $p_0(v) = \frac{1}{|V_L|}$ denotes random jumping probability similar to that in PageRank; and μ is the damping weight set as 0.85 following most PageRank-based models. The probability of traveling to leader v can accumulate as its weight increases during random walk, and leaders already having high weight can "absorb" weights from other leaders with high similarity to it, thus avoids redundancy.

For any $v \in V_L$, the update function for its ranking score at the t-th iteration $R_t(v)$ is formulated as:

$$R_t(v) = \sum_{u \in V_L} p_t(u \to v) R_{t-1}(u) \tag{3}$$

It has been proved that the Markov chain is ergodic, thus can converge to a stationary distribution (Mei et al., 2010), which determines the final rankings for leaders.

4.2 Soft-LeadSum Model

As a two-step summarization system, the performance of LeadSum relies on the leader detection, which might be error-prone. Followers misidentified as leaders participating in leader ranking brings risks to extract real followers into summary. Also, leaders misclassified as followers may leave out strong summary candidates. To reduce such error propagation, we enhance Basic-LeadSum by using an even-length random walk with adjustment steps that sample from leader probabilities given all the reposting messages, which is referred to as Soft-LeadSum.

Different from Basic-LeadSum, every message on repost tree T, no matter detected as a leader or a follower, participates in ranking process of Soft-LeadSum. In other words, in the random walk, visitor wanders on a complete graph $G = (V, E')$ whose vertex set V is identical to repost tree T, and $E' = \{(u, v) | u \in V, v \in V, \text{ and } u \neq v\}$ represents the edge set. Therefore, this makes it possible to include true leaders misclassified as followers by leader detection module into summary.

However, allowing all messages to participate in ranking also increases the risk of selecting real followers. To avoid this problem, Soft-LeadSum is composed of two types of walks on G, namely WALK-1 and WALK-2. In WALK-1, visitor moves based on content similarities between messages, which follows transition probabilities similar to equation (1), but is specifically given as:

$$p_t(u \to v) = (1 - \mu) \cdot \frac{1}{|V|} + \mu \cdot \frac{p_0(u \to v) N_{t-1}(v)}{Z(u)} \tag{4}$$

where $u, v \in V$, $p_0(u \to v)$ is proportional to content similarity between u and v similar to Basic-LeadSum, and $Z(u)$ is the normalizing factor.

WALK-2 attempts to avoid selecting true followers by adopting a sampling process, whose result determines the next vertex on G to be visited. Suppose the current vertex being visited is u, then we sample from $p_L(u)$, i.e., the probability of u being a leader. Practically, $p_L(u)$ can be estimated with the average of u's marginal probabilities as a leader over all root-to-leaf paths passing u on T output by the leader detection module. If u is sampled to be a leader, we claim that leader detection is correct and the visitor stays; otherwise, u is sampled as a follower, indicating that leader detection module misclassifies u, so the visitor should go to u's leader. Here we assume that a follower u's leader is its nearest ancestor leader on T as shown by the dotted lines in Figure 1. Based on such simplification, we let the visitor trace back one by one along the path on T from u to root and sample from their leader probabilities until a node v is sampled as a leader and then we determine v as u's leader.

So for any u's ancestor v, the probability of v being u's leader is:

$$\begin{aligned} &Pr\{v \text{ is } u\text{'s leader}\} \\ &= p_L(v)(1 - p_L(u) - \sum_{w \in \mathcal{P}(v,u)} Pr\{w \text{ is } u\text{'s leader}\}) \\ &= p_L(v) \prod_{w \in \mathcal{P}(v,u) \bigcup \{u\}} (1 - p_L(w)) \end{aligned} \tag{5}$$

where $\mathcal{P}(v, u)$ is the set of nodes between v and u on v-to-u path of repost tree, i.e., $\mathcal{P}(v, u) = \{w \in V | w \text{ is } v\text{'s descendant and } u\text{'s ancestor on } T\}$. In particular, we assume that $p_L(r) = 1$ so as to stop the sampling process when the visitor arrives at root r.

Therefore, WALK-2's transition probabilities can be calculated as follows:

$$q(u \to v) = \begin{cases} p_L(v) & \text{if } v = u; \\ Pr\{v \text{ is } u\text{'s leader}\} & \text{if } v \text{ is } u\text{'s ancestor}; \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

Algorithm 1 shows the ranking process of Soft-LeadSum, during which the visitor walks on G following WALK-1 and WALK-2 alternately. The fact that WALK-1 is ergodic ensures the ergodicity and convergency of the algorithm. In implementation, we set max iteration N=1000 empirically which is large enough to ensure convergence, or stop random walk process in advance when the condition of convergence is met, i.e., the change of Euclidean difference of ranking scores for three consecutive iterations are all less than 1e-6.

Soft-LeadSum can reduce the impact of errors made by leader detection on summarization due to the following two reasons: 1) It allows all messages to participate in ranking process, thus permits those leaders leaving out by leader detection module to be selected into summary; 2) With

Algorithm 1 Algorithm of Soft-LeadSum

Input: T, G, μ=0.85, max iteration N, length cut-off n
Output: Summary with n microblog messages

For all $v \in V$, initialize $R_0(v) = p_0(v) = \frac{1}{|V|}$
Initialize WALK-1's transition probabilities $p_0(u \rightarrow v)$ with normalized cosine similarity between u and v.
Calculate WALK-2's transition probabilities $q(u \rightarrow v)$ by equation (5) and (6).
Initialize current_walk="WALK-1"
for $t = 1$ to N and not converged **do**
  for all $v \in V$ **do**
    if current_walk=="WALK-1" **then**
      Update $p_t(u \rightarrow v)$ by equation (4)
      Update $R_t(v)$ as follows:
        $R_t(v) = \sum_{u \in V} R_{t-1}(u) \cdot p_t(u \rightarrow v)$
      Set current_walk="WALK-2"
    end if
    if current_walk=="WALK-2" **then**
      Update $R_v(v)$ as follows:
        $R_t(v) = \sum_{u \in V} R_{t-1}(u) \cdot q(u \rightarrow v)$
      Set current_walk="WALK-1"
    end if
  end for
end for
Sort all $v \in V$ by $R_N(v)$ in descending order
Pick the top-n messages as summary

WALK-2 sampling from leader probabilities, it also reduces the risk of including real followers into summary.

5 Experiments and Results

To evaluate the two modules in our repost tree summarization system, i.e., CRF-based model for leader detection and LeadSum model for summarization, we conducted two sets of experiments based on microblog posts data collected from Sina Weibo, which has a similar market penetration as Twitter (Rapoza, 2011)[2]. Microblog messages on Sina Weibo are in Chinese and we use FudanNLP (Qiu et al., 2013) for text preprocessing including word segmentation and POS tagging.

5.1 Experiment for Leader Detection

In this experiment, we evaluated the performance of CRF model for leader detection task.

5.1.1 Data Collection and Setup

We first crawled 1,300 different repost trees using the public PKUVIS toolkit (Ren et al., 2014). Given an original microblog post, the toolkit can automatically crawl its complete repost tree. For each tree, we randomly selected one path and further formed a set with 1,300 repost tree paths, which ensures that paths have different roots and the dataset can cover a wide variety of context information.

Then three annotators were invited to label each repost as a leader or a follower in the context of its repost tree path independently. The average Cohen's Kappa of each two of the three annotators was 0.52, which is considered good agreement (Fleiss et al., 2013). Then, we used the labels agreed by at least two annotators as the ground truth. The training and test of the leader detection models were conducted on this corpus.

We compared the performance of CRF-based leader detection model with four baselines: Random Classifier (RC) as a weak baseline; two state-of-the-art point-wise supervised models Logistic Regression (LR) and Support Vector Machine (SVM); and an effective context sensitive model SVM^{hmm}. We applied LibLinear toolkit (Fan et al., 2008) to implement LR and SVM with linear kernel. SVM^{hmm} was implemented by SVM^{struct} toolkit (Joachims et al., 2009). And CRF's implementation was based on CRF++[3]. For all the baselines, we used features listed in Table 1. The hyper-parameters of all leader detection models were tuned to the same extent based on 5-fold cross validation (with 1 fold as development set). The evaluation metrics were precision, recall and F1 score for the detected leaders.

	Cross-validation			Held-out		
	Prec	Rec	F1	Prec	Rec	F1
Random	29.8	49.5	37.3	31.6	49.6	38.6
LR	70.5	66.3	68.4	70.4	66.2	68.2
SVM	70.9	66.9	68.8	68.9	66.2	67.5
SVM^{hmm}	74.8	65.5	69.8	69.3	70.1	69.7
CRF	**75.5**	**72.0**	**73.7**	**71.1**	**70.7**	**70.9**

Table 2: The performance of leader detection (%)

5.1.2 Results

Table 2 shows the comparison result of 5-fold cross validation on 1,000 repost tree paths and held-out experiment on 300 complete fresh paths.

Among all baselines, SVM^{hmm} performed the best, which indicates the effectiveness of incorporating structure information for leader detection. And among context-sensitive models, both SVM^{hmm} and CRF were competitive. CRF outperformed SVM^{hmm} slightly with 5.6% and 1.7% improved F1 score in cross validation and held-out

[2]The datasets are available at http://www1.se.cuhk.edu.hk/~lijing/data/repost_tree_summ.zip

[3]http://taku910.github.io/crfpp/

experiments, respectively. In spite of their comparable performance, our framework applies CRF instead of SVM^{hmm} for leader detection because of its probabilistic nature, which can be exploited by the sampling process in Soft-LeadSum to reduce the propagation of classification error to the summarization stage. Section 5.2.2 shows the relevant experiment.

5.2 Experiment for Summarization

In this experiment, we evaluated end-to-end performance of our basic and soft LeadSum summarization models by comparing them with state-of-the-art microblog summarizers.

5.2.1 Data Collection and Evaluation Metrics

There is no public editorial repost tree dataset. Therefore, we manually selected 10 hot events taking place during January 2nd – July 28th 2014, and then used the PKUVIS toolkit (Ren et al., 2014) to crawl the complete repost trees for all the events given the corresponding original posts. Table 3 shows the details about the repost tree corpus[4]. Note that this repost tree corpus has no overlap with the repost tree path dataset for learning leader detection models in Section 5.1.1.

After that, we invited three experienced editors to write summaries for each repost tree. To ensure the quality of reference summaries, we first extracted a list of frequent nouns from each repost tree and generalized 7 to 10 topics based on the nouns list, which provided a high-level overview of a repost tree to editors. Then, our guideline required editors to read all repost microblogs ordered sequentially on a repost tree. For every message, its entire repost tree path was also provided as supplementary context information. When finished reading, editors wrote down one or two sentences to summarize each topic in the list.

We utilized ROUGE-N metric (Lin, 2004) for benchmark, which is a standard for evaluating automatic summaries based on N-gram overlapping between a generated summary and a reference. Specifically, ROUGE-1 and ROUGE-2 F1-measure were used as our evaluation metrics. Lin et al. (2004) has demonstrated that ROUGE-2 correlates well with humans in summarizing formal texts. And ROUGE-1 is a better alternative in evaluating summaries for short and informal microblog messages (Inouye and Kalita, 2011; Chang et al., 2013).

In our human-generated summaries, the average inter-annotator-agreement by ROUGE-1 is 0.431, which means each pair of manual summaries have no more than 50% words overlap on average even written under topic constraints. This indicates that microblog repost tree summarization is generally a difficult task. The reason is that repost trees have complex structure, and editors could hardly reconstruct the repost trees even though they went through all the microblogs. Therefore, in evaluation for each tree, we computed the average ROUGE F1 score between the model-generated summary and the three human-generated summaries.

5.2.2 Results

In each automatic summarizer, we selected the top-10 ranked reposts to form a summary. We compared the end-to-end performance with the following baseline systems:

- **RandSum:** RandSum is a weak baseline that randomly selects reposts into summaries.
- **RepSum:** RepSum ranks and selects messages simply by their reposts count, i.e., the size of their subtrees, based on reposting relations.
- **UserRankSum:** UserRankSum ranks and selects reposts by their authors' follower count based on user following relations.
- **LeadProSum:** LeadProSum ranks and selects reposting messages by their marginal probabilities as leaders determined by our CRF-based leader detection model.
- **SVDSum:** SVDSum adopts the Singular Value Decomposition (SVD) to discover hidden sub-topics for summarization (Gong and Liu, 2001). Reposting messages are ranked according to latent semantic analysis with SVD on term-message matrix.
- **DivRankSum:** DivRankSum directly applies DivRank (Mei et al., 2010) algorithm to rank all messages unaware of leaders and followers. A similar model is also reported in Yan et al. (2011). Following their work, we set damping weight as 0.85.
- **UserInfSum:** Chang et al. (2013) ranks messages utilizing Gradient Boosted Decision Tree (GBDT) algorithm with text, popularity, temporal and user influence signals to summarize Twitter context tree. In particular, without the interaction data with external users, we utilize users' fol-

[4] All descriptions are English translations of the root microblogs originally in Chinese.

Name	# of nodes	# of nodes with comments	Height	Description
Tree (I)	21,353	15,409	16	HKU dropping out student wins the college entrance exam again.
Tree (II)	9,616	6,073	11	German boy complains hard schoolwork in Chinese High School.
Tree (III)	13,087	9,583	8	Movie Tiny Times 1.0 wins high grossing in criticism.
Tree (IV)	12,865	7,083	8	"I am A Singer" states that singer G.E.M asking for resinging conforms to rules.
Tree (V)	10,666	7,129	8	Crystal Huang clarified the rumor of her derailment.
Tree (VI)	21,127	15,057	11	Germany routs Brazil 7:1 in World-Cup semi-final.
Tree (VII)	18,974	12,399	13	The pretty girl pregnant with a second baby graduated with her master degree.
Tree (VIII)	2,021	925	18	Girls appealed for equality between men and women in college admission
Tree (IX)	9,230	5,408	14	Violent terrorist attack in Kunming railway station.
Tree (X)	10,052	4,257	25	MH17 crash killed many top HIV researchers.

Table 3: Description of repost tree summarization corpus consisting of 10 hot events

	ROUGE-1			ROUGE-2		
	F1	σ	SIG	F1	σ	SIG
RandSum	.159	.046	**‡	.037	.009	**‡
RepSum	.162	.071	**‡	.030	.016	**‡
UserRankSum	.292	.066	‡	.087	.028	†
LeadProSum	.270	.119	‡	.064	.038	‡
SVDSum	.222	.070	**‡	.048	.032	**‡
DivRankSum	.159	.079	**‡	.029	.018	**‡
UserInfSum	.272	.091	‡	.071	.028	‡
B-LS+SVM^{hmm}	.301	.031	‡	.085	.020	†
B-LS+CRF	.300	.029	‡	.082	.016	‡
S-LS+CRF	**.351**	.027	NA	**.105**	.018	NA

Remarks:
B-LS: Basic-LeadSum model; S-LS: Soft-LeadSum model
F1: F1-measure of ROUGE-1 or ROUGE-2
σ: Standard deviation of F1-measure over 10 repost trees
SIG: Significance indicator of F1-measure based on one-tailed pairwise t-test:
– Significantly different with B-LS+CRF: * ($p < 0.1$); ** ($p < 0.05$)
– Significantly different with S-LS+CRF: † ($p < 0.1$); ‡ ($p < 0.05$)

Table 4: Comparison of different summarizers

lower count to approximate user influence. GBDT implementation is based on RankLib[5], and as a supervised method, UserInfSum is evaluated with 10-fold cross validation.

In addition, we observed that SVM^{hmm} is a competitive baseline for leader detection (see Table 2). So we also study its impact on the Basic-LeadSum model. Note that SVM^{hmm} cannot be combined with Soft-LeadSum since it is not probabilistic.

Table 4 shows the result of overall comparisons. We have the following observations:

• RepSum utilized trivial structure information, i.e., the size of sub-tree, and its performance was poor, which was even worse than RandSum on ROUGE-2. This implies that messages with a lot of reposts may not be good candidates as other reasons may lead to their popularity, e.g., a good posting time or sense of humor.

• UserRankSum performed the best on ROUGE-1&2 among all baseline summarizers, which confirms that user following relations can indeed be a strong signal in microblog summarization. UserRankSum is even slightly better than Basic-LeadSum on ROUGE-2. But, it does not perform consistently well for all repost trees, evidenced as the large standard deviation on ROUGE-1&2. This suggests that the user following relations cannot always effectively indicate salient candidates. It may not work for repost trees where authors have similar number of following users, or reposts of influential users contain nothing salient.

• LeadProSum achieved the second best performance among all unsupervised baselines, which indicates that the marginal probabilities as leaders can signal good summary candidates. This also confirms that leaders contain salient contents and should be distinguished from followers in summarization.

• Utilizing either SVM^{hmm} or CRF as leader detection model to filter out followers, Basic-LeadSum almost doubled the ROUGE-1 and tripled the ROUGE-2 scores compared to DivRankSum's performance. This indicates that differentiating leaders and followers is very helpful to summarization.

• Basic-LeadSum performed better than all baselines on ROUGE-1&2 except for a marginal drop compared to UserRankSum on ROUGE-2. But the differences with UserRankSum, LeadProSum and UserInfSum are not statistically significant. This may be ascribed to the error propagated from leader detection module to summarization process.

• Soft-LeadSum outperformed all the baselines with a large margin on ROUGE-1&2, including supervised summarizer UserInfSum. The one-tailed pairwise t-test indicates that all the improvements over baselines are significant at the 95% confidence level except for UserRankSum with 90% confidence level on ROUGE-2. This confirms the effectiveness of our framework for producing high-quality repost tree summaries.

• The supervised model UserInfSum did not

[5] http://sourceforge.net/p/lemur/wiki/RankLib/

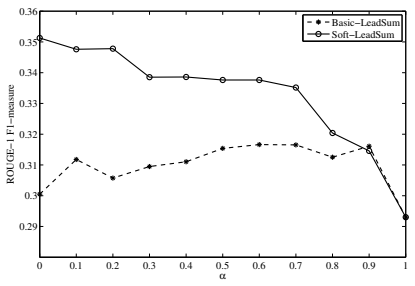


Figure 2: The impact of α on the ROUGE-1 F1-measure of combined models

perform quite well. The reason is that the model needs large amount of user interaction data external to the tree which are not readily available, and also it might be overfitting to the limited number of training instances.

• Basic-LeadSum with CRF and SVM^{hmm} had very close ROUGE-1&2 scores. Basic-LeadSum+SVM^{hmm} is even slightly better than Basic-LeadSum+CRF. Though SVM^{hmm} was marginally worse in leader detection experiment (Table 2), we can conclude that SVM^{hmm} is a comparable alternative as the leader detection module for Basic-LeadSum.

• Among our models, Soft-LeadSum significantly outperformed both Basic-LeadSum with CRF and that with SVM^{hmm}. This implies that sampling steps in the enhanced random walk of Soft-LeadSum is effective in reducing the impact of leader detection error on summarization.

5.3 Discussion

From Table 4, we observed that user following relations used by UserRankSum is a strong signal for microblog summarization. A natural question is: "Can the user following relations commonly used for modeling user influence be complementary to the content-level structure information used in our summariztaion models?"

We thus linearly combine the normalized ranking scores of LeadSum and UserRankSum using the formula $\alpha * \mathbf{u} + (1-\alpha) * \mathbf{l}$, where $\mathbf{u}$ and $\mathbf{l}$ denote the UserRankSum and LeadSum ranking scores, respectively. Figure 2 demonstrates the impact of α on our basic and soft LeadSum model with CRF.

Clearly, Basic-LeadSum can benefit from user influence information by incorporating UserRankSum scores into it. From the incremental trend of summarization performance with the increase of α for $\alpha \in [0, 0.9]$, we can conclude that user influence is helpful to it. This is because Basic-LeadSum is not sufficiently robust to the errors cascaded from leader detection module, thus user-level structures can have the chance to compensate these errors for content-level structures.

Incorporating the same information into Soft-LeadSum cannot improve its performance regardless of the value of α. This implies that content-level structures, i.e., message content and reposting relations together, are better indicative of good summary candidates. When these features are appropriately modeled by Soft-LeadSum, user influence, a traditionally well-known strong signal, cannot provide extra benefit at all.

6 Conclusion and Future Work

This work presents a study for microblog repost tree summarization, whose output can provide important clues for event analysis on microblogging platforms. Conventional works considering only plain text streams is insufficient to summarize noisy repost trees. We propose a novel summarization system by effectively differentiating leader and follower messages on repost tree based on content-level structure information. Firstly, a leader detection model categorizes each repost on repost tree path as a leader or a follower. Then, a random-walk variant summarization model called LeadSum is proposed to rank and select salient microblog messages on the basis of leader detection result. To reduce errors cascaded from leader detection, we enhance LeadSum based on an even-length random walk by sampling from leader probabilities for improving summarization. Based on real-world microblog post dataset, the experimental results confirm that our proposed framework is effective for repost tree summarization by the end-to-end comparison with the state-of-the-art baselines.

Constrained by the amount of annotation, we adopt this two-step framework and an unsupervised summarization algorithm. With the development of our corpora, we plan to explore the usefulness of supervised structure learning approaches, such as tree-structured CRF (Tang et al., 2006; Mensink et al., 2013), to integrate leader detection and summarization into a unified framework, and make global inference for important leaders by capturing various non-linear dependencies.

References

Yasemin Altun, Ioannis Tsochantaridis, and Thomas Hofmann. 2003. Hidden markov support vector machines. In *Proceedings of the Twentieth International Conference on Machine Learning, ICML*, pages 3–10.

Deepayan Chakrabarti and Kunal Punera. 2011. Event summarization using tweets. In *Proceedings of the Fifth International Conference on Weblogs and Social Media*, pages 66–73.

Yi Chang, Xuanhui Wang, Qiaozhu Mei, and Yan Liu. 2013. Towards twitter context summarization with user influence models. In *Sixth ACM International Conference on Web Search and Data Mining, WSDM*, pages 527–536.

Dipanjan Das and André FT Martins. 2007. A survey on automatic text summarization. *Literature Survey for the Language and Statistics II course at CMU*, 4:192–195.

Yajuan Duan, Zhimin Chen, Furu Wei, Ming Zhou, and Heung-Yeung Shum. 2012. Twitter topic summarization by ranking tweets using social influence and content quality. In *Proceedings of the 24th International Conference on Computational Linguistics, COLING*, pages 763–780.

Günes Erkan and Dragomir R. Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *J. Artif. Intell. Res. (JAIR)*, 22:457–479.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874.

Joseph L Fleiss, Bruce Levin, and Myunghee Cho Paik. 2013. *Statistical methods for rates and proportions*. John Wiley & Sons.

Yihong Gong and Xin Liu. 2001. Generic text summarization using relevance measure and latent semantic analysis. In *Proceedings of the 24th Annual International ACM Conference on Research and Development in Information Retrieval, SIGIR*, pages 19–25.

David Inouye and Jugal K. Kalita. 2011. Comparing twitter summarization algorithms for multiple post summaries. In *Privacy, Security, Risk and Trust (PASSAT) and 2011 IEEE Third Inernational Conference on Social Computing (SocialCom)*, pages 298–306.

Thorsten Joachims, Thomas Finley, and Chun-Nam John Yu. 2009. Cutting-plane training of structural svms. *Machine Learning*, 77(1):27–59.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning, ICML*, pages 282–289.

Liangda Li, Ke Zhou, Gui-Rong Xue, Hongyuan Zha, and Yong Yu. 2009. Enhancing diversity, coverage and balance for summarization through structure learning. In *Proceedings of the 18th International Conference on World Wide Web, WWW*, pages 71–80.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Proceedings of the ACL-04 Workshop*, pages 74–81.

Fei Liu, Yang Liu, and Fuliang Weng. 2011. Why is sxsw trending?: exploring multiple text sources for twitter topic summarization. In *Proceedings of the Workshop on Languages in Social Media*, pages 66–75. Association for Computational Linguistics.

Xiaohua Liu, Yitong Li, Furu Wei, and Ming Zhou. 2012. Graph-based multi-tweet summarization using social signals. In *Proceedings of the 24th International Conference on Computational Linguistics*, pages 1699–1714.

He Liu, Hongliang Yu, and Zhi-Hong Deng. 2015. Multi-document summarization based on two-level sparse representation model. In *Proceedings of the Twenty-Ninth Conference on Artificial Intelligence, AAAI*, pages 196–202.

Rui Long, Haofen Wang, Yuqiang Chen, Ou Jin, and Yong Yu. 2011. Towards effective event detection, tracking and summarization on microblog data. In *Web-Age Information Management - 12th International Conference, WAIM*, pages 652–663.

Qiaozhu Mei, Jian Guo, and Dragomir R. Radev. 2010. Divrank: the interplay of prestige and diversity in information networks. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining,KDD*, pages 1009–1018.

Xinfan Meng, Furu Wei, Xiaohua Liu, Ming Zhou, Sujian Li, and Houfeng Wang. 2012. Entity-centric topic-oriented opinion summarization in twitter. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD*, pages 379–387.

Thomas Mensink, Jakob J. Verbeek, and Gabriela Csurka. 2013. Tree-structured CRF models for interactive image labeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(2):476–489.

Xipeng Qiu, Qi Zhang, and Xuanjing Huang. 2013. Fudannlp: A toolkit for chinese natural language processing. In *Proceedings of Annual Meeting of the Association for Computational Linguistics, ACL*, pages 49–54.

Dragomir R. Radev, Eduard H. Hovy, and Kathleen McKeown. 2002. Introduction to the special issue on summarization. *Computational Linguistics*, 28(4):399–408.

Dragomir R. Radev, Timothy Allison, Sasha Blair-Goldensohn, John Blitzer, Arda Çelebi, Stanko Dimitrov, Elliott Drábek, Ali Hakim, Wai Lam, Danyu Liu, Jahna Otterbacher, Hong Qi, Horacio Saggion, Simone Teufel, Michael Topper, Adam Winkel, and Zhu Zhang. 2004. MEAD - A platform for multidocument multilingual text summarization. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation, LREC*.

Kenneth Rapoza. 2011. China's weibos vs us's twitter: And the winner is.

Donghao Ren, Xin Zhang, Zhenhuang Wang, Jing Li, and Xiaoru Yuan. 2014. Weiboevents: A crowd sourcing weibo visual analytic system. In *IEEE Pacific Visualization Symposium, PacificVis*, pages 330–334.

Kevin Dela Rosa, Rushin Shah, Bo Lin, Anatole Gershman, and Robert Frederking. 2011. Topical clustering of tweets. *Proceedings of the ACM SIGIR: SWSM*.

Beaux Sharifi, Mark-Anthony Hutton, and Jugal Kalita. 2010. Automatic summarization of twitter topics. In *National Workshop on Design and Analysis of Algorithm*.

Chao Shen, Fei Liu, Fuliang Weng, and Tao Li. 2013. A participant-based approach for event summarization using twitter streams. In *Proceedings of Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, HLT-NAACL*, pages 1152–1162.

Hiroya Takamura, Hikaru Yokono, and Manabu Okumura. 2011. Summarizing a document stream. In *Advances in Information Retrieval - 33rd European Conference on IR Research, ECIR*, pages 177–188.

Jie Tang, MingCai Hong, Juan-Zi Li, and Bangyong Liang. 2006. Tree-structured conditional random fields for semantic annotation. In *The Proceedings of 5th International Semantic Web Conference, ISWC*, pages 640–653.

Rui Yan, Liang Kong, Congrui Huang, Xiaojun Wan, Xiaoming Li, and Yan Zhang. 2011. Timeline generation through evolutionary trans-temporal summarization. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing, EMNLP*, pages 433–443.

Intra-sentential Zero Anaphora Resolution using Subject Sharing Recognition

Ryu Iida Kentaro Torisawa Chikara Hashimoto
Jong-Hoon Oh Julien Kloetzer
National Institute of Information and Communications Technology
Kyoto 619-0289, Japan
{ryu.iida,torisawa,ch,rovellia,julien}@nict.go.jp

Abstract

In this work, we improve the performance of intra-sentential zero anaphora resolution in Japanese using a novel method of recognizing subject sharing relations. In Japanese, a large portion of intra-sentential zero anaphora can be regarded as subject sharing relations between predicates, that is, the subject of some predicate is also the unrealized subject of other predicates. We develop an accurate recognizer of subject sharing relations for pairs of predicates in a single sentence, and then construct a *subject shared predicate network*, which is a set of predicates that are linked by the subject sharing relations recognized by our recognizer. We finally combine our zero anaphora resolution method exploiting the subject shared predicate network and a state-of-the-art ILP-based zero anaphora resolution method. Our combined method achieved a significant improvement over the the ILP-based method alone on intra-sentential zero anaphora resolution in Japanese. To the best of our knowledge, this is the first work to explicitly use an independent subject sharing recognizer in zero anaphora resolution.

1 Introduction

In 'pro-dropped' languages such as Japanese, Chinese and Italian, pronouns are often unrealized in text. For example, the subject of *nomu* (take) is omitted in example (1).

(1) *Tom$_i$-wa* *infuruenza-ni* *natta-node* ,
Tom-TOP flu-IOBJ had-since punc
(ϕ_i-ga) *kusuri-o* ***non-da*** .
he$_i$-SUBJ medicine-OBJ took period
Since Tom$_i$ had the flu, (he$_i$) took medicine.

Such unrealized pronouns are regarded as *zero anaphors*, which are indicated using ϕ in literature, like ϕ_i-ga in example (1). Zero anaphor refers to its antecedent somewhere. This phenomenon of the reference is called *zero anaphora*. In Japanese, about 60% of subjects appear as zero anaphors in newspaper articles (Iida et al., 2007b), and thus zero anaphora resolution is an essential task for developing highly accurate machine translation and information extraction systems.

In this paper, we propose a novel method of resolving intra-sentential zero anaphora, in which a subject zero anaphor refers to its antecedent inside *a single sentence*. This work does not address inter-sentential zero anaphora, in which a zero anaphor in a sentence refers to its antecedent in *another sentence*. The novelty of our method is in the use of *subject sharing relations*, which are relations between two predicates that share a subject by (zero) anaphora or coreference. For example, in example (2), there are two subject sharing relations for predicate pairs, *advance-plan* and *plan-dispatch*, as illustrated in Figure 1.

(2) *seifu$_i$-wa* *(ϕ_i-ga)* *hisaichi-ni*
government$_i$-TOP it$_i$-SUBJ disaster site-IOBJ
50 nin-o ***hakensuru*** *koto-o* *(ϕ_i-ga)*
50 people-OBJ dispatch COMP-OBJ it$_i$-SUBJ
keikakusi *junbisagyo-o* ***susumeru*** .
plan preparation-OBJ advance period
The government$_i$ plans that (it$_i$) will dispatch 50 people to the disaster site and (it$_i$) is advancing its preparations.

The most straightforward method to recognize subject sharing relations is to apply a (zero) anaphora resolution system to a sentence and detect such relations by recognizing (zero) anaphora, like the relations represented by *seifu$_i$* and two zero anaphors ϕ_i in Figure 1. However, to our surprise, we found that a simple supervised classifier that exploits the local contexts surrounding

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 2179–2189,
Lisbon, Portugal, 17-21 September 2015.

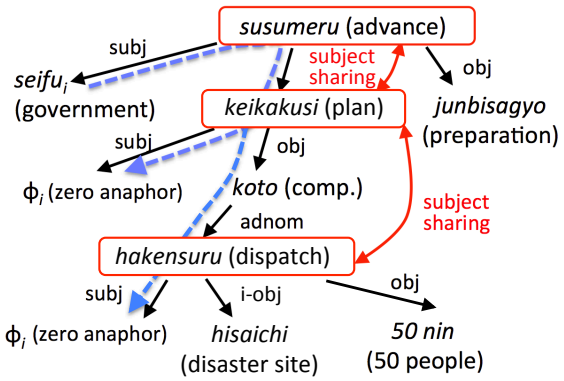


Figure 1: Example of subject shared predicate network

predicates achieved a higher accuracy than that of the straightforward method. This suggests that just *propagating* the realized subject of a predicate to the subject zero anaphor of other predicates through recognized subject sharing relations (e.g., propagating subject *government* of *advance* to the subject positions of *plan* and *dispatch* in Figure 1) might lead to a higher accuracy in zero anaphora resolution than the existing zero anaphora resolution methods. In addition, a large portion of zero anaphora can be regarded as subject sharing relations (e.g., 39% of the intra-sentential zero anaphora in the NAIST Text Corpus (Iida et al., 2007b) are such cases). Hence, just by combining our subject zero anaphora method with an existing general anaphora resolution method that covers other types of anaphora, significant improvement of accuracy over all types of anaphora might be achieved. This paper empirically shows that this is actually the case through a series of experiments in which we combine our method with an existing ILP-based zero anaphora resolution method (Iida and Poesio, 2011).

Our subject zero anaphora resolution method constructs a *subject shared predicate network (SSPN)*, which is a network of predicates in which subject sharing predicates are linked, from the results of our accurate *pairwise subject sharing recognizer*, which detects the predicate pairs that share a subject. Zero anaphora resolution is done by propagating the realized subject of a predicate to the subject zero anaphor of other predicates in the SSPN. An important point here is that SSPN was introduced to solve the issue related to our pairwise subject sharing recognizer. Our recognizer is applied only to the restricted pairs of predicates in a sentence, such as predicates that have a *direct* dependency relation between them. This is because our current recognizer cannot achieve high accuracy for *any* pair of predicates. In Figure 1, for instance, our recognizer can detect a subject sharing relation between *advance* and *plan* and another between *plan* and *dispatch*, but it cannot detect one between *advance* and *dispatch*. However, in the SSPN, the undetected relations can be derived by connecting the two detected ones, and in the zero anaphora resolution subject *government* of *advance* can be successfully propagated to the subject position of *dispatch*.

The rest of our paper is organized as follows. In Section 2, we briefly overview previous work on zero anaphora resolution. In Section 3, we overview the procedure of our zero anaphora resolution method. We explain the three types of subject sharing relations on which we focus and propose a method of pairwise subject sharing recognition for the three types in Section 4. We evaluate how effectively our method recognizes subject sharing relations for these types in Section 5. After that, we investigate the impact of explicitly introducing SSPNs in Section 6 and compare our zero anaphora resolution method with a state-of-the-art ILP-based method on the task of intra-sentential subject zero anaphora resolution in Section 7. Finally, in Section 8 we summarize this work and discuss future directions.

2 Related work

Traditional approaches to zero anaphora resolution are based on manually created heuristic rules (Kameyama, 1986; Walker et al., 1994; Okumura and Tamura, 1996; Nakaiwa and Shirai, 1996), which are mainly motivated by the rules and preferences introduced in Centering Theory (Grosz et al., 1995). However, the research trend of zero anaphora resolution has shifted from such rule-based approaches to machine learning-based approaches because in machine learning we can easily integrate many different types of information, such as morpho-syntactic, semantic and discourse-related information. Researchers have developed methods of zero anaphora resolution for Chinese (Zhao and Ng, 2007; Chen and Ng, 2013), Japanese (Seki et al., 2002; Isozaki and Hirao, 2003; Iida et al., 2007a; Taira et al., 2008; Sasano et al., 2008; Sasano et al., 2009; Imamura

et al., 2009; Watanabe et al., 2010; Hayashibe et al., 2011; Iida and Poesio, 2011; Yoshikawa et al., 2011; Hangyo et al., 2013; Yoshino et al., 2013) and Italian (Iida and Poesio, 2011). One critical issue in zero anaphora resolution is optimizing the outputs of sub-problems (e.g., zero anaphor detection and antecedent identification). Recent works by Watanabe et al. (2010), Iida and Poesio (2011) and Yoshikawa et al. (2011) revealed that joint inference improves the overall performance of zero anaphora resolution. We employed one of these works as a baseline in Section 6.

Concerning subject sharing recognition, related methods have been explored for pronominal anaphora (Yang et al., 2005) or coreference resolution (Bean and Riloff, 2004; Bansal and Klein, 2012). In these methods, the semantic compatibility between the contexts surrounding an anaphor and its antecedent (e.g., the compatibility of verbs *kidnap* and *release* given some arguments) was automatically extracted from raw texts in an unsupervised manner and used as features in a machine learning-based approach. However, because the automatically acquired semantic compatibility is not always true or applicable in the context of any pair of an anaphor and its antecedent, the effectiveness of the compatibility features might be weakened. In contrast, we accurately recognize the explicit subject sharing relations and directly use them for propagating the subject of some predicate to the empty subject position of other predicates instead of indirectly using the relations as features.

3 Zero anaphora resolution using subject shared predicate network

In this section, we first give an overview of the procedure of our zero anaphora resolution method. Intra-sentential zero anaphora resolution in our method is performed in the following five steps, as depicted in Figure 2.

Step 1 The pairwise subject sharing relations between two predicates in a sentence are recognized by our subject sharing recognizer.

Step 2 A subject shared predicate network (SSPN) is constructed based on the results of pairwise subject sharing recognition.

Step 3 For each predicate in the set of the subject shared predicates in the SSPN, a subject is detected by our subject detector, if one exists.

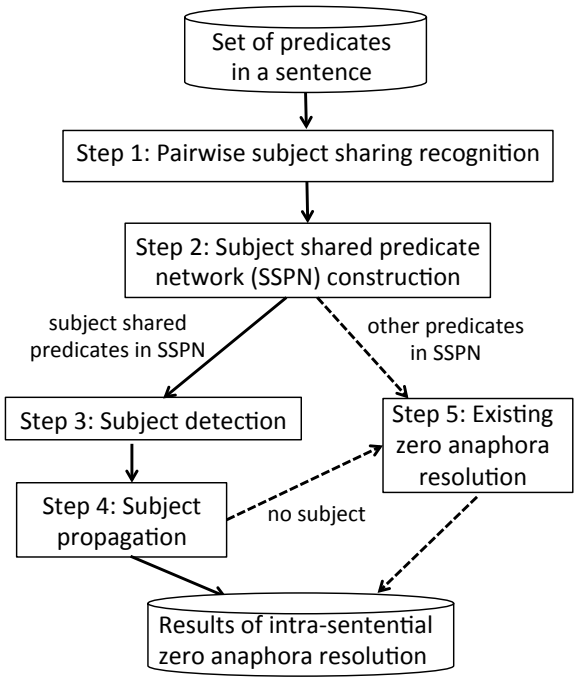


Figure 2: Procedure of our zero anaphora resolution method

Step 4 If a subject is detected, it is propagated to the empty subject position of each predicate in the subject shared predicates in the SSPN.

Step 5 For resolving the potential zero anaphora that were not resolved until Step 4, we apply the existing ILP-based method (Iida and Poesio, 2011).

We define subject sharing relations as follows. Two predicates have a subject sharing relation if and only if they share the same subject that is referred to by (zero) anaphora or coreference. Note that the shared subject does not need to be realized in the text; it can appear as inter-sentential zero anaphora or exophora. In Step 1, the pairwise subject sharing relations between two predicates are recognized, but recognizing the relations between any two predicates in a sentence remains difficult. We thus focus on some typical types of predicate pairs. The details of the predicate pair types will be explained in Section 4.1.

Given the results of pairwise subject sharing recognition, we construct an SSPN in Step 2. In an SSPN, every predicate in a sentence is a node and only the predicate pairs that were judged to be subject sharing are connected by a link. The major advantage of explicitly constructing an SSPN is that it enables us to resolve zero anaphora even

if a predicate with a subject zero anaphor does not have any direct subject sharing relation with a predicate with a subject, like predicates *susumeru* (advance) and *hakensuru* (dispatch) in Figure 1. By traversing the paths of the subject sharing relations in the SSPN, such predicates can be connected to successfully propagate the subject. The effect of introducing SSPNs is empirically evaluated in Section 6.

For use in Step 3, we create a subject detector, which judges whether an argument to a predicate is its subject using SVM^{light} [1], an implementation of Support Vector Machine (Vapnik, 1998), with a polynomial kernel of 2nd degree. The training instances of the subject detector are extracted from the predicate-argument relations[2] in the NAIST Text Corpus. The numbers of positive and negative instances are 35,304 and 104,250 respectively. As features, we used the morpho-syntactic information about the lemmas of the predicate and its argument and the functional words following the predicate and its argument. The results of subject detection with 5-fold cross-validation demonstrate that our subject detector accurately detects subjects with performances of 0.949 in recall, 0.855 in precision, and 0.899 in F-score.

Note that our subject detector checks whether each predicate in an SSPN has a syntactic subject among its arguments. An SSPN can include more than one predicate, and each predicate may have its own subject[3]. In this step, if two or more distinct subjects are detected for predicates in an SSPN, we use the most likely subject (i.e., the subject with the highest SVM score outputted by our subject detector) for subject propagation. Note that subject propagation is not performed if the subject position of a predicate is already filled.

Up to this point, the zero anaphora of the following three cases cannot be resolved: (i) no subject was detected for any predicate in a group linked by the subject sharing relations in the SSPN, (ii) no subject sharing relation was recognized for a predicate in the SSPN and (iii) non-

[1] http://svmlight.joachims.org/

[2] Note that if a predicate appears in a relative clause and a noun modified by the clause is the *semantic* subject of the predicate, the noun is not regarded as subject by our subject detector.

[3] The subject sharing recognizer is likely to regard two predicates, each of which has its own subject, as non-subject sharing predicate pairs, but it is still logically possible that they are judged as subject sharing predicate pairs hence as a part of an SSPN.

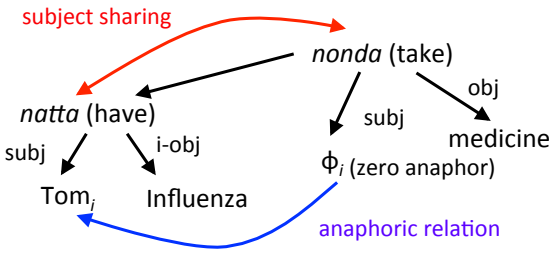


Figure 3: Example of DEP type

subject arguments were omitted as zero anaphors. To resolve zero anaphora in these cases, we apply a state-of-the-art ILP-based zero anaphora resolution method (Iida and Poesio, 2011) in Step 5. This method determines zero anaphor and its antecedent by joint inference using the results of subject detection, zero anaphor detection and intra- and inter-sentential antecedent identification. In the original method by Iida and Poesio (2011), the inter-sentential zero anaphora was resolved, but in this work we focus on intra-sentential zero anaphora. To adapt their method for our problem setting, we simply removed the inter-sentential antecedent identification model from their method.

4 Pairwise subject sharing recognition

A key component in our zero anaphora resolution method is pairwise subject sharing recognition. In this work, we focus on three types of subject sharing relations (DEP, ADJ and PNP types) as a first step because the instances belonging to the three types occupy 62% of intra-sentential zero anaphora that can be regarded as subject sharing. We developed a method that recognizes each subject sharing type and evaluate it.

4.1 Three types of subject sharing relations

We first describe the three types of subject sharing relations we focus on.

DEP A typical type of subject sharing relation is one between two predicates that have a syntactical dependency relation. The relation between two predicates, *natta* (have) and *nonda* (take), in example (1) in Section 1 is classified as this type because the two predicates have the same subject *Tom*$_i$ (ϕ_i), as illustrated in Figure 3. We call this type of subject sharing the DEP type.

ADJ This type is a subject sharing relation between two *adjacent* predicates, i.e., a predicate

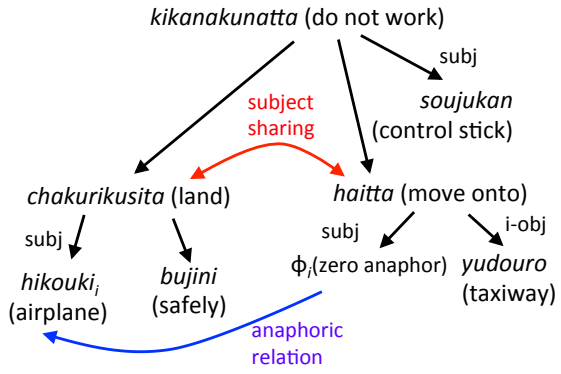


Figure 4: Example of ADJ type

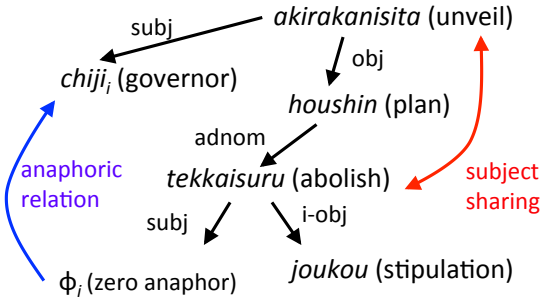


Figure 5: Example of PNP type

pairs that do not have any other predicate between them in the surface order of a sentence. Although two adjacent predicates in a sentence tend to share the same subject, they sometimes cannot be captured as the DEP type due to a long-distance dependency between predicates. For example, in example (3), two adjacent predicates, *land* and *move onto*, have the same subject but not a direct dependency relation, as illustrated in Figure 4.

(3) *hikouki-wa* *bujini* ***chakurikusi-ta-ga***
airplane-TOP safely land-PAST-but

(ϕ$_i$-ga) *yudouro-ni* ***hait-ta-atoni***
it$_i$-SUBJ taxiway-IOBJ move onto-PAST-after

soujukan-ga *kikanakunat-ta* .
control stick-SUBJ do not work-PAST period
The airplane safely landed, but its control stick did not work after (it$_i$) moved onto the taxiway.

To cover such cases, we also take into account the subject sharing relations of the ADJ type in which two predicates appear adjacently in the surface order.

PNP In addition to the above two types of relations, in Japanese predicate pairs often have a subject sharing relation when one of the predicates syntactically depends on a noun (or noun phrase) that in turn syntactically depends on the other predicate. Example (4) is classified as such a type because noun *houshin* (plan) is placed between two predicates, *akirakanisita* (unveil) and *tekkaisuru* (abolish), in the dependency path and predicates share subject *chiji* (governor), as illustrated in Figure 5.

(4) *chiji$_i$-wa* *(ϕ$_i$-ga)* *joukou-o*
governor-TOP he$_i$-SUBJ stipulation-OBJ

tekkaisuru *houshin-o* ***akirakanisi-ta*** .
abolish plan-TOP unveil-PAST period
The governor$_i$ unveiled his plan under which (he$_i$) will abolish the stipulation.

We call this type of subject sharing relation the PNP type.

In this work, we solve the problem of subject sharing recognition as a binary classification problem in which we classify whether two predicates share the same subject. We solve this problem using a supervised approach. We independently extract the training instances for each type from a corpus to which (zero) anaphora, coreference and subjects were annotated. The binary labels of the training instances are classified into the positive class if the subject of the two predicates in an instance is shared by coreference or (zero) anaphora, and negative otherwise. To create a classifier, we use SVM^{light} and experiment with both a linear kernel and a polynomial kernel of 2nd degree.

As features, we use the feature set shown in Table 1. Even though these features look simple, we expect them to work well to capture the characteristics of each subject sharing type. For example, as shown in example (5), the (subject) case marker of the argument (mother-SUBJ) between two predicates *natta* (have) and *katta* (buy) is a good indicator of non-subject sharing.

(5) *Tom$_i$-ga* *infuruenza-ni* ***natta-node*** ,
Tom-SUBJ flu-IOBJ had-since punc

haha-ga *kusuri-o* ***katta*** .
mother-SUBJ medicine-OBJ buy-PAST period
Since Tom had the flu, his mother bought medicine.

For recognizing the PNP type of subject sharing relations, whether certain nouns appear between two predicates is an important clue, e.g., *koto* (complementizer) in example (6) and *nouryoku* (ability) in example (7).

Name	Description
PoS_i (PoS_j)	PoS of p_i (p_j)
lemma_i (lemma_j)	lemma of p_i (p_j)
func_w_i (func_w_j)	function words following p_i (p_j)
case_i (case_j)	case marker of arguments of p_i (p_j)
btw_case	case marker of arguments that appeared between p_i and p_j
NpPoS*	PoS of *np*
Np_lemma*	lemma of *np*
func_w_{np}*	function words following *np*
case_{np}*	case marker of dependents of *np*
n_class*	noun class of *np* based on Kazama and Torisawa (2008)

p_i and p_j stand for the left and right predicates in predicate pairs. *np* is the noun phrase between p_i and p_j. b_i (b_j) stands for the *bunsetsu*-unit[4] including p_i (p_j). The features marked with * are only used for PNP type.

Table 1: Features of subject sharing recognition

(6) *seifu$_i$-wa* *(ϕ_i-ga)* *sono isetsu-o*
government-TOP it_i-SUBJ the relocation-OBJ
mitomeru ***koto-o*** *kime-ta* .
admit COMP-OBJ decide-PAST period
The government$_i$ decided that (it_i) admits the relocation.

(7) *sono fune-wa* *(ϕ_i-ga)* *hayaku*
the ship-TOP it_i-SUBJ fast
hashiru ***nouryoku-o*** *motteiru* .
run ability-OBJ have period
The ship$_i$ has an ability that (it_i) runs fast.

To robustly capture this characteristic, we use as features the discrete classes created by the noun clustering algorithm proposed by Kazama and Torisawa (2008). It follows the distributional hypothesis, which states that semantically similar words tend to appear in similar contexts (Harris, 1954). By treating the syntactic dependency relations between words as 'contexts,' the clustering method defines a probabilistic model of noun-verb dependencies with hidden classes:

$$p(n, \langle v, r \rangle) = \sum_c p(n|c)p(\langle v, r \rangle|c)p(c)$$

where n is a noun, v is a verb or noun on which n depends by grammatical relation r (post-positions in Japanese), and c is a hidden class. The dependency relation frequencies were obtained from a 600-million page web corpus, and model parameters $p(n|c)$, $p(\langle v, r \rangle|c)$ and $p(c)$ were estimated using the EM algorithm (Hofmann, 1999). We clustered one million nouns into 500 discrete classes by assigning noun n to class c when the model parameter $p(c|n) > \theta$ ($\theta = 0.2$).

[4]A bunsetsu-unit is a Japanese base phrase consisting of at least one content word optionally followed by functional words.

5 Experiment 1: pairwise subject sharing recognition

We first empirically evaluate the performance of our pairwise subject sharing recognition for the DEP, ADJ and PNP types.

5.1 Experimental setting

The training data for the subject sharing recognizer were generated from the NAIST Text Corpus 1.4 (Iida et al., 2007b), in which (zero) anaphora, coreference and subjects were manually annotated. We automatically extracted pairs of predicates from the corpus. Since the original NAIST Text Corpus has a wide variety of annotation noise, we cleaned it up by the following strategy. According to the annotation scheme in the NAIST Text Corpus, predicate-argument relations were annotated for the 'bare predicates' even if the predicates appear in passive or causative sentences. In such cases, the annotation was difficult and caused inconsistencies because the annotators needed to imagine the predicate-argument relations for predicates that are not explicitly written, considering case alternation caused by changes of voice and so on. As such, to achieve a higher level of consistency, we modified the annotation scheme for predicate-argument relations by considering 'surface predicates' and re-annotated predicate-argument relations in passive and causative cases, thus reducing the risk of inconsistent annotations caused by case alternation.

type	method	Recall	Precision	F-score
DEP	baseline	0.161	0.505	0.244
	proposed (linear)	0.545	0.719	0.620
	proposed (poly-2d)	0.578	0.732	**0.646**
ADJ	baseline	0.143	0.414	0.212
	proposed (linear)	0.011	0.604	0.021
	proposed (poly-2d)	0.285	0.713	**0.407**
PNP	baseline	0.154	0.329	0.210
	proposed (linear)	0.028	0.844	0.053
	proposed (poly-2d)	0.159	0.723	**0.260**

Table 2: Results of subject sharing recognition

Another important point is that in the NAIST Text Corpus, if the antecedent of a zero anaphor is not explicitly written in the corpus, it is simply annotated as 'exophoric', and the subject sharing relations between two predicates whose subject was annotated as exophoric cannot be captured. In contrast, in our cleaning procedure, the annotators additionally annotated such 'exophoric' subject sharing relations to take into account all subject sharing relations in the corpus.

The predicates in the corpus and their dependency relations were detected based on the outputs of a Japanese dependency parser, J.DepP[5] (Yoshinaga and Kitsuregawa, 2009). We obtained 49,313 predicate pairs for the DEP type, 86,728 for the ADJ type, and 27,117 for the PNP types. The numbers of positive instances of DEP, ADJ and PNP types are 9,524, 13,104, and 2,363 respectively. To evaluate the subject sharing recognition, we conducted 5-fold cross-validation using these predicate pairs and measured the performance using recall, precision and F-score.

Note that we also evaluated a baseline method that recognizes subject sharing relations using the results of the state-of-the-art zero anaphora resolution method (Iida and Poesio, 2011) and the subject detector at Step 3 in Section 3.

5.2 Results: subject sharing recognition

We measured the performances of the baseline and our subject sharing recognition method using recall, precision and F-score for each of the three types of subject sharing relations, which are shown in Table 2. The results demonstrate that all of the proposed classifiers solved the problems with high precision. In particular, for each type, the classifier using a polynomial kernel achieved more than 70% precision. We thus used the classifiers with a polynomial kernel for evaluations in Section 6. The results also show that the classifier using a polynomial kernel for each type outperformed the baseline method based on the state-of-the-art zero anaphora resolution method. That is, the direct subject sharing recognition using our classifiers has the potential to lead to a significant improvement in zero anaphora resolution, which we confirm through the experiments in Section 7.

Table 2 also shows that the classifier for the DEP type outperformed those for all of the other types in F-score. The difference reflects the wider variations of the problems in both ADJ and PNP compared to the case of DEP. For example, to recognize the PNP type of subject sharing relation, our classifier needs to appropriately learn the complicated relationship between two predicates and the noun that intervenes between them, a problem we do not need to consider for the DEP type.

6 Experiment 2: intra-sentential zero anaphora resolution between subjects

We next investigate the effect of introducing SSPNs. In this experiment, we evaluated the performance of intra-sentential zero anaphora resolution only between subjects, i.e., the positive instances used in this experiment were limited to the cases where the antecedent of a zero anaphor is the realized subject of a predicate. We evaluated a method of zero anaphora resolution using only SSPNs, where intra-sentential zero anaphora is resolved by the first four steps (Steps 1 to 4) in Section 3. We compared it to a baseline that only used the results of pairwise subject sharing recognition without SSPNs: if the subject sharing relation between two predicates is recognized by our pairwise subject sharing recognizer and a

[5]http://www.tkl.iis.u-tokyo.ac.jp/~ynaga/jdepp/

	Recall	Precision	F-score
DEP w/o SSPN	0.259	0.744	0.385
DEP with SSPN	0.284	0.744	0.411
ADJ w/o SSPN	0.182	0.554	0.274
ADJ with SSPN	0.193	0.561	0.288
PNP w/o SSPN	0.034	0.757	0.064
PNP with SSPN	0.033	0.780	0.064
DEP+ADJ w/o SSPN	0.315	0.602	0.413
DEP+ADJ with SSPN	0.354	0.604	0.447
DEP+PNP w/o SSPN	0.293	0.746	0.421
DEP+PNP with SSPN	0.324	0.749	0.453
ADJ+PNP w/o SSPN	0.191	0.558	0.285
ADJ+PNP with SSPN	0.203	0.566	0.299
DEP+ADJ+PNP w/o SSPN	0.324	0.604	0.422
DEP+ADJ+PNP with SSPN	0.365	0.607	**0.456**

Table 3: Results of intra-sentential zero anaphora resolution between subjects

single subject is detected by our subject detector for one of the two predicates, then the subject fills the empty subject position of the other predicate. Note that in this baseline method, *transitive* subject propagation through more than one subject sharing relation is not performed. Also, if multiple subjects are detected for a predicate, we used the most likely subject to fill the subject position of the predicate, as in our method.

We conducted 5-fold cross-validation using the modified version of the NAIST Text Corpus presented in Section 5.1. In this evaluation, we used the 8,473 subject zero anaphors that refer to the *subject* antecedents (46% of all the *intra-sentential subject zero anaphora*, in which a subject zero anaphor refers to the antecedent that are not limited to subject) in the corpus. We measured the performance using recall, precision and F-score for each of the three types of subject sharing relations and their combinations. When combining more than one subject sharing recognizer in our method, we construct the SSPN using the subject sharing relations recognized by at least one of those recognizers for transitive subject propagation. On the other hand, in the baseline method, the SSPN was not constructed and zero anaphoric relations were identified using only the outputs of our subject detector and one of those recognizers.

The experimental results shown in Table 3 clearly demonstrate that the method with SSPNs for each type or a combination of the three types consistently outperformed that without SSPNs except for the PNP type. This result suggests that multi-step propagation of subjects through more than one subject sharing relation, as done in SSPNs, is an effective way to propagate a subject to a subject position that cannot be reached by a single subject sharing relation. Our results also show that the F-score is improved by combining different types of subject sharing relations, and the best F-score, 0.456, was achieved when we used all types of relations, i.e., in the case of DEP+ADJ+PNP with SSPNs.

7 Experiment 3: intra-sentential subject zero anaphora resolution

Finally, we evaluate the performance of intra-sentential subject zero anaphora resolution. In the previous section, we evaluated just a part of our method, i.e., from Step 1 to Step 4 presented in Section 3. In this section, we evaluate the whole method, i.e., from Step 1 to Step 5, against 18,324 subject zero anaphors, which are all subject zero anaphors annotated in our modified version of the NAIST Text Corpus. As a baseline, we employed Iida and Poesio (2011)'s method that was tuned for intra-sentential zero anaphora resolution. The baseline method solves the problems by applying only Step 5 in Section 3 to all the predicates.

Our results in Table 4 show that all the methods using either each type or a combination of the three types significantly outperformed the baseline[6]. The best performing method was DEP+PNP, which achieved 0.380 in F-score, which is 3.6%

[6]The significance was tested using McNemar's testing ($p < 0.01$).

	Recall	Precision	F-score
Baseline (Step 5)	0.345	0.344	0.344
+DEP with SSPN	0.388	0.363	0.375
+ADJ with SSPN	0.374	0.347	0.360
+PNP with SSPN	0.351	0.347	0.349
+DEP+ADJ with SSPN	0.399	0.355	0.376
+DEP+PNP with SSPN	0.394	0.366	**0.380**
+ADJ+PNP with SSPN	0.376	0.347	0.361
+DEP+ADJ+PNP with SSPN	0.401	0.356	0.377

Table 4: Results of intra-sentential subject zero anaphora resolution (Steps 1 to 5 vs. Step 5)

	Recall	Precision	F-score
Baseline (Step 5)	0.345	0.344	**0.344**
DEP with SSPN	0.131	0.744	0.223
ADJ with SSPN	0.089	0.561	0.154
PNP with SSPN	0.015	0.780	0.030
DEP+ADJ with SSPN	0.164	0.604	0.258
DEP+PNP with SSPN	0.150	0.749	0.250
ADJ+PNP with SSPN	0.094	0.566	0.161
DEP+ADJ+PNP with SSPN	0.169	0.607	0.264

Table 5: Results of intra-sentential subject zero anaphora resolution (Steps 1 to 4 vs. Step 5)

higher than the baseline. This suggests that our method exploiting subject sharing relations and SSPNs has a positive impact on accuracy of *general* intra-sentential zero anaphora resolution methods because about 84% of zero anaphors of general intra-sentential zero anaphora appear as subject zero anaphor in our corpus.

We also estimate how accurately the method using only the SSPNs evaluated in Section 6 resolves intra-sentential subject zero anaphora in comparison to the baseline method. The results are shown in Table 5 and demonstrate that the performance of all the methods without Step 5 does not reach that of the baseline method in F-score. However, they retain high precision that ranges from 60% to 75%, preserving more than 10% of the recall on the DEP, DEP+ADJ, DEP+PNP and DEP+ADJ+PNP methods. Actually, in some of the potential applications of zero anaphora resolution, such as information extraction, methods with high precision and low recall are preferable to ones with low precision and high recall. Our methods with SSPNs alone might be usable in such applications because of their high precision.

8 Conclusion

In this paper, we introduced a subject shared predicate network (SSPN), which is a network of predicates that are linked by subject sharing relations for resolving typical intra-sentential zero anaphora. In our zero anaphora resolution method, zero anaphoric relations are identified by propagating a subject through subject sharing *paths* in the SSPN. To construct SSPNs, we developed a novel method of pairwise subject sharing recognition using the local contexts that surround two predicates and demonstrated that it can accurately recognize subject sharing relations. We combined our method of intra-sentential zero anaphora resolution with Iida and Poesio (2011)'s method and achieved significantly better F-score than Iida and Poesio (2011)'s method alone.

As future work, we are planning to use commonsense knowledge, such as causality (Hashimoto et al., 2014) and script-like knowledge (Sano et al., 2014), that has been automatically acquired from big data for accurate subject sharing recognition to improve inter-sentential zero anaphora resolution for cases not focused on in this work.

References

Mohit Bansal and Dan Klein. 2012. Coreference semantics from web features. In *Proceedings of the*

50th Annual Meeting of the Association for Computational Linguistics, pages 389–398.

David Bean and Ellen Riloff. 2004. Unsupervised learning of contextual role knowledge for coreference resolution. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 297–304.

Chen Chen and Vincent Ng. 2013. Chinese zero pronoun resolution: Some recent advances. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1360–1365.

Barbara J. Grosz, Scott Weinstein, and Aravind K. Joshi. 1995. Centering: A framework for modeling the local coherence of discourse. *Computational Linguistics*, 21(2):203–225.

Masatsugu Hangyo, Daisuke Kawahara, and Sadao Kurohashi. 2013. Japanese zero reference resolution considering exophora and author/reader mentions. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 924–934.

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

Chikara Hashimoto, Kentaro Torisawa, Julien Kloetzer, Motoki Sano, István Varga, Jong-Hoon Oh, and Yutaka Kidawara. 2014. Toward future scenario generation: Extracting event causality exploiting semantic relation, context, and association features. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 987–997.

Yuta Hayashibe, Mamoru Komachi, and Yuji Matsumoto. 2011. Japanese predicate argument structure analysis exploiting argument position and type. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 201–209.

Thomas Hofmann. 1999. Probabilistic latent semantic indexing. In *Processing of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 50–57.

Ryu Iida and Massimo Poesio. 2011. A cross-lingual ILP solution to zero anaphora resolution. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 804–813.

Ryu Iida, Kentaro Inui, and Yuji Matsumoto. 2007a. Zero-anaphora resolution by learning rich syntactic pattern features. *ACM Transactions on Asian Language Information Processing*, Volume 6. Issue 4, Article 12.

Ryu Iida, Mamoru Komachi, Kentaro Inui, and Yuji Matsumoto. 2007b. Annotating a Japanese text corpus with predicate-argument and coreference relations. In *Proceedings of the ACL Workshop: 'Linguistic Annotation Workshop'*, pages 132–139.

Kenji Imamura, Kuniko Saito, and Tomoko Izumi. 2009. Discriminative approach to predicate-argument structure analysis with zero-anaphora resolution. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 85–88.

Hideki Isozaki and Tsutomu Hirao. 2003. Japanese zero pronoun resolution based on ranking rules and machine learning. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, pages 184–191.

Megumi Kameyama. 1986. A property-sharing constraint in centering. In *Proceedings of the 24th Annual Meeting of the Association for Computational Linguistics*, pages 200–206.

Jun'ichi Kazama and Kentaro Torisawa. 2008. Inducing gazetteers for named entity recognition by large-scale clustering of dependency relations. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 407–415.

Hiromi Nakaiwa and Satoshi Shirai. 1996. Anaphora resolution of Japanese zero pronouns with deictic reference. In *Proceedings of the 16th International Conference on Computational Linguistics*, pages 812–817.

Manabu Okumura and Kouji Tamura. 1996. Zero pronoun resolution in Japanese discourse based on Centering Theory. In *Proceedings of the 16th International Conference on Computational Linguistics*, pages 871–876.

Motoki Sano, Kentaro Torisawa, Julien Kloetzer, Chikara Hashimoto, István Varga, and Jong-Hoon Oh. 2014. Million-scale derivation of semantic relations from a manually constructed predicate taxonomy. In *Proceedings of the 25th International Conference on Computational Linguistics*, pages 1423–1434.

Ryohei Sasano, Daisuke Kawahara, and Sadao Kurohashi. 2008. A fully-lexicalized probabilistic model for Japanese zero anaphora resolution. In *Proceedings of the 22nd International Conference on Computational Linguistics*, pages 769–776.

Ryohei Sasano, Daisuke Kawahara, and Sadao Kurohashi. 2009. The effect of corpus size on case frame acquisition for discourse analysis. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 521–529.

Kazuhiro Seki, Atsushi Fujii, and Tetsuya Ishikawa. 2002. A probabilistic method for analyzing Japanese anaphora integrating zero pronoun detection and resolution. In *Proceedings of the 19th International Conference on Computational Linguistics*, pages 911–917.

Hirotoshi Taira, Sanae Fujita, and Masaaki Nagata. 2008. A Japanese predicate argument structure analysis using decision lists. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 523–532.

Vladimir N. Vapnik. 1998. *Statistical Learning Theory*. Adaptive and Learning Systems for Signal Processing, Communications, and Control. John Wiley & Sons.

Marilyn Walker, Sharon Cote, and Masayo Iida. 1994. Japanese discourse and the process of centering. *Computational Linguistics*, 20(2):193–233.

Yotaro Watanabe, Masayuki Asahara, and Yuji Matsumoto. 2010. A structured model for joint learning of argument roles and predicate senses. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 98–102.

Xiaofeng Yang, Jian Su, and Chew Lim Tan. 2005. Improving pronoun resolution using statistics-based semantic compatibility information. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 165–172.

Katsumasa Yoshikawa, Masayuki Asahara, and Yuji Matsumoto. 2011. Jointly extracting Japanese predicate-argument relation with Markov logic. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 1125–1133.

Naoki Yoshinaga and Masaru Kitsuregawa. 2009. Polynomial to linear: Efficient classification with conjunctive features. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1542–1551.

Koichiro Yoshino, Shinsuke Mori, and Tatsuya Kawahara. 2013. Predicate argument structure analysis using partially annotated corpora. In *Proceedings of the 6th International Joint Conference on Natural Language Processing*, pages 957–961.

Shanheng Zhao and Hwee Tou Ng. 2007. Identification and resolution of Chinese zero pronouns: A machine learning approach. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 541–550.

Estimation of Discourse Segmentation Labels from Crowd Data

Ziheng Huang
Department of Statistics
Columbia University
zh2220@columbia.edu

Jialu Zhong
Department of Computer Science
Columbia University
jialu.zhong@columbia.edu

Rebecca J. Passonneau
Center for Computational Learning Systems
Columbia University
becky@ccls.columbia.edu

Abstract

For annotation tasks involving independent judgments, probabilistic models have been used to infer ground truth labels from data where a crowd of many annotators labels the same items. Such models have been shown to produce results superior to taking the majority vote, but have not been applied to sequential data. We present two methods to infer ground truth labels from sequential annotations where we assume judgments are not independent, based on the observation that an annotator's segments all tend to be several utterances long. The data consists of crowd labels for annotation of discourse segment boundaries. The new methods extend Hidden Markov Models to relax the independence assumption. The two methods are distinct, so positive labels proposed by both are taken to be ground truth. In addition, results of the models are checked using metrics that test whether an annotator's accuracy relative to a given model remains consistent across different conversations.

1 Introduction

A single, spontaneous, spoken interaction can consist of multiple activities, such as to plan a future event, to complain about a past situation, or to carry out a transaction that might consist of subtasks. Speakers shift from one activity to the next with more or less awareness and explicit demarcation. To treat such conversational activities as a sequence of discrete units is a convenient oversimplification that is often resorted to (Bokaei et al., 2015; Galley et al., 2003; Passonneau and Litman, 1997). Systems that provide automated access to spoken language data often rely on segmentation of spoken discourse into sequential units for summarization (Wang and Cardie, 2012; Dielmann and Renals, 2005) or information retrieval (Ward et al., 2015). Research on the organization of spoken discourse also relies directly or indirectly on identification of such units to detect agreement among participants (Hillard et al., 2003; Somasundaran et al., 2007; Germesin and Wilson, 2009), multiparty meeting action items (Purver et al., 2007), decisions (Fernández et al., 2008), or answers to questions (Sun and Chai, 2007; Bosma, 2005). To support such research, there is a need for annotation methods to segment conversational interaction into sequential, multi-utterance units. We present and compare two methods to derive such data from crowdsourced annotations.

Crowdsourced annotation, where each item is labeled by a crowd of many independent annotators, is becoming more common in natural language processing. Examples include word sense (Bruce and Wiebe, 1999; Snow et al., 2008; Passonneau and Carpenter, 2014), named entities (Finin et al., 2010), and several other tasks in (Snow et al., 2008), including textual entailment. Three advantages to corpus annotation through application of a probabilistic model to crowdsourced labels, rather than reliance on interannotator agreement computed for a small number of trained annotators, are higher quality, lower cost, and a posterior probability for each ground truth label (Sheng et al., 2008; Snow et al., 2008; Passonneau and Carpenter, 2014). The latter serves as a confidence measure, which contrasts with interannotator agreement measures and with majority-voted labels, neither of which provides quality information for the ground truth labels on individual items. Previous work has demonstrated that model estimation of ground truth labels from crowd labels produces results superior to the crowd's majority vote, due to differences among annotators in the quality of their labels (Dawid and Skene, 1979; Snow et al., 2008; Passonneau and Carpenter, 2014). No previous work, however, provides model-based estimation of labels for sequential annotation from crowd labels.

For the discourse segmentation data presented here, annotators were presented with audio files of conversations and corresponding transcriptions into utterances. The annotation task was to identify each utterance that completes a discourse segment spanning one or more utterances, based on the speakers' conversational activities or intentions, as in (Passonneau and Litman, 1997). The annotations from y annotators for a conversation with x utterances can be represented as a $y \times x$ matrix, with cell values $n_{ij} \in \{0, 1\}$ to represent the binary segment boundary label assigned by annotator y_i at utterance x_j. Figure 1 illustrates part of such a matrix. The eight annotators for this conversation are on the y-axis and utterances 80 through 180 are on the

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 2190–2200,
Lisbon, Portugal, 17-21 September 2015.

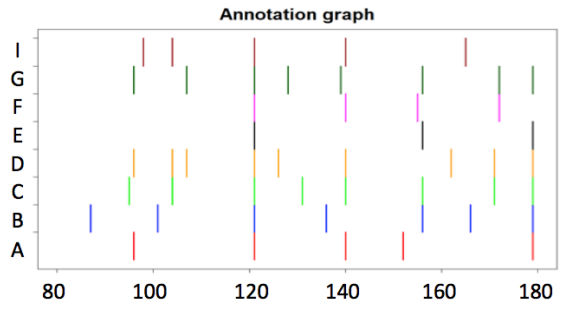


Figure 1: Annotation labels from eight annotators (A-G, I) on utterances 80 through 180 of a sample conversation. Vertical bars represent positive labels, with a different color for each annotator. Annotator H did not do this conversation.

x-axis. Colored bars represent positive labels, and each color represents a distinct annotator. The label distribution shown here is typical of our dataset: an annotator's positive labels are typically separated by several utterances, and annotators agree much more often on non-boundaries than on boundaries. Full consensus on a positive label is rare, but does occur. Here, all eight annotators assigned a positive label at utterance 120, six at utterance 178, and five at utterance 140.

Our work assumes that unobserved true labels condition the annotators' observed labels, and can be modeled as hidden states in a Markov-type process. Because an annotator rarely assigns positive labels for adjacent utterances, we assume that neither the true labels nor the observations are conditionally independent, and therefore are not generated by a simple Markov process. Our first model adapts the Double Chain Markov Model (Berchtold, 1999), designed to account for such cases. We then propose a second model that assumes that each annotator's labels are drawn from a Bernoulli distribution, that annotator performance is a parameter of the model, and that the state transitions are conditioned by an empirical distribution of discourse segment lengths. The two methods are quite distinct. Each thus serves as an evaluation of the other. The segment boundaries proposed by both models include all the majority vote cases, and in addition, cases voted on by a minority of relatively accurate annotators. We take segment boundaries proposed by both methods as ground truth. To further assess the results of the models, we assume that an annotator's accuracies should be consistent across the conversations she annotates.

2 Related Work

Previous work on annotation of discourse into linear segments has used a variety of methods to derive ground truth segment boundaries. In (Passonneau and Litman, 1997), seven annotators annotated narrative monologues for segments based on speaker intention. Agreement levels for ground truth boundaries were based on statistical significance using Cochran's Q. In (Galley et al., 2003), three annotators segmented the ICSI meeting corpus into topical units, and majority agreement was taken as ground truth. A functional segmentation of meetings from the AMI multiparty meeting corpus based on involved participants was segmented by one annotator and finalized by a second annotator (Bokaei et al., 2015). Task-based segmentation of patron-librarian interactions (Passonneau et al., 2011) measured agreement among two annotators using Krippendorff's Alpha at an average of 0.77 (Krippendorff, 1980). The annotation task here mostly closely resembles (Passonneau and Litman, 1997), and uses a similar number of annotators. No prior work, however, applies a probabilistic model to crowd labels for discourse segmentation.

Estimation of ground truth from crowd labels has been applied to many tasks, but is especially useful where judgments are subjective, making ground truth difficult to arrive at. Application areas include disease prevalence estimation (Albert and Dodd, 2008), identification of craters in images of Venus (Smyth et al., 1995), curation of biological data (Rzhetsky et al., 2009), computer vision (Whitehill et al., 2009), patient history (Dawid and Skene, 1979), and clinical reports (2010). Smyth et al. (1995), Rogers et al., and (2010) and Raykar et al. (2010) discuss the advantages of probabilistically annotated corpora over majority vote. Much of this work is motivated by the observation that annotators have different accuracies, and the fact that when annotators have known accuracies it can be shown that a majority of inaccurate annotators can be wrong (Raykar et al., 2010; Passonneau and Carpenter, 2014). Equally important, information from inaccurate annotators informs the model inference. For example, an inaccurate annotator might be biased towards label m whenever the true label is z.

Dawid and Skene (1979) present a joint model of true labels, observed labels, and annotator performance. Perhaps its first application to NLP data was the Bruce and Wiebe (1999) investigation of word sense. It has also been applied to more fine-grained word sense with a direct comparison to trained annotator labels in (Passonneau and Carpenter, 2014). Snow et al. (2008) showed that application of the same model to noisy crowd annotations produced data of equal quality to five distinct published gold standards. Hovy et al. (2013) apply a simple and effective model to identify untrustworthy annotators and test it on the same datasets used in (Snow et al., 2008). As they point out, when ties occur among an even number of annotators, it's necessary to resort to a tie-breaking procedure, e.g., for utterance 155 in Figure 1 where four annotators assign a positive label and four do not.

In experiments on an existing dataset of word sense annotation, Dligach et al. (2010) compare singly annotated data with doubly annotated adjudicated data, using trained annotators. They find that with the same amount of data, machine learning performance improves with the doubly annotated adjudicated data by

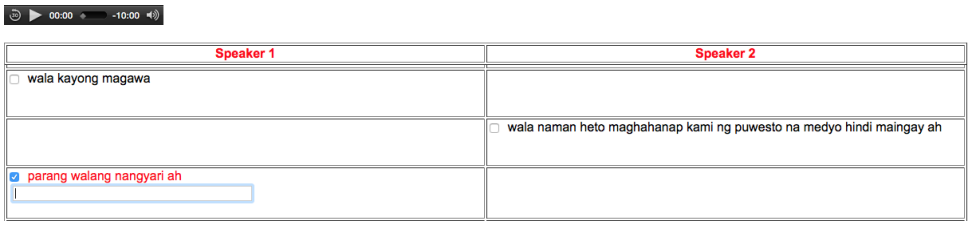


Figure 2: The annotation interface presented the audio control button on the upper left and the transcript below, with a scroll bar (not shown). Utterances from the two speakers are on the right and left sides, respectively. Each utterance had a checkbox; when selected, a textbox appeared to allow annotators to enter their segment descriptions.

a small amount, but that investing in more singly annotated labels leads to greater improvements. Their results on trained annotators, however, would not apply to our use case involving untrained annotators. In previous work, we found the cost per ground truth label of singly annotated data with trained annotators to be more than twice that for multiply annotated data with twenty untrained annotators (Passonneau and Carpenter, 2014). Half that many would have been sufficient for the Dawid & Skene model used there, which would reduce the cost by half again as much.[1]

3 Data and Annotation Task

The data consists of digital recordings and transcripts of fifty telephone calls between family members and friends who were native speakers of Tagalog. These were collected for the Babel program, sponsored by the Intelligence Advanced Research Projects Activity (IARPA). The calls ranged in length from about seven to ten minutes (μ = 9.67 minutes, σ=0.68 minutes). Transcripts provided by IARPA had an average of 364.66 utterances (min=239; max=475; σ=60.80).

The annotations were collected using Amazon Mechanical Turk. The task name and instructions were in English. The instructions were provided through a short video and text. Proficiency in Tagalog was assessed through a vocabulary test. Those who passed the vocabulary test were paid to do an initial annotation so we could ensure they understood the task. The initial task was based on a short Tagalog conversation that had been translated, annotated by a bilingual speaker of Tagalog and English, and verified by Passonneau. Annotators who understood the task and whose labels and descriptions seemed reasonable were admitted into the pool of annotators. A pool of nine annotators completed the qualifications. Each conversation was annotated by at least five annotators. Altogether, annotators assigned 5,567 labels to 164,097 utterances. Annotators' segments had a mean length of 21.85 utterances with a high standard deviation (σ = 19.32).

The interface designed for the annotation task is shown in Figure 2. Through the interface, annotators could read the transcript of a recorded conversation, and could play, pause or stop the audio. Each utterance had a checkbox for assigning a positive label if the annotator judged it to be the end of a segment. As shown, selection of a checkbox opened a text box for the annotator to enter a brief description of the segment. Table 1 in section 8 illustrates the descriptions assigned by six annotators to several segments.

4 Assumptions

Given the many labels from annotators, our goal is to estimate a ground truth label for each utterance position, where the label values represent a binary classification of segment boundaries. Our two models each assume there is a hidden *true* label that conditions an annotator's observed labels, and that can be estimated from the observed labels. How well the estimated ground truth fits the data thus depends on how well the model assumptions accord with the phenomenon of interest. The models do not account for annotator differences in the level of granularity they apply; cf. the contrast between lumpers and splitters in taxonomic classification of the natural world (Branch, 2014). Further, neither model takes linguistic features into account that annotators consider in deciding on segments, such as speaker attitude towards utterance content or speaker role in the conversational activity (Niekrasz and Moore, 2009). We find, however, much agreement between the two models on the proposed segment boundaries, and leave for future work the question of whether more complex models could accoount for differences in granularity or utterance features.

As discussed in section 2, we assume that annotators are not equally accurate, and that a probabilistic model based on the distribution of observed labels can do better than majority vote. Inspired by the type of probabilistic model proposed in (Dawid and Skene, 1979) and extended in (Bruce and Wiebe, 1999; Passonneau and Carpenter, 2014), annotator accuracy is a parameter of our second model. As described in detail in subsequent sections, the two models proposed here rely on distinct assumptions and inference methods. They nevertheless propose many of the same labels. We take each model to provide independent evidence for the ground truth labels, thus the final labels are those voted on by both models.

[1] Twenty labels per item were collected in order to provide tight estimates for item difficulty. This, however, requires a model with a parameter for item difficulty, which had not yet been implemented for this data.

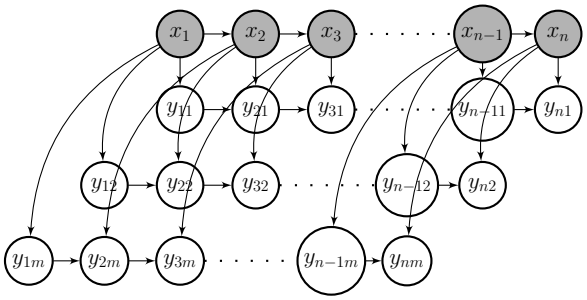


Figure 3: Graphical model of Double Chain Dynamic Hidden Markov Model for a conversation with m annotators and n utterances. The x_t are the hidden states, and the y_{jl} are the observed labels from annotator l at utterance j.

In addition, we assume that annotators' accuracies should be relatively consistent across conversations, and we measure how well each model's results support this assumption. We base the assumption on the observation that the annotation task is the same for all conversations, and an annotator's relative ability to do the task should not change significantly. The annotators all had the same initial training, and did about the same number of conversations. The conversations all had similar conditions of collection, similar participants, and similarly mundane topics and conversational activities that most annotators would be familiar with. The subjects that were discussed included parties, watching tv, siblings, money, jobs, spouses, medical issues, birthdays, and so on.

5 Double Chain Dynamic Hidden Markov Model

The first model we propose combines the Double Chain Markov Model (Berchtold, 1999) and dynamic Bayesian networks (Martinez and Sucar, 2008). The double chaining involves the dependence of observations on immediately prior observations. Figure 3 shows that for all $y_{tl}, t \geq 2$, observation y_{tl} depends on observation $y_{(t-1)l}$. The emission matrix at the first utterance x_1 is thus a 2×2 matrix, while all subsequent emission matrices are $2 \times 2 \times 2$. As in (Martinez and Sucar, 2008), the observed states can be regarded as a composition of m independent chains, where m is the number of annotators for the conversation. Also, the l^{th} annotator's observation at the t^{th} utterance depends not only on the same hidden state x_t, but also on the last observation $y_{(t-1)l}$.

Assume in a conversation, there are m annotators and n utterances. The model $\Theta = \{\pi, \gamma, A, B\}$ can be described as follows:

- a set of hidden states, i.e the true labels: $x_t \in \{0, 1\}$, $t \in \{1, 2, \ldots, n\}$. $x_t = 1$ represents the t^{th} utterance is a true boundary and 0 otherwise;
- a set of observed variables: $y_{tl} \in \{0, 1\}$, $l \in \{1, 2, \ldots, m\}$ annotators, $t \in \{1, 2, \ldots, n\}$ utterances. $y_{tl} = 1$ represents that the l^{th} annotator annotates t^{th} utterance to a true boundary and 0 otherwise;
- Θ is a vector of parameters. To be more specific, the elements are:
  - the probability of the initial hidden state: $\pi_{x_1}, x_1 \in \{0, 1\}$. Note $\pi_0 + \pi_1 = 1$.
  - the probabilities of the initial emission matrix. Note that the initial emission matrix is a 2×2 matrix: $\gamma_l \in \{c_{x_1, y_{1l}}\}$, $x_1, y_{1l} \in \{0, 1\}, l \in \{1, 2, \ldots, m\}$. For annotator l, $c_{x_1, y_{1l}}$ is the probability of emitting from x_1 to y_{1l}.
  - the transition matrix between hidden states, $A \in \{a_{x_{t-1}, x_t}\}$, $x_{t-1}, x_t \in \{0, 1\}$, $t \in \{2, 3, \ldots, n\}$. a_{x_{t-1}, x_t} is the probability of transitioning from x_{t-1} to x_t.
  - the emission matrices, $B_l \in \{b_{x_t, y_{(t-1)l}, y_{tl}}\}$, $x_t, y_{(t-1)l}, y_{tl} \in \{0, 1\}, l \in \{1, 2, \ldots, m\}$, $t \in \{2, 3, \ldots, n\}$. Note that the emission matrix is a $2 \times 2 \times 2$ matrix as each observed state depends on current hidden state as well as the previous observation, i.e., $b_{x_t, y_{(t-1)l}, y_{tl}}$ is the probability of emitting from x_t to y_{tl} and transitioning from $y_{(t-1)l}$ to y_{tl}.

A graphical sketch of the DCD HMM model is shown in Figure 3. The target function $F = P(x, y|\Theta)$ is:

$$F = \pi_{x_1} \prod_{l=1}^{m} c_{x_1, y_{1l}} \prod_{t=2}^{n} a_{x_{t-1}, x_t} \prod_{l=1}^{m} \prod_{t=2}^{n} b_{x_t, y_{(t-1)l}, y_{tl}}$$

We can derive a marginal distribution over y and have the *likelihood* as:

$$L(\Theta) = P(y|\Theta) = \sum_{x} P(x, y|\Theta)$$

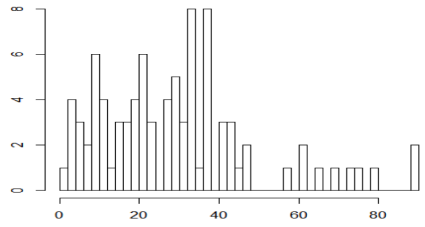
Our goal is to find the parameters (Θ) that maximize the above function. Bayes Net Toolbox for Matlab (Murphy, 2001) is used for the inference. Expectation-Maximization (EM) with Junction Tree inference for the E-step is used for learning the parameters. The Junction Tree Algorithm is a method to calculate marginals by propagation on the graph. It runs as follows: 1) Initialize: Pick a proper root and initialize all variables; 2) Collect: Pass message from each child of a node through separators to the parent node and update the node with collected evidence; 3) Distribute: Send back message to each child of the node through separators and update the child with distributed evidence; 4) Normalize: Normalize cliques connected by a separator so they agree with each other: e.g., for {AB} and {BC}, if we have $\sum_A \{AB\} = \sum_C \{AB\}$, propagation is complete.

After convergence from EM, junction tree propagation is again used for inference, and the model produces

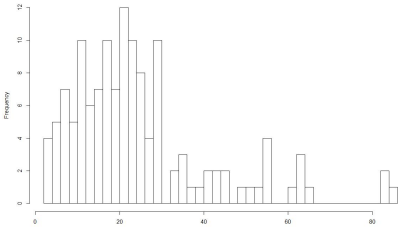
a probability for each ground truth label. We take the label to be positive if the posterior probability is greater than 0.5; as shown in section 8, probabilities tend to be very high or very low.

6 Interval-dependent HMM

The second model, Interval-dependent HMM, imposes a constraint on the state transitions between two positive labels based on the empirical distribution of intervals between observed labels. Initially, we examined known distributions. The Poisson, for example, represents the probability of events in an interval as an average rate. The model based on the Poisson did not perform particularly well. Histograms of interval sizes from different conversations have similar shapes, however, as illustrated in Figure 4. Although more of the probability is towards 20 to 40 utterances in Figure 4a, and between 15 and 35 utterances in Figure 4b, we assume these small differences in the two distributions are mainly due to sampling variation. As discussed in preceding sections, the model we present here assumes that the probability of a true label at time t_i is a function of the interval length $t_i - t_j$, where t_j is the most recent time of a true label. The observed data for all annotators on all conversations provides a set of time intervals to construct the empirical distribution.



(a) First sample conversation



(b) Second sample conversation

Figure 4: Histograms of interval lengths between all observed labels for two conversations.

To assess whether we have sufficient data to reliably construct the empirical distribution, we performed fifty iterations of random divisions of the data into two samples. For each pair of samples, we measured the maximum distance between pairs of cumulative distribution function (CDF) curves, and used the two-sample Kolmogorov-Smirnov test to measure the goodness of fit of the two curves. Figure 5 shows an example comparison of two CDF curves which have a maximum gap

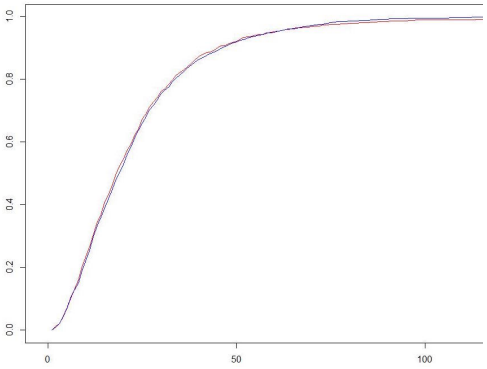


Figure 5: A plot of two CDF curves for a random split of the data. The curves are almost identical; the maximum gap is 0.0175. A two sample K-S test has a p-value of 0.79.

of 0.0175 and a K-S p-value of 0.7866. The mean maximum distance between pairs of curves was 0.014, with a standard deviation of 0.009, both of which are quite small. The p-values for the K-S test ranged from 0.4 to 0.96, which fail to reject the hypothesis that the pairs of samples are from the same distribution. While the two measures are not conclusive evidence that we have sufficient data to construct the empirical distribution, they are supportive. Further, reliance on estimates of the empirical distribution are preferable to a known distribution that does not fit the data, such as the Poisson.

The model can be described as follows:

- the observations $Y_{ij} \in 0, 1, i \in 1, 2, \cdots, N, j \in 1, 2, \cdots, J$;
- the true labels $Z_i \in 0, 1, i \in 1, 2, \cdots, N$;
- the 2×2 annotator performance matrices B_j;
- the initial state probability π

Given N utterances, J annotators, the initial state probability π and four cells in each annotator's performance matrix B_j, where B_{j11} represents the true positives (the probability that given a ground truth positive label, annotator a_j assigns a positive label), B_{j10} represents false negatives, B_{j01} represents false positives, and B_{j00} represents true negatives. $\pi = 1$ is the probability that the first hidden state is a boundary and $\pi = 2$ means it is not. Our objective is to find the parameter vector $\theta = (\pi, B)$ that maximize the likelihood $P(Y|\theta)$, and to use this θ to estimate the true labels Z.

To solve:

$$\underset{\theta}{\text{Argmax}} \log\left[P(Y|\theta)\right] = \underset{\theta}{\text{Argmax}} \log\left[\sum_{Z} P(Y, Z|\theta)\right]$$

we use expectation-maximization (EM).

E step First, we should find the lower bound of our optimization object: $\underset{\theta}{\text{Argmax}} \log\left[\sum_{Z} P(Y, Z|\theta)\right]$; by

Jensen's inequality, we have:

$$\log\left[\sum_Z P(Y,Z|\theta)\right] = \log\left[\sum_Z \frac{P(Y,Z|\theta)}{Q_\theta(Z)} Q_\theta(Z)\right] \geq \sum_Z Q_\theta(Z)\log\left[\frac{P(Y,Z|\theta)}{Q_\theta(Z)}\right]$$

$Q_\theta(Z)$ is a function of θ which satisfies that $\sum_Z Q_\theta(Z) = 1$. The equality holds if and only if

$$\frac{P(Y,Z|\theta)}{Q_\theta(Z)} = c \qquad \text{for all Z}$$

Note that c is a constant. In the E step we need to calculate the Q function to maintain the equality. By straightforward algebra, we get $Q_\theta = P(Z|Y,\theta)$.

M step In this part, we should maximize our lower bound:

$$\underset{\theta}{\text{Argmax}} \sum_Z Q_{\theta^{(n)}}(Z)\log\left[\frac{P(Y,Z|\theta)}{Q_{\theta^{(n)}}(Z)}\right]$$

Since $\log[Q_\theta(Z)]$ is a term not related to θ, $P(Z|Y,\theta) \propto P(Z,Y|\theta)$. Our problem becomes:

$$\underset{\theta}{\text{Argmax}} \sum_Z P(Y,Z|\theta^{(n)})\log[P(Y,Z|\theta)]$$

$\theta^{(n)}$ is the parameter we get from the last iteration, and the Q function is fixed in this M step. We cannot use the forward-backward algorithm to optimize, because the first order Markov property does not hold: $P(Z_i = 1)$ is a function of the last positive label $Z_j = 1$ at time j such that $j < i$, and for all k such that $j < k < i$, $Z_k = 0$. To make use of the Markov property, we rely on a hidden variable U_i to save the interval length between i and j. The hidden parameter space is then expanded to $X_t = (Z_t, U_t)$, where U_t denotes the size of the interval between the current position t_i and the most recent t_j with a positive label. If the true label $Z_{t_i} = 0$, then $U_{t_i} = t_i - t_j$, and if $Z_{t_i} = 1$, then $U_{t_i} = 0$. This gives $t+1$ possible states for each t: the t states for $Z_t = 0$, and one state for $Z_t = 1$.

In this problem, given a length N conversation, there are $N+1$ hidden states at each moment. $X_t = 1$ means $(Z_t = 1, U_t = 0)$, $X_t = 2$ means $(Z_t = 0, U_t = 1)$, $X_t = 3$ means $(Z_t = 0, U_t = 2)$, and so on.

The transition matrix at each t for the cases represented by $P(X_t = k|X_{t-1} = l)$, which is with size $(t+1)\times(t+2)$, will necessarily be very sparse. For example, given an empirical function $f(n) = P(x = n|x \geq n)$, the transition matrix from $t = 4$ to $t = 5$ can be written:

$$\begin{pmatrix} f(1) & 1-f(1) & 0 & 0 & 0 & 0 \\ f(2) & 0 & 1-f(2) & 0 & 0 & 0 \\ f(3) & 0 & 0 & 1-f(3) & 0 & 0 \\ f(4) & 0 & 0 & 0 & 1-f(4) & 0 \\ f(5) & 0 & 0 & 0 & 0 & 1-f(5) \end{pmatrix}$$

After this transformation, X_{t+1} is independent to all X_k for any $k < t$ provided that X_t is given. With X as the new hidden state, we can estimate the HMM parameter by adding some constraints. Replacing the Z in the object function with X, we can rewrite the object function as:

$$\begin{aligned}
&\sum_X P(Y,X|\theta^{(n)})\log[P(Y,X|\theta)] \\
=&\sum_X P(Y,X|\theta^{(n)})\left[\text{Log}P(X_1) + \sum_{t=1}^{N-1}\log P(X_{t+1}|X_t) + \sum_{t=1}^{N}\log P(Y_t|X_t)\right] \\
=&\sum_X P(Y,X|\theta^{(n)})\left[\log[\pi_{X_1}] + \sum_{t=1}^{N-1}\log\left[A_{X_t,X_{t+1}}\right] + \sum_{t=1}^{N}\log[B_{X_t,Y_t}]\right]
\end{aligned}$$

The object is split into three independent parts: the first part is for the initial state distribution π, the second for the transition probability matrix A, and the third is the emission matrix B. For the first term, because in the moment $t = 1$, X_t can just be 1 or 2, we have the optimization problem:

$$\begin{aligned}
&\underset{\pi}{\text{Argmax}} \sum_{i=1}^{2} P(Y, X_1 = i|\theta^{(n)})\log[\pi_i] \\
&\text{s.t} \qquad \pi_1 + \pi_2 = 1 \\
&\qquad \pi_3 = \pi_4 = ... = \pi_{N+1} = 0
\end{aligned}$$

We can easily solve this optimization problem by the Lagrange multiplier: we have the update formula:

$$\pi_1^{(n+1)} = P(X_1 = 1|Y,\theta^{(n)})$$

$$\pi_2^{(n+1)} = P(X_1 = 2|Y,\theta^{(n)})$$

$$\pi_i^{(n+1)} = 0 \qquad \text{for} \quad i > 2$$

Both can be solved by the traditional forward-backward algorithm after this transformation. $\theta^{(n)}$ is the parameter set we get from the last iteration.

The second term can be ignored, since we use the known empirical distribution as the transition matrix; it is therefore a constant term.

The third term can be rewritten as:

$$\begin{aligned}
&\sum_{t=1}^{N} P(Y,X|\theta^{(n)})\log[B_{X_t,Y_t}] \\
=&\sum_{t=1}^{N}\sum_{i=1}^{N+1}\sum_{j=1}^{J}\sum_{k=0}^{1} I(Y_{t,j} = k)P(X_t = i, Y|\theta^{(n)}) \\
&\log[B_{j,i,k}]
\end{aligned}$$

So our problem is:

$$\underset{B}{\text{Argmax}} \sum_{t=1}^{N} \sum_{j=1}^{J} \sum_{i=1}^{N+1} \sum_{k=0}^{1} I(Y_{t,j} = k)$$
$$P(X_t = i, Y|\theta^{(n)}) \log B_{j,i,k}$$
$$\text{s.t} \quad \sum_{k=0}^{1} B_{j,i,k} = 1 \qquad \text{For all i, j}$$
$$B_{j,i_1,k} = B_{j,i_2,k} \qquad \text{For all j, k and } i_1, i_2 \geq 2$$

The second constraint here means that, if this is not a true boundary, a given annotator j will have the same emission matrix no matter what U is. This optimization can also be solved by Lagrange multiplier, where the update formula is as follows. For $i = 1$:

$$B_{j,i=1,k}^{(n+1)} = \frac{\sum_{t=1}^{N} P(Y, Z_t = 1|\theta^{(n)}) I(Y_{t,j} = k)}{\sum_{t=1}^{N} P(Y, Z_t = 1|\theta^{(n)})}$$

For any $i \neq 1$, the matrix B is the same given j:

$$B_{j,i\neq 1,k}^{(n+1)} = \frac{\sum_{t=1}^{N} P(Y, Z_t \neq 1|\theta^{(n)}) I(Y_{t,j} = k)}{\sum_{t=1}^{N} P(Y, Z_t \neq 1|\theta^{(n)})}$$

Now we have the update function for θ. After convergence, we will have π and B. It is straightforward to transfer these parameters for the new space to our original HMM problem. This completes the M step.

7 Model Checking

No ground truth labels are available to evaluate our models. We check the model results, however, in three ways. One, we consider labels proposed by both models to be stronger evidence than labels proposed only by one. Two, we measure the consistency of annotators on the assumption that the same annotator should have relatively consistent performance across conversations, relative to the same model. The third way we can check the models is to examine the descriptive labels that annotators assign to segments to determine whether descriptions for the same segment from different annotators are consistent. In this section, we describe the two consistency metrics.

We measure how consistently the label quality from annotator a_i surpasses that for $a_j, i \neq j$, for all pairs of annotators using a metric to measure inconsistency and strength of inconsistency (I&SI) (de Vries, 1998). We also apply a variant we refer to as Directional Consistency (DC), which takes into account how often annotator a_i surpasses annotator a_j. To measure annotators' performance relative to the inferred true labels, we use F-score, the harmonic mean of recall and precision. Recall is the ratio of true positives to the sum of true positives and false negatives; precision is the ratio of true positives to the sum of true positives and false positives. A square matrix of annotator *dominance* is first constructed to give a count of how many conversations there are where a_i has a higher F measure than $a_j, i \neq j$. A linear dominance ordering $>$ of all annotators has an inconsistency score I that is incremented by 1 for each pair of annotators where $a_i > a_j$ in the linear ordering and $(a_i, a_j) \neq (a_j, a_i)$ in the matrix. I is minimal if no other ordering has fewer inconsistencies. The strength of the inconsistency IS for a linear ordering is incremented by the difference in rank between a_i and a_j for every inconsistent pair in the linear ordering. The I&SI method finds an ordering that minimizes I and SI. To check the results of our models, we compare the I&SI value of the dominance matrix associated with the model results against a simulated random matrix. If the model results are significantly more consistent than the simulation, the model produces a consistent ranking of annotators.

We propose a Directional Consistency index ($DC \in [0, 1]$) which considers the number of times a_i has a higher F measure than a_j (Leiva et al., 2008). Where X is the dominance matrix:

$$DC = \frac{\sum_{i=1}^{n} \sum_{j=i+1}^{n} |x_{ij} - x_{ji}|}{N}$$
$$N = \sum_{i=1}^{n} \sum_{j=1, j\neq i}^{n} x_{ij}$$

DC values closer to zero indicate less consistency in differences among annotators, and the converse for values closer to 1. High DC values for the results of our models thus indicates better performance of the model in predicting consistent annotator behavior.

8 Results and Model Checking

The results consist of the true labels assigned by each model to each conversation, and estimates of the annotators' performance relative to the model's ground truth labels. Note that as the conversation is not a parameter of either model, after estimation of the empirical distribution of segment lengths, the data for each conversation is treated separately.

To provide a concrete illustration, we first review the data for a typical conversation. Table 1 presents the segments derived from both models for an extract from conversation 945, which had six annotators, and the annotator's segment descriptions. We selected a conversation with an even number of annotators to illustrate that an arbitrary choice must be made, given a 50/50 vote split. We take ties as true positives to provide a more conservative baseline. We first discuss this example conversation in detail to explain the kinds of cases where the models differ from majority voting. Then we present summary results on the fifty conversations for majority voting compared with the two models.

In Table 1 a description at n gives the annotator's interpretation of the kind of conversational activity that ends with n. When annotators agree on a positive label that ends a segment, they might not agree on the utterance that starts the segment, so their descriptions will not necessarily be about the same segments. From

Utt	Description
191	C: S1 and S2 are talking about the status of their children's studies I: S1 and S2 are talking about their children's education
216	A: S1 and S2 spoke about their children's studies E: S2 then shared that he's going to Laguna-Muntinlupa tomorrow. S1 said that S2 has many orders. S2 shared that he's striving hard in order for her kids to graduate college. . . . The two laughed at each other about S1's children not getting traits from S1 I: S1 and S2 are talking about who their children took after
217	C: S1 and S2 are joking about the traits their children got from them D: They are talking about S1s daughter that she is good at academics and that she got her being smart from her mom and nothing from S1. S1 said even if she got nothing from him as long as she will just study hard its okay
241	A: S1 and S2 spoke about their time of sleeeping C: S1 and S2 tell each other what time they usually go to sleep D: They are talking about the time that they go to sleep. S2 said sometimes by ten, eleven or twelve midnight. S1 said sometimes he goes out one in the morning. Sometimes he goes to sleep at ten or eleven in the evening too

Table 1: Annotator descriptions for conversation 945 for a sequence of four segment boundaries hypothesized by both models. A description from annotator a_i at utterance n indicates a_i assigned a positive label, and gives the annotator's interpretation of the kind of interaction that ended at utterance n. Underlined utterance numbers indicate cases where at least six annotators assigned a positive label.

the table, however, we see a a pattern that is consistent for most of the data: abstracting over the descriptions gives a good indication of what's going on in the segments that are defined by the positive labels assigned by both models. The descriptions from C and I at 191, for example, describe the first segment as the speakers talking about their children's education. A's similar description at 216 indicates that A ended the segment later than C and I. E and I describe the second segment as being about the children, including who they take after. C's description about who the children take after occurs at a later utterance. The third segment goes into detail about the children's traits, and the fourth is about what time the speakers go to sleep.

Across all fifty conversations, ID HMM assigns more positive labels than the majority, and DCD HMM assigns more than ID HMM. Totals for each labeling

Method	Total
Majority	683
ID HMM	991
DCD HMM	1324

Table 2: Total positive labels assigned by each method.

Utt	Annotators	DCD HMM	ID HMM
11	2 (A,I)	1.00	0.99
42	3 (B,E,I)	1.00	1.00
43	3 (A,C,D)	1.00	1.00
67	6 (A,B,C,D,E,I)	1.00	1.00
114	2 (A,C)	1.00	0.98
126	2 (D,E)	1.00	1.00
127	1 (C)	**0.63**	*0.02*
144	2 (A,D)	1.00	0.98
147	2 (C,I)	0.90	0.65
191	2 (C,I)	0.90	0.73
216	3 (A,E,I)	1.00	1.00
217	2 (C,D)	1.00	0.66
241	3 (A,C,D)	1.00	1.00
276	3 (B,C,D)	1.00	1.00
282	1 (A)	**1.00**	*0.29*
300	*5* (A,B,D,E,I)	1.00	1.00
356	2 (C,I)	**0.98**	*0.27*
357	4 (A,B,D,E)	1.00	1.00

Table 3: Comparison of positive predictions from majority voting (N=8, underlined; ties are taken as positive), DDC HMM (N=18), and the ID HMM (N=15) for conversation 945. Probabilities in bold are for boundaries proposed by only one model; italics are for probabilities below the 0.5 threshold to be considered true boundaries.

method are in Table 2. Wherever the majority vote predicts a true label, both models always do. If ID HMM posits a boundary at an utterance, DCD HMM also does, but DCD HMM predicts additional ones. Because all the ID HMM labels are also identified by DCD HMM, these are the final labels we propose.

Table 3 shows the positive labels predicted for conversation 945 by majority vote, and by our two models. Column one is the utterance number, and again, underlining indicates cases where the voted baseline would assign a positive label. Column two lists the annotators who assigned a positive label, and columns three and four show the posteriors assigned by the two models; for all utterances not listed in the table, the posteriors are below 0.5. Low posteriors for ID HMM where DCD HMM proposed a boundary are in italics.

Ann	Maj	DCD HMM	ID HMM
A	0.68	0.71	0.68
B	0.57	0.36	0.38
C	0.40	0.63	0.46
D	0.59	0.56	0.55
E	0.73	0.50	0.52
I	0.43	0.55	0.49

Table 4: F-measure for annotators in conversation 945 for majority vote labels and both models; recall that the true labels for each model are different, and that DCD HMM hypothesizes more true labels than ID HMM.

For each model, the annotator can be ranked by the F-scores relative to the model predictions. When one of the models agrees with a minority of annotators,

Model	I&SI	DC
Majority	I=1, SI=3, p=0.008	p=0.0600
DCD HMM	I=2, SI=5, p=0.02	p=0.0014
ID HMM	I=0, SI=0, p=0	p=0.0001

Table 5: Consistency of annotators

the minority consists of the annotators considered by the model to have higher performance, as given by F-measure. The three sets of F-scores for the six annotators in 945 are shown in Table 4. Annotator performance given the two models is very similar; the Pearson correlation is 0.80. F-scores based on the majority baseline, however, do not correlate well with DCD ($\rho = -0.5$) or ID ($\rho = 0.49$). In eight cases where DCD posits a true label for conversation 945, and only 2 annotators voted positive, the pair never includes B, the least accurate annotator by DCD (see Table 4), and always includes one of the top three annotators (A,C,D). In the two cases where only one annotator voted positive, it was A or C, one of the two top DCD HMM annotators. Both models consider A to be the best annotator. C is relatively good in the DCD HMM model and relatively poor in the ID HMM model.

I&SI tests whether there exists a linear ordering of the annotators such that their relative performance across conversations is consistent. DC tests whether an ordering $a_i > a_j$ is based on relatively more frequent dominance of a_i over a_j. Table 5 shows that majority vote and the two models produce results that lead to high I&SI consistency, based on the statistically significant p-values. The majority vote p-value for DC, however, is not statistically significant. By the more stringent DC measure, the labels from the two HMM variants are superior to the majority vote labels.

The list of descriptions from annotators at utterance n represents the semantics of the hypothesized segment ending at n. Semantic consistency for a given segment serves as another check on the output of the model, because the human descriptions of the activity within the segment do not conflict. In general, this is the case for both models, but less so for DCD HMM. For conversation 945 illustrated in Table 3, there are three positive labels proposed by DCD HMM that are missing from the ID HMM predictions. These are at 127 where only annotator C had a positive label, 282 where only annotator A had a positive label, and 356 where annotators C and I had a positive label. Annotators B, C and D, for example, describe a segment ending at utterance 276 as the speakers discussing Facebook, whereas annotator A locates the end of the Facebook segment at utterance 282. The DCD HMM model posits a true label at 276 but not at 282, in contast to ID HMM.

9 Discussion

The two models for estimating ground truth labels from crowd labels advance previous work on probabilistic models for annotation by handling sequential data. We have argued that for our data, the Markov assumption must be relaxed. The two models handle this in distinct ways. The first model assumes that each state can be decomposed into multiple aspects, and that states and observations are conditionally dependent on the previous point in time. The second model builds in a parameter for annotator performance, as in previous work that adopts the Dawid and Skene (1979) model. Both assign more ground truth labels than majority voting, and avoid the problem with the majority vote method of ties where there are an even number of annotators. The results of the two models are very similar, but DCD HMM hypothesizes more boundaries, and therefore ranks some annotators differently.

Here we check the models by comparing them to each other, through analysis of each annotator's consistency across multiple conversations, and through inspection of the semantics of annotators' descriptions. Our future work will use the models generatively to predict a subset of the data for a given annotator, based on a model fit to all but the held out data. To do so, we would extend the models with an additional parameter for the conversation, to account for the observation that while all conversations seem to fit the same empirical distribution, there are differences across conversations.

10 Conclusion

Annotation and machine learning of discourse segmentation covers several types of units, including topical segments (Galley et al., 2003), meeting units in which action items are identified or decisions made (Purver et al., 2007; Fernández et al., 2008), transaction subtasks for ordering library books (Passonneau et al., 2014), or speaker involvement (Bokaei et al., 2015). This work relies on manual transcription, and draws on many sources of knowledge for machine learned models, including turn-taking, prosody, and linguistic features. The segmentation annotation can be linear (Galley et al., 2003; Bokaei et al., 2015; Passonneau and Litman, 1997; Passonneau et al., 2014) or hierarchical (Purver et al., 2007; Fernández et al., 2008; Passonneau et al., 2011). The differences in methods and results across this body of work, points to a need for more datasets for research on the organization of discourse into activity units. The results presented here support this research agenda by providing a reliable and cost-effective method to estimate ground truth discourse segment labels from crowd labels.

Acknowledgments

The authors thank Bob Carpenter for discussions during the early stages of the data analysis, and for helpful feedback on the paper. We thank the IARPA Babel program manager, Mary Harper, for giving us permission to annotate the Babel data.

References

Paul S. Albert and Lori E. Dodd. 2008. On estimating diagnostic accuracy from studies with multiple raters and partial gold standard evaluation. *Journal of the American Statistical Association*, 103(481):61–73.

André Berchtold. 1999. The Double Chain Markov Model. *Communications in Statistics: Theory and Methods*, 28(348):2569–2589.

Mohammad Hadi Bokaei, Hossein Sameti, and Yang Liu. 2015. Linear discourse segmentation of multiparty meetings based on local and global information. *IEEE Transactions on Audio, Speech and Language Processing*, 23(11):1879–1891.

Wauter Bosma. 2005. Extending answers using discourse structure. In *RANLP Workshop on Crossing Barriers in Text Summarization Research*.

Glenn Branch. 2014. Whence lumpers and splitters? `http://ncse.com/blog/2014/11/whence-lumpers-splitters-0016004`, December.

Rebecca F. Bruce and Janyce M. Wiebe. 1999. Recognizing subjectivity: a case study of manual tagging. *Natural Language Engineering*, 1(1):1–16.

A. P. Dawid and A. M. Skene. 1979. Maximum likelihood estimation of observer error-rates using the EM algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):20–28.

Han de Vries. 1998. Finding a dominance order most consistent with a linear hierarchy: a new procedure and review. *Animal Behavior*, 55:827–843.

Alfred Dielmann and Steve Renals. 2005. Multistream dynamic Bayesian network for meeting segmentation. In Samy Bengio and Herv Bourlard, editors, *Machine Learning for Multimodal Interaction*, volume 3361 of *Lecture Notes in Computer Science*, pages 76–86. Springer Berlin Heidelberg.

Dmitriy Dligach, Rodney D. Nielsen, and Martha Palmer. 2010. To annotate more accurately or to annotate more. In *Proceedings of the Fourth Linguistic Annotation Workshop (LAW IV)*, pages 64–72.

Raquel Fernández, Matthew Frampton, Patrick Ehlen, Matthew Purver, and Stanley Peters. 2008. Modelling and detecting decisions in multi-party dialogue. In *Proceedings of the 9th SIGdial Workshop on Discourse and Dialogue*.

Tim Finin, Will Murnane, Anand Karandikar, Nicholas Keller, Justin Martineau, and Mark Dredze. 2010. Annotating named entities in twitter data with crowdsourcing. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, CSLDAMT '10, pages 80–88, Stroudsburg, PA, USA. Association for Computational Linguistics.

Michel Galley, Kathleen R. McKeown, Eric Fosler-Lussier, and Hongyan Jing. 2003. Discourse segmentation of multi-party conversation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 562–569, Sapporo, Japan, July. Association for Computational Linguistics.

Sebastian Germesin and Theresa Wilson. 2009. Agreement detection in multiparty conversation. In *Proceedings of the 2009 International Conference on Multimodal Interfaces*, ICMI-MLMI '09, pages 7–14, New York, NY, USA. ACM.

Dustin Hillard, Mari Ostendorf, and Elizabeth Shriberg. 2003. Detection of agreement vs. disagreement in meetings: Training with unlabeled data. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology: Companion Volume of the Proceedings of HLT-NAACL 2003–short Papers - Volume 2*, NAACL-Short '03, pages 34–36, Stroudsburg, PA, USA. Association for Computational Linguistics.

Dirk Hovy, Taylor Berg-Kirkpatrick, Ashish Vaswani, and Eduard Hovy. 2013. Learning whom to trust with MACE. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 1120–1130, Atlanta, Georgia, June. Association for Computational Linguistics.

Klaus Krippendorff. 1980. *Content analysis: An introduction to its methodology*. Sage Publications, Beverly Hills, CA.

David Leiva, Antonio Solanas, and Lluis Salafranca. 2008. Testing reciprocity in social interactions: A comparison between the directional consistency and skew-symmetry statistics. *Behavior Research Methods*, 40(2):626–634.

Miriam Martinez and L. Enrique Sucar. 2008. Learning dynamic Naive Bayesian classifiers. In *Proceedings of the Twenty-First International FLAIRS Conference*, pages 655–659.

Kevin Murphy. 2001. Bayes Net toolbox for Matlab. *Computing Science and Statistics*, 33(2):1024–1034.

John Niekrasz and Johanna Moore. 2009. Participant subjectivity and involvement as a basis for discourse segmentation. In *Proceedings of the SIGDIAL 2009 Conference*, pages 54–61, London, UK, September. Association for Computational Linguistics.

Rebecca J. Passonneau and Bob Carpenter. 2014. The benefits of a model of annotation. *Transactions of the Association for Computational Linguistics*, 2:311326.

Rebecca J. Passonneau and Diane Litman. 1997. Discourse segmentation by human and automated means. *Computational Linguistics*, 23.1:103–139. Special Issue on Empirical Studies in Discourse Interpretation and Generation.

Rebecca J. Passonneau, Irene Alvarado, Phil Crone, and Simon Jerome. 2011. PARADISE-style evaluation of a human-human library corpus. In *Proceedings of the SIGDIAL 2011 Conference*, pages 325–331, Portland, Oregon, June. Association for Computational Linguistics.

Rebecca J. Passonneau, Boxuan Guan, Cho Ho Yeung, Yuan Du, and Emma Conner. 2014. Aspectual properties of conversational activities. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 228–237, Philadelphia, PA, U.S.A., June. Association for Computational Linguistics.

Matthew Purver, John Dowding, John Niekrasz, Patrick Ehlen, Sharareh Noorbaloochi, and Stanley Peters. 2007. Detecting and summarizing action items in multi-party dialogue. In *Proceedings of the 8th SIGdial Workshop on Discourse and Dialogue*.

Vikas C. Raykar, Shipeng Yu, Linda H. Zhao, Gerardo Hermosillo Valadez, Charles Florin, Luca Bogoni, and Linda Moy. 2010. Learning from crowds. *Journal of Machine Learning Research*, 11:1297–1322.

Simon Rogers, Mark Girolami, and Tamara Polajnar. 2010. Semi-parametric analysis of multi-rater data. *Statistical Computing*, 20:317–334.

Andrey Rzhetsky, Hagit Shatkay, and W. John Wilbur. 2009. How to get the most out of your curation effort. *PLoS Computational Biology.*, 5(5):1–13.

Victor S. Sheng, Foster Provost, and Panagiotis G. Ipeirotis. 2008. Get another label? Improving data quality and data mining using multiple, noisy labelers. In *Proceedings of the Fourteenth ACM International Conference on Knowledge Discovery and Data Mining (KDD)*.

Padhraic Smyth, Usama Fayyad, Michael Burl, Pietro Perona, and Pierre Baldi. 1995. Inferring ground truth from subjectively-labeled images of Venus. In *Advances in Neural Information Processing Systems 7*, pages 1085–1092. MIT Press.

Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Y. Ng. 2008. Cheap and fast - but is it good? Evaluating non-expert annotations for natural language tasks. In *Proceedings of Empirical Methods in Natural Language Processing (EMNLP)*, pages 254–263, Honolulu.

Swapna Somasundaran, Josef Ruppenhofer, and Janyce Wiebe. 2007. Detecting arguing and sentiment in meetings. In *Proceedings of the 8th SIGdial Workshop on Discourse and Dialogue (SIGdial 07)*, page 2634.

Mingyu Sun and Joyce Y. Chai. 2007. Discourse processing for context question answering based on linguistic knowledge. *Know.-Based Syst.*, 20(6):511–526, August.

Lu Wang and Claire Cardie. 2012. Focused meeting summarization via unsupervised relation extraction. In *Proceedings of the 13th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, SIGDIAL '12, pages 304–313, Stroudsburg, PA, USA. Association for Computational Linguistics.

Nigel G. Ward, Steven D. Werner, Fernando Garcia, and Emilio Sanchis. 2015. A prosody-based vector-space model of dialog activity for information retrieval. *Speech Communication*, 28:85–96.

Jacob Whitehill, Paul Ruvolo, Tingfan Wu, Jacob Bergsma, and Javier Movellan. 2009. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. In *Proceedings of the 24th Annual Conference on Advances in Neural Information Processing Systems*.

Comparing Word Representations for Implicit Discourse Relation Classification

Chloé Braud
ALPAGE, Univ Paris Diderot
& INRIA Paris-Rocquencourt
75013 Paris - France
chloe.braud@inria.fr

Pascal Denis
MAGNET, INRIA Lille Nord-Europe
59650 Villeneuve d'Ascq - France
pascal.denis@inria.fr

Abstract

This paper presents a detailed comparative framework for assessing the usefulness of unsupervised word representations for identifying so-called implicit discourse relations. Specifically, we compare standard one-hot word pair representations against low-dimensional ones based on Brown clusters and word embeddings. We also consider various word vector combination schemes for deriving discourse segment representations from word vectors, and compare representations based either on all words or limited to head words. Our main finding is that denser representations systematically outperform sparser ones and give state-of-the-art performance or above without the need for additional hand-crafted features.

1 Introduction

Identifying discourse relations is an important task, either to build a discourse parser or to help other NLP systems such as text summarization or question-answering. This task is relatively straightforward when a discourse connective, such as but or because, is used (Pitler and Nenkova, 2009). The identification becomes much more challenging when such an overt marker is lacking, and the relation needs to be inferred through other means. In (1), the presence of the pair of verbs (rose,tumbled) triggers a *Contrast* relation. Such relations are extremely pervasive in real text corpora: they account for about 50% of all relations in the Penn Discourse Treebank (Prasad et al., 2008).

(1) [Quarterly revenue **rose** 4.5%, to $2.3 billion from $2.2 billion]$_{arg1}$ [For the year, net income **tumbled** 61% to $86 million, or $1.55 a share]$_{arg2}$

Automatically classifying implicit relations is difficult in large part because it relies on numerous factors, ranging from syntax, and tense and aspect, to lexical semantics and even world knowledge (Asher and Lascarides, 2003). Consequently, a lot of previous work on this problem have attempted to incorporate some of these information into their systems. These assume the existence of syntactic parsers and lexical databases of various kinds, which are available but for a few languages, and they often involve heavy feature engineering (Pitler et al., 2009; Park and Cardie, 2012). While acknowledging this knowledge bottleneck, this paper focuses on trying to predict implicit relations based on easily accessible lexical features, targeting in particular simple word-based features, such as pairs like (rose,tumbled) in (1).

Most previous studies on implicit relations, going back to (Marcu and Echihabi, 2002), incorporate word-based information in the form of word pair features defined across the pair of text segments to be related. Such word pairs are often encoded in a *one-hot* representation, in which each possible word pair corresponds to a single component of a very high-dimensional vector. From a machine learning perspective, this type of sparse representation makes parameter estimation extremely difficult and prone to overfitting. It also makes it difficult to achieve any interesting semantic generalization. To see this, consider the distance (e.g., Euclidean or cosine) induced by such representation. Assuming for simplicity that one characterizes each pair of discourse segments via their main verbs, the corresponding one-hot encoding for the pair (rose,tumbled) would be at equal distance from the synonymic pair (went_up,lost) and the antonymic pair (went_down,gained), as all three vectors are orthogonal to each others.

Various attempts have been made at reducing sparsity of lexical features. Recently, Ruther-

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 2201–2211,
Lisbon, Portugal, 17-21 September 2015.

ford and Xue (2014) proposed to use Brown clusters (Brown et al., 1992) for this task, in effect replacing each token by its cluster binary code. These authors conclude that these denser, cluster-derived representations significantly improve the identification of implicit discourse relations and report the best performance to date using also additional features. Unfortunately, their claim is somewhat weakened by the fact that they fail to compare the use of their cluster word pairs against other types of word representations, including one-hot encodings of word pairs or other low-dimensional word representations. This work also leaves other important questions open. In particular, it is unclear whether all word pairs constructed over the two discourse segments are truly informative and should be included in the model. Given that word embeddings capture latent syntactic and semantic information, yet another important question is to which extent the use of these representations dispenses us from using additional hand-crafted syntactic and semantic features.

This paper fills these gaps and significantly extends the work of (Rutherford and Xue, 2014) by explicitly comparing various types of word representations and vector composition methods. Specifically, we investigate three well-known word embeddings, namely Collobert and Weston (Collobert and Weston, 2008), hierarchical log-bilinear model (Mnih and Hinton, 2007) and Hellinger Principal Component Analysis (Lebret and Collobert, 2014), in addition to Brown cluster-based and standard one-hot representations. All these word representations are publicly available for English and can be easily acquired for other languages just using raw text data, thus alleviating the need for hand-crafted lexical databases. This makes our approach easily extendable to resource-poor languages. In addition, we also investigate the issue of which specific words need to be fed to the model, by comparing using just pairs of verbs against all pairs of words, and how word representations should be combined over discourse segments, comparing component-wise product against simple vector concatenation.

2 Word Representations

A word representation associates a word to a mathematical object, typically a high-dimensional vector in $\{0,1\}^{|\mathcal{V}|}$ or $\mathbb{R}^{|\mathcal{V}|}$, where $\mathcal{V}$ is a base vocabulary. Each dimension of this vector corresponds to a feature which might have a syntactic or semantic interpretation. In the following, we review different types of word representations used in NLP.

2.1 One-hot Word Representations

Given a particular NLP problem, the crudest and yet most common type of word representation consists in mapping each word into a one-hot vector, wherein each observed word corresponds to a distinct vector component. More formally, let $\mathcal{V}$ denote the set of all words found in the texts and w a particular word in $\mathcal{V}$. The one-hot representation of w is the d-dimensional indicator vector, noted $\mathbb{1}_w$, such that $d = |\mathcal{V}|$: that is, all of this vector's components are 0's but for one 1 component corresponding to the word's index in $\mathcal{V}$. It is easy to see that this representation is extremely sparse, and makes learning difficult as it mechanically blows up the parameter space of the model.

2.2 Clustering-based Word Representations

An alternative to these very sparse representations consists in learning word representations in an unsupervised fashion using clustering. An example of this approach are the so-called *Brown clusters* induced using the Brown hierarchical clustering algorithm (Brown et al., 1992) with the goal of maximizing the mutual information of bigrams. As a result, each word is associated to a binary code corresponding to the cluster it belongs to. Given the hierarchical nature of the algorithm, one can create word classes of different levels of granularity, corresponding to bit codes of different sizes. The less clusters, the less fine-grained the distinctions between words but the less sparsity. Note that this kind of representations also yields one-hot encodings but on a much smaller vocabulary size (i.e., the number of clusters). Brown clusters have been used for several NLP tasks, including NER, chunking (Turian et al., 2010), parsing (Koo et al., 2008) and implicit discourse relation classification (Rutherford and Xue, 2014).

2.3 Dense Real-Valued Representations

Another approach to induce word representations from raw text is to learn *distributed* word representations (aka word embeddings), which are dense, low-dimensional, and real-valued vectors. These are typically learned using neural language models (Bengio et al., 2003). Each dimension corresponding to a latent feature of the word that captures paradigmatic information. An example of such

embeddings are the so-called Collobert and Weston embeddings (Collobert and Weston, 2008). The embeddings are learned discriminatively by minimizing a loss between the current n-gram and a corrupted n-gram whose last word comes from the same vocabulary but is different from the last word of the original n-gram. Another example are the Hierarchical log-bilinear embeddings (Mnih and Hinton, 2007) induced using a probabilistic and linear neural model, with a hierarchical principle used to speed up the model evaluation. The embeddings are obtained by concatenating the embeddings of the $n-1$ words of a n-gram and learning the embedding of the last word.

A final approach is based on the assumption that words occurring in similar contexts tend to have similar meanings. Building word *distributional* representations is done by computing the raw cooccurrence frequencies between each word and the $|\mathcal{D}|$ words that serve as context, with $\mathcal{D}$ generally smaller than the overall vocabulary, then applying some transformation (e.g. TF-IDF). As $|\mathcal{D}|$ is generally too large to form a tractable representation, a dimensionality reduction algorithm is used to end up with $p \ll |\mathcal{D}|$ dimensions. Like for distributed representations, we end up with a dense low-dimensional real-valued vector for each word. A recent example of such approach is the Hellinger PCA embeddings of (Lebret and Collobert, 2014) which were built using Principal Component Analysis based on Hellinger distance as dimensionality reduction algorithm. An important appeal of these representations is that they are much less time-consuming to train than the ones based on neural language models while allowing similar performance (Lebret and Collobert, 2014).

3 Segment Pair Representations

We now turn to the issue of combining the word representations as described in section 2 into composite vectors corresponding to implicit discourse classification instances. Schematically, the representations employed for pairs of discourse segments differ along three main dimensions. First, we compare the use of a single word per segment (roughly, the two main verbs) against that of all the words contained in the two segments. Second, we compare the use of sparse (i.e., one-hot) vs. dense representations for words. As discussed, Brown cluster bit representations are a special (i.e., low-dimensional) version of one-hot encoding. Third, we use two different types of combinations of word vectors to yield segment vector representations: concatenation and Kronecker product. The proposed framework is therefore much more general than the one given in previous work such as (Rutherford and Xue, 2014).

3.1 Notation

Our classification inputs are pairs of text segments, the two arguments of the relation to be predicted. Let $S_1 = \{w_{1_1}, \ldots, w_{1_n}\}$ denote the n words that make up the first segment and $S_2 = \{w_{2_1}, \ldots, w_{2_m}\}$ the m words in the second segment. That is, we regard segments as bags of words. Let $\mathcal{V}$ again denote the word vocabulary, that is the set of all words found in the segments. Sometimes, we will find it useful to refer to a particular subset of $\mathcal{V}$. Let $\mathrm{head}(\cdot)$ refer to the function that extracts the head word of segment S,[1] and $\mathcal{V}_h \subseteq \mathcal{V}$ the set of head words. As our goal is to compare different feature representations, we define Φ as a generic feature function mapping pairs of segments to a d-dimensional real vector:

$$\begin{aligned}\Phi: \ & \mathcal{V}^n \times \mathcal{V}^m \rightarrow \mathbb{R}^d \\ & (S_1, S_2) \mapsto \Phi(S_1, S_2)\end{aligned}$$

The goal of learning is to acquire for each relation a linear classification function $f_{\boldsymbol{w}}(\cdot)$, parametrized by $\boldsymbol{w} \in \mathbb{R}^d$, mapping $\Phi(S_1, S_2)$ into $\{-1, +1\}$.

Recall that $\mathbb{1}_w$ refers to the d-dimensional one-hot encoding for word $w \in \mathcal{V}$. Let us also denote by $\oplus$ and $\otimes$ the vector concatenation operator and the Kronecker product, respectively. Note that doing a Kronecker product on two vectors $\boldsymbol{u} \in \mathbb{R}^m$ and $\boldsymbol{v} \in \mathbb{R}^n$ is equivalent to doing the outer product $\boldsymbol{u}\boldsymbol{v}^\top \in \mathbb{R}^{m \times n}$. Finally, the $\mathrm{vec}(\cdot)$ operator converts a $m \times n$ matrix into an $mn \times 1$ column vector by stacking its columns.

3.2 Representations based on head words

One of the simplest representation one can construct for a pair of segments (S_1, S_2) is to consider only their head words: $h_1 = \mathrm{head}(S_1)$ and $h_2 = \mathrm{head}(S_2)$. In this simple scenario, two main questions that remain are: (i) which vector representations do we use for h_1 and h_2, and (ii) how do we combine these representations. An important criterion for word vector combination is that they retain text ordering information between text segments which really matters for this task.

[1] Head extraction will be detailed in section 4.1.

Thus, inverting the order between two main verbs (e.g., push and fall) will often lead to distinct discourse relation being inferred, as some relations are asymmetric (e.g., *Result* or *Explanation*).

One-hot representations Starting again with the simplest case, one can use the one-hot encodings corresponding to the two head words, $\mathbb{1}_{h_1}$ and $\mathbb{1}_{h_2}$ respectively, and combine them using either concatenation or product, leading to our two first feature mappings:

$$\Phi_{h,\mathbb{1},\oplus}(S_1, S_2) = \mathbb{1}_{h_1} \oplus \mathbb{1}_{h_2}$$
$$\Phi_{h,\mathbb{1},\otimes}(S_1, S_2) = \text{vec}(\mathbb{1}_{h_1} \otimes \mathbb{1}_{h_2})$$

Note that $\Phi_{h,\mathbb{1},\oplus}(S_1, S_2)$ lives in $\{0,1\}^{2|\mathcal{V}_h|}$ and $\Phi_{h,\mathbb{1},\otimes}$ in $\{0,1\}^{|\mathcal{V}_h|^2}$. The latter representation amounts to assigning one 1 component for each pair of words in $\mathcal{V}_h \times \mathcal{V}_h$, and is the sparsest representation one can construct from head words alone. In some sense, it is also the most expressive in that we learn one parameter for each word pair, hence capturing interaction between words across segments. By contrast, $\Phi_{h,\mathbb{1},\oplus}(S_1, S_2)$ doesn't explicitly model word interaction across discourse segments, treating each word in a given segment (left or right) as a separate dimension.

Dense representations Alternatively, one can represent head words through their real low-dimensional embeddings. Let $\boldsymbol{M}$ denote a $n \times p$ real matrix, wherein the i^{th} row corresponds to the p-dimensional embedding of the i^{th} word of $\mathcal{V}_h$, with $p \ll |\mathcal{V}_h|$.[2] Using this notation, one can derive the word embeddings of the head words h_1 and h_2 from their one-hot representations using simple matrix multiplication: $\boldsymbol{M}^\top \mathbb{1}_{h_1}$ and $\boldsymbol{M}^\top \mathbb{1}_{h_2}$, respectively. Concatenation and product yield two new feature mappings, respectively:

$$\Phi_{h,\boldsymbol{M},\oplus}(S_1, S_2) = \boldsymbol{M}^\top \mathbb{1}_{h_1} \oplus \boldsymbol{M}^\top \mathbb{1}_{h_2}$$
$$\Phi_{h,\boldsymbol{M},\otimes}(S_1, S_2) = \text{vec}(\boldsymbol{M}^\top \mathbb{1}_{h_1} \otimes \boldsymbol{M}^\top \mathbb{1}_{h_2})$$

These new representations live in a much lower dimensional real spaces: $\Phi_{h,\boldsymbol{M},\oplus}(S_1, S_2)$ lives in $\mathbb{R}^{2p}$ and $\Phi_{h,\boldsymbol{M},\otimes}(S_1, S_2)$ in $\mathbb{R}^{p^2}$.

3.3 Representations based on all words

The various segment-pair representations that we derived from pairs of head words can be generalized to the case in which we keep all the words in each segment. The additional issue in this context is in the combination of the different word vector representations within and across the two segments, and that of normalizing the segment vectors thus obtained. For simplicity, we assume that the representation for each segment is computed by summing over the pairs of words vectors composing the segments.

One-hot representations Following this approach and recalling that S_1 contains n words, while S_2 has m words, one can construct one-hot encodings for segment pairs as follows:

$$\Phi_{all,\mathbb{1},\oplus}(S_1, S_2) = \sum_i^n \sum_j^m \mathbb{1}_{w_{1_i}} \oplus \mathbb{1}_{w_{2_j}}$$
$$\Phi_{all,\mathbb{1},\otimes}(S_1, S_2) = \sum_i^n \sum_j^m \text{vec}(\mathbb{1}_{w_{1_i}} \otimes \mathbb{1}_{w_{2_j}})$$

If used without any type of frequency thresholding, these mappings result in very high-dimensional feature representations living in $\mathbb{Z}_{\geq 0}^{2|\mathcal{V}|}$ and $\mathbb{Z}_{\geq 0}^{|\mathcal{V}|^2}$, respectively. Interestingly, note that the feature mapping $\Phi_{all,\mathbb{1},\otimes}(S_1, S_2)$ corresponds to the standard segment-pair representation used in many previous work, as (Marcu and Echihabi, 2002; Park and Cardie, 2012).

Dense representations We can apply the same composition operations to denser representations, yielding two new mappings:

$$\Phi_{all,\boldsymbol{M},\oplus}(S_1, S_2) = \sum_{i,j}^{n,m} \boldsymbol{M}^\top \mathbb{1}_{w_{1_i}} \oplus \boldsymbol{M}^\top \mathbb{1}_{w_{2_j}}$$
$$\Phi_{all,\boldsymbol{M},\otimes}(S_1, S_2) = \sum_{i,j}^{n,m} \text{vec}(\boldsymbol{M}^\top \mathbb{1}_{w_{1_i}} \otimes \boldsymbol{M}^\top \mathbb{1}_{w_{2_j}})$$

Like their head word versions, these vectors live in $\mathbb{R}^{2p}$ and $\mathbb{R}^{p^2}$, respectively.

Vector Normalization Normalization is important as unnormalized composite vectors are sensitive to the number of words present in the segments. The first type of normalization we consider is to simply convert our vector representation into vectors on the unit hypersphere: this is achieved by dividing each vector by its L_2 norm.

Another type of normalization is obtained by inverting the order of summation and concatenation in the construction of composite vectors. Instead

[2] For now, we assume that $n = \mathcal{V}_h$ which is unrealistic. See section 4.1 for a discussion of unknown words.

of summing over concatenated pairs of word vectors across the two segments, one can first sum individual word vectors within each segment, then concatenate the two segment vectors. One can thus use mapping $\Phi'_{all,\mathbb{1},\oplus}$ in lieu of $\Phi_{all,\mathbb{1},\oplus}$:

$$\Phi'_{all,\mathbb{1},\oplus}(S_1, S_2) = \sum_i^n \mathbb{1}_{w_{1_i}} \oplus \sum_j^m \mathbb{1}_{w_{2_j}}$$

It should be clear that $\Phi'_{all,\mathbb{1},\oplus}$ provides a normalized version of $\Phi_{all,\mathbb{1},\oplus}$ as this latter mapping amounts to weighted versions of the former:

$$\Phi_{all,\mathbb{1},\oplus}(S_1, S_2) = m \sum_i^n \mathbb{1}_{w_{1_i}} \oplus n \sum_j^m \mathbb{1}_{w_{2_j}}$$

4 Experiment Settings

Through the comparative framework described in section 3, our objective is to assess the usefulness of different vectorial representations for pairs of discourse segments. Specifically, we want to establish whether dense representations are better than sparse ones, and whether certain word pairs are more relevant than others, which resource and which combination schemes are more adapted to the task, and, finally, whether standard features derived from external databases are still relevant in the presence of dense representations.

4.1 Feature Set

Our main features are primarily lexical in nature and based on surface word forms. These are defined either on *all* words used in the relation arguments or only on their heads.

Head Extraction Heads of discourse segments are first extracted using Collins syntactic head rules[3]. In order to retrieve the semantic predicate, we define a heuristic which looks for the past participle of auxiliaries, the adjectival or nominal attribute of copula, the infinitive complementing "have to" forms and the first head of coordination conjunctions. In case of multiple subtrees, we look for the head of the first independent clause, or, failing that, of the first phrase.

Word Representations We use either one-hot encodings or use word embeddings to build denser representations as described in section 3. The Brown clusters (*Brown*), Collobert-Weston (*CnW*) representations, and the hierarchical log-bilinear (*HLBL*) embeddings correspond to the versions implemented in (Turian et al., 2010)[4]. They have been built on Reuters English newswire with case left intact. We test versions with 100, 320, 1000 and 3,200 clusters for *Brown*, with 25, 50, 100 and 200 dimensions for *CnW* and with 50 and 100 dimensions for *HLBL*. The Hellinger PCA (*H-PCA*) embeddings come from (Lebret and Collobert, 2014)[5] and have been built over the entire English Wikipedia, the Reuters corpus and the Wall Street Journal with all words in lower case. The vocabulary corresponds to the words that appear at least 100 times and normalized frequency is computed with the 10,000 most frequent words as context words. We test versions with 50, 100 and 200 dimensions for *H-PCA*. The coverage of each resource is presented in table 1.

	# words	# missing words	
		All words	Head words
HLBL	246,122	5,439	171
CnW	268,810	5,638	171
Brown	247,339	5,413	171
H-PCA	178,080	7,042	190

Table 1: Word embeddings and Brown clusters lexicon coverage.

When presenting our results, we distinguish between systems based on one-hot encoding built from raw tokens (*one-hot*) or Brown clusters (*Brown*). We group the systems that use embeddings under *Embed.* When relevant, we indicate the number of dimensions (e.g. *Brown 3,200* is the system using Brown clusters with 3,200 clusters). We use the symbols defined in section 3 to represent the operation linking the arguments representations (e.g. *one-hot* $\oplus$ corresponds to the transformation defined by $\Phi_{h,\mathbb{1},\oplus}$ when using heads and by $\Phi_{all,\mathbb{1},\oplus}$ when using all words).

Vocabulary Sizes For one-hot encoding, the case is left intact. We ignore the unknown words when using the Brown clusters following (Rutherford and Xue, 2014). For the word embeddings, we use the mean of the vectors of all words.

In order to give an idea of the sparsity of the one-hot encodings, note that we have $|\mathcal{V}| = 33,649$ different tokens considering all implicit examples without filtering. The Brown clusters

[3] https://github.com/jkkummerfeld/nlp-util

[4] http://metaoptimize.com/projects/wordreprs/

[5] http://lebret.ch/words/

merge these tokens into $3,190$ codes (for $3,200$ clusters), 393 ($1,000$ clusters), 59 (320 clusters) or 16 (100 clusters). For heads, we count $5,615$ different tokens which correspond to $1,988$ codes for $3,200$ clusters and roughly the same number for the others. For the dense representations, the vocabulary size is twice the number of dimensions of the embedding, thus from 50 to 400, or the square of this number, thus from 625 to $40,000$.

Other Features We experiment with additional features commonly used for this task: productions rules, average verb phrases length, Levin verb classes, modality, polarity, General Inquirer tags, number, first last and first three words. These feature templates are well described in (Pitler et al., 2009; Park and Cardie, 2012). They all correspond to a one-hot encoding, except average verb phrases length which is continuous. We thus concatenate these features to the lexical ones.

4.2 Model

We use the same classification algorithm for comparing all the described feature configurations. Specifically, we train a Maximum Entropy (ME) classifier (aka, logistic regression).[6] As in previous studies, we build one binary classifier for each relation. In order to deal with class imbalance, we use a sample weighting scheme: each sample receives a weight inversely proportional to the frequency of its class in the training set. We optimize the hyper-parameters of the algorithm (i.e., the regularization norm: L_1 or L_2, and its strength) and a filter on the features on the development set, based on the F1 score. Note that filtering is pointless for purely dense representations. We test statistical significancy of the results using t-test and Wilcoxon test on a split of the test set in 20 folds.

Previous studies have tested several algorithms generally concluding that Naive Bayes (NB) gives the best performance (Pitler et al., 2009; Rutherford and Xue, 2014). We found that, when the hyper-parameters of ME are well tuned, the performance are comparable to NB if not better. Note that NB cannot be used with word embeddings representations as it does not handle negative value. Concerning the class imbalance issue, the downsampling scheme is the most spread since (Pitler et al., 2009) but it has been shown that oversampling and instance weighting lead to better performance (Li and Nenkova, 2014a).

[6] We use the implementation provided in Scikit-Learn (Pedregosa et al., 2011), available at: `http://scikit-learn.org/dev/index.html`.

Relation	Train	Dev	Test
Temporal	665	93	68
Contingency	3,281	628	276
Comparison	1,894	401	146
Expansion	6,792	1,253	556
Total	12,632	2,375	1,046

Table 2: Number of examples in train, dev, test.

4.3 Penn Discourse Treebank

We use the Penn Discourse Treebank (Prasad et al., 2008), a corpus annotated at the discourse level upon the Penn Treebank, giving access to a gold syntactic annotation, and composed of articles from the Wall Street Journal. Five types of examples are distinguished: implicit, explicit, alternative lexicalizations, entity relations, and no relation. Each example could carry multiple relations, up to four for implicit ones, and the relations are organized into a three-level hierarchy.

We keep only true implicit examples and only the first annotation. We focus on the top level relations which correspond to general categories included in most discursive frameworks. Finally, in order to make comparison easier, we choose the most spread split of the data, used in (Pitler et al., 2009; Park and Cardie, 2012; Rutherford and Xue, 2014) among others. The amount of data for training (sections $2-21$), development ($00, 01, 23, 24$) and evaluation ($21, 22$) is summarized in table 2.

5 Results

We first discuss the models that use only lexical features, defined either over all the words that appear in the arguments or only the head words. We then compare our best performing lexical configurations with the ones that also integrate additional standard features, and to state-of-the-art systems.

5.1 Word Pairs over the Arguments

Our first finding in this setting is that feature configurations that employ unsupervised word representations almost systematically outperform those that use raw tokens. This is shown in the left part of table 3. Although the optimal word representation differs from one relation to another, it is always a dense representation that achieves the

	All words				Head words only			
Representation	*Temp.*	*Cont.*	*Compa.*	*Exp.*	*Temp.*	*Cont.*	*Compa.*	*Exp.*
One-hot $\otimes$	21.14	50.36	34.80	59.43	11.96	43.24	17.30	**69.21**
One-hot $\oplus$	23.04	51.31	34.06	58.96	23.01	49.40	29.23	59.08
Brown $3,200$ $\otimes$	20.38	50.95	34.85	61.23	11.98	43.77	16.75	68.76
Best *Brown* $\otimes$	15.52	**53.85****	30.90	61.87	22.91	45.74	25.83	68.76
Best *Brown* $\oplus$	**27.96****	49.48	31.19	**67.42****	21.84	47.36	27.52	61.38
Best *Embed.* $\otimes$	22.97	52.76**	**34.99**	61.87	**23.88**	**51.29**	**30.59**	58.59
Best *Embed.* $\oplus$	25.98*	52.50	33.15	60.17	22.48	47.48	29.82	57.45

Table 3: F1 score for systems using all words and only heads for *Temporal* (*Temp.*), *Contingency* (*Cont.*), *Comparison* (*Compa.*) and *Expansion* (*Exp.*). * $p \leq 0.1$, ** $p \leq 0.05$ compared to *One-hot* $\otimes$ with t-test and Wilcoxon ; for head words, all the improvements observed against *One-hot* $\otimes$ are significant.

best F1 score. Our baselines correspond to multiplicative and additive one-hot encodings, noted *One-hot* $\otimes$ and *One-hot* $\oplus$, the former being the most commonly used in previous work. These are strong baselines in the sense they have been obtained after optimizing a frequency cut-off. Our best systems based on dense representations correspond to significant improvements in terms of F1 of about 8 points for *Expansion*, 7 points for *Temporal* and 3.5 for *Contingency*. The gains for *Comparison* are not statistically significant. All these results are obtained using the normalization to unit vectors possibly combined to the concatenation-specific normalization described in §3.3.

Comparing Dense Representations The best results are obtained using the Brown clusters (*Brown*) showing that this resource merges words in a way that is relevant to the task. Strikingly, the Brown configuration used in (Rutherford and Xue, 2014) (*One-hot Brown* $3,200$ $\otimes$) does not do better than the raw word pair baselines, except for *Expansion*. Recall that these authors did not explicitly provide this comparison. While doing a little worse, word embeddings (*Embed.*) also yield significant improvements for *Temporal* and *Contingency*, and smaller improvements for the others. This suggests that, even if they were not built based on discursive criteria, the latent dimensions encode word properties that are relevant to their rhetorical function. The superiority of Brown clusters over word embeddings is in line with the conclusions in (Turian et al., 2010) for two rather different NLP tasks (i.e., NER and chunking).

Turian et al. (2010) showed that the optimal word embedding is task dependent. Our experiments suggest that it is relation dependent: the best scores are obtained with *HLBL* for *Temporal*, *CnW* for *Contingency*, *H-PCA* for *Expansion* and *CnW* (Best *Embed.* $\otimes$) and *HPCA* (Best *Embed.* $\oplus$) for *Comparison*. This again demonstrates that these four relations have to be considered as four distinct tasks. Identifying temporal or causal links is indeed sensitive to very different factors, the former relying more on temporal expressions and temporal ordering of events whereas the latter relies on lexical and encyclopaedic knowledge on events. We think that this also explains that the behavior of the F1 against the optimal number of clusters for *Expansion* really differs from what we observed for the other relations: only 100 clusters for the best concatenated system and 320 for the best multiplicative one. *Expansion* is the least semantically marked relation and thus takes less advantage of fine-grained semantic groupings.

Comparing Word Combinations Comparing concatenated configurations ($\oplus$ systems) against multiplicative ones ($\otimes$ systems), we first note that for raw tokens the concatenated form (*one-hot* $\oplus$) yields results that are comparable, and sometimes better, than the standard multiplicative system (*one-hot* $\otimes$), while failing to explicitly model word pair interactions. With Brown clusters, the concatenated form *Best Brown* $\oplus$ lead to better F1 scores than *Best Brown* $\otimes$ except for *Contingency*.

When comparing performance on dev set, we found that the differences between concatenated and multiplicative forms for *Brown* (excluding *Expansion* for now) depend on the number of clusters used. Turian et al. (2010) found that the more clusters, the better the performance. This is also the case here with concatenated forms, but not with multiplicative forms. In that case, F1 increases un-

til $1,000$ clusters and then decreases. There is indeed a trade-off between expressivity and sparsity: having too few clusters means that we loose important distinctions, but having too many clusters leads to a loss in generalization. A similar trend is also found with word embeddings.

5.2 Head Words Only

Considering the right part of table 3, the first finding is that performance of systems that use only head words decrease compared to those using all words, but much more so with the baseline *One-hot* $\otimes$ than with other representations. *One-hot* $\otimes$ has very poor performance for most relations, losing between 7 and 17 points in F_1 score. The performance loss is much less striking with *One-hot* $\oplus$ and with denser representations, which are again the best performing. The only exception is *Expansion* whose precision however increases. As said, this relation is the less semantically marked, making it less likely to take advantage of the use of word representations. The best performance in this setting are obtained with word embeddings (not *Brown*) with significant gain from 8 to 13 points in F1 for most relations. Moreover, the best systems are all based on the multiplicative form confirming that this is a better way of representing pairs than simple concatenation when the number of initial dimensions is not too large.

5.3 Adding Other Features

Finally, we would like to assess how much improvement can still be obtained by adding other standard features, such as those in §4.1, to word-based features. Conversely, we want to evaluate how far we are from state-of-the-art performance by just using word representations. We compare our results with those presented in (Rutherford and Xue, 2014) and in (Ji and Eisenstein, 2014), both systems deal with sparsity either by using Brown clusters or by learning task-dependent representations. To make comparison easier we reproduce the experiments in (Rutherford and Xue, 2014) with Naive Bayes (NB)[7] and Maximum Entropy (ME) but without their coreference features and using gold syntactic parse. These correspond to the "repr." lines in table 4. We attribute the small differences observed with NB by the lack of coreference features and/or the use of different filtering thresholds. Concerning the difference between NB and ME, the only obvious issue is the low F1 score for *Expansion*: the system built using NB predicts all examples as positive thus leading to a high F1 score whereas the other one produces more balanced predictions, meaning neither systems is truly satisfactory. Finally, we give results using the traditional one-encoding based on word pairs plus additional features (*One-hot* $\otimes$ + additional features). These results are summarized in table 4, also including the best results of our experiments without additional features ("only").

Our first finding is that the addition of extra features to our previous word-based only configuration appears to outperform state-of-the art results for *Temporal* and *Contingency*, thus giving the best performance to date on these relations. These improvements are significant compared to our reproduced systems. Note that we also outperform the task-dependent embeddings of Ji and Eisenstein (2014), except for *Expansion*. Our tentative explanation for this is that these authors included Entity relations and coreference features. Note that our system corresponding to a reproduction of (Rutherford and Xue, 2014) gives results similar to the baseline using raw word pairs (*One-hot* $\otimes$ + additional features) showing that their improvements were due to other factors, the optimized filter threshold and the coreference features.

Overall, the addition of these hand-crafted features to our best systems do not provide improvements as high as one might have hoped. While improvements are significant compared to our reproduced systems, they are not with respect to the best systems given in table 3. When using all words, we only have a tendency toward significant improvement for *Contingency*[8]. These very small differences demonstrate that semantic and syntactic properties encoded in these features are already taken into account into the unsupervised word representations.

6 Related Work

Automatically identifying implicit relations is challenging due to the complex nature of the predictors. Previous studies have thus used many features relying on several external resources (Pitler et al., 2009; Park and Cardie, 2012; Biran and McKeown, 2013) as the MPQA lexicon (Wilson et al., 2005) or the General Inquirer lexicon (Stone and Hunt, 1963), or on constituent or dependency

[7] Implemented in Scikit-Learn, we optimized the hyper-parameter corresponding to the smoothing.

[8] $p = 0.135$ with ttest and $p = 0.061$ with Wilcoxon.

	Temporal	*Contingency*	*Comparison*	*Expansion*
System	F1	F1	F1	F1
(Ji and Eisenstein, 2014)	26.91	51.39	35.84	**79.91**
(Rutherford and Xue, 2014)	28.69	54.42	**39.70**	70.23
repr. (Rutherford and Xue, 2014) NB	28.05	52.95	37.38	70.23
repr. (Rutherford and Xue, 2014) ME	24.79	53.39	36.46	50.00
One-hot $\otimes$ all tokens + add. features	23.26	54.41	34.34	62.57
Best all tokens only	27.96	53.85	34.99	67.42
Best all tokens + add. features	**29.30**	**55.76**	36.36	61.76

Table 4: Systems using additional features ("+ add.features"), state-of-the art results either reported or reproduced ("repr.") using Naive Bayes ("NB") or logistic regression ("ME") and best system from previous table ("only").

parsers (Li and Nenkova, 2014b; Lin et al., 2009). Feature selection methods have been proved necessary to handle all of these features (Park and Cardie, 2012; Lin et al., 2009). Interestingly, Park and Cardie (2012) conclude on the worthlessness of word pair features, given the existence of such resources. We showed that provided unsupervised word representations, the opposite was in fact true, as dense word representations capture a lot of syntactic and semantic information.

The major problem of standard word pair representations is their sparsity. A line of work is to deal with this issue by adding automatically annotated data from explicit examples (Marcu and Echihabi, 2002), possibly using some kind of filtering or adaptation methods (Pitler et al., 2009; Biran and McKeown, 2013; Braud and Denis, 2014). Another line of work propose to make use of dense representations as Brown clusters in (Rutherford and Xue, 2014). These authors claim that this resource provides word representations that are relevant to the task, a conclusion that we considerably refined. Ji and Eisenstein (2014) propose to learn a distributed representation from the syntactic trees representing each argument in way that is more directly related to the task. Although this is an attractive idea, the score on top level PDTB relations are mostly below those reported by (Rutherford and Xue, 2014), possibly because their representations are learned on a rather small corpus, the PDTB itself, whereas building this kind of representation requires massive amount of data.

Our work also relates to studies comparing unsupervised representations for other NLP tasks such as name entity recognition, chunking (Turian et al., 2010), sentiment analysis (Lebret and Collobert, 2014) or POS tagging (Stratos and Collins, 2005). In particular, we found some similarities between our conclusions and those in (Turian et al., 2010). Our comparison is slightly richer in that it includes different methods of vector compositions and add an extra distributional representation to our comparison (namely, H-PCA).

7 Conclusions and Future Work

In this paper, we show that one can reach state-of-the-art results for implicit discourse relation identification using only shallow lexical features and existing unsupervised word representations thus contradicting previous conclusions on the worthlessness of these features. We carefully assess the usefulness of word representations for discourse by comparing various formulations and combination schemes, demonstrating the inadequacy of the previously proposed strategy based on Brown clusters and the distinctive relevance of head words, and by establishing that the created dense representations already provide most of the semantic and syntactic information relevant to the task thus alleviating the need for traditional external resources.

In future work, we first plan to extend our comparative framework to a larger set of relations and to other languages. We also want to explore methods for learning embeddings that are directly related to the task of discourse relation classification, potentially using existing embeddings as initialization (Labutov and Lipson, 2013). It is also clear that seeing discourse segments as bag of words is too simplistic, we would like to investigate ways of learning adequate segment-wide embeddings.

References

N. Asher and A. Lascarides. 2003. *Logics of Conversation*. Cambridge University Press.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155.

Or Biran and Kathleen McKeown. 2013. Aggregated word pair features for implicit discourse relation disambiguation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*.

Chloé Braud and Pascal Denis. 2014. Combining natural and artificial examples to improve implicit discourse relation identification. In *Proceedings of the 25th International Conference on Computational Linguistics*.

Peter F. Brown, Peter V. deSouza, Robert L. Mercer, Vincent J. Della Pietra, and Jenifer C. Lai. 1992. Class-based n-gram models of natural language. *Computational Linguistics*, 18:467–479.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning*.

Yangfeng Ji and Jacob Eisenstein. 2014. One vector is not enough: Entity-augmented distributional semantics for discourse relations. *Transactions of the Association for Computational Linguistics*.

Terry Koo, Xavier Carreras, and Michael Collins. 2008. Simple semi-supervised dependency parsing. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*.

Igor Labutov and Hod Lipson. 2013. Re-embedding words. In *Proceedings of the 51th Annual Meeting of the Association for Computational Linguistics*.

Rémi Lebret and Ronan Collobert. 2014. Word emdeddings through hellinger PCA. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*.

Junyi Jessy Li and Ani Nenkova. 2014a. Addressing class imbalance for improved recognition of implicit discourse relations. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue*.

Junyi Jessy Li and Ani Nenkova. 2014b. Reducing sparsity improves the recognition of implicit discourse relations. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue*.

Ziheng Lin, Min-Yen Kan, and Hwee Tou Ng. 2009. Recognizing implicit discourse relations in the penn discourse treebank. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*.

Daniel Marcu and Abdessamad Echihabi. 2002. An unsupervised approach to recognizing discourse relations. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*.

Andriy Mnih and Geoffrey Hinton. 2007. Three new graphical models for statistical language modelling. In *Proceedings of the 24th International Conference on Machine Learning*.

Joonsuk Park and Claire Cardie. 2012. Improving implicit discourse relation recognition through feature set optimization. In *Proceedings of the 13th Annual Meeting of the Special Interest Group on Discourse and Dialogue*.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Emily Pitler and Ani Nenkova. 2009. Using syntax to disambiguate explicit discourse connectives in text. In *Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the AFNLP*.

Emily Pitler, Annie Louis, and Ani Nenkova. 2009. Automatic sense prediction for implicit discourse relations in text. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*.

Rashmi Prasad, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Livio Robaldo, Aravind Joshi, and Bonnie Webber. 2008. The penn discourse treebank 2.0. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation*.

Attapol Rutherford and Nianwen Xue. 2014. Discovering implicit discourse relations through brown cluster pair representation and coreference patterns. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*.

Philip J. Stone and Earl B. Hunt. 1963. A computer approach to content analysis: Studies using the general inquirer system. In *Proceedings of the Spring Joint Computer Conference*.

Karl Stratos and Michael Collins. 2005. Simple semi-supervised pos tagging. In *Proceedings of NAACL Workshop on Vector Space Modeling for NLP*.

Joseph Turian, Lev-Arie Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*.

Better Document-level Sentiment Analysis from RST Discourse Parsing*

Parminder Bhatia and **Yangfeng Ji** and **Jacob Eisenstein**
School of Interactive Computing
Georgia Institute of Technology
Atlanta, GA 30308
parminder.bhatia243@gmail.com, {jiyfeng,jacobe}@gatech.edu

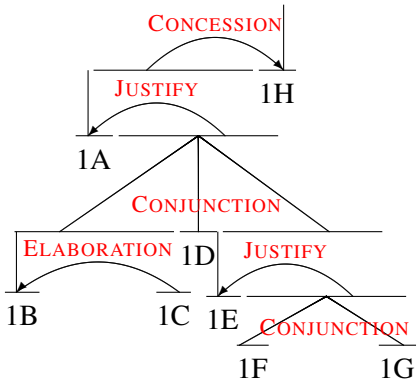
Abstract

Discourse structure is the hidden link between surface features and document-level properties, such as sentiment polarity. We show that the discourse analyses produced by Rhetorical Structure Theory (RST) parsers can improve document-level sentiment analysis, via composition of local information up the discourse tree. First, we show that reweighting discourse units according to their position in a dependency representation of the rhetorical structure can yield substantial improvements on lexicon-based sentiment analysis. Next, we present a recursive neural network over the RST structure, which offers significant improvements over classification-based methods.

1 Introduction

Sentiment analysis and opinion mining are among the most widely-used applications of language technology, impacting both industry and a variety of other academic disciplines (Feldman, 2013; Liu, 2012; Pang and Lee, 2008). Yet sentiment analysis is still dominated by bag-of-words approaches, and attempts to include additional linguistic context typically stop at the sentence level (Socher et al., 2013). Since document-level opinion mining inherently involves multi-sentence texts, it seems that analysis of document-level structure should have a role to play.

A classic example of the potential relevance of discourse to sentiment analysis is shown in Figure 1. In this review of the film *The Last Samurai*, the positive sentiment words far outnumber the negative sentiment words. But the discourse structure — indicated here with Rhetorical Structure Theory (RST; Mann and Thompson, 1988) — clearly favors the final sentence, whose polarity is negative. This example is illustrative in more than one way: it was originally identified by Voll and Taboada (2007), who found that *manually-annotated* RST parse trees improved lexicon-based sentiment analysis, but that automatically-generated parses from the SPADE parser (Soricut and Marcu, 2003), which was then state-of-the-art, did not.



[It could have been a great movie]^{1A} [It does have beautiful scenery,]^{1B} [some of the best since Lord of the Rings.]^{1C} [The acting is well done,]^{1D} [and I really liked the son of the leader of the Samurai.]^{1E} [He was a likable chap,]^{1F} [and I hated to see him die.]^{1G} [But, other than all that, this movie is nothing more than hidden rip-offs.]^{1H}

Figure 1: Example adapted from Voll and Taboada (2007).

Since this time, RST discourse parsing has improved considerably, with the best systems now yielding 5-10% greater raw accuracy than SPADE, depending on the metric. The time is therefore right to reconsider the effectiveness of RST for document-level sentiment analysis. In this paper, we present two different ways of combining RST discourse parses with sentiment analysis. The methods are both relatively simple, and

*Code is available at https://github.com/parry2403/R2N2

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 2212–2218,
Lisbon, Portugal, 17-21 September 2015.

can be used in combination with an "off the shelf" discourse parser. We consider the following two architectures:

- Reweighting the contribution of each discourse unit, based on its position in a dependency-like representation of the discourse structure. Such weights can be defined using a simple function, or learned from a small of data.

- Recursively propagating sentiment up through the RST parse, in an architecture inspired by recursive neural networks (Smolensky, 1990; Socher et al., 2011).

Both architectures can be used in combination with either a lexicon-based sentiment analyzer, or a trained classifier. Indeed, for users whose starting point is a lexicon-based approach, a simple RST-based reweighting function can offer significant improvements. For those who are willing to train a sentiment classifier, the recursive model yields further gains.

2 Background

2.1 Rhetorical Structure Theory

RST is a compositional model of discourse structure, in which elementary discourse units (EDUs) are combined intro progressively larger discourse units, ultimately covering the entire document. Discourse relations may involve a *nucleus* and a *satellite*, or they may be *multinuclear*. In the example in Figure 1, the unit $1C$ is the satellite of a relationship with its nucleus $1B$; together they form a larger discourse unit, which is involved in a multinuclear CONJUNCTION relation.

The nuclearity structure of RST trees suggests a natural approach to evaluating the importance of segments of text: satellites tend to be less important, and nucleii tend to be more important (Marcu, 1999). This idea has been leveraged extensively in document summarization (Gerani et al., 2014; Uzêda et al., 2010; Yoshida et al., 2014), and was the inspiration for Voll and Taboada (2007), who examined intra-sentential relations, eliminating all words except those in the top-most nucleus within each sentence. More recent work focuses on reweighting each discourse unit depending on the relations in which it participates (Heerschop et al., 2011; Hogenboom et al., 2015). We consider such an approach, and compare it with a compositional method, in which sentiment polarity is propagated up the discourse tree.

Marcu (1997) provides the seminal work on automatic RST parsing, but there has been a recent spike of interest in this task, with contemporary approaches employing discriminative learning (Hernault et al., 2010), rich features (Feng and Hirst, 2012), structured prediction (Joty et al., 2015), and representation learning (Ji and Eisenstein, 2014; Li et al., 2014). With many strong systems to choose from, we employ the publicly-available DPLP parser (Ji and Eisenstein, 2014),[1]. To our knowledge, this system currently gives the best F-measure on relation identification, the most difficult subtask of RST parsing. DPLP is a shift-reduce parser (Sagae, 2009), and its time complexity is linear in the length of the document.

2.2 Sentiment analysis

There is a huge literature on sentiment analysis (Pang and Lee, 2008; Liu, 2012), with particular interest in determining the overall sentiment polarity (positive or negative) of a document. Bag-of-words models are widely used for this task, as they offer accuracy that is often very competitive with more complex approaches. Given labeled data, supervised learning can be applied to obtain sentiment weights for each word. However, the effectiveness of supervised sentiment analysis depends on having training data in the same domain as the target, and this is not always possible. Moreover, in social science applications, the desired labels may not correspond directly to positive or negative sentiment, but may focus on other categories, such as politeness (Danescu-Niculescu-Mizil et al., 2013), narrative frames (Jurafsky et al., 2014), or a multidimensional spectrum of emotions (Kim et al., 2012). In these cases, labeled documents may not be available, so users often employ a simpler method: counting matches against lists of words associated with each category. Such lists may be built manually from introspection, as in LIWC (Tausczik and Pennebaker, 2010) and the General Inquirer (Stone, 1966). Alternatively, they may be induced by bootstrapping from a seed set of words (Hatzivassiloglou and McKeown, 1997; Taboada et al., 2011). While lexicon-based methods may be less accurate than supervised classifiers, they are easier to apply to

[1] https://github.com/jiyfeng/DPLP

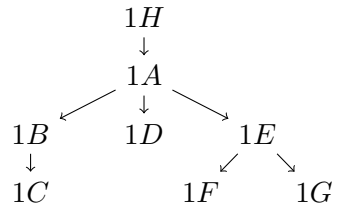


Figure 2: Dependency-based discourse tree representation of the discourse in Figure 1

new domains and problem settings. Our proposed approach can be used in combination with either method for sentiment analysis, and in principle, could be directly applied to other document-level categories, such as politeness.

2.3 Datasets

We evaluate on two review datasets. In both cases, the goal is to correctly classify the opinion polarity as positive or negative. The first dataset is comprised of 2000 movie reviews, gathered by Pang and Lee (2004). We perform ten-fold cross-validation on this data. The second dataset is larger, consisting of 50,000 movie reviews, gathered by Socher et al. (2013), with a predefined 50/50 split into training and test sets. Documents are scored on a 1-10 scale, and we treat scores ≤ 4 as negative, ≥ 7 as positive, and ignore scores of 5-6 as neutral — although in principle nothing prevents extension of our approaches to more than two sentiment classes.

3 Discourse depth reweighting

Our first approach to incorporating discourse information into sentiment analysis is based on quantifying the importance of each unit of text in terms of its discourse depth. To do this, we employ the *dependency-based discourse tree (DEP-DT)* formulation from prior work on summarization (Hirao et al., 2013). The DEP-DT formalism converts the constituent-like RST tree into a directed graph over elementary discourse units (EDUs), in a process that is a close analogue of the transformation of a headed syntactic constituent parse to a syntactic dependency graph (Kübler et al., 2009).

The DEP-DT representation of the discourse in Figure 1 in shown in Figure 2. The graph is constructed by propagating "head" information up the RST tree; if the elementary discourse unit e_i is the satellite in a discourse relation headed by e_j, then there is an edge from e_j to e_i. Thus, the "depth" of each EDU is the number of times in which it is embedded in the satellite of a discourse relation. The exact algorithm for constructing DEP-DTs is given by Hirao et al. (2013).

Given this representation, we construct a simple linear function for weighting the contribution of the EDU at depth d_i:

$$\lambda_i = \max(0.5, 1 - d_i/6). \quad (1)$$

Thus, at $d_i = 0$, we have $\lambda_i = 1$, and at $d_i \geq 3$, we have $\lambda_i = 0.5$. Now assume each elementary discourse unit contributes a prediction $\psi_i = \boldsymbol{\theta}^\top \boldsymbol{w}_i$, where $\boldsymbol{w}_i$ is the bag-of-words vector, and θ is a vector of weights, which may be either learned or specified by a sentiment lexicon. Then the overall prediction for a document is given by,

$$\Psi = \sum_i \lambda_i(\boldsymbol{\theta}^\top \boldsymbol{w}_i) = \boldsymbol{\theta}^\top(\sum_i \lambda_i \boldsymbol{w}_i). \quad (2)$$

Evaluation We apply this approach in combination with both lexicon-based and classification-based sentiment analysis. We use the lexicon of Wilson et al. (2005), and set $\theta_j = 1$ for words marked "positive", and $\theta_j = -1$ for words marked negative. For classification-based analysis, we set $\boldsymbol{\theta}$ equal to the weights obtained by training a logistic regression classifier, tuning the regularization coefficient on held-out data.

Results are shown in Table 1. As seen in the comparison between lines B1 and D1, discourse depth weighting offers substantial improvements over the bag-of-words approach for lexicon-based sentiment analysis, with raw improvements of $4-5\%$. Given the simplicity of this approach — which requires only a sentiment lexicon and a discourse parser — we strongly recommend the application of discourse depth weighting for lexicon-based sentiment analysis at the document level. However, the improvements for the classification-based models are considerably smaller, less than 1% in both datasets.

4 Rhetorical Recursive Neural Networks

Discourse-depth reweighting offers significant improvements for lexicon-based sentiment analysis, but the improvements over the more accurate classification-based method are meager. We therefore turn to a data-driven approach for combining sentiment analysis with rhetorical structure theory, based on recursive neural networks (Socher et al.,

	Pang & Lee	Socher et al.
Baselines		
B1. Lexicon	68.3	74.9
B2. Classifier	82.4	81.5
Discourse depth weighting		
D1. Lexicon	72.6	78.9
D2. Classifier	82.9	82.0
Rhetorical recursive neural network		
R1. No relations	83.4	85.5
R2. With relations	84.1	85.6

Table 1: Sentiment classification accuracies on two movie review datasets (Pang and Lee, 2004; Socher et al., 2013), described in Section 2.3.

2011). The idea is simple: recursively propagate sentiment scores up the RST tree, until the root of the document is reached. For nucleus-satellite discourse relations, we have:

$$\Psi_i = \tanh(K_n^{(r_i)}\Psi_{n(i)} + K_s^{(r_i)}\Psi_{s(i)}), \quad (3)$$

where i indexes a discourse unit composed from relation r_i, $n(i)$ indicates its nucleus, and $s(i)$ indicates its satellite. Returning to the example in Figure 1, the sentiment score for the discourse unit obtained by combining $1B$ and $1C$ is obtained from $\tanh(K_n^{(\text{elaboration})}\Psi_{1B} + K_s^{(\text{elaboration})}\Psi_{1C})$. Similarly, for multinuclear relations, we have,

$$\Psi_i = \tanh(\sum_{j \in n(i)} K_n^{(r_i)}\Psi_j). \quad (4)$$

In the base case, each elementary discourse unit's sentiment is constructed from its bag-of-words, $\Psi_i = \boldsymbol{\theta}^\top \boldsymbol{w}_i$. Because the structure of each document is different, the network architecture varies in each example; nonetheless, the parameters can be reused across all instances.

This approach, which we call a Rhetorical Recursive Neural Network (R2N2), is reminiscent of the compositional model proposed by Socher et al. (2013), where composition is over the constituents of the syntactic parse of a sentence, rather than the units of a discourse. However, a crucial difference is that in R2N2s, the elements Ψ and K are *scalars*: we do not attempt to learn a latent distributed representation of the sub-document units. This is because discourse units typically comprise multiple words, so that accurate analysis of the sentiment for elementary discourse units is not so difficult as accurate analysis of individual words. The scores for individual discourse units can be computed from a bag-of-words classifier, or, in future work, from a more complex model such as a recursive or recurrent neural network.

While this neural network structure captures the idea of compositionality over the RST tree, the most deeply embedded discourse units can be heavily down-weighted by the recursive composition (assuming $K_s < K_n$): in the most extreme case of a right-branching or left-branching structure, the recursive operator may be applied N times to the most deeply embedded EDU. In contrast, discourse depth reweighting applies a uniform weight of 0.5 to all discourse units with depth ≥ 3. In the spirit of this approach, we add an additional component to the network architecture, capturing the bag-of-words for the entire document. Thus, at the root node we have:

$$\Psi_{\text{doc}} = \gamma\boldsymbol{\theta}^\top(\sum_i \boldsymbol{w}_i) + \Psi_{\text{rst-root}}, \quad (5)$$

with $\Psi_{\text{rst-root}}$ defined recursively from Equations 3 and 4, $\boldsymbol{\theta}$ indicating the vector of per-word weights, and the scalar γ controlling the tradeoff between these two components.

Learning R2N2 is trained by backpropagating from a hinge loss objective; assuming $y_t \in \{-1, 1\}$ for each document t, we have the loss $L_t = (1 - y_t\Psi_{\text{doc},t})_+$. From this loss, we use backpropagation through structure to obtain gradients on the parameters (Goller and Kuchler, 1996). Training is performed using stochastic gradient descent. For simplicity, we follow Zirn et al. (2011) and focus on the distinction between contrastive and non-contrastive relations. The set of contrastive relations includes CONTRAST, COMPARISON, ANTITHESIS, ANTITHESIS-E, CONSEQUENCE-S, CONCESSION, and PROBLEM-SOLUTION.

Evaluation Results for this approach are shown in lines R1 and R2 of Table 1. Even without distinguishing between discourse relations, we get an improvement of more than 3% accuracy on the Stanford data, and 0.5% on the smaller Pang & Lee dataset. Adding sensitivity to discourse relations (distinguishing $K^{(r)}$ for contrastive and non-contrastive relations) offers further improvements on the Pang & Lee data, outperforming the baseline classifier (D2) by 1.3%.

The accuracy of discourse relation detection is only 60% for even the best systems (Ji and Eisen-

stein, 2014), which may help to explain why relations do not offer a more substantial boost. An anonymous reviewer recommended evaluating on gold RST parse trees to determine the extent to which improvements in RST parsing might transfer to downstream document analysis. Such an evaluation would seem to require a large corpus of texts with both gold RST parse trees and sentiment polarity labels; the SFU Review Corpus (Taboada, 2008) of 30 review texts offers a starting point, but is probably too small to train a competitive sentiment analysis system.

5 Related Work

Section 2 mentions some especially relevant prior work. Other efforts to incorporate RST into sentiment analysis have often focused on intra-sentential discourse relations (Heerschop et al., 2011; Zhou et al., 2011; Chenlo et al., 2014), rather than relations over the entire document. Wang et al. (2012) address sentiment analysis in Chinese. Lacking a discourse parser, they focus on explicit connectives, using a strategy that is related to our discourse depth reweighting. Wang and Wu (2013) use manually-annotated discourse parses in combination with a sentiment lexicon, which is automatically updated based on the discourse structure. Zirn et al. (2011) use an RST parser in a Markov Logic Network, with the goal of making polarity predictions at the sub-sentence level, rather than improving document-level prediction. None of the prior work considers the sort of recurrent compositional model presented here.

An alternative to RST is to incorporate "shallow" discourse structure, such as the relations from the Penn Discourse Treebank (PDTB). PDTB relations were shown to improve sentence-level sentiment analysis by Somasundaran et al. (2009), and were incorporated in a model of sentiment flow by Wachsmuth et al. (2014). PDTB relations are often signaled with explicit discourse connectives, and these may be used as a feature (Trivedi and Eisenstein, 2013; Lazaridou et al., 2013) or as posterior constraints (Yang and Cardie, 2014). This prior work on discourse relations within sentences and between adjacent sentences can be viewed as complementary to our focus on higher-level discourse relations across the entire document.

There are unfortunately few possibilities for direct comparison of our approach against prior work. Heerschop et al. (2011) and Wachsmuth et al. (2014) also employ the Pang and Lee (2004) dataset, but neither of their results are directly comparable: Heerschop et al. (2011) exclude documents that SPADE fails to parse, and Wachsmuth et al. (2014) evaluates only on individual sentences rather than entire documents. The only possible direct comparison is with very recent work from Hogenboom et al. (2015), who employ a weighting scheme that is similar to the approach described in Section 3. They evaluate on the Pang and Lee data, and consider only lexicon-based sentiment analysis, obtaining document-level accuracies between 65% (for the baseline) and 72% (for their best discourse-augmented system). Table 1 shows that fully supervised methods give much stronger performance on this dataset, with accuracies more than 10% higher.

6 Conclusion

Sentiment polarity analysis has typically relied on a "preponderance of evidence" strategy, hoping that the words or sentences representing the overall polarity will outweigh those representing counterpoints or rhetorical concessions. However, with the availability of off-the-shelf RST discourse parsers, it is now easy to include document-level structure in sentiment analysis. We show that a simple reweighting approach offers robust advantages in lexicon-based sentiment analysis, and that a recursive neural network can substantially outperform a bag-of-words classifier. Future work will focus on combining models of discourse structure with richer models at the sentence level.

Acknowledgments Thanks to the anonymous reviewers for their helpful suggestions on how to improve the paper. The following members of the Georgia Tech Computational Linguistics Laboratory offered feedback throughout the research process: Naman Goyal, Vinodh Krishan, Umashanthi Pavalanathan, Ana Smith, Yijie Wang, and Yi Yang. Several class projects in Georgia Tech CS 4650/7650 took alternative approaches to the application of discourse parsing to sentiment analysis, which was informative to this work; thanks particularly to Julia Cochran, Rohit Pathak, Pavan Kumar Ramnath, and Bharadwaj Tanikella. This research was supported by a Google Faculty Research Award, by the National Institutes of Health under award number R01GM112697-01, and by the Air Force Office of Scientific Research. The content is solely the responsibility of the authors and does not necessarily represent the official views of these sponsors.

References

José M Chenlo, Alexander Hogenboom, and David E Losada. 2014. Rhetorical structure theory for polarity estimation: An experimental study. *Data & Knowledge Engineering*.

Cristian Danescu-Niculescu-Mizil, Moritz Sudhof, Dan Jurafsky, Jure Leskovec, and Christopher Potts. 2013. A computational approach to politeness with application to social factors. In *Proceedings of the Association for Computational Linguistics (ACL)*, pages 250–259, Sophia, Bulgaria.

Ronen Feldman. 2013. Techniques and applications for sentiment analysis. *Communications of the ACM*, 56(4):82–89.

Vanessa Wei Feng and Graeme Hirst. 2012. Text-level Discourse Parsing with Rich Linguistic Features. In *Proceedings of the Association for Computational Linguistics (ACL)*, Jeju, Korea.

Shima Gerani, Yashar Mehdad, Giuseppe Carenini, Raymond T Ng, and Bita Nejat. 2014. Abstractive summarization of product reviews using discourse structure. In *Proceedings of the Association for Computational Linguistics (ACL)*, Baltimore, MD.

Christoph Goller and Andreas Kuchler. 1996. Learning task-dependent distributed representations by backpropagation through structure. In *Neural Networks, IEEE International Conference on*, pages 347–352. IEEE.

Vasileios Hatzivassiloglou and Kathleen R. McKeown. 1997. Predicting the semantic orientation of adjectives. In *Proceedings of the Association for Computational Linguistics (ACL)*, pages 174–181, Madrid, Spain.

Bas Heerschop, Frank Goossen, Alexander Hogenboom, Flavius Frasincar, Uzay Kaymak, and Franciska de Jong. 2011. Polarity analysis of texts using discourse structure. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 1061–1070. ACM.

Hugo Hernault, Helmut Prendinger, David A. duVerle, and Mitsuru Ishizuka. 2010. HILDA: A Discourse Parser Using Support Vector Machine Classification. *Dialogue and Discourse*, 1(3):1–33.

Tsutomu Hirao, Yasuhisa Yoshida, Masaaki Nishino, Norihito Yasuda, and Masaaki Nagata. 2013. Single-document summarization as a tree knapsack problem. In *Proceedings of Empirical Methods for Natural Language Processing (EMNLP)*, pages 1515–1520.

Alexander Hogenboom, Flavius Frasincar, Franciska de Jong, and Uzay Kaymak. 2015. Using rhetorical structure in sentiment analysis. *Communications of the ACM*, 58(7):69–77.

Yangfeng Ji and Jacob Eisenstein. 2014. Representation learning for text-level discourse parsing. In *Proceedings of the Association for Computational Linguistics (ACL)*, Baltimore, MD.

Shafiq Joty, Giuseppe Carenini, and Raymond Ng. 2015. CODRA: A novel discriminative framework for rhetorical analysis. *Computational Linguistics*, 41(3).

Dan Jurafsky, Victor Chahuneau, Bryan R Routledge, and Noah A Smith. 2014. Narrative framing of consumer sentiment in online restaurant reviews. *First Monday*, 19(4).

Suin Kim, JinYeong Bak, and Alice Haeyun Oh. 2012. Do you feel what i feel? social aspects of emotions in twitter conversations. In *Proceedings of the International Conference on Web and Social Media (ICWSM)*, Menlo Park, California. AAAI Publications.

Sandra Kübler, Ryan McDonald, and Joakim Nivre. 2009. Dependency parsing. *Synthesis Lectures on Human Language Technologies*, 1(1):1–127.

Angeliki Lazaridou, Ivan Titov, and Caroline Sporleder. 2013. A bayesian model for joint unsupervised induction of sentiment, aspect and discourse representations. In *Proceedings of the Association for Computational Linguistics (ACL)*, pages 1630–1639, Sophia, Bulgaria.

Jiwei Li, Rumeng Li, and Eduard Hovy. 2014. Recursive deep models for discourse parsing. In *Proceedings of Empirical Methods for Natural Language Processing (EMNLP)*.

Bing Liu. 2012. Sentiment analysis and opinion mining. *Synthesis Lectures on Human Language Technologies*, 5(1):1–167.

William Mann. 1984. Discourse structures for text generation. In *Proceedings of the 10th International Conference on Computational Linguistics and 22nd annual meeting on Association for Computational Linguistics*, pages 367–375. Association for Computational Linguistics.

Daniel Marcu. 1997. The rhetorical parsing of natural language texts. In *Proceedings of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 96–103.

Daniel Marcu. 1999. The automatic construction of large-scale corpora for summarization research. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 137–144. ACM.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the Association for Computational Linguistics (ACL)*, pages 271–278.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Foundations and trends in information retrieval*, 2(1-2):1–135.

Kenji Sagae. 2009. Analysis of Discourse Structure with Syntactic Dependencies and Data-Driven Shift-Reduce Parsing. In *Proceedings of the 11th International Conference on Parsing Technologies (IWPT'09)*, pages 81–84, Paris, France, October. Association for Computational Linguistics.

Paul Smolensky. 1990. Tensor product variable binding and the representation of symbolic structures in connectionist systems. *Artificial intelligence*, 46(1):159–216.

Richard Socher, Cliff C. Lin, Andrew Y. Ng, and Christopher D. Manning. 2011. Parsing Natural Scenes and Natural Language with Recursive Neural Networks. In *Proceedings of the International Conference on Machine Learning (ICML)*, Seattle, WA.

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of Empirical Methods for Natural Language Processing (EMNLP)*.

Swapna Somasundaran, Galileo Namata, Janyce Wiebe, and Lise Getoor. 2009. Supervised and unsupervised methods in employing discourse relations for improving opinion polarity classification. In *Proceedings of Empirical Methods for Natural Language Processing (EMNLP)*, Singapore.

Radu Soricut and Daniel Marcu. 2003. Sentence level discourse parsing using syntactic and lexical information. In *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 149–156.

Philip J. Stone. 1966. *The General Inquirer: A Computer Approach to Content Analysis*. The MIT Press.

Maite Taboada, Julian Brooke, Milan Tofiloski, Kimberly Voll, and Manfred Stede. 2011. Lexicon-based methods for sentiment analysis. *Computational linguistics*, 37(2):267–307.

Maite Taboada. 2008. SFU review corpus. `http://www.sfu.ca/~mtaboada/research/SFU_Review_Corpus.html`.

Yla R Tausczik and James W Pennebaker. 2010. The psychological meaning of words: LIWC and computerized text analysis methods. *Journal of Language and Social Psychology*, 29(1):24–54.

Rakshit Trivedi and Jacob Eisenstein. 2013. Discourse connectors for latent subjectivity in sentiment analysis. In *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 808–813, Stroudsburg, Pennsylvania. Association for Computational Linguistics.

Vinícius Rodrigues Uzêda, Thiago Alexandre Salgueiro Pardo, and Maria Das Graças Volpe Nunes. 2010. A comprehensive comparative evaluation of rst-based summarization methods. *ACM Transactions on Speech and Language Processing (TSLP)*, 6(4):4.

Kimberly Voll and Maite Taboada. 2007. Not all words are created equal: Extracting semantic orientation as a function of adjective relevance. In *Proceedings of Australian Conference on Artificial Intelligence*.

Henning Wachsmuth, Martin Trenkmann, Benno Stein, and Gregor Engels. 2014. Modeling review argumentation for robust sentiment analysis. In *Proceedings of the International Conference on Computational Linguistics (COLING)*.

Fei Wang and Yunfang Wu. 2013. Exploiting hierarchical discourse structure for review sentiment analysis. In *Proceedings of the Conference on Asian Language Processing (IALP)*, pages 121–124.

Fei Wang, Yunfang Wu, and Likun Qiu. 2012. Exploiting discourse relations for sentiment analysis. In *Proceedings of the International Conference on Computational Linguistics (COLING)*, pages 1311–1320, Mumbai, India.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of Empirical Methods for Natural Language Processing (EMNLP)*, pages 347–354.

Bishan Yang and Claire Cardie. 2014. Context-aware learning for sentence-level sentiment analysis with posterior regularization. In *Proceedings of the Association for Computational Linguistics (ACL)*, Baltimore, MD.

Yasuhisa Yoshida, Jun Suzuki, Tsutomu Hirao, and Masaaki Nagata. 2014. Dependency-based Discourse Parser for Single-Document Summarization. In *Proceedings of Empirical Methods for Natural Language Processing (EMNLP)*.

Lanjun Zhou, Binyang Li, Wei Gao, Zhongyu Wei, and Kam-Fai Wong. 2011. Unsupervised discovery of discourse relations for eliminating intra-sentence polarity ambiguities. In *Proceedings of Empirical Methods for Natural Language Processing (EMNLP)*, pages 162–171.

Cäcilia Zirn, Mathias Niepert, Heiner Stuckenschmidt, and Michael Strube. 2011. Fine-grained sentiment analysis with structural features. In *Proceedings of the International Joint Conference on Natural Language Processing (IJCNLP)*, pages 336–344, Chiang Mai, Thailand.

Closing the Gap: Domain Adaptation from Explicit to Implicit Discourse Relations

Yangfeng Ji Gongbo Zhang Jacob Eisenstein
School of Interactive Computing
Georgia Institute of Technology
{jiyfeng,gzhang64,jacobe}@gatech.edu

Abstract

Many discourse relations are explicitly marked with discourse connectives, and these examples could potentially serve as a plentiful source of training data for recognizing implicit discourse relations. However, there are important linguistic differences between explicit and implicit discourse relations, which limit the accuracy of such an approach. We account for these differences by applying techniques from domain adaptation, treating implicitly and explicitly-marked discourse relations as separate domains. The distribution of surface features varies across these two domains, so we apply a marginalized denoising autoencoder to induce a dense, domain-general representation. The label distribution is also domain-specific, so we apply a resampling technique that is similar to instance weighting. In combination with a set of automatically-labeled data, these improvements eliminate more than 80% of the transfer loss incurred by training an implicit discourse relation classifier on explicitly-marked discourse relations.

1 Introduction

Discourse relations reveal the structural organization of text, potentially supporting applications such as summarization (Louis et al., 2010; Yoshida et al., 2014), sentiment analysis (Somasundaran et al., 2009), and coherence evaluation (Lin et al., 2011). While some relations are signaled **explicitly** with connectives such as *however* (Pitler et al., 2008), many more are **implicit**. Expert-annotated datasets of implicit discourse relations are expensive to produce, so it would be preferable to use weak supervision, by automatically labeling instances with explicit connectives (Marcu and Echihabi, 2003).

However, Sporleder and Lascarides (2008) show that models trained on explicitly marked examples generalize poorly to implicit relation identification. They argued that explicit and implicit examples may be linguistically dissimilar, as writers tend to avoid discourse connectives if the discourse relation could be inferred from context (Grice, 1975). Similar observations are made by Rutherford and Xue (2015), who attempt to add automatically-labeled instances to improve supervised classification of implicit discourse relations.

In this paper, we approach this problem from the perspective of domain adaptation. Specifically, we argue that the reason that automatically-labeled examples generalize poorly is due to domain mismatch from the explicit relations (**source** domain) to the implicit relations (**target** domain). We propose to close the gap by using two simple methods from domain adaptation: (1) feature representation learning: mapping the source domain and target domain to a shared latent feature space; (2) resampling: modifying the relation distribution in the explicit relations to match the distribution over implicit relations. Our results on the Penn Discourse Treebank (Prasad et al., 2008) show that these two methods improve the performance on unsupervised discourse relation identification by more than 8.4% on average F_1 score across all relation types, an 82% reduction on the transfer loss incurred by training on explicitly-marked discourse relations.

2 Related Work

Marcu and Echihabi (2003) train a classifier for implicit intra-sentence discourse relations from

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 2219–2224,
Lisbon, Portugal, 17-21 September 2015.

explicitly-marked examples in the rhetorical structure theory (RST) treebank, where the relations are automatically labeled by their discourse connectives: for example, labeling the relation as CONTRAST if the connective is *but*. However, Sporleder and Lascarides (2008) argue that explicitly marked relations are too different from implicit relations to serve as an adequate supervision signal, obtaining negative results in segmented discourse representation theory (SDRT) relations.

More recent work has focused on the Penn Discourse Treebank (PDTB), using explicitly-marked relations to supplement, rather than replace, a labeled corpus of implicit relations. For example, Biran and McKeown (2013) collect word pairs from arguments of explicit examples to help the supervised learning on implicit relation identification. Lan et al. (2013) present a multi-task learning framework, using explicit relation identification as auxiliary tasks to help main task on implicit relation identification. Rutherford and Xue (2015) explore several selection heuristics for adding automatically-labeled examples from Gigaword to their system for implicit relation detection, obtaining a 2% improvement in Macro-F_1. Our work differs from these previous efforts in that we focus exclusively on training from automatically-labeled explicit instances, rather than supplementing a training set of manually-labeled implicit examples.

Learning good feature representations (Ben-David et al., 2007) and reducing mismatched label distributions (Joshi et al., 2012) are two main ways to make a domain adaptation task successful. Structural correspondence learning is an early example of representation learning for domain adaptation (Blitzer et al., 2006); we build on the more computationally tractable approach of marginalized denoising autoencoders (Chen et al., 2012). Instance weighting is an approach for correcting label distribution mismatch (Jiang and Zhai, 2007); we apply a simpler approach of resampling the source domain according to an estimate of the target domain label distribution.

3 Domain Adaptation for Implicit Relation Identification

We employ two domain adaptation techniques: learning feature representations, and resampling to match the target label distribution.

3.1 Learning feature representation: Marginalized denoising autoencoders

The goal of feature representation learning is to obtain dense features that capture feature correlations between the source and target domains. Denoising autoencoders (Glorot et al., 2011) do this by first "corrupting" the original data, $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ into $\tilde{\boldsymbol{x}}_1, \ldots, \tilde{\boldsymbol{x}}_n$, either by adding Gaussian noise (in the case of real-valued data) or by randomly zeroing out features (in the case of binary data). We can then learn a function to reconstruct the original data, thereby capturing feature correlations and improving resilience to domain shift.

Chen et al. (2012) propose a particularly simple and elegant form of denoising autencoder, by marginalizing over the noising process. Their single-layer marginalized denoising autoencoder (mDA) solves the following problem:

$$\min_{\mathbf{W}} E_{\tilde{\boldsymbol{x}}_i|\boldsymbol{x}_i}[\|\boldsymbol{x}_i - \mathbf{W}\tilde{\boldsymbol{x}}_i\|^2] \quad (1)$$

where the parameter $\mathbf{W} \in \mathbb{R}^{d\times d}$ is a projection matrix. After learning the projection matrix, we use $\tanh(\mathbf{W}\boldsymbol{x})$ as the representation for our relation identification task.

Usually, $\boldsymbol{x}_i \in \mathbb{R}^d$ is a sparse vector with more than 10^5 dimensions. Solving the optimization problem defined in equation 1 will produce a $d \times d$ dense matrix $\mathbf{W}$, and is prohibitively expensive. We employ the trick proposed by Blitzer et al. (2006) to select κ *pivot* features to be reconstructed. We then split all features into non-overlapping subsets of size $\leq K$. Then, a set of projection matrices are learned, so as to transform each feature subset to the pivot feature set. The final projection matrix $\mathbf{W}$ is the stack of all projection matrices learned from the feature subsets.

3.2 Handling mismatched label distributions: Resampling with minimal supervision

There is a notable mismatch between the relation distributions for implicit and explicitly-marked discourse relations in the Penn Discourse Treebank: as shown in Figure 1, the EXPANSION and CONTINGENCY relations comprise a greater share of the implicit relations, while the TEMPORAL and COMPARISON relations comprise a greater share of the explicitly-marked discourse relations. Such label distribution mismatches can be a major source of transfer loss across domains, and therefore, reducing this mismatch can be an easy

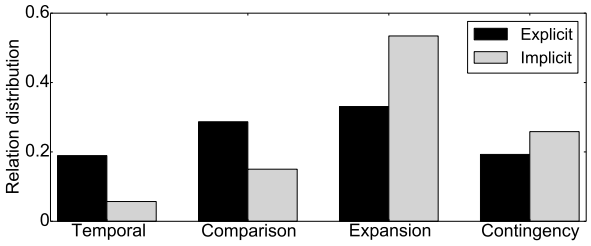


Figure 1: The relation distributions of training examples from the source domain (explicitly-marked relations) and target domain (implicit relations) in the PDTB.

way to obtain performance gains in domain adaptation (Joshi et al., 2012). Specifically, our goal is to modify the relation distribution in the source domain (explicitly-marked relations) and make it as similar as possible to the target domain (implicit relations). Given the label distribution from the target domain, we resample training examples from the source domain with replacement, in order to match the label distribution in the target domain. As this requires the label distribution from the target domain, it is no longer purely unsupervised domain adaptation; instead, we call it *resampling with minimal supervision*.

It may also be desirable to ensure that the source and target training instances are similar in terms of their observed features; this is the idea behind the *instance weighting* approach to domain adaptation (Jiang and Zhai, 2007). Motivated by this idea, we require that sampled instances from the source domain have a cosine similarity of at least τ with at least one target domain instance (Rutherford and Xue, 2015).

4 Experiments

Our experiments test the utility of the two domain adaptation methods, using the Penn Discourse Treebank (Prasad et al., 2008) and some extra-training data collected from a external resource.

4.1 Experimental setup

Datasets The test examples are implicit relation instances from section 21-22 in the PDTB. For the domain adaptation setting, the training set consists of the explicitly-marked examples extracted from sections 02-20 and 23-24, and the development set consists of the explicit relations from sections 21-22. All relations in the explicit examples are automatically labeled by using the connective-to-relation mapping from Table 2 in (Prasad et al., 2007), where we only keep the majority relation type for every connective. For each identified connective, we use its annotated arguments in the PDTB. As an upper bound, we also train a supervised discourse relation classifier, using the implicit examples in sections 02-20 and 23-24 as the training set, and using sections 00-01 as the development set. Following prior work (Pitler et al., 2009; Park and Cardie, 2012; Biran and McKeown, 2013), we consider the first-level discourse relations in the PDTB — Temporal (TEMP.), Comparison (COMP.), Expansion (EXP.) and Contingency (CONT.). We train binary classifiers and report F_1 score on each binary classification task. Extension of this approach to multi-class classification is important, but since this is not the setting considered in most of the prior research, we leave it for future work.

The true power of learning from automatically labeled examples is that we could leverage much larger datasets than hand-annotated corpora such as the Penn Discourse Treebank. To test this idea, we collected 1,000 news articles from CNN.com as extra training data. Explicitly-marked discourse relations from this data are automatically extracted by matching the PDTB discourse connectives (Prasad et al., 2007). For this data, we also need to extract the arguments of the identified connectives: for every identified connective, the sentence following this connective is labeled as Arg2 and the preceding sentence is labeled as Arg1, as suggested by Biran and McKeown (2013). In a pilot study we found that larger amounts of additional training data yielded no further improvements, which is consistent with the recent results of Rutherford and Xue (2015).

Model selection We use a linear support vector machine (Fan et al., 2008) as the classification model. Our model includes five tunable parameters: the number of pivot features κ, the size of the feature subset K, the noise level for the denoising autoencoder q, the cosine similarity threshold for resampling τ, and the penalty parameter C for the SVM classifier. We consider $\kappa \in \{1000, 2000, 3000\}$ for pivot features and $C \in \{0.001, 0.01, 0.1, 1.0, 10.0\}$ for penalty parameters, $q \in \{0.90, 0.95, 0.99\}$ for noise levels. To reduce the free parameters, we set $K = 5\kappa$ and simply fix the cosine similarity threshold $\tau =$

Surface Features	+Rep. Learning	+Resampling	Relations				Average F_1
			TEMP.	COMP.	EXP.	CONT.	
Implicit → Implicit							
1. FULL			24.15	28.87	68.84	43.45	41.32
Explicit [PDTB] → Implicit							
2. FULL	No	No	17.13	20.54	50.55	36.14	31.04
3. FULL	No	Yes	15.38	23.88	62.04	35.29	34.14
4. FULL	Yes	No	17.53	22.77	50.85	36.43	31.90
5. FULL	Yes	Yes	17.05	22.00	63.51	38.23	35.20
6. PIVOT	No	No	17.33	23.89	53.53	36.22	32.74
7. PIVOT	No	Yes	17.73	25.39	62.65	36.02	35.44
8. PIVOT	Yes	No	18.66	25.86	63.37	38.87	36.69
9. PIVOT	Yes	Yes	19.26	25.74	68.08	41.39	38.62
Explicit [PDTB + CNN] → Implicit							
10. PIVOT	Yes	Yes	20.35	26.32	68.92	42.25	39.46

Table 1: Performance of cross-domain learning for implicit discourse relation identification.

0.85; pilot studies found that results are not sensitive to the value of τ across a range of values.

Features All features are motivated by prior work on implicit discourse relation classification: from each training example with two arguments, we extract (1) Lexical features, including word pairs, the first and last words from both arguments (Pitler et al., 2009); (2) Syntactic features, including production rules from each argument, and the shared production rules between two arguments (Lin et al., 2009); (3) Other features, including modality, Inquirer tags, Levin verb classes, and argument polarity (Park and Cardie, 2012). We re-implement these features as closely as possible to the cited works, using the Stanford CoreNLP Toolkit to obtain syntactic annotations (Manning et al., 2014).

The FULL feature set for domain adaptation is constructed by collecting all features from the training set, and then removing features that occur fewer than ten times. The PIVOT feature set includes κ high-frequency features from the FULL feature set. To focus on testing the domain adaptation techniques, we use the same FULL and PIVOT set for all four relations, and leave feature set optimization for each relation as a future work (Park and Cardie, 2012). To obtain the upper bound, we employ the same feature categories and frequency threshold to extract features from the in-domain data, hand-annotated implicit discourse relations. To include the representations from the marginalized denoising autoencoder for relation identification, we concatenate them with the original surface feature representations of the same examples.

4.2 Experimental results

In experiments, we start with surface feature representations as baselines, then incorporate the two domain adaptation techniques incrementally. As shown in line 2 of Table 1, the performance is poor if directly applying a model trained on the explicit examples with the FULL feature set, which is consistent with the observations of Sporleder and Lascarides (2008): there is a 10.28% absolute reduction on average F_1 score from the upper bound obtained with in-domain supervision (line 1). With mDA, the overall performance increases by 0.86% (line 4); resampling gives a further 4.16% improvement mainly because of the performance gain on the EXP. relation (line 5). The resampling method itself (line 3) also gives a better overall performance then mDA (line 4). However, the F_1 scores on the TEMP. and CONT. are even worse than the baseline (line 2).

Surface representations with the FULL feature set were found to cause serious overfitting in the experiments. To deal with this problem, we propose to use only κ pivot features, which gives a stronger baseline of the cross-domain relation identification, as shown in line 6. Then, by incorporating resampling and feature representation

learning individually, the average F_1 increases from 32.74% to 35.44% (line 7) and 36.69% (line 8) respectively. The combination of these two domain adaptation techniques boosts the average F_1 further to 38.62% (line 9). The additional CNN training data further improves performance to 39.46% (line 10). This represents an 8.42% improvement of average F_1 from the original result (line 2), for more than 80% reduction on the transfer loss incurred by training on explicit discourse relations.

An additional experiment is to use automatic argument extraction in both the PDTB and the CNN data, which would correspond to more truly unsupervised domain adaptation. (Recall that in the CNN data, we used adjacent sentences as argument spans, while in the PDTB data, we use expert annotations.) When using adjacent sentences as argument spans in both datasets, the average F_1 is 38.52% for the combination of representation learning and resampling. Compared to line 10, this is a 0.94% performance drop, indicating the importance of argument identification in the PDTB data. In future work we may consider better heuristics for argument extraction, such as obtaining automatically-labeled examples only from those connectors for whom the arguments usually are the adjacent sentences; for example, the connector *nonetheless* usually connects adjacent spans (e.g., *Bob was hungry. Nonetheless he gave Tina the burger.*), while the connector *even though* may connect two spans that follow the connector in the same sentence (e.g., *Even though Bob was hungry, he gave Tina the burger.*).

5 Conclusion

We have presented two methods — feature representation learning and resampling — from domain adaptation to close the gap of using explicit examples for unsupervised implicit discourse relation identification. Experiments on the PDTB show the combination of these two methods eliminates more than 80% of the transfer loss caused by training on explicit examples, increasing average F_1 from 31% to 39.5%, against a supervised upper bound of 41.3%. Future work will explore the combination of this approach with more sophisticated techniques for instance selection (Rutherford and Xue, 2015) and feature selection (Park and Cardie, 2012; Biran and McKeown, 2013), while also tackling the more difficult problems of multi-class relation classification and fine-grained level-2 discourse relations.

Acknowledgments This research was supported by a Google Faculty Research Award to the third author. The following members of the Georgia Tech Computational Linguistics Laboratory offered feedback throughout the research process: Parminder Bhatia, Naman Goyal, Vinodh Krishan, Umashanthi Pavalanathan, Ana Smith, Yijie Wang, and especially Yi Yang. Thanks to the reviewers for their constructive and helpful suggestions.

References

Shai Ben-David, John Blitzer, Koby Crammer, Fernando Pereira, et al. 2007. Analysis of representations for domain adaptation. In *Neural Information Processing Systems (NIPS)*, Vancouver.

Or Biran and Kathleen McKeown. 2013. Aggregated word pair features for implicit discourse relation disambiguation. In *Proceedings of the Association for Computational Linguistics (ACL)*, pages 69–73, Sophia, Bulgaria.

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *Proceedings of Empirical Methods for Natural Language Processing (EMNLP)*, pages 120–128.

Minmin Chen, Z. Xu, Killian Weinberger, and Fei Sha. 2012. Marginalized denoising autoencoders for domain adaptation. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A library for large linear classification. *The Journal of Machine Learning Research*, 9:1871–1874.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proceedings of the International Conference on Machine Learning (ICML)*, Seattle, WA.

H Paul Grice. 1975. Logic and Conversation. In P. Cole and J. L. Morgan, editors, *Syntax and Semantics Volume 3: Speech Acts*, pages 41–58. Academic Press.

Jing Jiang and ChengXiang Zhai. 2007. Instance weighting for domain adaptation in nlp. In *Proceedings of the Association for Computational Linguistics (ACL)*, Prague.

Mahesh Joshi, William W Cohen, Mark Dredze, and Carolyn P Rosé. 2012. Multi-domain learning: when do domains matter? In *Proceedings of the*

2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pages 1302–1312. Association for Computational Linguistics.

Man Lan, Yu Xu, and Zhengyu Niu. 2013. Leveraging Synthetic Discourse Data via Multi-task Learning for Implicit Discourse Relation Recognition. In *Proceedings of the Association for Computational Linguistics (ACL)*, pages 476–485, Sophia, Bulgaria.

Ziheng Lin, Min-Yen Kan, and Hwee Tou Ng. 2009. Recognizing implicit discourse relations in the penn discourse treebank. In *Proceedings of Empirical Methods for Natural Language Processing (EMNLP)*, pages 343–351, Singapore.

Ziheng Lin, Hwee Tou Ng, and Min-Yen Kan. 2011. Automatically Evaluating Text Coherence Using Discourse Relations. In *Proceedings of the Association for Computational Linguistics (ACL)*, pages 997–1006, Portland, OR.

Annie Louis, Aravind Joshi, and Ani Nenkova. 2010. Discourse indicators for content selection in summarization. In *Proceedings of the 11th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 147–156. Association for Computational Linguistics.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60.

Daniel Marcu and Abdessamad Echihabi. 2003. An unsupervised approach to recognizing discourse relations. In *Proceedings of the Association for Computational Linguistics (ACL)*, pages 368–375.

Joonsuk Park and Claire Cardie. 2012. Improving Implicit Discourse Relation Recognition Through Feature Set Optimization. In *Proceedings of the 13th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 108–112, Seoul, South Korea, July. Association for Computational Linguistics.

Emily Pitler, Mridhula Raghupathy, Hena Mehta, Ani Nenkova, Alan Lee, and Aravind Joshi. 2008. Easily identifiable discourse relations. In *Proceedings of the International Conference on Computational Linguistics (COLING)*, pages 87–90, Manchester, UK.

Emily Pitler, Annie Louis, and Ani Nenkova. 2009. Automatic Sense Prediction for Implicit Discourse Relations in Text. In *Proceedings of the Association for Computational Linguistics (ACL)*, Suntec, Singapore.

Rashmi Prasad, Eleni Miltsakaki, Nikhil Dinesh, Alan Lee, Aravind Joshi, Livio Robaldo, and Bonnie L Webber. 2007. The Penn Discourse Treebank 2.0 annotation manual. The PDTB Research Group.

Rashmi Prasad, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Livio Robaldo, Aravind Joshi, and Bonnie Webber. 2008. The Penn Discourse Treebank 2.0. In *Proceedings of LREC*.

Attapol Rutherford and Nianwen Xue. 2015. Improving the Inference of Implicit Discourse Relations via Classifying Explicit Discourse Connectives. In *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 799–808, Denver, CO, May–June.

Swapna Somasundaran, Galileo Namata, Janyce Wiebe, and Lise Getoor. 2009. Supervised and unsupervised methods in employing discourse relations for improving opinion polarity classification. In *Proceedings of Empirical Methods for Natural Language Processing (EMNLP)*, Singapore.

Caroline Sporleder and Alex Lascarides. 2008. Using automatically labelled examples to classify rhetorical relations: An assessment. *Natural Language Engineering*, 14(3):369–416.

Yasuhisa Yoshida, Jun Suzuki, Tsutomu Hirao, and Masaaki Nagata. 2014. Dependency-based Discourse Parser for Single-Document Summarization. In *Proceedings of Empirical Methods for Natural Language Processing (EMNLP)*.

Wikification of Concept Mentions within Spoken Dialogues Using Domain Constraints from Wikipedia

Seokhwan Kim, Rafael E. Banchs, Haizhou Li
Human Language Technology Department
Institute for Infocomm Research
Singapore 138632
{kims,rembanchs,hli}@i2r.a-star.edu.sg

Abstract

While most previous work on Wikification has focused on written texts, this paper presents a Wikification approach for spoken dialogues. A set of analyzers are proposed to learn dialogue-specific properties along with domain knowledge of conversations from Wikipedia. Then, the analyzed properties are used as constraints for generating candidates, and the candidates are ranked to find the appropriate links. The experimental results show that our proposed approach can significantly improve the performances of the task in human-human dialogues.

1 Introduction

Linking mentions in natural language to the relevant concepts in knowledge-bases plays a key role in better understanding the meanings of expressions as well as further populating knowledge-bases with less human effort. Especially, Wikipedia has been widely used as a major target resource for linking. Most previous work on this Wikipedia-based linking task called Wikification (Mihalcea and Csomai, 2007) has focused on resolving ambiguities and variabilities of the expressions in written texts including newswire collections (McNamee and Dang, 2009; Ji et al., 2010; Ji et al., 2014) or microblog posts (Genc et al., 2011; Cassidy et al., 2012; Guo et al., 2013; Huang et al., 2014).

But writing and reading are not the only ways for exchange of information, since many communications between people in real life are performed through spoken dialogues also. Thus, we could expect to improve the understanding capabilities of applications based on Wikification and broaden the coverage of the contents in knowledge-bases, if Wikification is successfully performed also for human-human spoken conversations.

In this work, we focus on the following differences between spoken dialogues and written texts as sources for Wikification. Firstly, at least two speakers are engaged in a dialogue session, while the texts in newswire or microblogs are mostly written by a single author. Thus, the viewpoint of each speaker should be considered separately or jointly depending on the situation. Secondly, the correspondence between mentions and concepts in spoken dialogues tends to be dependent not only on the contexts explicitly mentioned in a given dialogue, but also on other information inferred by speakers based on their background knowledge. The other difference is that spoken utterances are more likely to be informal and noisy than written sentences, which makes expressions more ambiguous and variable.

To solve these issues, we propose a three step approach for Wikification on spoken dialogues. At the first step, a set of classifiers are used for analyzing the dialogue-specific aspects of a given mention. According to the analyzed results, the criteria in selecting concept candidates is determined, and then a ranking is performed on the filtered candidates to identify the concept that is the most relevant to the mention.

While many researchers have worked on linking named-entities (Bunescu and Pasca, 2006; Cucerzan, 2007; McNamee and Dang, 2009; Han and Sun, 2011; Han et al., 2011; Ji et al., 2014) or other types of concept mentions (Mihalcea and Csomai, 2007; Milne and Witten, 2008; Ferragina and Scaiella, 2010; Ratinov et al., 2011; Mendes et al., 2011; Cheng and Roth, 2013) to the relevant articles in Wikipedia, all the noun phrases including not only named entities or base noun phrases, but also complex or recursive noun phrases in a dialogue are considered as instances to be linked in this work. For the concept candidates, we divide every article into sub-sections and consider each section as a unit along with article-level concepts.

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 2225–2229,
Lisbon, Portugal, 17-21 September 2015.

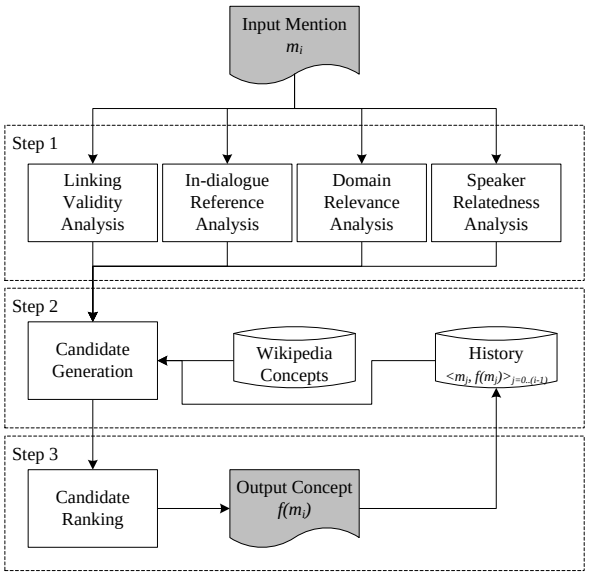


Figure 1: Overall architecture of three-step approach for Wikification on spoken dialogues

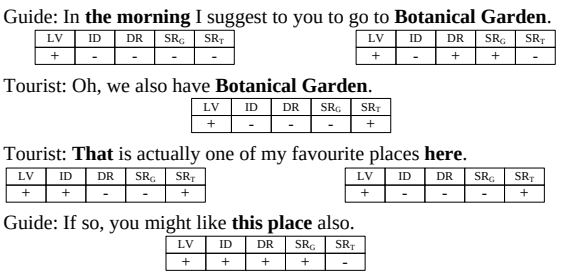


Figure 2: Examples of annotations for mention analysis: SR_G and SR_T denote guide and tourist relatedness, respectively.

2 Method

2.1 Mention Analysis

The first step in our proposed approach (Figure 1) is analyzing the following four types of binary properties of a given mention: linking validity (LV), in-dialogue reference (ID), domain relevance (DR), and speaker relatedness (SR).

Linking validity of the mention is determined by the decision whether it is matched with any Wikipedia concept or not. Since only the mentions assigned with positive validity values are proceeded to the further processes, this classification can be considered as a joint task for target mention identification and NIL detection.

Another type of analysis focuses on the references between the mention and the linking history. If the mention is matched with one in the set of concepts for the previous mentions in the same session, it has a positive value for the in-dialogue reference property.

The other two types of properties are defined for indicating the relevances of the mention to the contents that are specific to the target domain or the profiles of each speaker in the conversation. For these analyses, the whole Wikipedia collection is partitioned into subsets according to the domain or speaker-relevances. In this work, the concepts in these subsets are automatically collected with no manual effort by utilizing the domain knowledge also from Wikipedia. First, we retrieve the 'List' or 'Index' pages in Wikipedia that are related to the topic or the profile of a speaker. Then, all the articles listed on these seed pages are collected and considered as the related concepts in the corresponding sets.

Since every property has a positive or a negative value as a result, each analysis can be considered as a binary classification problem. In this work, we train support vector machines (SVM) (Cortes and Vapnik, 1995) from the dialogues annotated with the corresponding labels as shown in Figure 2 based on the features listed in Table 1.

2.2 Candidate Generation

After analyzing the above property values of a given mention, a set of concepts to be disambiguated are selected from Wikipedia. These candidates are retrieved from a Lucene [1] index on the whole Wikipedia collection with the fields of article title, section title, redirection, category, and body texts. Each query to the search engine is prepared with the combination of the mention phrase and its analyzed properties as constraints for filtering. If the value for in-dialogue reference is positive, the searching is restricted to the set of concepts linked with the previous mentions in the same session. Similarly, the domain relevance and speaker relatedness values provide the filtering condition within the corresponding subsets introduced in Section 2.1.

One practical issue on this candidate generation step is how to combine the multiple constraints when we have more than one positive properties for a given mention. The simplest way is taking the intersection of the corresponding constraints. However, we should consider the fact that the properties assigned automatically can be erroneous, since none of the analyzer is perfect.

[1] http://lucene.apache.org/

Level	Name	Description
Mention	SP	the speaker who spoke that mention
	WM	word n-grams within the mention
	LM	lemma n-grams within the mention
	PM	POS n-grams within the mention
	NE	named entities within the mention
	NP	base noun phrases within the mention
Utterance	BW	the words before the mention
	AW	the words after the mention
	BL	the lemmas before the mention
	AL	the lemmas after the mention
	BP	the POS tags before the mention
	AP	the POS tags after the mention
	IU	whether the mention previously occurs in the same utterance
Dialogue	EO	whether the phrase is previously mentioned in the dialogue history
	EOS	whether the phrase is previously mentioned by the same speaker in the history
Wikipedia	IW	whether the phrase is a title of an entry in Wikipedia
	IWD	whether the phrase is a title of an entry in the set of domain-relevant concepts
	IWS_k	whether the phrase is a title of an entry in the set of k-th speaker-relevant concepts

Table 1: List of features for training the models for mention analysis

For the noisy cases, the intersection-based filtering could be risky, because the errors are also jointly accumulated. To circumvent the impact of errors from the previous step, we also try to use the union of the constraints and compare it with the intersection case later in Section 3.

2.3 Candidate Ranking

In this work, linking a given mention to its most relevant concept is determined by ranking SVM (Joachims, 2002) which is a pairwise ranking algorithm learned from the ranked lists. For each pair of a mention m in the training data and its candidate concept c, the ranking score $s(m, c)$ is assigned as follows:

$$s(m,c) = \begin{cases} 4 & \text{if } c \text{ is the exactly same as } f(m), \\ 3 & \text{if } c \text{ is the parent article of } f(m), \\ 2 & \text{if } c \text{ belongs to the same article} \\ & \text{but different section of } f(m), \\ 1 & \text{otherwise.} \end{cases}$$

where $f(m)$ is the annotation of m in the training dataset. The list of candidates assigned with their scores provides the relative orders for a given mention, and it can be converted into a set of pairwise constraints which are trained by ranking SVM with the features in Table 2.

Name	Description
SP	the speaker who spoke that mention
WM	word n-grams within the surface of m
WT	word n-grams within the title of c
EMT	whether the surface of m is same as the title of c
EMR	whether the surface of m is same as one of re-directions to c
MIT	whether the surface of m is a sub-string of the title of c
TIM	whether the title of c is a sub-string of the m's surface form
MIR	whether the surface of m is a sub-string of a re-directed title to c
RIM	whether a re-directed title to c is a sub-string of the m's surface form
PMT	similarity score based on edit distance between the surface of m and the title of c
PMR	maximum similarity score between the surface of m and the redirected titles to c
OC	whether c previously occurred in the full dialogue history
OC_w	whether c occurred within w previous turns with $w \in \{1, 3, 5, 10\}$

Table 2: List of features for training the ranking SVM model

3 Evaluation

3.1 Data

To demonstrate the effectiveness of our approach to Wikification on spoken dialogues, we performed experiments on a dialogue corpus which consists of 35 sessions collected from human-human conversations in English about tourism in Singapore between actual tour guides in Singapore and tourists from the Philippines. All the recorded dialogues with the total length of 21 hours were manually transcribed, then the 31,034 utterance were pre-processed by Stanford CoreNLP toolkit [2]. Each noun phrase in the constituent trees provided by the parser is considered as an instance for Wikification and manually annotated with the corresponding concept in Wikipedia. 34,949 mentions have been linked to the concepts in Wikipedia.

As a pool for candidate generation, we built a Lucene index based on Wikipedia database dump as of January 2015 which has 4,797,927 articles and 25,577,464 sections in total. From this collection, 11,128 and 27,186 articles have been considered as Singapore-related and Philippines-related concepts, respectively, for the filtering based on domain and speaker relevances.

[2] http://nlp.stanford.edu/software/corenlp.shtml

Features	LV	ID	SR_G	SR_T
M	86.29	69.15	**71.10**	**72.94**
M+U	**86.90**	70.43	70.43	68.85
M+D	86.17	71.09	70.56	71.52
M+W	86.21	68.96	70.66	71.86
M+U+D	86.82	**72.37**	70.12	68.30
M+U+W	86.84	70.13	70.19	68.78
M+U+D+W	86.77	72.20	69.94	68.10

Table 3: Comparisons of the performances in F-measure of mention analyzers with different combinations of features: M,U,D,W denotes mention-level, utterance-level, dialogue-level, and Wikipedia features, respectively

3.2 Mention Analysis

Based on the annotated dialogues, we built four mention analyzers for LV, ID, SR_G, and SR_T, where SR_G is for the guides and SR_T is for the tourists in the conversations. In this work, only the information where each speaker is from was considered as a profile to analyze the speaker-related properties. Since all the guides participated in the data collection are from Singapore and the main topic of the conversations is also about Singapore, we omitted DR which should have the same results as SR_G in the experiments.

For each analyzer, we trained the SVM models using SVM^{light} [3] with the features in Table 1. All the evaluations were performed in five-fold cross validation to the manual annotations with precision, recall, and F-measure.

Table 3 compares the performances of the seven combinations of feature sets for each analyzer. Based on these results, we selected the model that achieved the best performance for each analyzer to process the mentions for the further steps.

3.3 Candidate Generation

For each mention in the corpus, we prepared four sets of candidates with different filtering constraints. While the first baseline set was retrieved with no filtering, the others were generated according to the procedure described in Section 2.2. When more than one positive values were provided from mention analyzers, intersection and union operators were applied for combining multiple constraints. In the last set, the property values manually annotated in the training data were considered as the correct constraints, which is intended for comparing with the others to investigate the influence of errors in mention analysis. For every set, we retrieved top 100 candidates satisfying the given constraints from the Lucene index with Wikipedia collection and added one more special candidate for NIL detection.

Method	P	R	F
No filtering	26.85	22.52	21.24
Intersection	44.37	27.35	33.84
Union	38.04	31.97	34.74
Manual (Oracle)	39.90	34.72	37.13

Table 4: Comparisons of the performances of Wikification on spoken dialogues

3.4 Candidate Ranking

For each set of candidates, we trained a ranking function using SVM^{rank}[4] with the features in Table 2. Both training and testing the ranking models were performed also based on five-fold cross validation with the same divisions as the former evaluation. After getting the ranking results, we took the top-ranked candidate for each list and considered it as a result of Wikification for the corresponding mention.

Table 4 compares the final performances of Wikification obtained by ranking on the candidates generated with different sets of constraints. Both approaches, intersection and union, outperformed the baseline by 12.60 and 13.50 in F-measure, respectively. While the intersection strategy contributed to produce more precise outputs than the others even including the case with manual filtering, the other proposed approach with union achieved more gain in recall with slightly better F-measure than the former one.

4 Conclusions

This paper presented a Wikification approach for spoken dialogues. In this approach, a set of dialogue-specific properties were analyzed for generating concept candidates. Then, supervised ranking was performed on these candidates to identify the relevant concepts. Experimental results show that the proposed constraints help to improve the performances of the task on spoken dialogues.

[3] http://svmlight.joachims.org/

[4] http://www.cs.cornell.edu/people/tj/svm_light/svm_rank.html

References

Razvan C. Bunescu and Marius Pasca. 2006. Using Encyclopedic Knowledge for Named entity Disambiguation. In *EACL*, volume 6, pages 9–16.

Taylor Cassidy, Heng Ji, Lev-Arie Ratinov, Arkaitz Zubiaga, and Hongzhao Huang. 2012. Analysis and Enhancement of Wikification for Microblogs with Context Expansion. In *COLING*, volume 12, pages 441–456.

Xiao Cheng and Dan Roth. 2013. Relational inference for wikification. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1787–1796.

Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine learning*, 20(3):273–297.

Silviu Cucerzan. 2007. Large-Scale Named Entity Disambiguation Based on Wikipedia Data. In *EMNLP-CoNLL*, volume 7, pages 708–716.

Paolo Ferragina and Ugo Scaiella. 2010. Tagme: on-the-fly annotation of short text fragments (by wikipedia entities). In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 1625–1628.

Yegin Genc, Yasuaki Sakamoto, and Jeffrey V Nickerson. 2011. Discovering context: Classifying tweets through a semantic transform based on wikipedia. *Foundations of Augmented Cognition*, page 484.

Stephen Guo, Ming-Wei Chang, and Emre Kiciman. 2013. To Link or Not to Link? A Study on End-to-End Tweet Entity Linking. In *HLT-NAACL*, pages 1020–1030.

Xianpei Han and Le Sun. 2011. A generative entity-mention model for linking entities with knowledge base. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 945–954. Association for Computational Linguistics.

Xianpei Han, Le Sun, and Jun Zhao. 2011. Collective entity linking in web text: a graph-based method. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 765–774.

Hongzhao Huang, Yunbo Cao, Xiaojiang Huang, Heng Ji, and Chin-Yew Lin. 2014. Collective tweet wikification based on semi-supervised graph regularization. *Proceedings of the ACL, Baltimore, Maryland*.

Heng Ji, Ralph Grishman, Hoa Trang Dang, Kira Griffitt, and Joe Ellis. 2010. Overview of the tac 2010 knowledge base population track. In *Third Text Analysis Conference (TAC 2010)*.

Heng Ji, H. T. Dang, J. Nothman, and B. Hachey. 2014. Overview of tac-kbp2014 entity discovery and linking tasks. In *Proc. Text Analysis Conference (TAC2014)*.

Thorsten Joachims. 2002. Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 133–142.

Paul McNamee and Hoa Trang Dang. 2009. Overview of the tac 2009 knowledge base population track. In *Text Analysis Conference (TAC)*, volume 17, pages 111–113.

Pablo N Mendes, Max Jakob, Andrés García-Silva, and Christian Bizer. 2011. Dbpedia spotlight: shedding light on the web of documents. In *Proceedings of the 7th International Conference on Semantic Systems*, pages 1–8.

Rada Mihalcea and Andras Csomai. 2007. Wikify!: linking documents to encyclopedic knowledge. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 233–242.

David Milne and Ian H. Witten. 2008. Learning to link with wikipedia. In *Proceedings of the 17th ACM conference on Information and knowledge management*, pages 509–518.

Lev Ratinov, Dan Roth, Doug Downey, and Mike Anderson. 2011. Local and global algorithms for disambiguation to wikipedia. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 1375–1384. Association for Computational Linguistics.

Shallow Convolutional Neural Network for Implicit Discourse Relation Recognition

Biao Zhang[1], **Jinsong Su**[1]*, **Deyi Xiong**[2], **Yaojie Lu**[1], **Hong Duan**[1] and **Junfeng Yao**[1]
Xiamen University, Xiamen, China 361005[1]
Soochow University, Suzhou, China 215006[2]
{zb, lyj}@stu.xmu.edu.cn, {jssu, hduan, yao0010}@xmu.edu.cn
dyxiong@suda.edu.cn

Abstract

Implicit discourse relation recognition remains a serious challenge due to the absence of discourse connectives. In this paper, we propose a Shallow Convolutional Neural Network (SCNN) for implicit discourse relation recognition, which contains only one hidden layer but is effective in relation recognition. The shallow structure alleviates the overfitting problem, while the convolution and nonlinear operations help preserve the recognition and generalization ability of our model. Experiments on the benchmark data set show that our model achieves comparable and even better performance when comparing against current state-of-the-art systems.

1 Introduction

As a crucial task for discourse analysis, discourse relation recognition (DRR) aims to automatically identify the internal structure and logical relationship of coherent text (e.g., TEMPORAL, CONTINGENCY, EXPANSION, etc). It provides important information to many other natural language processing systems, such as question answering (Verberne et al., 2007), information extraction (Cimiano et al., 2005), machine translation (Guzmán et al., 2014) and so on. Despite great progress in explicit DRR where the discourse connectives (e.g., "*because*", "*but*" et al.) explicitly exist in the text (Miltsakaki et al., 2005; Pitler et al., 2008), implicit DRR remains a serious challenge because of the absence of discourse connectives (Prasad et al., 2008).

Conventional methods for implicit DRR directly rely on feature engineering, wherein researchers generally exploit various hand-crafted features, such as words, part-of-speech tags and production rules (Pitler et al., 2009; Lin et al., 2009; Louis et al., 2010; Wang et al., 2012; Park and Cardie, 2012; McKeown and Biran, 2013; Lan et al., 2013; Versley, 2013; Braud and Denis, 2014; Rutherford and Xue, 2014). Although these methods have proven successful, these manual features are labor-intensive and weak to capture intentional, semantic and syntactic aspects that govern discourse coherence (Li et al., 2014), thus limiting the effectiveness of these methods.

Recently, deep learning models have achieved remarkable results in natural language processing (Bengio et al., 2003; Bengio et al., 2006; Socher et al., 2011b; Socher et al., 2011a; Socher et al., 2013; Li et al., 2013; Kim, 2014). However, to the best of our knowledge, there is little deep learning work specifically for implicit DRR. The neglect of this important domain may be due to the following two reasons: (1) discourse relation distribution is rather unbalanced, where the generalization of deep models is relatively insufficient despite their powerful studying ability; (2) training dataset in implicit DRR is relatively small, where overfitting problems become more prominent.

In this paper, we propose a Shallow Convolutional Neural Network (SCNN) for implicit DRR, with only one simple convolution layer on the top of word vectors. On one hand, the network structure is simple, thereby overfitting issue can be alleviated; on the other hand, the convolution operation and nonlinear transformation help preserve the recognition ability of SCNN. This makes our model able to generalize better on the test dataset. We performed evaluation for English implicit DRR on the PDTB-style corpus. Experimental results show that the proposed method can obtain comparable even better performance when compares against several baselines.

2 Model

In Penn Discourse Treebank (PDTB) (Prasad et al., 2008), implicit discourse relations are anno-

*Corresponding author

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 2230–2235,
Lisbon, Portugal, 17-21 September 2015.

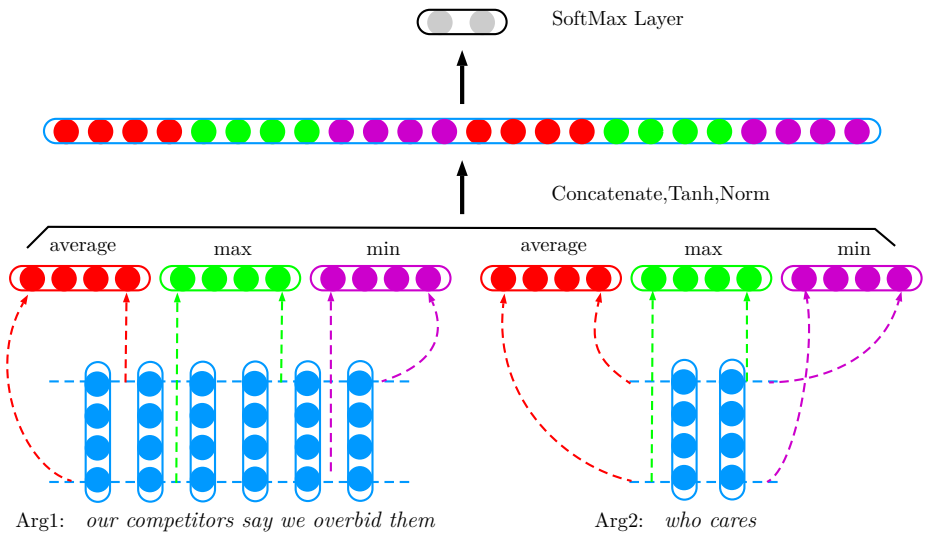


Figure 1: SCNN model architecture visualized with an instance.

tated with connective expressions that best convey implicit relations between two neighboring arguments, e.g.

Arg1: (But) *our competitions say we overbid them*

Arg2: *who cares*

the connective "*But*", which is annotated manually, is used to express the inferred COMPARISON relation.

We learn a classifier for implicit DRR based on convonlutional neural network. The overall model architecture is illustrated in Figure 1.[1] In our model, each word in vocabulary V corresponds to a d-dimensional dense, real-valued vector, and all words are stacked into a word embedding matrix $L \in \mathbb{R}^{d \times |V|}$, where $|V|$ is the vocabulary size.

Given an ordered list of n words in an argument, we retrieve the i-th word representation $x_{v_i} \in \mathbb{R}^d$ from L with its corresponding vocabulary index v_i. All word vectors in the argument produce the following output matrix:

$$\mathrm{X} = (x_{v_1}, x_{v_2}, \ldots, x_{v_n}) \quad (1)$$

Following previous work (Collobert et al., 2011; Socher et al., 2011a), for each row r in X, we explore three convolutional operations to obtain three convolution features *average*, *min* and *max* as follows:

$$c_r^{avg} = \frac{1}{n} \sum_i^n X_{r,i} \quad (2)$$

$$c_r^{min} = \min\left(X_{r,1}, X_{r,2}, \ldots, X_{r,n}\right) \quad (3)$$

$$c_r^{max} = \max\left(X_{r,1}, X_{r,2}, \ldots, X_{r,n}\right) \quad (4)$$

In this way, SCNN is able to capture almost all important information inside X (one with the highest, lowest and average values). Besides, each convolution operation naturally deals with variable argument lengths (Note that $c \in \mathbb{R}^d$). Back to Figure 1, we present c^{avg}, c^{min} and c^{max} with red, purple and green color respectively.

After obtaining the features of both arguments, we concatenate all of them into one vector, and then apply $tanh$ transformation and length normalization successively to generate the hidden layers:

$$a = \left[c_{Arg1}^{avg}; c_{Arg1}^{min}; c_{Arg1}^{max}; c_{Arg2}^{avg}; c_{Arg2}^{min}; c_{Arg2}^{max}\right] \quad (5)$$

$$h = \frac{\tanh(a)}{\|\tanh(a)\|} \quad (6)$$

where $h \in \mathbb{R}^{6d}$ is the hidden layer representation. The normalization operation scales the components of a feature vector to unit length. This, to some extent, eliminates the manifold differences among different features.

Upon the hidden layer, we stack a Softmax layer for relation recognition,

$$y = f(Wh + b) \quad (7)$$

where f is the softmax function, $W \in \mathbb{R}^{l \times 6d}$ is the parameter matrix, $b \in \mathbb{R}^l$ is the bias term, and l is the relation number.

To assess how well the predicted relation y represents the real relation, we supervise it with the

[1] For better illustration, we assume that the dimension of word vectors is 4 throughout this paper.

gold relation g in the annotated training corpus using the traditional cross-entropy error,

$$E(y, g) = -\sum_{j}^{l} g_j \times \log(y_j) \quad (8)$$

Combined with the regularization error, the joint training objective function is

$$J(\theta) = \frac{1}{m}\sum_{t=1}^{m} E(y_t, g_t) + \frac{\lambda}{2}\|\theta\|^2 \quad (9)$$

where m is the number of training instances, y_t is the t-th predicted distribution, λ is the regularization coefficient and θ is parameters, including L, W and b.[2]

To train SCNN, we first employ the toolkit *Word2Vec*[3] (Mikolov et al., 2013) to initialize the word embedding matrix L using a large-scale unlabeled data. Then, L-BFGS algorithm is applied to fine-tune the parameters θ.

3 Experiments

We conducted a series of experiments on English implicit DRR task. After a brief description of the experimental setup and the baseline systems, we further investigated the effectiveness of our method with deep analysis.

3.1 Setup

For comparison with other systems, we formulated the task as four separate one-against-all binary classification problems: one for each top level sense of implicit discourse relations (Pitler et al., 2009).

We used the *PDTB 2.0* corpus[4] (Prasad et al., 2008) for evaluation. The PDTB corpus contains discourse annotations over 2,312 Wall Street Journal articles, and is organized in different sections. Following Pitler et al. (2009), we used sections 2-20 as training set, sections 21-22 as test set, and sections 0-1 as development set for parameter optimization. For each relation, we randomly extracted the same number of positive and negative instances as training data, while all instances in sections 21 and 22 are used as our test set. The statistics of various data sets is listed in Table 1.

We tokenized PDTB corpus using *Stanford NLP Tool*[5]. For all experiments, we empirically set

[2]The bias terms b is not regularized. We preserve it in the equation just for clarity.

[3]https://code.google.com/p/word2vec/

[4]http://www.seas.upenn.edu/ pdtb/

[5]http://nlp.stanford.edu/software/corenlp.shtml

Relation	Positive/Negative Sentences		
	Train	Dev	Test
COMP.	1942/1942	197/986	152/894
CONT.	3342/3342	295/888	279/767
EXP.	7004/7004	671/512	574/472
TEMP.	760/760	64/1119	85/961

Table 1: Statistics of positive and negative instances in training (Train), development (Dev) and test (Test) sets. COMP.=COMPARISON, CONT.=CONTINGENCY, EXP.=EXPANSION and TEMP.=TEMPORAL

d=128 and λ=$1e^{-4}$. Besides, the unlabeled data for word embedding initialization contains 1.02M sentences with 33.5M words.

3.2 Baselines

We compared our model against the following baseline methods:

- **SVM:** This method learns a support vector machine (SVM) classifier with the labeled data.
- **TSVM:** This method learns a transductive SVM (TSVM) classifiers given the labeled data and unlabeled data. We extracted unlabeled data from above-mentioned 1.02M sentences. After filtering the noise ones, we finally obtained 0.11M unlabeled instances, each of which contains only two clauses.
- **RAE:** This method learns a recursive autoencoder (RAE) classifier with the labeled data. We first utilized standard RAEs to represent arguments, and then stacked a Softmax layer upon them. The hyperparameters were set as follows: word dimension 64, balance factor for reconstruction error 0.10282 and regularization factor $1e^{-5}$. Word embeddings are initialized via Word2Vec.

Rutherford and Xue (2014) show that Brown cluster pair feature is very impactful in implicit DRR (Rutherford and Xue, 2014). This feature is superior to one-hot representation for the interactions between two arguments, such as cross-argument word pair features in our baseline methods. We therefore conducted two additional experiments for comparison:

- **Add-Bro:** This method learns an SVM classifier using baseline system features along with the Brown cluster pair feature.
- **No-Cro:** This method learns an SVM classifier on Add-Bro's features without cross-

Relation	Model	Precision	Recall	Accuracy	MacroF1
COMP. vs Other	**SVM**	22.22	60.53	63.48	32.51
	TSVM	20.53	66.45	57.74	31.37
	Add-Bro	22.79	64.47	63.10	33.68
	No-Cro	22.89	67.76	62.14	**34.22**
	RAE	18.38	62.50	54.21	28.40
	SCNN-No-Norm	21.07	54.61	63.67	30.40
	SCNN	22.00	67.76	60.42	33.22
CONT. vs Other	**SVM**	39.70	67.03	64.05	49.87
	TSVM	38.72	67.03	62.91	49.08
	Add-Bro	39.14	72.40	62.62	50.82
	No-Cro	39.50	74.19	62.81	51.56
	RAE	37.55	68.10	61.28	48.41
	SCNN-No-Norm	39.02	71.33	62.62	50.44
	SCNN	39.80	75.29	63.00	**52.04**
EXP. vs Other	**SVM**	66.35	60.10	61.38	63.07
	TSVM	66.48	61.15	61.76	63.70
	Add-Bro	65.89	58.89	60.71	62.19
	No-Cro	66.73	61.15	61.95	63.82
	RAE	58.24	70.29	56.02	63.67
	SCNN-No-Norm	59.39	74.39	58.03	66.05
	SCNN	56.29	91.11	56.30	**69.59**
TEMP. vs Other	**SVM**	15.76	68.24	67.78	25.61
	TSVM	16.26	77.65	65.68	26.88
	Add-Bro	15.10	68.24	66.25	24.73
	No-Cro	13.89	64.71	64.53	22.87
	RAE	10.02	60.00	52.96	17.17
	SCNN-No-Norm	18.26	67.06	72.94	28.71
	SCNN	20.22	62.35	76.95	**30.54**

Table 2: Performance comparison of different systems on the test set.

argument word pair features.

In addition, to further verify the effectiveness of normalization, we also compared against SCNN model without normalization (**SCNN-No-Norm**).

Throughout our experiments, we used the toolkit *SVM-light*[6] (Joachims, 1999) in all the SVM-related experiments. Following previous work (Pitler et al., 2009; Lin et al., 2009), we adopted the following features for baseline methods:

Bag of Words: Three binary features that check whether a word occurs in *A*rg1, *A*rg2 and both arguments.

Cross-Argument Word Pairs: We group all words from *A*rg1 and *A*rg2 into two sets W_1,W_2 respectively, then extract any possible word pair $(w_i, w_j)(w_i \in W_1, w_j \in W_2)$ as features.

Polarity: The count of positive, negated positive, negative and neutral words in *A*rg1 and *A*rg2 according to the MPQA corpus (English). Their cross products are used as features.

First-Last, First3: The first and last words of each argument, the pair of the first words in two arguments, the pair of the last words in two arguments, and the first three words of each argument are used as features.

Production Rules: We extract all production rules from syntactic trees of arguments. We defined three binary features for each rule to check whether this rule appear in *A*rg1, *A*rg2 and both arguments.

Dependency Rules: We also extracted all dependency rules from dependency trees of arguments. Similarly, we defined three binary features for each rule to check whether this rule appear in *A*rg1, *A*rg2 and both arguments.

In order to collect bag of words, production rules, dependency rules, and cross-argument word pairs, we used a frequency cutoff of 5 to remove rare features, following Lin et al. (2009).

3.3 Results and Analysis

All models are evaluated by assessing the accuracy and F1 scores on account of the imbalance in test set. Besides, for better analysis, we also provided the precision and recall results.

Table 2 summarizes the performance of different models. On the whole, the F1 scores for implicit DRR are relatively low on average: COMP., CONT., EXP. and TEMP. about 32%, 50%, 65% and 28% respectively. This illustrates the difficulty in implicit DRR. Although we ex-

[6]http://svmlight.joachims.org/

pected unlabeled data could obtain improvement, we observed negative results appeared in **TSVM**: COMP. and CONT. dropped 1.14% and 0.79% respectively[7]. The F1 scores of TEMP. and EXP. are improved (1.27% and 0.63% respectively). The main reason may be that our unlabeled data is not strictly from the discourse corpus.

Incorporating Brown cluster pair features enhances the recognition of COMP. and CONT.. Particllarly, **No-Cro** achieves the best result in COMP. 34.22%. But we found no consistent improvement in EXP. and TEMP.: **No-Cro** loses 2.74% in TEMP.; **Add-Bro** loses 0.88% and 2.12% in EXP. and TEMP. respectively. This result is inconsistent with the finding of Rutherford and Xue (2014). The reason may lie in the training strategy, where we used sampling to solve the problem of unbalanced dataset while they reweighted training samples.

Compared with SVM-based models, **RAE** performs poorly in three relations, except EXP. which has the largest training dataset. Maybe **RAE** needs more labeled training data for better results. However, SCNN models perform remarkably well, producing comparable and even better results. Without normalization, **SCNN-No-Norm** gains 0.57%, 2.98% and 3.1% F1 scores for CONT., EXP. and TEMP. respectively, but loses 2.11% for COMP.. We obtain further improvement using **SCNN** with normalization: 0.71%, 2.17%, 6.52% and 4.93% for COMP., CONT., EXP. and TEMP. respectively. This suggests that normalization is useful for generalization of shallow models.

From Table 2, we found that our models do not achieve consistent improvements in precision, but benefit greatly from the gains of recall. Besides, our model works quite well for small dataset (Both accuracy and F1 score are improved in TEMP.). All of these demonstrate that our model is suitable for implicit DRR.

4 Conclusion and Future Work

In this paper, we have presented a convolutional neural network based approach to learn better DRR classifiers. The method is simple but effective for relation recognition. Experiment results show that our approach achieves satisfactory performance against the baseline models.

In the future, we will verify our model on other languages, for example, Chinese and Arabic. Besides, since our model is general to classification problems, we would like to investigate its effectiveness on other similar tasks, such as sentiment classification and movie review classification, etc.

[7]Without special illustration, all improvements and declines are against **SVM**.

Acknowledgments

The authors were supported by National Natural Science Foundation of China (Grant Nos 61303082 and 61403269), Natural Science Foundation of Jiangsu Province (Grant No. BK20140355), Natural Science Foundation of Fujian Province of China (Grant No. 2013J01250), the Special and Major Subject Project of the Industrial Science and Technology in Fujian Province 2013 (Grant No. 2013HZ0004-1), and 2014 Key Project of Anhui Science and Technology Bureau (Grant No. 1301021018). We thank the anonymous reviewers for their insightful comments. We are also grateful to Kaixu Zhang for his valuable suggestions.

References

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *JMLR*, 3:1137–1155.

Yoshua Bengio, Holger Schwenk, Jean-Sébastien Senécal, Fréderic Morin, and Jean-Luc Gauvain. 2006. Neural probabilistic language models. In *Innovations in Machine Learning*, pages 137–186. Springer Berlin Heidelberg.

Chloé Braud and Pascal Denis. 2014. Combining natural and artificial examples to improve implicit discourse relation identification. In *Proc. of COLING*, pages 1694–1705.

Philipp Cimiano, Uwe Reyle, and Jasmin Šarić. 2005. Ontology-driven discourse analysis for information extraction. *Data & Knowledge Engineering*, pages 59–83.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *JMLR*, pages 2493–2537.

Francisco Guzmán, Shafiq Joty, Lluís Màrquez, and Preslav Nakov. 2014. Using discourse structure improves machine translation evaluation. In *Proc. of ACL*, pages 687–698. Association for Computational Linguistics.

T. Joachims. 1999. Making large-scale SVM learning practical. In *Advances in Kernel Methods - Support Vector Learning*, chapter 11, pages 169–184. MIT Press, Cambridge, MA.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proc. of EMNLP*, pages 1746–1751. Association for Computational Linguistics.

Man Lan, Yu Xu, and Zhengyu Niu. 2013. Leveraging synthetic discourse data via multi-task learning for implicit discourse relation recognition. In *Proc. of ACL*, pages 476–485. Association for Computational Linguistics.

Peng Li, Yang Liu, and Maosong Sun. 2013. Recursive autoencoders for ITG-based translation. In *Proc. of EMNLP*, pages 567–577. Association for Computational Linguistics.

Jiwei Li, Rumeng Li, and Eduard Hovy. 2014. Recursive deep models for discourse parsing. In *Proc. of EMNLP*, pages 2061–2069. Association for Computational Linguistics.

Ziheng Lin, Min-Yen Kan, and Hwee Tou Ng. 2009. Recognizing implicit discourse relations in the Penn Discourse Treebank. In *Proc. of EMNLP*, pages 343–351. Association for Computational Linguistics.

Annie Louis, Aravind Joshi, Rashmi Prasad, and Ani Nenkova. 2010. Using entity features to classify implicit discourse relations. In *Proc. of SIGDIAL*, pages 59–62. Association for Computational Linguistics.

Kathleen McKeown and Or Biran. 2013. Aggregated word pair features for implicit discourse relation disambiguation. In *Proc. of ACL*, pages 69–73. The Association for Computational Linguistics.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. *CoRR*, abs/1310.4546.

Eleni Miltsakaki, Nikhil Dinesh, Rashmi Prasad, Aravind Joshi, and Bonnie Webber. 2005. Experiments on sense annotations and sense disambiguation of discourse connectives. In *Proc. of TLT2005*.

Joonsuk Park and Claire Cardie. 2012. Improving Implicit Discourse Relation Recognition Through Feature Set Optimization. In *Proc. of SIGDIAL*, pages 108–112. Association for Computational Linguistics.

Emily Pitler, Mridhula Raghupathy, Hena Mehta, Ani Nenkova, Alan Lee, and Aravind K Joshi. 2008. Easily identifiable discourse relations. *Technical Reports (CIS)*.

Emily Pitler, Annie Louis, and Ani Nenkova. 2009. Automatic sense prediction for implicit discourse relations in text. In *Proc. of ACL-AFNLP*, pages 683–691. Association for Computational Linguistics.

Rashmi Prasad, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Livio Robaldo, Aravind K Joshi, and Bonnie L Webber. 2008. The penn discourse treebank 2.0. In *Proc. of LREC*. Citeseer.

Attapol Rutherford and Nianwen Xue. 2014. Discovering implicit discourse relations through brown cluster pair representation and coreference patterns. In *Proc. of EACL*, pages 645–654. Association for Computational Linguistics.

Richard Socher, Eric H. Huang, Jeffrey Pennin, Christopher D Manning, and Andrew Y. Ng. 2011a. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Proc. of NIPS*, pages 801–809. Curran Associates, Inc.

Richard Socher, Cliff Chiung-Yu Lin, Andrew Y. Ng, and Christopher D. Manning. 2011b. Parsing natural scenes and natural language with recursive neural networks. In *Proc. of ICML*, pages 129–136. Omnipress.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proc. of EMNLP*, pages 1631–1642. Association for Computational Linguistics.

Suzan Verberne, Lou Boves, Nelleke Oostdijk, and Peter-Arno Coppen. 2007. Evaluating discourse-based answer extraction for why-question answering. In *Proc. of SIGIR*, pages 735–736. ACM.

Yannick Versley. 2013. Subgraph-based classification of explicit and implicit discourse relations. In *Proc. of IWCS*, pages 264–275. Association for Computational Linguistics.

Xun Wang, Sujian Li, Jiwei Li, and Wenjie Li. 2012. Implicit discourse relation recognition by selecting typical training examples. In *Proc. of COLING*, pages 2757–2772.

On the Role of Discourse Markers for Discriminating Claims and Premises in Argumentative Discourse

Judith Eckle-Kohler[1] **Roland Kluge**[3] **Iryna Gurevych**[1,2]
[1] UKP Lab, Technische Universität Darmstadt
[2] UKP Lab, German Institute for Educational Research
[3] Real-Time Systems Lab, Technische Universität Darmstadt
http://www.ukp.tu-darmstadt.de

Abstract

This paper presents a study on the role of discourse markers in argumentative discourse. We annotated a German corpus with arguments according to the common claim-premise model of argumentation and performed various statistical analyses regarding the discriminative nature of discourse markers for claims and premises. Our experiments show that particular semantic groups of discourse markers are indicative of either claims or premises and constitute highly predictive features for discriminating between them.

1 Introduction

The growing field of argumentation mining in NLP develops methods to automatically analyze argumentative discourse for enhanced Information Extraction.

Terminology: We understand argumentation as a rhetorical arrangement of *arguments* with the intent to convince or persuade the reader of a particular state of affairs. Following previous work in argumentation mining (e.g., (Palau and Moens, 2009; Florou et al., 2013; Peldszus and Stede, 2013a; Stab and Gurevych, 2014)), we define an *argument* as the pairing of a single *claim* (an arguable text unit) and a (possibly empty) set of *premises*, which each either support or attack the claim (Besnard and Hunter, 2008). We subsume claims and premises under the term *argument unit* (AU).

Discourse markers in argumentative discourse: Since an argumentation line can only be captured in the context of a coherent text, argumentation mining is closely related to automated discourse analysis (Cabrio et al., 2013), which aims at identifying discourse functions of text segments, and discourse relations (DRs) between adjacent text segments (Webber et al., 2012). Often, so-called discourse markers (DMs) are used to signal DRs. The following example shows that DMs act as lexical markers in argumentative discourse as well: the DM *however* (marking the DR *contrast*) positions the claim in (1) in the overall argumentation line, while in (2), the DMs *as* (marking *reason*) and *also* (marking *elaboration*) connect the premises with each other and with the claim.

> (1) *However*, staying down is pointless from a pedagogic point of view.
>
> (2) *As* the students get frustrated, their performance generally does not improve. *Also*, the point of repeating all courses because of only one or two bad grades is arguable.

DMs belong to the word classes of conjunctions and adverbs (also called discourse particles) and are semantically characterized in traditional grammar books. The correspondence between DM semantics and DR semantics has received considerable attention in previous research in linguistics, most of which is based on corpora annotated with DRs (Carlson et al., 2003; Wolf and Gibson, 2005; Prasad et al., 2008). In contrast, the role of DMs as potential lexical signals in *argumentative* discourse is not well-understood, yet. While Stab and Gurevych (2014) used DMs as features for classifying AUs into different types, they did not analyze the semantics of DMs with respect to AU classification or considered different DM resources.

As far as we are aware, there is no prior work performing a detailed investigation on the role of DMs as lexical signals for discriminating between the two fundamental argumentative roles of AUs, i.e., between claims and premises.

Our contribution: In this paper, we address this gap by analyzing the role of DMs for the automatic discrimination of claims and premises in a

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 2236–2242,
Lisbon, Portugal, 17-21 September 2015.

new dataset of argumentative texts annotated with arguments. More specifically, we (i) present the results of an annotation study that we performed to annotate a German dataset of news documents with arguments, and (ii) performed a detailed investigation of the role of DMs in our annotated dataset which highlights correspondences between DM semantics and semantic properties of claims and premises, and shows that DMs are highly predictive features for automatically discriminating premises and claims.

2 Related Work

Related to our work are prior investigations (i) on the relationship of DMs and DRs, and (ii) on the role of DMs in classification tasks.

2.1 Relation of DMs and DRs

Previous work on the relation between DMs and DRs is mostly based on corpora annotated with DRs (Wolf et al., 2004; Taboada, 2006; Prasad et al., 2008), most notably the Penn Discourse Treebank (PDTB) for English (Prasad et al., 2008). The PDTB is annotated with DRs, such as *contrast* or *result*, and the corresponding DMs, even if they were not realized in the text. For instance, the *contrast* relation can be expressed by the DMs *however* and *but*. DRs that are lexically signaled by DMs in the text are called *explicit* DRs. Some DMs are highly polysemous, e.g., *while* appears in 12 DRs in the PDTB.

Asr and Demberg (2013) analyzed the DMs and their corresponding DRs annotated in the PDTB and addressed the question, which information is conveyed by discourse connectives in the context of human sentence processing, i.e., how they contribute in the process of inferring a particular DR.

Taboada (2006) performed a corpus-based analysis of DRs annotated in the Rhetorical Structure Theory (RST) Discourse Treebank (Carlson et al., 2003). The most frequent relation in the RST Discourse Treebank is *concession*, and this relation also received particular attention in the corpus linguistics literature: Taboada and Gómez-González (2012) present a corpus-based comparative study of DMs that express *concession* across English and Spanish in different genres. A classification of DMs signaling *concession* across English and German is presented by Grote et al. (1997). They also point out the importance of *concession* in argumentative discourse: DMs expressing *concession* are often used to introduce counter-arguments in an argumentation line.

2.2 DMs in Classification

There is previous work in sentiment classification and argumentation mining using DMs as features, as well as work in predicting DMs for natural language generation tasks, such as abstractive summarization. Taboada et al. (2011) successfully employed discourse particles as features for the calculation of polarity scores in automated sentiment analysis. They focused on particles acting as intensifiers, which modify the semantic intensity of the lexical item they refer to.[1] Mukherjee and Bhattacharyya (2012) demonstrated that using discourse connectives as features in a system for sentiment classification of Twitter posts significantly improves classification accuracy over state-of-the art systems not considering DMs as features.

In argumentation mining, Stab and Gurevych (2014) experimented with different types of features, including DMs from the PDTB annotation guidelines, to classify text units into the classes non-argumentative, major claim, claim, and premise. While the PDTB DMs appeared to be not helpful for discriminating between argumentative and non-argumentative text units, they were useful to distinguish between the classes premise and claim.

Patterson and Kehler (2013) describe a classification model, trained and evaluated on PDTB data, for predicting whether or not a DR is signaled by an explicit DM. The most predictive features in their model are discourse level features that encode dependencies between neighboring DRs given by the overall discourse structure. Other highly predictive features turned out to be the DMs themselves, because DMs vary as to their rates of being realized explicitly.[2]

3 Annotation Study

For our study, we used a dataset of 88 documents in German – mainly news (ca. 83% of the documents) – from seven current topics related to the German educational system (e.g., mainstreaming, staying down at school). The documents were manually selected from a focused crawl and the top 100 search engine hits per topic (Vovk, 2013).

[1] Amplifiers such as *very* increase the semantic intensity, while downtoners such as *slightly* decrease it.

[2] In the PDTB, DMs are also given for implicit DRs.

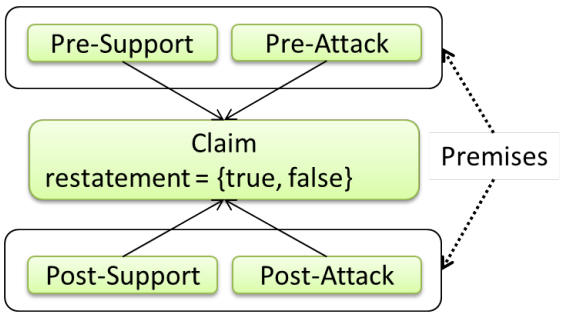


Figure 1: Claim-Premise scheme.

3.1 Annotation Scheme

Since argumentation theory offers a wide range of models, we performed a pre-study on a held-out development set to develop the annotation scheme. We found that most arguments consisted of adjacent claims and premises, related by a support or attack relation; premises and claims were rarely nested. Therefore, we decided to use a simplified claim-premises scheme (see Figure 1), using the terminology set up in Section 1.

Our scheme models an argument as a linked set of AUs and encodes the argumentative support and attack relations by assigning an argumentative role to *non-nested* (e.g., adjacent) AUs. The non-nested structure in combination with the *pre-* and *post-* prefixes of premises (indicating if a premise precedes or follows the claim) makes sure that all premises are correctly attached to their respective claims.

For claims, we also annotated whether the claim is a restatement. In total, we distinguish six argumentative roles as shown in Figure 1: *claim, restatement, pre-claim support, post-claim support, pre-claim attack and post-claim attack*. Annotators could freely select annotation boundaries, but we encouraged them to annotate clauses or sentences.

In the main study, three annotators annotated the remaining 80 documents (3 863 sentences) belonging to six topics. On average, each annotator marked 1 708 AUs (53 % premises, 47 % claims) and 783 arguments (2.2 AUs per argument). An average claim spans 1.1 sentences, whereas an average premise spans 2.2 sentences. On average, 74 % of the tokens are covered by an AU, indicating that the documents are highly argumentative.

	$A_{o,t}$	κ_t	$A_{o,s}$	κ_s	α_u
all	61.0	44.2	60.9	45.2	40.2
prem. / claims	62.7	45.1	62.6	46.3	41.7
AU / non-AU	79.0	50.0	77.0	50.0	56.6

Table 1: Inter-annotator agreement scores (percentages): A_o – observed agreement, κ_t– token-based kappa; κ_s– sentence-based kappa; α_u– unitized alpha.

3.2 Inter-annotator Agreement

Like in other discourse annotation tasks, there is no straightforward way to compute inter-annotator agreement (IAA) due to free annotation boundaries, see for instance (Miltsakaki et al., 2004; Wilson and Wiebe, 2003). We selected sentence-based *kappa* κ_s, token-based *kappa* κ_t and *Krippendorff's unitized alpha* α_u as IAA metrics.[3] The *token-based kappa* κ_t metric treats each token as annotation item and, thereon, calculates Fleiss' kappa (Fleiss, 1971). The tokens are labeled with the IOB scheme, i.e., every token is annotated as inside an AU (I), beginning of an AU (B) or outside an AU (O). In contrast, *Krippendorff's unitized alpha* α_u operates on whole annotation spans instead of isolated tokens. For comparison, we also calculated the observed agreement A_o (token-based and sentence-based).

Table 1 summarizes the IAA scores for three scenarios: (i) considering all six labels and non-AUs, (ii) only premise, claim and non-AUs, and (iii) AUs versus non-AUs. The IAA scores are in line with previous results: Peldszus and Stede (2013b) report κ=38.4 % and α=42.5 % for their sentence-level annotation study, which used artificially created documents.

4 Experiments

This section describes the experiments we performed to understand the role of DMs for the automatic discrimination of claims and premises. Since the IAA was not substantial, we performed all experiments on three separate datasets, one per annotator.

[3] We calculated the metrics using DKPro Statistics (Meyer et al., 2014).

4.1 Semantically Categorized DMs

Our first set of experiments aimed at understanding the correspondence of DM semantics and semantic aspects of claims and premises. For this, we used semantically characterized lists of DMs compiled from three different sources: (i) 28 semantically categorized particles from a German grammar (Helbig and Buscha, 1996, pp. 481–484), (ii) 51 discourse connectives based on a manual translation of the DMs in Appendix B of the PDTB annotation guidelines, and (iii) a large German lexicon of about 170 DMs called DiMLex[4] (Stede, 2002).

First, we applied a two-sample statistical test to find out if the two classes *claim* and *premise* are significantly different regarding the number of occurrences of individual DMs and semantic groups of DMs (based on the semantic information given in our lists). We chose Fisher's exact test (Fisher, 1932), a non-parametric randomization test that makes no assumptions about the underlying probability distribution of the DMs.[5] Lacking DR annotations, we counted surface word forms of single word and continuous multi-word DMs in the AUs annotated in our dataset.[6] For each semantically categorized DM, we computed a contingency table containing the number of observed occurrences in the two samples of claims and premises. For each semantic group of DMs given in DiMLex and PDTB, we calculated the per-group contingency table by adding up the contingency tables for each DM in that particular group.

The results of the significance tests revealed both single DMs and semantic groups of DMs that occur with significantly different frequency in claims or premises. The following individual DMs and semantic groups of DMs appeared to be indicative for claims and premises (note that the sentence-initial variants are counted separately):

Claims: four DMs expressing *result, comparison, contrast*:

- *also* (therefore), *Doch* (However), *jedoch* (though), *sondern* (but), as well as the amplifier *ganz* (quite);
- semantic groups of DMs from PDTB: *comparison* (expressing *concession* and *contrast*) and *result*.

Premises: three DMs expressing *cause, reason, elaboration, alternative*:

- *Denn* (As), *oder* (or), *und* (and), as well as the downtoner *etwa* (roughly);
- semantic groups of DMs from PDTB: *alternative* (e.g., *or*), *reason* (e.g., *because*), and from DiMLex, the group *sequence* (e.g., *then*).

In the group of high-frequent but non-indicative DMs are DMs expressing *elaboration* (*Auch* (Also), *Und* (And), *So* (Therefore)), *sequence* (*dann* (then)), and *contrast* (*aber/Aber* (but/But)), as well as some highly ambiguous particles (e.g., *immer* (always), *schon* (already)).

Second, we ranked the DMs according to their Information Gain (IG) using Weka (Hall et al., 2009). For this, we mapped each DM to a boolean feature indicating if it is present in an AU or not. We considered all the DMs from the three resources as features and ranked them by IG separately for each annotator. The resulting ranking revealed additional DMs indicative for premises, e.g. further DMs expressing *elaboration* and the downtoner *nur* (only).

Our findings are in accordance to the ability of a claim to act as conclusion or result of an argument, and to the role of premises as providing support for a claim by giving reasons and elaborating on them. Moreover, we found that particular intensifying discourse particles play an important role in discriminating claims and premises: Downtoners seem to be significant for premises, amplifiers for claims. Finally, semantic groups of DMs expressing *concession* appeared to be significant for claims, since claims often introduce a counter-argument in the overall argumentation structure (Grote et al., 1997).

4.2 Classification

The significance test and the IG ranking evaluated the DMs in isolation. In order to examine the predictiveness of all DMs *in combination* for claims vs. premises, we used a classification model. Given an AU, the model predicts its argumentative role, i.e., claim vs. premise. In this experiment, we used a list of single word DMs, extracted from the three DM resources described above, as boolean features (DMres).

[4] https://github.com/discourse-lab/dimlex

[5] We used the implementation given in http://stat.ethz.ch/R-manual/R-patched/library/stats/html/fisher.test.html and a p-value of 0.05.

[6] Sentence initial DMs were counted separately to capture DRs being signaled by a sentence initial position.

For comparison, we used two sets of data-driven DMs extracted from the Tiger corpus (Brants et al., 2004), a German newspaper corpus: one set containing the 350 most frequent conjunctions and adverbs (DMtiger), and another set containing the 300 most frequent non-open-class words (NOCtiger, excluding the word classes of nouns, main verbs and adjectives).[7] In addition, we considered the top 1 800 unigrams (with minimum frequency 5) as baseline features.

We trained three Machine Learning algorithms (Naive Bayes (NB), Random Forests (RF) and Support Vector Machine (SVM)) on the three datasets annotated by the annotators A1, A2 and A3, using 10-fold cross-validation and the DKPro TC framework (Daxenberger et al., 2014). For the classification experiments, we used all 88 documents in the annotated corpus (including the documents from the pre-study).

Table 2 summarizes the results. All classifiers show significant improvement compared to the majority class baseline (MC), indicating that DMs might be useful as predictive features for discriminating claims and premises. The DMs extracted from Tiger did not improve the performance consistently for all three datasets, showing that the coverage of the manually compiled DMs is good. Using the NOCtiger set, however, improved the performance by up to 4 pp., compared to DMres; the improvement of NOCtiger over DMres is statistically significant for NB across all datasets A1 - A3 (using Fisher's exact test and a p-value of 0.05). These results suggest that not only DMs, but also other function words, as well as auxiliaries and modals, play an important role in discriminating claims and premises.

While the unigram baseline also outperforms DMres, it is on par with NOCtiger (no significant improvement for any of NB, RF, SVM across the three datasets), but at the cost of a much larger feature space and a model less able to generalize to other datasets from the news domain.

	A1	**A2**	**A3**
MC	53.04	52.05	51.71
DMres-NB	64.74	65.21	64.53
DMres-RF	64.89	63.61	63.50
DMres-SVM	68.50	66.31	67.06
DMtiger-NB	67.81	65.65	66.65
DMtiger-RF	64.65	61.12	63.44
DMtiger-SVM	70.37	65.59	67.57
NOCtiger-NB	70.86	68.80	68.87
NOCtiger-RF	67.06	64.71	65.51
NOCtiger-SVM	71.80	68.25	68.41
unigram-NB	72.79	69.69	69.60
unigram-RF	70.32	68.75	68.15
unigram-SVM	71.21	69.97	71.14

Table 2: Accuracy (percent) using different feature sets.

5 Summary

Our goal was to understand some of the lexical-semantic characteristics of claims and premises with a focus on DMs acting as lexical signals for the argumentative role of AUs. Such insights into the way claims and premises are signaled by lexical markers can be exploited not only for the automatic analysis of argumentative discourse, but also for language generation tasks such as creating abstractive summaries of argumentative documents.

We investigated the role of a large set of DMs in argumentative discourse based on a German dataset annotated with arguments and identified semantic groups of DMs that are indicative of either claims or premises. These semantic groups also shed light on semantic aspects of claims and premises. Our classification model shows that DMs are important features for the discrimination of claims and premises. In order to foster further research on DMs in argumentative discourse, we publicly released the annotation guidelines, as well as the semantically categorized DM lists used in our experiments.[8]

Acknowledgments

This work has been supported by the Volkswagen Foundation as part of the Lichtenberg-Professorship Program under grant No. I/82806, by the German Institute for Educational Research (DIPF), and by the German Research Foundation under grant No. GU 798/17-1. We thank the anonymous reviewers for their valuable comments. We also thank Georg Weber for his contributions to the annotation study.

[7] We used DKPro Core (Eckart de Castilho and Gurevych, 2014) to pre-process Tiger.

[8] https://www.ukp.tu-darmstadt.de/data/argumentation-mining/argument-annotated-news-articles/

References

Fatemeh Torabi Asr and Vera Demberg. 2013. On the Information Conveyed by Discourse Markers. In *Proceedings of the Fourth Annual Workshop on Cognitive Modeling and Computational Linguistics (CMCL)*, pages 84–93, Sofia, Bulgaria.

Philippe Besnard and Anthony Hunter. 2008. *Elements of argumentation*, volume 47. MIT press Cambridge.

Sabine Brants, Stefanie Dipper, Peter Eisenberg, Silvia Hansen, Esther König, Wolfgang Lezius, Christian Rohrer, George Smith, and Hans Uszkoreit. 2004. TIGER: linguistic interpretation of a German corpus. *Research on Language and Computation*, 2(4):597–620.

Elena Cabrio, Sara Tonelli, and Serena Villata. 2013. From discourse analysis to argumentation schemes and back: Relations and differences. In João Leite, TranCao Son, Paolo Torroni, Leon Torre, and Stefan Woltran, editors, *Computational Logic in Multi-Agent Systems*, volume 8143 of *Lecture Notes in Computer Science*, pages 1–17. Springer Berlin Heidelberg.

Lynn Carlson, Daniel Marcu, and Mary Ellen Okurowski, 2003. *Building a discourse-tagged corpus in the framework of rhetorical structure theory*, chapter 5, pages 85–112. Springer Berlin Heidelberg.

Johannes Daxenberger, Oliver Ferschke, Iryna Gurevych, and Torsten Zesch. 2014. DKPro TC: A Java-based Framework for Supervised Learning Experiments on Textual Data. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics (ACL): System Demonstrations*, pages 61–66, Baltimore, MD, USA.

Richard Eckart de Castilho and Iryna Gurevych. 2014. A broad-coverage collection of portable NLP components for building shareable analysis pipelines. In *Proceedings of the Workshop on Open Infrastructures and Analysis Frameworks for HLT (OIAF4HLT) at COLING 2014*, pages 1–11, Dublin, Ireland.

Ronald Aylmer Fisher. 1932. *Statistical Methods for Research Workers*. Oliver and Boyd, London.

Joseph L. Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological Bulletin*, 76(5):378–382.

Eirini Florou, Stasinos Konstantopoulos, Antonis Koukourikos, and Pythagoras Karampiperis. 2013. Argument extraction for supporting public policy formulation. In *Proceedings of the 7th Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities*, pages 49–54, Sofia, Bulgaria.

Brigitte Grote, Nils Lenke, and Manfred Stede. 1997. Ma(r)king concessions in English and German. *Discourse Processes*, 24(1):87–118.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. 2009. The WEKA data mining software: an update. *ACM SIGKDD explorations newsletter*, 11(1):10–18.

Gerhard Helbig and Joachim Buscha. 1996. *Deutsche Grammatik. Ein Handbuch für den Ausländerunterricht.* Langenscheidt.

Christian M. Meyer, Margot Mieskes, Christian Stab, and Iryna Gurevych. 2014. DKPro Agreement: An Open-Source Java Library for Measuring Inter-Rater Agreement. In *Proceedings of the 25th International Conference on Computational Linguistics (COLING): System Demonstrations*, pages 105–109, Dublin, Ireland.

Eleni Miltsakaki, Rashmi Prasad, Aravind Joshi, and Bonnie Webber. 2004. Annotating discourse connectives and their arguments. In *Proceedings of the HLT/NAACL Workshop on Frontiers in Corpus Annotation*, pages 9–16, Boston, Massachusetts, USA.

Subhabrata Mukherjee and Pushpak Bhattacharyya. 2012. Sentiment analysis in twitter with lightweight discourse analysis. In *Proceedings of the 24th International Conference on Computational Linguistics (COLING)*, pages 1847–1864, Mumbai, India.

Raquel Mochales Palau and Marie-Francine Moens. 2009. Argumentation mining: The detection, classification and structure of arguments in text. In *Proceedings of the 12th international conference on artificial intelligence and law*, pages 98–107, New York, NY, USA.

Gary Patterson and Andrew Kehler. 2013. Predicting the Presence of Discourse Connectives. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 914–923, Seattle, Washington, USA.

Andreas Peldszus and Manfred Stede. 2013a. From Argument Diagrams to Argumentation Mining in Texts: A Survey. *International Journal of Cognitive Informatics and Natural Intelligence (IJCINI)*, 7(1):1–31.

Andreas Peldszus and Manfred Stede. 2013b. Ranking the annotators: An agreement study on argumentation structure. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 196–204, Sofia, Bulgaria.

Rashmi Prasad, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Livio Robaldo, Aravind Joshi, and Bonnie Webber. 2008. The Penn Discourse TreeBank 2.0. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC)*, pages 28–30, Marrakech, Morocco.

Christian Stab and Iryna Gurevych. 2014. Identifying Argumentative Discourse Structures in Persuasive Essays. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 46–56, Doha, Qatar.

Manfred Stede. 2002. DiMLex: A Lexical Approach to Discourse Markers. In Alessandro Lenci and Vittorio Di Tomaso, editors, *Exploring the Lexicon. Theory and Computation*. Edizioni Dell'Orso, Alessandria.

Maite Taboada and María de los Angeles Gómez-González. 2012. Discourse markers and coherence relations: Comparison across markers, languages and modalities. *Linguistics and the Human Sciences*, 6(1-3):17–41.

Maite Taboada, Julian Brooke, Milan Tofiloski, Kimberly Voll, and Manfred Stede. 2011. Lexicon-based methods for sentiment analysis. *Computational Linguistics*, 37(2):267–307.

Maite Taboada. 2006. Discourse Markers as Signals (or Not) of Rhetorical Relationsteu. *Journal of Pragmatics*, 38(4):567–592.

Artem Vovk. 2013. Discovery and Analysis of Public Opinions on Controversial Topics in the Educational Domain. Master's thesis, Ubiquitious Knowledge Processing Lab, TU Darmstadt.

Bonnie Webber, Mark Egg, and Valia Kordoni. 2012. Discourse structure and language technology. *Natural Language Engineering*, 18:437–490, 10.

Theresa Wilson and Janyce Wiebe. 2003. Annotating opinions in the world press. In *Proceedings of the 4th SIGdial Workshop on Discourse and Dialogue*, pages 13–22, Hannover, Germany.

Florian Wolf and Edward Gibson. 2005. Representing Discourse Coherence: A Corpus-Based Study. *Computational Linguistics*, 31(2):249–288.

Florian Wolf, Edward Gibson, Amy Fisher, and Meredith Knight. 2004. Discourse Graphbank. Linguistic Data Consortium.

Fatal or not? Finding errors that lead to dialogue breakdowns in chat-oriented dialogue systems

Ryuichiro Higashinaka[1], **Masahiro Mizukami**[2], **Kotaro Funakoshi**[3], **Masahiro Araki**[4], **Hiroshi Tsukahara**[5], **Yuka Kobayashi**[6]
[1]NTT Corporation, [2]Nara Institute of Science and Technology
[3]Honda Research Institute Japan, [4]Kyoto Institute of Technology
[5]Denso IT Laboratory, Inc., [6]Toshiba Corporation

Abstract

This paper aims to find errors that lead to dialogue breakdowns in chat-oriented dialogue systems. We collected chat dialogue data, annotated them with dialogue breakdown labels, and collected comments describing the error that led to the breakdown. By mining the comments, we first identified error types. Then, we calculated the correlation between an error type and the degree of dialogue breakdown it incurred, quantifying its impact on dialogue breakdown. This is the first study to quantitatively analyze error types and their effect in chat-oriented dialogue systems.

1 Introduction

Chat-oriented or open-domain dialogue systems have recently been attracting attention from social and entertainment aspects (Bickmore and Cassell, 2001; Banchs and Li, 2012; Wilcock and Jokinen, 2013). However, since they need to deal with open-domain utterances, which current natural language processing techniques are not mature enough to handle appropriately, the system inevitably makes errors. This discourages users from talking to the system, leading to dialogue breakdowns in conversation (Martinovsky and Traum, 2003). Here, dialogue breakdowns denote points in dialogue where users are unable to continue the conversation.

This paper aims to find errors that lead to dialogue breakdowns in chat-oriented dialogue systems. Our approach is two-fold: (1) identify error types in chat-oriented dialogue systems, and (2) calculate their impact on dialogue breakdown. For (1), we first collect chat dialogues between an automated system and users, annotate the dialogues with dialogue breakdown labels, and collect comments that describe the error that led to the breakdowns. After that, we apply automatic clustering methods to the comments to obtain clusters of comments, each of which corresponds to a particular error type. For (2), we calculate the correlation between an error type and the degree of dialogue breakdown it incurs. This way, we can quantify the impact of an error type on dialogue breakdown. By our approach, we hope to distinguish fatal errors from non-fatal ones, providing useful guidelines for dialogue system developers to avoid breakdowns in chat-oriented dialogue systems.

2 Data Collection

For data collection, we asked dialogue researchers and their collaborators in Japan to use our chat-oriented dialogue system. The system is text-based and enables chatting in Japanese. It was built by wrapping a chat API provided by NTT DOCOMO (Onishi and Yoshimura, 2014). Since the API works on the basis of (Higashinaka et al., 2014), which uses a number of natural language processing techniques to understand and generate utterances, we expected to obtain a wide variety of dialogues, and hence, a variety of errors.

A total of 116 users chatted with the system, resulting in 1,146 dialogues. Here, each dialogue was controlled to be 21 utterances long: one system prompt with ten utterances each from the user and system. Then, we randomly sampled 100 dialogues (called the *init100* data set) for dialogue breakdown annotation. Twenty-four annotators subjectively labeled each system utterance in init100 with one of the following three dialogue breakdown labels:

(NB) Not a breakdown: The conversation is easy to continue after the system utterance in question.

(PB) Possible breakdown: The conversation is difficult to continue smoothly after the system utterance in question.

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 2243–2248,
Lisbon, Portugal, 17-21 September 2015.

	Utterance	NB	PB	B	C	Comments
S0	Hello! Talk to me anytime.	24	0	0	0	
U1	It's hot today, isn't it?					
S1	Please tell me your favorite movie genre.	20	2	2	2	User utterance is not considered. / The system ignored user's intended topic.
U2	I like action movies.					
S2	Action movies, I like them.	21	3	0	0	
U3	What kind of movies do you like?					
S3	I like Japanese ones.	23	1	0	0	
U4	That's great.					
S4	Thank you.	24	0	0	0	
U5	Do you know what movie will be aired on Friday night?					
S5	Yes, yes.	4	11	9	4	The system did not mention a movie name. / It answered only "yes" but without saying a movie name. / This is not an answer to the question. / The conversation is not valid.
U6	I haven't heard of that movie.					
S6	That's good.	3	10	11	4	This is not a response to a question. / Strange answer. / It is not clear what is good. / The conversation is not valid.
U7	Oh, is it good? I will watch it then.					
S7	That's good!	18	3	3	1	It is not clear what is good.
U8	Recently, I have had lots of free time.					
S8	That's great.	15	7	2	0	
U9	Are you busy?					
S9	Indeed so.	24	0	0	0	
U10	Take care. Have some rest.					
S10	I see.	10	9	5	2	The system should thank the user. / Strange answer.

Figure 1: Example dialogue with the number of breakdown labels for each system utterance. S and U stand for system and user utterances, respectively. C stands for the number of comments provided. The dialogue was originally in Japanese and was translated by the authors.

(B) Breakdown: The conversation is difficult to continue at all after the system utterance in question.

The annotators were instructed to provide voluntary comments to describe the errors that led to the breakdowns. Table 1 shows the statistics of the dialogue breakdown labels and comments. Figure 1 shows an example dialogue with the number of breakdown labels and comments for each system utterance. In this example, S5 and S6 were annotated with nine and eleven breakdown labels, respectively, both having four comments.

The inter-annotator agreement of dialogue breakdown annotation in Fleiss' kappa was 0.276, which seems relatively low. One reason for this is obviously the subjective nature of the task. Another possible reason is that we intentionally did not set rigid guidelines for dialogue breakdown annotation so as to explore possible error types in chat-oriented dialogue systems. When we merge PB and B and make it a two-class annotation, the agreement becomes 0.396 (moderate agreement), showing that the subjects share some common conception about dialogue breakdown.

Breakdown label	# of labels	# of comments
NB	14,212	57
PB	5,322	1,818
B	4,466	1,511

Table 1: Statistics related to breakdown labels and comments in init100 data set. Note that NB also had some opinions as comments.

3 Analysis

3.1 Automatic clustering of comments

We first need to identify the error types in chat-oriented dialogue systems. For this, we applied an automatic clustering method to the comments to obtain clusters of comments. Our idea is that, since each comment describes a particular error that led to a breakdown, a cluster of comments is likely to represent an error type. Since the number of clusters is difficult to know in advance, we turn to a non-parametric Bayesian method called the Chinese restaurant process (CRP) as a clustering method. CRP can infer the number of clusters automatically from data (Pitman, 1995).

We applied CRP to the 1,511 comments given

ID	Size	Interpretation	Representative words in the cluster
2	259	General quality	dialogue, well-formed, consideration, conversation
0	194	Not understandable	understand, meaning, what
6	148	Ignore user utterance	ignore, user, utterance
7	134	Ignore user question	answer, question, partner, respond
1	113	Unclear intention	unclear, intention, meaning, utterance
8	107	Contradiction	doubtful, change, negative (as opposed to positive), previous
16	100	Analysis failure	analysis, recognition, related, understand
4	95	Inappropriate answer	response, inappropriate, invalid, answer
5	77	Repetition	say, add, mind, strange, tired
3	54	Grammatical error	(words specific to particular grammar usage)
14	53	Expression error	(words specific to particular expressions)
11	39	Topic-change error	change, topic, sudden, mismatch, response
12	38	Violation of common sense	match, flow, common sense, against, connection
13	36	Word usage error	(words specific to particular word usage)
10	35	Diversion	story, different
15	25	Mismatch in conversation	match, story
9	4	Social error	(no words)

Table 2: Clusters by CRP of 1,511 comments given to breakdowns. The clusters are ordered by size.

to breakdown labels (B labels). For the clustering, we used the same procedure as (Higashinaka et al., 2011); each datum (comment) was represented by a word frequency vector, and the probability that it belonged to a particular cluster was calculated by using the likelihood that the words are generated from the word distribution of that cluster. The hyper-parameters α and β were both set to 0.1 and the number of iterations for Gibbs sampling was 10,000. See (Higashinaka et al., 2011) for the details of the procedure.

Table 2 shows the clustering results. We obtained 17 clusters. For each cluster, we mined representative words by a log-likelihood ratio test, which uses a two-by-two matrix to test the independence of a word to a particular cluster. By looking at the representative words and also the raw comments, we came up with the interpretations of the clusters as indicated in the table. Although we do not go into the details of the clusters one by one, each cluster seems to successfully represent a certain error type in chat-oriented dialogue systems. We also applied CRP to the 3,329 comments given to PB and B to obtain similar clusters except that we additionally had clusters whose interpretations are as follows: inability to handle invalid user input, missing topic, missing information, mismatch in response, no reaction, and no information. They account for about 13.3% of the comments and mostly concern missing elements (such as missing arguments) in dialogue. Since such missing elements can be complemented by follow-on utterances in dialogue, they only appear in the comments for PB; they do not lead to an immediate dialogue breakdown.

To further categorize the clusters, we applied a hierarchical clustering (an agglomerative clustering) to the clusters. Here, a cluster was represented by the word frequency vector of all comments contained in the cluster, and the similarity of the clusters was calculated by cosine similarity of word frequency vectors. For the linkage criterion, we used Ward's method. Figure 2 shows the clustering results. The figure indicates that there are the following eight main error categories (E1–E8):

(E1) Clusters 2 and 16 concern the general ability of a system.

(E2) Clusters 7, 5, and 8 relate to context awareness: the ability to recognize when it is asked a question and to recognize what has been said before.

(E3) Clusters 13, 3, and 14 concern the language generation (surface realization) ability.

(E4) Clusters 4 and 6 concern the response ability: the ability to answer questions and to create utterances relevant to the previous user utterance.

(E5) Cluster 1 relates to the exhibition of an intention or a plan: the ability to make clear the

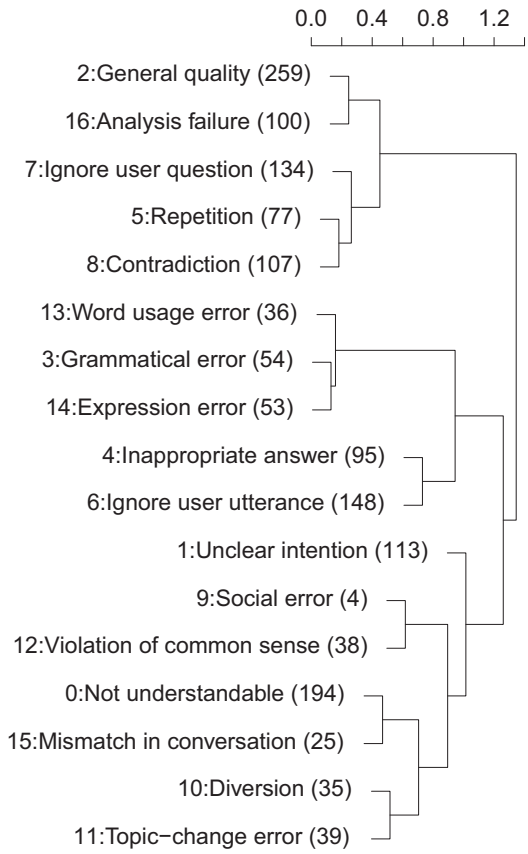


Figure 2: Hierarchical clustering applied to the obtained clusters. The numbers in parentheses denote cluster size.

purpose of an utterance.

(E6) Clusters 9 and 12 relate to the social ability: the ability not to offend users or say things that are not socially acceptable.

(E7) Clusters 0 and 15 concern the understandability of an utterance: the ability to generate utterances that have clear meanings in the context of the conversation.

(E8) Clusters 10 and 11 relate to the awareness of current topics.

3.2 Analyzing the impact of error types

Having identified the error types and error categories, we investigated their impact on dialogue breakdown. For this purpose, we examined the correlation between an error type and its degree of breakdown: the higher the correlation, the more it is related to dialogue breakdown. Specifically, we calculated the correlation ratio (η) between the existence of a comment belonging to a particular cluster (error type) and the number of breakdown labels (B labels). Note that the correlation ratio is equivalent to Pearson's correlation coefficient except that it can be applied to categorical data. The η ranges from 0 to 1. For calculation, we first extracted data that had one or more B labels and one or more corresponding comments (we had 556 such samples in our data). Then, we calculated the correlation ratios.

ID	Interpretation	Cat	η
0	Not understandable	E7	0.38
7	Ignore user question	E2	0.37
2	General quality	E1	0.36
1	Unclear intention	E5	0.36
6	Ignore user utterance	E4	0.24
13	Word usage error	E3	0.18
16	Analysis failure	E1	0.17
4	Inappropriate answer	E4	0.17
3	Grammatical error	E3	0.15
12	Violation of common sense	E6	0.15
8	Contradiction	E2	0.14
5	Repetition	E2	0.11
9	Social error	E6	0.10
10	Diversion	E8	0.09
11	Topic-change error	E8	0.06
15	Mismatch in conversation	E7	0.06
14	Expression error	E3	0.02

Table 3: Correlation ratio (η) between the existence of a comment of a cluster (error type) and the number of breakdown labels. "Cat" denotes the error category of an error type.

Table 3 shows the correlation ratios for the error types. Clearly, not all error types have the same level of correlation. At the top of the table, there are four salient error types with similar η values: "Not understandable", "Ignore user question", "General quality", and "Unclear intention". Putting aside "General quality", which seems to concern the overall dialogue ability, the error types that we need to consider as fatal would seem to be the other three. Other errors seem to be less important with lower η values. "Expression error", which concerns the use of unnatural expressions, was found the least important.

When we look at the error categories, we can see an interesting result that it is NOT the error category that determines the fatality of errors but the specificity of error types. For example, "Not understandable" and "Mismatch in conversation" are both under error category E7 but have totally different effects on perceived breakdown. The same can be said for error types in E2.

Note that, although the values of correlation ratio seem rather low, the correlation often becomes low when it comes to subjective judgments (Higashinaka et al., 2004). Considering that we deal with chat-oriented dialogues, which are less restricted than task-oriented ones, we consider the current values of correlation ratio to be acceptable. Here, the important finding is that several error types are comparatively more important than the others.

4 Related work

Few studies have analyzed breakdowns in conversation. One exception is the study by Martinovsky and Traum (2003), who discussed possible causes of breakdowns they observed. Our work is different in that we systematically identify error types and quantitatively evaluate their effect. Our work can be seen as listing up errors in dialogue systems. A number of studies have aimed to create a taxonomy of errors (Bernsen et al., 1996; Möller, 2002; Paek, 2003), but their taxonomies are created manually and focus on task-oriented dialogue systems.

5 Summary and future work

By processing dialogue data with dialogue breakdown annotations and comments, this paper identified 17 error types that can be further categorized into eight error categories. By calculating correlation ratios, we discovered three error types that can be fatal: "Not understandable", "Ignore user question", and "Unclear intention". To avoid dialogue breakdowns, it is suggested that we need to make clear the meanings of system utterances, not ignore user questions, and show some intention behind system utterances. The findings will be useful for dialogue system developers who want to realize smooth human-machine interaction in chat-oriented dialogue systems and possibly in dialogue design as a whole.

For future work, we plan to consider ways to improve systems on the basis of our findings and also verify the generality of the results on data using other systems. To accurately detect dialogue breakdowns, dialogue systems researchers will need to collaborate. To this end, we are planning to organize an evaluation workshop on dialogue breakdown detection. For use in the evaluation workshop as well as in dialogue research in general, we have released our data with all the annotations and comments to the public.[1]

Acknowledgments

The "Project Next NLP" project in Japan is addressing the problems related to analyzing errors in natural language processing systems. One subgroup, comprising more than 32 researchers from 15 institutions, is collaborating in the performance of a dialogue task. We thank the members of this sub-group for the data collection, annotation, and fruitful discussions. We also thank NTT DOCOMO for letting us use their chat API for data collection.

References

Rafael E Banchs and Haizhou Li. 2012. IRIS: a chat-oriented dialogue system based on the vector space model. In *Proc. the ACL 2012 System Demonstrations*, pages 37–42.

Niels Ole Bernsen, Hans Dybkjaer, and Laila Dybkjaer. 1996. Principles for the design of cooperative spoken human-machine dialogue. In *Proc. ICSLP*, volume 2, pages 729–732.

Timothy W. Bickmore and Justine Cassell. 2001. Relational agents: a model and implementation of building user trust. In *Proc. CHI*, pages 396–403.

Ryuichiro Higashinaka, Noboru Miyazaki, Mikio Nakano, and Kiyoaki Aikawa. 2004. Evaluating discourse understanding in spoken dialogue systems. *ACM Transactions on Speech and Language Processing (TSLP)*, 1:1–20.

Ryuichiro Higashinaka, Noriaki Kawamae, Kugatsu Sadamitsu, Yasuhiro Minami, Toyomi Meguro, Kohji Dohsaka, and Hirohito Inagaki. 2011. Unsupervised clustering of utterances using non-parametric bayesian methods. In *Proc. INTERSPEECH*, pages 2081–2084.

Ryuichiro Higashinaka, Kenji Imamura, Toyomi Meguro, Chiaki Miyazaki, Nozomi Kobayashi, Hiroaki Sugiyama, Toru Hirano, Toshiro Makino, and Yoshihiro Matsuo. 2014. Towards an open-domain conversational system fully based on natural language processing. In *Proc. COLING*, pages 928–939.

Bilyana Martinovsky and David Traum. 2003. The error is the clue: Breakdown in human-machine interaction. In *Proc. ISCA Workshop on Error Handling in Spoken Dialogue Systems*, pages 11–16.

Sebastian Möller. 2002. A new taxonomy for the quality of telephone services based on spoken dialogue systems. In *Proc. SIGDIAL*, pages 142–153.

[1] https://sites.google.com/site/dialoguebreakdowndetection/

Kanako Onishi and Takeshi Yoshimura. 2014. Casual conversation technology achieving natural dialog with computers. *NTT DOCOMO Technical Jouranl*, 15(4):16–21.

Tim Paek. 2003. Toward a taxonomy of communication errors. In *Proc. ISCA Workshop on Error Handling in Spoken Dialogue Systems*, pages 53–58.

Jim Pitman. 1995. Exchangeable and partially exchangeable random partitions. *Probability theory and related fields*, 102(2):145–158.

Graham Wilcock and Kristiina Jokinen. 2013. Wikitalk human-robot interactions. In *Proc. ICMI*, pages 73–74.

Learning Word Meanings and Grammar for Describing Everyday Activities in Smart Environments

Muhammad Attamimi[1] **Yuji Ando**[1] **Tomoaki Nakamura**[1] **Takayuki Nagai**[1]
Daichi Mochihashi[2] **Ichiro Kobayashi**[3] **Hideki Asoh**[4]
[1] The University of Electro-Communications, 1-5-1 Chofugaoka, Chofu-shi, Tokyo, Japan
[2] Institute of Statistical Mathematics, 10-3 Midori-cho, Tachikawa, Tokyo, Japan
[3] Ochanomizu University, 2-1-1 Otsuka Bunkyo-ku Tokyo, Japan
[4] National Institute of Advanced Industrial Science and Technology,
1-1-1 Umezono, Tsukuba, Ibaraki, Japan
{m_att, ando, naka_t}@apple.ee.uec.ac.jp, tnagai@ee.uec.ac.jp,
daichi@ism.ac.jp, koba@is.ocha.ac.jp, h.asoh@aist.go.jp

Abstract

If intelligent systems are to interact with humans in a natural manner, the ability to describe daily life activities is important. To achieve this, sensing human activities by capturing multimodal information is necessary. In this study, we consider a smart environment for sensing activities with respect to realistic scenarios. We next propose a sentence generation system from observed multimodal information in a bottom up manner using multilayered multimodal latent Dirichlet allocation and Bayesian hidden Markov models. We evaluate the grammar learning and sentence generation as a complete process within a realistic setting. The experimental result reveals the effectiveness of the proposed method.

1 Introduction

Describing daily life activities is an important ability of intelligent systems. In fact, we can use this ability to achieve a monitoring system that is able to report on an observed situation, or create an automatic diary of a user. Recently, several studies have been performed to generate sentences that describe images using Deep Learning (Vinyals et al., 2014; Fang et al., 2014; Donahue et al., 2014; Kiros et al., 2015). Although these results were good, we are interested in unsupervised frameworks. This is necessary to achieve a system that can adapt to the user, that is, one that can learn a user-unique language and generate it automatically. Moreover, the use of crowdsourcing should be avoided to respect the privacy of the user. Regarding this, studies on sentence generation from RGB videos have been discussed in (Yu and Siskind, 2013; Regneri et al., 2013). A promising result for language learning has been shown in (Yu and Siskind, 2013) and a quite challenging effort to describe cooking activities was made in (Regneri et al., 2013). However, these studies rely only on visual information, while we aim to build a system that is able to describe everyday activities using multimodal information. To realize such systems, we need to consider two problems. The first problem is the sensing of daily life activities. In this paper, we utilize a smart house (Motooka et al., 2010) for sensing human activities. Thanks to the smart house, multimodal information such as visual, motion, and audio data can be captured. The second problem to be tackled is verbalization of the observed scenes. To solve this problem, a multilayered multimodal latent Dirichlet allocation (mMLDA) was proposed in (Attamimi et al., 2014).

In this paper, we propose a sentence generation system from observed scenes in a bottom up manner using mMLDA and a Bayesian hidden Markov model (BHMM) (Goldwater and Griffiths, 2007). To generate sentences from scenes, we need to consider the words that represent the scenes and their order. Here, mMLDA is used to infer words for given scenes. To determine the order of words, inspired by (Kawai et al., 2014), a probabilistic grammar that considers syntactic information is learned using BHMM. In this study, the order of concepts is generated by sampling the learned grammar. The word selection for each generated concept is then performed using the observed data. Moreover, a language model that represents the relationship between words is also used to calculate

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 2249–2254,
Lisbon, Portugal, 17-21 September 2015.

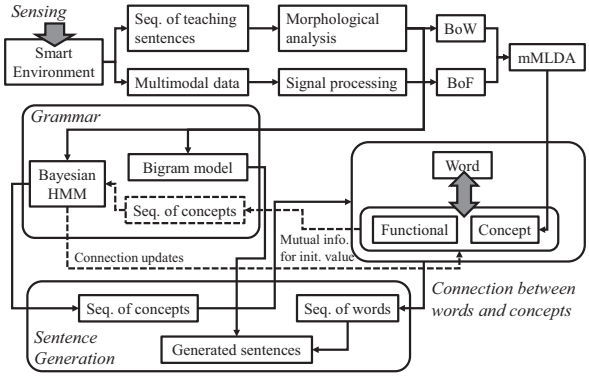


Figure 1: Language learning and sentence generation system.

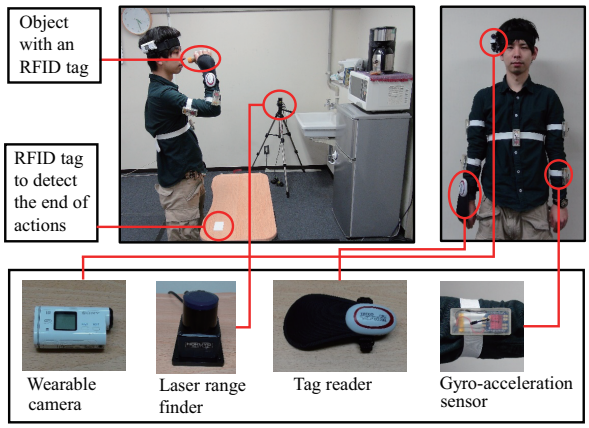


Figure 2: Multimodal information acquisition.

the transition probability between them. Considering the transition probability at word level, a lattice of word candidates corresponding to the concept sequence can be generated. Therefore, sentence generation can be thought of as a problem of finding the word sequence that has the highest probability from the lattice of word candidates, which can be solved by the Viterbi algorithm. Finally, sampling from grammar is performed multiple times to generate sentence candidates and select the most probable one.

2 Proposed method

2.1 Overview

Figure 1 illustrates the overall system of proposed language learning and sentence generation. In this study, we use a smart environment for sensing multimodal information. The system shown in Figure 2 is part of a smart house (Motooka et al., 2010) that is used to capture multimodal information. Here, an RFID tag is attached to an object

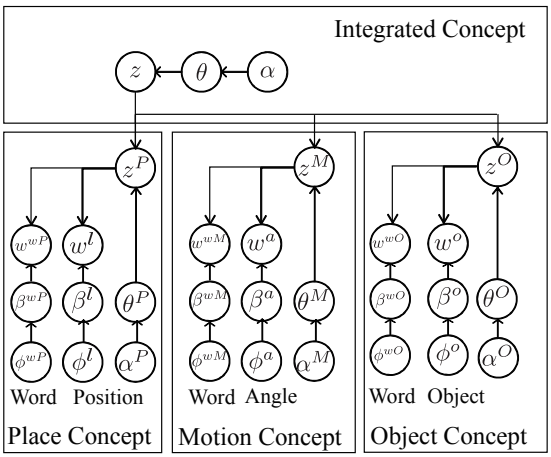


Figure 3: Graphical model of mMLDA.

to enable the object information to be read using a wearable tag reader. To capture motion, five sensors that consist of 3-axis acceleration with 3-axis gyroscope sensors are attached to the upper body, as shown in Figure 2. Moreover, a particle filter-based human tracker (Glas et al., 2007) applied to four laser range finders is used to estimate the location of a person while performing an action. This is a setup designed to demonstrate that language can be learned and generated from real human actions. Ultimately, our goal is sensing based on image recognition.

The acquired multimodal data is then processed, which results in a bag-of-words model (BoW) and bag-of-features model (BoF) (Csurka et al., 2004). Using mMLDA (see section 2.2), various concepts can be formed from the multimodal data. Given teaching sentences, the connection between words and concepts can be learned based on mMLDA and BHMM which is learned with mutual information (MI) as the initial value. On the other hand, the bigram model of words is calculated and used as the score when reordering words inferred from multimodal information using grammar. A morphological analyzer for parsing words in a sentence is also necessary in the proposed system. We use publicly available parser MeCab (Kudo et al., 2004). In the future, we plan to use the unsupervised morphological analysis technique proposed in (Mochihashi et al., 2009).

2.2 mMLDA

Figure 3 shows the graphical model of mMLDA used in this paper. Here, z represents the integrated category (concept), whereas z^O, z^M, and z^P represent the object, mo-

tion, and place concepts, respectively. In the bottom layer (lower panel of Figure 3), $w^m \in \{w^o, w^{wO}, w^a, w^{wM}, w^l, w^{wP}\}$ represents the multimodal information obtained from each object, motion, and place. Here, w^o, w^a, and w^l denote multimodal information obtained respectively from the object used in an action, motion of a person while using the object, and location of the action. Further, $w^{wC} \in \{w^{wO}, w^{wM}, w^{wP}\}$ denotes word information obtained from teaching sentences. Observation information is acquired by using the system shown in Figure 2. A brief explanation of each observation is as follows.

For object information, an N_o-dimensional vector $w^o = (o_1, o_2, \cdots, o_{N_o})$ is used, where N_o denotes the number of objects. In this vector, o_* takes a value of 0 or 1, where o_i is set to 1 if an object with index i is observed. Moreover, all of the teaching sentences are segmented into words and represented by a BoW as word information. Here, motion is segmented according to the object used. A sequence of 15-dimensional feature vectors for each motion is acquired. Using BoF, the acquired feature vectors are vector quantized, resulting in a 70-dimensional vector. The acquired two dimensional of human positions are processed using BoF to construct a 10-dimensional vector as place information.

In mMLDA, latent variables that represent upper and lower concepts z and $z^C \in \{z^O, z^M, z^P\}$ are learned simultaneously. Gibbs sampling is applied to the marginalized posterior probability of latent variables to learn the model from observed data w^m (Attamimi et al., 2014).

2.3 Language learning and generation

2.3.1 Word inference

In this study, word information is obtained from teaching sentences and employed for all concepts, as shown in Figure 3. Considering that appropriate words to express each concept exist, a criterion to measure the correlation between words and concepts is needed. At the start of grammar learning, MI, which can measure the mutual dependence of two stochastic variables, is used. Therefore, a word is considered to express a category when the MI between the word and category is large. On the other hand, a word with small MI is identified as a functional word. This determination is used as an initial value in the syntactic learning and needs not be strictly determined. Once the grammar is learned, we can utilize BHMM's parameters $P(w^w|c)$ to infer a word w^w from observed data $\boldsymbol{w}^m_{\text{obs}}$ as $\hat{P}(w^{wC}|\boldsymbol{w}^m_{\text{obs}}, c) \propto \max_k P(w^{wC}|c)P(w^{wC}|k)P(k|\boldsymbol{w}^m_{\text{obs}}, c)$, where $P(w^{wC}|k)$ and $P(k|\boldsymbol{w}^m_{\text{obs}}, c)$ can be estimated from mMLDA (Attamimi et al., 2014) and k is category of concept $c' \in \{object, motion, place\}$ and $c \in \{c', functional\}$. It should be note that $P(w^{wC}|k)$ and $P(k|\boldsymbol{w}^m_{\text{obs}}, c)$ are considered as uniform distribution for "functional" since they cannot be inferred from observed data using mMLDA. In this case, we can rely on syntactic information which is learned by BHMM.

2.3.2 Grammar learning using BHMM

Thanks to mMLDA and BHMM, appropriate words to represent the observed information can be inferred. Given an input consisting of a teaching sentence of a sequence of words, a BHMM can infer a sequence of concepts. In the learning phase, the MI results of concept selection for each word are used as the initial values of the BHMM. Here, grammar is defined as the concept transition probability $P(C_t|C_{t-1})$, which is estimated using Gibbs sampling, where $C_t \in c$ represents the corresponding concepts of the t-th word in the sentence. In addition, a language model that represent the bigram model of words in the teaching sentences is also used for generating sentences.

Motion	Object	Place	Motion	Object	Place	Motion	Object	Place
Drink (1)	Juice (1)	Sofa (1)	Wipe (7)	Dustcloth (9)	Kitchen (4)	Write on (12)	Notebook (16)	Bedroom (5)
	Tea (4)	Dining room (2)		Tissue (10)	Dining room (2)		Textbook (17)	Sofa (1)
Eat (2)	Cookies (2)	Dining room (2)	Turn on (8)	Remote control	Living room (3)		Refrigerator (18)	Kitchen (4)
	Chocolate (3)	Living room (3)		(air conditioner) (11)	Bedroom (5)	Open (13)	Microwave (19)	Kitchen (4)
Shake (3)	Tea (4)	Sofa (1)	Open (turn) (9)	Tea (4)	Living room (3)		Closet (20)	Bedroom (5)
	Dressing (5)	Kitchen (4)		Honey (6)	Dining room (2)	Read (14)	Textbook (17)	Bedroom (5)
Pour (4)	Tea (4)	Kitchen (4)	Wrap (10)	Plastic wrap (12)	Dining room (2)		Magazine (21)	Sofa (1)
	Juice (1)	Living room (3)		Aluminum foil (13)	Kitchen (4)	Spray (15)	Deodorizer (22)	Living room (3)
Put on (5)	Dressing (5)	Dining room (2)		Shirt (14)	Bedroom (5)			Bedroom (5)
	Honey (6)	Kitchen (4)	Hang (11)				Scourer (23)	Kitchen (4)
Throw (6)	Ball (7)	Sofa (1)		Parka (15)	Living room (3)	Scrub (16)	Sponge (24)	Kitchen (4)
	Plushie (8)	Bedroom (5)						

Table 1: Object, motion, and place correspondences (numbers in parentheses represent the category index).

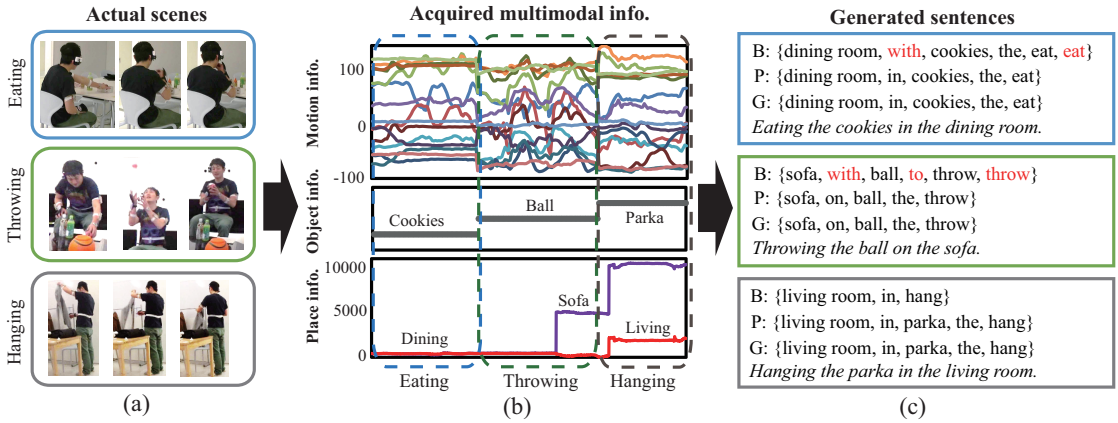


Figure 4: Examples of: (a) actual images, (b) captured multimodal information, and (c) generated sentences. In each image, B, P, and G indicate the sentence structure in Japanese grammar generated by the baseline method, proposed method, and correct sentence, respectively; whereas the bottom line gives the meaning of the generated sentence. Words marked in red have been incorrectly generated.

2.3.3 Sentence generation of observed scenes

First, concepts are sampled from the begin of sentence "BOS" until the end of sentence "EOS" according to the learned grammar N times. Let the n-th ($n \in \{1, 2, \cdots, N\}$) sequence of concepts that excludes "BOS" and "EOS" be $\boldsymbol{C}^n = \{C_1^n, \cdots, C_t^n, \cdots, C_{T_n}^n\}$, where, T_n denotes the number of sampled concepts, which corresponds to the length of a sampled sentence.

Next, the word that corresponds to concept C_t^n is estimated. Here, for a given observed information $\boldsymbol{w}_{\text{obs}}^m$, the top-$K$ words that correspond to concept C_t^n and have high probabilities $\boldsymbol{w}_t^n = \{w_{t1}^n, w_{t2}^n, \cdots, w_{tK}^n\}$ are used. Hence, the set of all words for a sequence of concepts $\boldsymbol{C}^n$ can be written as $\boldsymbol{W}^n = \{\boldsymbol{w}_1^n, \boldsymbol{w}_2^n, \cdots, \boldsymbol{w}_{T_n}^n\}$. Therefore, K^{T_n} number of patterns for a candidate of the sentence can be considered for $\boldsymbol{C}^n$ and the corresponding words $\boldsymbol{W}^n$. Each candidate for sentence S^n is selected from these patterns and has the following probability:

$$P(S^n|\boldsymbol{C}^n, \boldsymbol{W}^n, \boldsymbol{w}_{obs}^m) \propto \prod_t P(C_t^n|C_{t-1}^n) P(w_t^n|\boldsymbol{w}_{obs}^m, C_t^n) P(w_t^n|w_{t-1}^n). \quad (1)$$

For observed information, the most probable sentence is selected from N sequences of concepts with sets of words. Here, the sentence $\hat{S}^n$ that maximizes Eq. (1) is determined for each sequence of concepts. Because many patterns of S^n exist, the Viterbi algorithm is applied to cut the computational cost and determine the most probable sentence. Thus, a set of sentences that consists of sentences with the highest probability for each sequence of concepts can be written as $\hat{\boldsymbol{S}} = \{\hat{S}^1, \cdots, \hat{S}^n, \cdots, \hat{S}^N\}$.

We can select the final sentence from $\hat{\boldsymbol{S}}$ by considering the most probable candidate. In fact, long sentences tend to have low probability and are less likely to be selected. To cope with this problem, adjustment coefficient $\ell(\hat{S}^n) = \frac{(L^{\max} - L_{\hat{S}^n})}{\sum_n^N L_{\hat{S}^n}} \sum_n^N \log P(\hat{S}^n|\boldsymbol{C}^n, \boldsymbol{W}^n, \boldsymbol{w}_{obs}^m)$ is introduced, where, $L_{\hat{S}^n}$ denotes the length of sentence $\hat{S}^n$ and $L^{\max}$ represents the maximum value of the sentence length in $\hat{\boldsymbol{S}}$. Using $\ell(\hat{S}^n)$, the logarithmic probability of the sentence can be calculated as $\log \bar{P}(\hat{S}^n|\boldsymbol{C}^n, \boldsymbol{W}^n, \boldsymbol{w}_{obs}^m) = \log P(\hat{S}^n|\boldsymbol{C}^n, \boldsymbol{W}^n, \boldsymbol{w}_{obs}^m) + \omega \ell(\hat{S}^n)$, where ω is a weight that controls the length of sentences. A large weight leads to longer sentences. The final sentence S is determined as $S = \text{argmax}_{\hat{S}^n \in \hat{\boldsymbol{S}}} \log \bar{P}(\hat{S}^n|\boldsymbol{C}^n, \boldsymbol{W}^n, \boldsymbol{w}_{obs}^m)$.

3 Experiments

The acquisition system shown in Figure 2 was used to capture multimodal information from human actions. Table 1 shows the actions that were performed by three subjects twice, resulting in a total of 195 multimodal data with 1170 sentences. We then divided the data into training data (99 multimodal data with 594 sentences) and test data (96 multimodal data with 576 sentences). Some examples of acquired multimodal data are shown in Figure 4(b). Using training data, various concepts were formed by mMLDA, and the categorization accuracies for object, motion, and place were respectively 100.00%, 52.53%, and 95.96%. Motion similarity was responsible for the false cat-

	♯ of words	Baseline	Proposed
w/o functional words	78	65.38%	**73.08%**
w functional words	98	–	**68.37%**

Table 2: Concepts selection results.

egorization of motion concepts. Since our goal is to generate sentences from observed scenes, these results are used as reference instead of comparing with the baseline.

To evaluate the concept selection of words, 98 words in teaching sentences were used. We compared the results of concept selection with hand-labeled ones. Table 2 shows the accuracy rate of concept selection. Here, we excluded the functional words (resulting in 78 words) for fair comparison with the baseline method (Attamimi et al., 2014). One can see that, better results can be achieved by the proposed method. It is clear that concept selection is improved by using the BHMM, indicating that a better grammar can be learned using this model.

Next, the learned grammar was used and sentences were generated. To reduce randomness of the results, sentence generation was conducted 10 times for each data. To verify sentence generation quantitatively, we evaluated the sentences automatically using BLEU score (Papineni et al., 2002). Figure 5 depicts the results of 2- to 4-gram of BLEU scores. Since functional words are not considered in (Attamimi et al., 2014), we used our grammar and performed sentence generation proposed in (Attamimi et al., 2014) as the baseline method. One can see from the figure that in all cases the BLEU scores of proposed method out performs the baseline method. It can be said that the sentences generated by the proposed method are of better quality than those generated by the baseline method.

Moreover, we also manually evaluated generated sentences by asking four subjects (i.e., college students who understand Japanese) whether the sentences were: correct both in grammar and meaning (E1), grammatically correct but incorrect in meaning (E2), grammatically incorrect but correct in meaning (E3), or incorrect both in grammar and meaning (E4). The average rates of E1, E2, E3, and E4 were shown in Table 3. We can see that the proposed method out performs the baseline method by providing high rates of E1 and E2; and low rates of E4. Because we want to generate sentences that explain actions, incorrect motion inference would lead to incorrect sentence generation. Examples of E2 are "Eating the plastic wrap in the dining room" and "Opening the dressing in the kitchen." One can see that these sentences are grammatically correct but do not express the scenes correctly because the words that represent the motion are incorrect. Hence, the misclassification that occurred in the motion concept formation was responsible for the incorrect meaning of the generated sentences. Figure 4(c) shows the sentences generated from the given scenes (Figure 4(a)). We can see that meaningful yet natural sentences that explain the observed scenes can be generated using the proposed method.

	Grammar	Meaning	Baseline	Proposed
E1	**correct**	**correct**	$(23.21 \pm 5.28)\%$	(45.39 ± 3.02)**%**
E2	**correct**	incorrect	$(35.07 \pm 9.32)\%$	(49.79 ± 3.77)**%**
E3	incorrect	**correct**	(11.34 ± 5.59)**%**	$(2.79 \pm 2.39)\%$
E4	incorrect	incorrect	(30.38 ± 10.54)**%**	$(2.03 \pm 2.10)\%$

Table 3: Evaluation results of generated sentences.

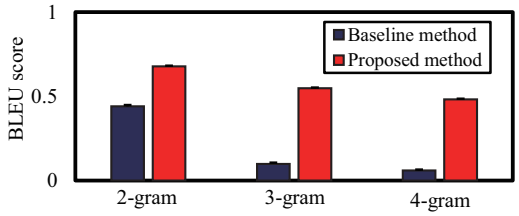


Figure 5: BLEU scores of generated sentences.

4 Conclusion

In this paper, we proposed an unsupervised method to generate natural sentences from observed scenes in a smart environment using mMLDA and BHMM. In the smart environment, multimodal information can be acquired for realistic scenarios. Thanks to mMLDA, various concepts can be formed and an initial determination of functional words can be made by assuming a weak connection of concepts and words calculated by MI. The possibility that grammar can be learned from BHMM by considering the syntactic information has also been shown. We conducted experiments to verify the proposed sentence generation, and promising preliminary results were obtained. In future work, we aim to implement a nonparametric Bayes model that will be able to estimate the number of concepts automatically.

Acknowledgments

This work is partly supported by JSPS KAKENHI 26280096.

References

Muhammad Attamimi, Muhammad Fadlil, Kasumi Abe, Tomoaki Nakamura, Kotaro Funakoshi, and Takayuki Nagai. 2014. *Integration of Various Concepts and Grounding of Word Meanings Using Multi-layered Multimodal LDA for Sentence Generation*. In Proc. of IEEE/RSJ International Conference on Intelligent Robots, pp.2194–2201.

Gabriella Csurka, Christopher R. Dance, Lixin Fan, Jutta Willamowski, Cédric Bray. 2004. *Visual Categorization with Bags of Keypoints*. In Proc. of ECCV International Workshop on Statistical Learning in Computer Vision.

Jeffrey Donahue, Lisa Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. 2014. *Long-term Recurrent Convolutional Networks for Visual Recognition and Description*. Technical Report No. UCB/EECS-2014-180.

Hao Fang, Saurabh Gupta, Forrest Iandola, Rupesh Srivastava, Li Deng, Piotr Dollár, Jianfeng Gao, Xiaodong He, Margaret Mitchell, John Platt, C. Lawrence Zitnick, and Geoffrey Zweig. 2014. *From Captions to Visual Concepts and Back*. In Proc. of IEEE International Conference on Computer Vision and Pattern Recognition.

Dylan F. Glas, Takahiro Miyashita, Hiroshi Ishiguro, and Norihiro Hagita. 2007. *Laser Tracking of Human Body Motion Using Adaptive Shape Modeling*. In Proc. of IEEE/RSJ International Conference on Intelligent Robots, pp.602–608.

Sharon Goldwater and Thomas L. Griffiths. 2007. *A Fully Bayesian Approach to Unsupervised Part-of-Speech Tagging*. In Proc. of the 45th Annual Meeting of the Association for Computational Linguistics, pp.744–751.

Yuji Kawai, Yuji Oshima, Yuki Sasamoto, Yukie Nagai, and Minoru Asada. 2014. *Computational Model for Syntactic Development: Identifying How Children Learn to Generalize Nouns and Verbs for Different Languages* In Proc. of Joint IEEE International Conferences on Development and Learning and Epigenetic Robotics, pp.78–84.

Ryan Kiros, Ruslan Salakhutdinov, and Richard S. Zemel. 2015. *Unifying Visual-Semantic Embeddings with Multimodal Neural Language Models*. In Trans. of the Association for Computational Linguistics.

Taku Kudo, Kaoru Yamamoto, and Yuji Matsumoto. 2004. *Appliying Conditional Random Fields to Japanese Morphological Analysis*. In Proc. of Conference on Empirical Methods in Natural Language Processing, pp.230–237.

Daichi Mochihashi, Takeshi Yamada, and Naonori Ueda. 2009. *Bayesian Unsupervised Word Segmentation with Nested Pitman-Yor Language Modeling* In Proc. of the Association for Computational Linguistics, pp.100–108.

Nobuhisa Motooka, Ichiro Shiio, Yuji Ohta, Koji Tsukada, Keisuke Kambara, and Masato Iguchi. 2010. *Ubiquitous Computing House Project: Design for Everyday Life*. *Journal of Asian Architecture and Building Engineering*, 8:77–82.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. *BLEU: a Method for Automatic Evaluation of Machine Translation*. In Proc. of the Association for Computational Linguistics, pp.311–318.

Michaela Regneri, Marcus Rohrbach, Dominikus Wetzel, Stefan Thater, Bernt Schiele, and Manfred Pinkal. 2013. *Grounding Action Descriptions in Videos*. Trans. of the Association for Computational Linguistics, 1:25–36.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2014. *Show and Tell: A Neural Image Caption Generator*. In arXiv:1411.4555 [cs.CV].

Haonan Yu and Jeffrey M. Siskind. 2013. *Grounded Language Learning from Video Described with Sentences*. In Proc. of the Association for Computational Linguistics, pp.53–63.

Discourse Element Identification in Student Essays based on Global and Local Cohesion

Wei Song[†], **Ruiji Fu**[‡], **Lizhen Liu**[†], **Ting Liu**[§]
[†]Information Engineering, Capital Normal University, Bejing 100048, China
[‡]Iflytek Research Beijing, Beijing 100083, China
[§]Harbin Institute of Technology, Harbin 150001, China
{wsong, lzliu}@cnu.edu.cn, rjfu@iflytek.com, tliu@ir.hit.edu.cn

Abstract

We present a method of using cohesion to improve discourse element identification for sentences in student essays. New features for each sentence are derived by considering its relations to global and local cohesion, which are created by means of cohesive resources and subtopic coverage. In our experiments, we obtain significant improvements on identifying all discourse elements, especially of +5% $F1$ score on *thesis* and *main idea*. The analysis shows that global cohesion can better capture *thesis statements*.

Element	Definition
Introduction (I)	introduces the background and/or grabs readers' attention
Prompt (P)	restates or summarize the prompt
Thesis (T)	states the author's main claim on the issue for which he/she is arguing
Main idea (M)	asserts foundational ideas or aspects that are related to the thesis
Supporting idea (S)	provides evidence to explain or support the thesis and main ideas
Conclusion (C)	concludes the whole essay or one of the main ideas
Other (O)	doesn't fit into the above elements or makes no meaningful contribution

Table 1: Definitions of discourse elements.

1 Introduction

Automatic discourse analysis of student essays can benefit many downstream applications such as essay rating, text organization assessment and writing instruction. In this paper we focus on identifying discourse elements for sentences in persuasive essays written by Chinese high school students. **Discourse elements** represent the contributions that sentences can make to text organization. Typical discourse elements and their functions in persuasive writing are summarized in Table 1.

Previous work mainly exploits the properties of a sentence itself or adjacent sentences for this task. In this work, we explore cohesion to express relations among sentences through the whole text. Cohesion can be defined as a set of resources linking within a text that organize the text together (Halliday and Hasan, 1976). It can be achieved through the use of reference, ellipsis, substitution, conjunction and lexical cohesion. Among them, lexical cohesion has been widely used for modeling local coherence and applied to related applications (Barzilay and Elhadad, 1999; Barzilay and Lapata, 2008; Galley et al., 2003; Hsueh et al., 2006; Filippova and Strube, 2006). Since cohesion is closely related to the structure of text (Morris and Hirst, 1991), it motivates us to explore similar techniques for discourse element identification. In addition, its ease of implementation is also attractive. Other options for representing text structure such as full-text discourse parsers (Marcu, 2000) may be not available or don't have satisfied performance, especially for non-English languages.

However, modeling local coherence alone is not adequate to distinguish discourse elements in persuasive essays. For example, a *main idea* may be followed by a *supporting idea* sentence. The two sentences can be coherent but their discourse elements are different. To deal with this, global cohesion should be exploited. Considering that in persuasive writing, *thesis*, *main ideas* and *conclusion*, which are termed *thesis statements* by Burstein et al. (2001), are expected to relate to each other (Higgins et al., 2004). It is likely that cohesive relations exist among them through the whole text.

We make a focused contribution by investigating global and local cohesive relations. We create sentence chains based on cohesive resources and examine whether the chains represent global cohesion or local cohesion. Our hypothesis is that global cohesion can better capture *thesis state-*

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 2255–2261,
Lisbon, Portugal, 17-21 September 2015.

Corpus	#Essays	Avg.#paras	Avg.#sents	Element distributions							Kappa
				I	P	T	M	S	C	O	
C1	367	7.4	22.7	0.077	0.080	0.087	0.135	0.514	0.095	0.008	0.94
C2	346	7.8	22.5	0.070	0.027	0.069	0.181	0.530	0.114	0.006	0.93
C3	197	9.1	27.7	0.082	–	0.045	0.187	0.571	0.106	0.007	0.91
Avg.	303	7.9	23.7	0.077	0.053	0.067	0.169	0.538	0.105	0.007	0.93

Table 2: Basic statistics of the annotated corpora. Para and sent are short for paragraph and sentence.

ments and help distinguishing them from overwhelming *supporting idea* sentences. Experiments were conducted on essays written by Chinese high school students in the mother tongue. The results confirm our hypothesis. Our method achieves significant improvements of +%5 $F1$ score on *thesis* and *main idea* sentences by adding cohesion features. The features related to global cohesion are most discriminative.

2 Data Annotation

We mainly use the discourse elements defined by Burstein et al. (2003b) except for adding a *prompt* element. The discourse element definitions are listed in Table 1. We asked two labelers from the college of liberal art of a university to conduct data annotation. Provided a detail manual with element definitions, explanations and examples, the labelers assigned a discourse element to each sentence.

We collected three corpora, two of which (C1 and C2) are prompt-directed and one (C3) is prompt-free. All essays were written by Chinese high school students in Chinese. The prompt-directed essays are samples of essays written by senior high school students when they were taking a mock examination. The students were required to write a pervasive essay related to a given prompt. The prompts of corpora C1 and C2 are different. The prompt-free essays in C3 were crawled from an online writing assistance website, where the essays were used as writing examples of persuasive essays written by high school students. The average essay lengths on three corpora are 795, 772 and 864 Chinese characters respectively. The other basic statistics of the annotated corpora are listed in Table 2.

During annotation, the labelers found cases of difficulties about ambiguous elements. For example, content about the prompt and the main thesis can be mentioned in the same sentence. In such cases, *thesis statements* have priority over other elements to be labeled, since identifying *thesis statements* is more important for some potential applications (Burstein et al., 2001).

From each corpus, 100 essays were labeled by both annotators for computing agreements, and the others were labeled independently. The label agreements measured with Kappa (Cohen et al., 1960) are high as shown in Table 2. The disagreements were resolved by discussion. The distributions of discourse elements are also shown in Table 2. We can see that they are imbalanced. The *supporting idea* sentences account for more than 53%, while the *thesis statements* account for only 34% in total. As a result, the distinction between minority *thesis statements* from overwhelming *supporting idea* sentences is a major challenge.

3 Discourse Element Identification

Identifying discourse elements in student essays can be seen as a functional segmentation of discourse (Webber et al., 2011). In this work, we focus on utilizing supervised feature-based machine learning models for this task.

3.1 Learning Models

Discourse element identification can be casted as a classification problem that sentences are classified independently using a classifier, e.g. naive Bayes (Burstein et al., 2001), decision tree (Burstein et al., 2003b) and Support Vector Machines (SVMs) (Stab and Gurevych, 2014). It can also be solved in a sequence labeling framework, which models the whole sentence sequence and captures the correlations among predictions. For example, Conditional Random Fields (CRFs) have be studied for similar task on argumentative zoning of scientific documents (Guo et al., 2011).

We will evaluate different types of features using two representative models respectively: the SVM model and the linear-chain CRF model.

3.2 Basic Features

Before feature extraction, sentence splitting, word segmentation,POS and NE tagging are done using a Chinese language processing toolkit (Che et al., 2010). Most basic features are adapted from previous work (Burstein et al., 2003a; Stab and Gurevych, 2014; Persing et al., 2010). For each sentence, the following feature sets are extracted.

Position features The relative position of its paragraph (first, last or body) in the essay and its relative position (first, last or body) in the paragraph are modeled as a set of binary features. The index of the sentence is also used as a feature.

Indicator features Cue words/phrases like "我认为(in my opinion)" and "总之(in conclusion)" are used as indicators. Partial indicators are adapted from the ones used by Persing et al. (2010). More Chinese specific indicators are then augmented manually. We use a binary feature denoting a reference to the first person ("我(I)","我们(We)") in the sentence. We also use a binary feature to indicate whether the sentence contains a modal verb like "应该(should)" and "希望(hope)".

Lexical features Binary features are modeled for all connectives and adverbs, which are identified based on POS tags.

Structural features The number of words, the number of clauses in the sentence and the number of sentences in the same paragraph are used as features. We also define binary features based on punctuation which indicate whether the sentence ends with a full stop, question mark, exclamation mark or no sentence-final punctuation.

Topic and prompt features For each sentence, the cosine similarities to the essay title and to the prompt are used as features.

4 Identification based on Cohesion

4.1 Cohesive Chains

We mainly exploit reference and lexical cohesion.

Creating identity chains Reference refers to resources for referring to a participant whose identity is recoverable (Schiffrin et al., 2008). We focus on person identities, because person names might be mentioned when describing facts. Firstly, we extract all nouns/entities with a POS/NE tag *person* as identities. Secondly, we conduct a simple third-person pronoun resolution by selecting the nearest proper antecedent identity within the same paragraph. Finally, an identity and all its anaphora together form an identity chain.

Creating lexical chains Lexical cohesion is referred to relations between text using lexical repetition, synonymy or near synonymy. We don't distinguish between systematic semantic relations and non-systematic semantic relations (Berzlánovich et al., 2008) nor use a thesaurus (Hirst and St-Onge, 1998). Instead, we compute the relatedness of two words based on their distributed representations, which are learned using the Word2Vec toolkit (Mikolov et al., 2013). The data for learning word representations consists of student essays and textbooks crawled from the Web. The vocabulary size is about $490k$.

We extract nouns, adjectives and verbs (excluding auxiliary verbs) instead of using nouns only (Hirst and St-Onge, 1998) for constructing lexical chains . We firstly cluster words into clusters in a graph based manner. Each word corresponds to a node in a graph. If the relatedness of two words is larger than a threshold T, they are considered as related and linked by an edge. After constructing all edges in this way, every connected subgraph forms a word cluster. Through the essay, all occurrences of the words from the same cluster form a lexical chain.

We discard identity and lexical chains that exist within single sentences, since they can't capture cohesive relations among sentences.

4.2 Global and Local Sentence Chains

We organize sentences based on cohesive chains. Sentences that contain members from the same identity chain or lexical chain form a sentence chain. The sentence chains represent cohesive relations among sentences.

In persuasive writing, discourse elements are commonly linked globally. For example, *main ideas* are usually related to each other because they are about different aspects of the main thesis, and *thesis* and *conclusion* should echo each other as well. Therefore, we attempt to explicitly categorize sentence chains into local chains and global chains based on subtopic coverage. A local chain represents sentences that share cohesive relations and gather locally within single subtopics. In contrast, a global chain represents sentences with cohesive relations distribute across multiple subtopics. Heuristically, we expect that *thesis statements* can be better captured by global chains, while sentences that state facts or provide evidences are associated to local chains.

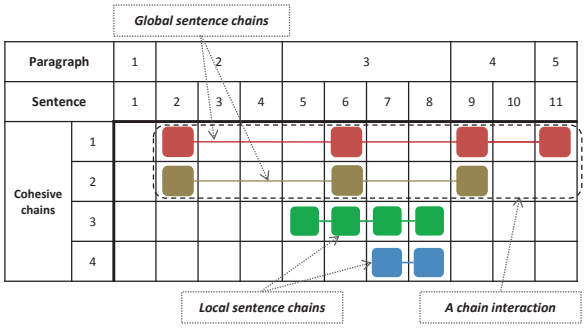


Figure 1: An illustration of global/local sentence chains. Each solid node in the grid indicates that a sentence contains a word from a cohesive chain.

Although subtopics can be identified by existing text segmentation algorithms (Hearst, 1997; Filippova and Strube, 2006), we observe that in student essays a subtopic boundary usually coincides with a paragraph boundary and almost all subtopic segments are within one paragraph and only a few of subtopics are within two or more paragraphs. Therefore, we simply assume that each paragraph corresponds to a subtopic. Based on the assumption, chain classification is approximated based on chain span over paragraphs. A sentence chain is classified as a global chain, if its members appear in at least N paragraphs, otherwise it is classified as a local chain. We set $N = 3$, which means a global chain would cover at least two subtopics considering most subtopics finish within two paragraphs. One sentence can be involved in multiple sentence chains. An illustration of global and local sentence chains is shown in Figure 1.

4.3 Cohesion Features from Sentence Chains

We develop cohesion features for a sentence from the sentence chains that involve it. Such features are beyond the intrinsic properties of the sentence itself but describe relations to other sentences.

Chain-type features We consider four combined types of sentence chains: *global-identity, local-identity, global-lexical* and *local-lexical* chains. The number of each type of chains that involve the sentence is used as a feature.

Global-title feature If the sentence is in a global sentence chain and the corresponding cohesive chain contains a word in the title, a binary feature *global-title* is set as *true*, otherwise set as *false*. Containing globally distributed title words is thought of as an indicator of *thesis statements*.

Interaction features Hasan (1984, 1985) defined that an interaction between two chains takes place when multiple members of a chain relate in the same way to more than one members of another chain, which can be used to distinguish central tokens from peripheral tokens. Hoey (1991) explored similar interactions to assess the centrality of sentences. This indicates that chain interactions might be signals of important content.

Similarly, we say two sentence chains interact with each other, if they have more than one sentence in common. An example is shown in Figure 1. Moreover, if two chains are both global chain, we term it a global interaction, otherwise a local interaction. The shared sentences by two chains are named as *global or local interaction sentences* accordingly. Two binary features are derived: the sentence is or not a *global-interaction* sentence and it is or not a *local-interaction* sentence.

Strength features We attempt to measure the overall strength of the sentence chains that involve the sentence. The features include the number of chains, the maximum and average number of covered sentences and paragraphs over chains, among which the ones related to paragraphs can be seen as measuring the global cohesion strength.

5 Evaluation

5.1 Settings

We evaluated the effectiveness of **Cohesion** features by comparing with the baseline that uses the **Basic** features introduced in Section 3.2.

We adopted $precision(P)$, $recall(R)$ and $F1$-measure($F1$) as evaluation metrics. The threshold T used to determine whether two words are related was set to 0.8 empirically. Because sentences with the discourse element *Other* are few, we didn't evaluate the performance on it.

We conducted experiments on three corpora respectively using 5-fold cross-validation. We compared various SVM classifiers with different kernels implemented in the LibSVM toolkit (Chang and Lin, 2011) and the linear-chain CRF model (Lafferty et al., 2001). When using CRF, the prediction of previous sentence is considered for current sentence. In our experiments, CRF achieves significant superior performance than SVM both when using basic features alone and after adding cohesion features. This indicates that incorporating correlations among sequential predictions are important for this task. Next, we only report the experimental results of using the CRF Model.

Element	Features	C1			C2			C3			
		P	R	$F1$	P	R	$F1$	P	R	$F1$	$avg. \triangle (F1)$
Introduction	Basic	84.5	89.6	86.8	82.2	80.7	81.5	80.6	90.1	85.0	+3.7
	+ Cohesion	87.2	90.8	88.8	85.6	84.8	85.2	87.3	94.4	90.6	
Prompt	Basic	89.7	86.9	88.2	77.2	69.0	72.5	–	–	–	+1.9
	+ Cohesion	91.1	89.2	90.1	82.0	69.1	74.4	–	–	–	
Thesis	Basic	76.5	69.0	72.4	69.9	61.1	64.9	73.3	57.5	64.0	+5.1
	+ Cohesion	78.3	73.1	75.5	75.4	63.8	68.6	77.3	68.9	72.7	
Main idea	Basic	71.4	59.1	64.5	69.0	60.9	64.6	69.4	54.0	60.7	+5.4
	+ Cohesion	75.7	65.3	70.0	73.6	61.3	66.8	75.7	64.3	69.4	
Supporting idea	Basic	86.1	91.4	88.6	83.8	89.6	86.6	83.8	90.5	87.0	+1.8
	+ Cohesion	88.0	92.3	90.1	84.2	91.6	87.7	87.7	92.2	89.9	
Conclusion	Basic	87.2	89.9	88.4	85.6	88.5	87.0	88.1	91.0	89.5	+2.2
	+ Cohesion	89.1	91.9	90.4	86.0	90.7	88.2	92.1	94.0	93.1	

Table 3: Experimental results on six discourse elements over three corpora using the CRF model.

5.2 Experimental Results

The experimental results on three corpora are shown in Table 3. We tested statistical significance for $F1$ scores and found that all improvements were significant with $p < 0.01$ based on the pairwise t-test. We can see that adding cohesion features obtain improvements on all discourse elements over three corpora. Especially, the cohesion features contribute to large improvements of +5.1% and +5.4% average $F1$ score on identifying *thesis* and *main idea* sentences. By analyzing the confusion matrix, we found that the improvements mainly come from more accurately distinguishing *thesis* and *main idea* sentences from *introduction* and *supporting idea* sentences.

We are interested in that which between the local and global cohesion contributes more to distinguish *thesis statements*. To this end, we used Area under the ROC Curve (AUC)(Swets, 1988) to measure the discriminative power of individual features. Larger AUC of a feature indicates better discriminative performance. The experiment was done on the dataset mixing all sentences from three corpora. We divided sentences into *thesis statements* and *non-thesis statements* according to their true element labels. As the results in Table 4 show, global cohesion related features are of higher rank with regard to the discriminative power. Local cohesion relates more to *non-thesis statements*, though it is not so discriminative as global cohesion. The results indicate that separating global cohesion from local cohesion help the distinction between *thesis statements* and others. Features related to identity chains alone don't show much discriminative ability. But they increase the macro $F1$ score by 0.9% in combination with other features.

Cohesion Feature	AUC
Global-lexical	0.712
Avg.#paras	0.670
Global-title	0.664
Max.#para	0.659
Global-interaction	0.654
Max.#sents	0.636
Avg.#sents	0.613
#Chains	0.601
Local-title	0.522
Global-identity	0.510
Local-identity	0.481
Local-interaction	0.476
Local-lexical	0.431

Table 4: Discriminative powers of individual features by Areas under ROC curve (AUC).

6 Conclusion

We have investigated the impact of cohesion for identifying discourse elements in student essays. Our method creates sentence chains by means of cohesive resources and separates global chains from local ones based on the subtopic coverage. New features for each sentence are derived from the properties of the sentence chains involving it. Experimental results show the effectiveness of cohesion features and the discriminative ability of global cohesion for identifying *thesis statements*.

Acknowledgments

This research is supported by the National Natural Science Foundation of China (No.61402304), the Beijing Municipal Natural Science Foundation (No.4154065) and the Humanity & Social Science General Project of Ministry of Education (No.14YJAZH046).

References

Regina Barzilay and Michael Elhadad. Using lexical chains for text summarization. *Advances in automatic text summarization*, pages 111–121, 1999.

Regina Barzilay and Mirella Lapata. Modeling local coherence: An entity-based approach. *Computational Linguistics*, 34(1):1–34, 2008.

Ildikó Berzlánovich, Markus Egg, and Gisela Redeker. Coherence structure and lexical cohesion in expository and persuasive texts. In *Proceedings of the Workshop Constraints in Discourse III*, pages 19–26, 2008.

Jill Burstein, Daniel Marcu, Slava Andreyev, and Martin Chodorow. Towards automatic classification of discourse elements in essays. In *Proceedings of the 39th annual Meeting on Association for Computational Linguistics*, pages 98–105. Association for Computational Linguistics, 2001.

Jill Burstein, Martin Chodorow, and Claudia Leacock. Criterionsm online essay evaluation: An application for automated evaluation of student essays. In *IAAI*, pages 3–10, 2003a.

Jill Burstein, Daniel Marcu, and Kevin Knight. Finding the write stuff: Automatic identification of discourse structure in student essays. *Intelligent Systems, IEEE*, 18(1):32–39, 2003b.

Chih-Chung Chang and Chih-Jen Lin. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27, 2011.

Wanxiang Che, Zhenghua Li, and Ting Liu. Ltp: A chinese language technology platform. In *Proceedings of the 23rd International Conference on Computational Linguistics: Demonstrations*, pages 13–16. Association for Computational Linguistics, 2010.

Jacob Cohen et al. A coefficient of agreement for nominal scales. *Educational and psychological measurement*, 20(1):37–46, 1960.

Katja Filippova and Michael Strube. Using linguistically motivated features for paragraph boundary identification. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 267–274. Association for Computational Linguistics, 2006.

Michel Galley, Kathleen McKeown, Eric Fosler-Lussier, and Hongyan Jing. Discourse segmentation of multi-party conversation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 562–569. Association for Computational Linguistics, 2003.

Yufan Guo, Anna Korhonen, and Thierry Poibeau. A weakly-supervised approach to argumentative zoning of scientific documents. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 273–283. Association for Computational Linguistics, 2011.

MAK Halliday and Ruqaiya Hasan. Cohesion in english (english language). *London, 1976; Martin JR*, 1976.

Ruqaiya Hasan. Coherence and cohesive harmony. *Understanding Reading Comprehesion: Cognition, Lanugage and The Structure of Prose*, 1984.

Ruqaiya Hasan. The texture of a text. *Language, Context and Text*, 1985.

Marti A. Hearst. Texttiling: Segmenting text into multi-paragraph subtopic passages. *Computational Linguistics*, pages 33–64, 1997.

Derrick Higgins, Jill Burstein, Daniel Marcu, and Claudia Gentile. Evaluating multiple aspects of coherence in student essays. In *HLT-NAACL*, pages 185–192, 2004.

Graeme Hirst and David St-Onge. Lexical chains as representations of context for the detection and correction of malapropisms. *WordNet: An electronic lexical database*, 305:305–332, 1998.

Michael Hoey. Patterns of lexis in text. 1991.

Pei-Yun Hsueh, Johanna D Moore, and Steve Renals. Automatic segmentation of multiparty dialogue. In *EACL*, 2006.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, pages 282–289. Morgan Kaufmann, 2001.

Daniel Marcu. *The theory and practice of discourse parsing and summarization*. MIT press, 2000.

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

Jane Morris and Graeme Hirst. Lexical cohesion computed by thesaural relations as an indicator of the structure of text. *Computational linguistics*, 17(1):21–48, 1991.

Isaac Persing, Alan Davis, and Vincent Ng. Modeling organization in student essays. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 229–239. Association for Computational Linguistics, 2010.

Deborah Schiffrin, Deborah Tannen, and Heidi E Hamilton. *The handbook of discourse analysis*. John Wiley & Sons, 2008.

Christian Stab and Iryna Gurevych. Identifying argumentative discourse structures in persuasive essays. In *EMNLP 2014*, pages 46–56, 2014.

John A Swets. Measuring the accuracy of diagnostic systems. *Science*, 240(4857):1285–1293, 1988.

Bonnie Webber, Markus Egg, and Valia Kordoni. Discourse structure and language technology. *Natural Language Engineering*, 18(4): 437–490, 2011.

Adapting Coreference Resolution for Narrative Processing

Quynh Ngoc Thi Do[1], **Steven Bethard**[2], **Marie-Francine Moens**[1]
[1]Katholieke Universiteit Leuven, Belgium
[2]University of Alabama at Birmingham, United States
quynhngocthi.do@cs.kuleuven.be
bethard@cis.uab.edu
sien.moens@cs.kuleuven.be

Abstract

Domain adaptation is a challenge for supervised NLP systems because of expensive and time-consuming manual annotated resources. We present a novel method to adapt a supervised coreference resolution system trained on newswire to short narrative stories without retraining the system. The idea is to perform inference via an Integer Linear Programming (ILP) formulation with the features of narratives adopted as soft constraints. When testing on the UMIREC[1] and N2[2] corpora with the-state-of-the-art Berkeley coreference resolution system trained on OntoNotes[3], our inference substantially outperforms the original inference on the CoNLL 2011 metric.

1 Introduction

Coreference resolution is the task of partitioning the set of mentions of discourse referents in a text into classes (or 'chains') corresponding to those referents (Stede, 2011). To solve the problem, contextual and grammatical clues, as well as semantic information and world knowledge are necessary for either learning-based (Bengtson and Roth, 2008; Stoyanov et al., 2010; Haghighi and Klein, 2010) or rule-based (Haghighi and Klein, 2009; Lee et al., 2011) coreference systems. These systems draw on diverse information sources and complex heuristics to resolve pronouns, model discourse, determine anaphoricity, and identify semantically compatible mentions. However, this leads to systems with many hetorogenous parts that can be difficult to interpret or modify.

Durrett and Klein (2013) propose a learning-based, mention-synchronous coreference system to tackle the various aspects of coreference by using the simplest possible set of features. Its advantage is that the system can both implicitly model important linguistic effects and capture other patterns in the data that are not easily teased out by hand. With a simple set of features including head/first/last words, preceding/following words, length, exact string match, head match, sentence/mention distance, gender, number etc. and an efficient training using conditional log-likelihood augmented with a parameterized loss function optimization they report state-of-the-art results on CoNLL 2011 data.

But while CoNLL 2011 training data (OntoNotes) includes a few different source domains (newswire, weblogs, etc.), we witness significant drops in performance when systems trained on CoNLL 2011 are applied to new target domains such as narratives. Some linguistic effects and patterns that are very important for the target domain were never seen in the source domain on which the model was trained. In such cases, when adapting a coreference system to a new domain, it is necessary to incorporate these more complex linguistic features and patterns into the model.

We propose a novel method to adopt the target domain's features to a supervised coreference system without retraining the model. We present a case of transferring the system of (Durrett and Klein, 2013), which is trained on OntoNotes, to short narrative stories. The idea is to perform inference via a linear programming formulation with the features of narratives adopted as soft constraints. Since the new features are incorporated only into the linear program, there is no need to retrain the original model. Our formulation models three phenomena that are important for short narrative stories: local discourse coherence, which we model via centering theory constraints, speaker-listener relations, which we model via direct speech act constraints, and character-naming, which we model via definite noun phrase and exact match constraints.

[1]http://dspace.mit.edu/handle/1721.1/57507
[2]http://dspace.mit.edu/handle/1721.1/85893
[3]https://catalog.ldc.upenn.edu/LDC2011T03

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 2262–2267,
Lisbon, Portugal, 17-21 September 2015.

We also suggest a method to compute back pointers (as defined in Durrett and Klein (2013)) globally.

2 Berkeley coreference system

Given N mentions $m_1, ..., m_N$ from a document x, each m_i has an associated random variable a_i taking values in the set of $\{1, ..., i-1, NEW\}$. This variable specifies m_i's selected antecedent or indicates that it begins a new coreference chain. We call a_i the *back pointer* of m_i. A setting of all the back pointers, denoted by $a = (a_1, ..., a_n)$, implies an unique set of coreference chains that serve as the system output.

A log-linear model of the conditional distribution $P(a|x) \propto \exp \sum_{i=1}^{n} \mathbf{f}(i, a_i, x)$ is used, where $\mathbf{f}(i, a_i, x)$ is a feature function that examines the coreference decision a_i for m_i with document context x. If $a_i = NEW$, the features indicate the suitability of the given mention to be anaphoric or not; when $a_i = j$ for some j, the features express aspects of the pairwise linkage, and examine relevant attributes of the anaphor i or the antecedent j. During training, the model is optimized with a parameterized loss function. The inference is simple and efficient: because $\log P(a|x)$ decomposes linearly over mentions, $a_i = \arg\max_{a_i} P(a_i|x)$ (Durrett and Klein, 2013).

3 Computing back pointers globally

A drawback of computing each a_i locally is that the system does not take into account constraints from mentions outside of the (mention, antecedent) pairs. For example, given three mentions m_1, m_2, m_3, if the system predicts that $a_2 = 1$ and $a_3 = 2$ (i.e., that m_2's antecedent is m_1 and m_3's antecedent is m_2), then m_3 will be automatically inferred as coreferent with m_1. But if there is a clear clue that m_1 and m_3 are not coreferent, leveraging this clue could help avoid the error of linking m_3 to m_2.

In this work, we perform inference via an ILP formulation which allows new linguistic features and patterns over mentions – not only (mention, antecedent) pairs – that were not part of training the original model to be adopted as constraints of the ILP problem.

Let $\mathbf{U}$ be the set of binary indicator variables corresponding to the values assigned to the back pointers. Specifically, $u_{ij} = 1$ iff $a_i = j$ and $u_{ii} = 1$ iff $a_i = NEW$.

$\mathbf{C}$ is the set of K binary constraint indicator variables indicating if linguistic constraints are violated. Specifically, $c_{k,i,j} = 1$ iff the linguistic constraint C_k is violated for the back pointer u_{ij}. Each C_k is associated with a penalty score ρ_k.

We aim to maximize the objective function:

$$\sum_{i=1}^{N} \sum_{j=1}^{i} u_{ij} P(a_i = j|x) - \sum_{k=1}^{K} \rho_k c_{k,i,j} \quad (1)$$

Subject to:

$$\forall i : \sum_{j=1}^{i} u_{ij} = 1$$

To incorporate coreference constraints, we introduce $\mathbf{V}$, a set of binary variables indicating if two mentions are in the same coreference chain. For each pair of $j < i$, a variable v_{ij} is added to the ILP model, where $v_{ij} = 1$ iff m_i and m_j are in the same chain. The definition of v_{ij} in terms of u_{ij} is encoded as the following ILP constraints:

$$\begin{aligned}
\forall j < i &: u_{ii} + v_{ij} \leq 1 \\
\forall j < i &: u_{ij} - v_{ij} \leq 0 \\
\forall k < j < i &: u_{ij} + v_{jk} - v_{ik} \leq 1 \\
\forall k < j < i &: u_{ij} - v_{jk} + v_{ik} \leq 1 \\
\forall j < k < i &: u_{ij} + v_{kj} - v_{ik} \leq 1 \\
\forall j < k < i &: u_{ij} - v_{kj} + v_{ik} \leq 1
\end{aligned}$$

For long texts, to reduce the complexity of the ILP problem, we set a threshold, $windows_v$, so that v_{ij} is only available if $i - windows_v \leq j$.

The framework of $\mathbf{V}$ variables allows coreference constraints to be adopted easily by any coreference resolution system that provides scores for each possible back pointer value. For example, consider the Stanford exact string match sieve, which "requires an exact string match between a mention and its antecedent" (Lee et al., 2011). If we want to encourage such matches, for each pair $j < i$ where the two nominal mentions m_i and m_j have an exact string match, we would introduce a constraint indicator variable $c_{exact,i,j}$ and add the constraint $v_{ij} + c_{exact,i,j} = 1$ to the ILP model. The result would be that when the exact match constraint is violated and some $v_{ij} = 0$, ILP would force the corresponding $c_{exact,i,j} = 1$ and the objective function would be reduced by ρ_{exact}.

ILP has been used previously to enforce global consistency in coreference resolution (Finkel and Manning, 2008; Denis and Baldridge, 2007; Barzilay and Lapata, 2006). These models were designed for an all-pairs classification approach to

coreference resolution, and are not directly applicable to the back pointer approach of (Durrett and Klein, 2013). But the back pointer approach allows features to be expressed more naturally using local context, rather than requiring, for example, judgments of whether two pronouns separated by many paragraphs are coreferent. Moreover, our ILP formulation is the only one to consider the problem of adapting to another domain and incorporating new features without retraining the original model.

4 Centering theory constraints

Pronouns, in particular, have a huge effect on information flow across sentences. Since they are almost void of meaning (only signal gender and number of the antecedent), the discourse referent to be picked up must be particularly salient, so that it can be readily identified by the reader (Stede, 2011). The discourse center hypothesis (Hudson-D'Zmura, 1988) states that at any point in discourse understanding, there is one single entity that is the most salient discourse referent at that point. This referent is called the center. Centering theory is a key element of the discourse center hypothesis used in anaphora resolution (Grosz et al., 1995). Beaver (2004) reformulates the centering theory in terms of Optimality Theory (Prince and Smolensky, 2004). Six ranked constraints – Agree, Disjoint, ProTop, FamDef, Cohere and Align – are used to make anaphora decisions. We adopt four of these constraints in our ILP model as follows:

Disjoint "Co-arguments of a predicate[4] are disjoint." For each $j < i$ such that m_i and m_j are subject and object arguments of a non-reflexive predicate, we introduce a constraint indicator variable $c_{disjoint,i,j}$, and add the ILP constraint $v_{ij} - c_{disjoint,i,j} = 0$.

ProTop "The *topic* of a sentence which is the entity referred to in both the current and the previous sentence, is pronominalized." If a sentence contains pronouns then at least one of its pronouns is coreferent with a mention in the previous sentence. For each sentence t containing pronouns, we introduce a constraint indicator variable $c_{protop,t,t-1}$, and add the ILP constraints:

$$\forall i \in \mathbf{P}_t, \forall j \in \mathbf{M}_{t-1} : v_{ij} + c_{protop,t,t-1} \leq 1$$

$$c_{protop,t,t-1} + \sum_{i \in \mathbf{P}_t} \sum_{j \in \mathbf{M}_{t-1}} v_{ij} \geq 1$$

[4] A word that evokes a semantic frame (event) in a sentence.

$\mathbf{P}_t$ is the set of all pronouns in sentence t. $\mathbf{M}_{t-1}$ is the set of all mentions in sentence $t-1$ [5].

FamDef "No new information about the referent is provided by the definite." We consider only pronouns here, though the original FamDef also includes definite descriptions and proper names (Beaver, 2004). For each pronoun m_i, we introduce a constraint indicator variable $c_{famdef,i,i}$ and add the ILP constraint $u_{ii} - c_{famdef,i,i} = 0$.

Align "The topic is in subject position." More specifically, the topic of a sentence is pronominalized and prefers the subject position over other positions. For each sentence containing only one pronoun m_i, if the previous sentence has only one verbal semantic frame and m_j is its subject, we introduce a constraint indicator variable $c_{align,i,j}$, and add the ILP constraint $v_{ij} + c_{align,i,j} = 1$.

Note: The ProTop, FamDef and Align constraints are not applied to sentences containing quotations.

5 Direct speech constraints

Direct speech acts (with quotation marks) are detected and attached to the closest verbal communication semantic frames. For each direct speech act q_t, we call the mentions m_{st}, m_{ot} the speaker and listener of q_t if they play the subject and object roles respectively in the semantic frame of q_t. We detect the set of subject pronouns[6] inside the quote marks of q_t and name it $\mathbf{S_t}$. The set of all mentions that refer to the speaker of q_t is $\text{SPEAKER}_t = \{m_{st}\} \cup \mathbf{S_t}$. For each $(m_i, m_j) \in \text{SPEAKER}_t \times \text{SPEAKER}_t$ with $i > j$, we introduce a constraint indicator variable $c_{subject,i,j}$, and add the ILP constraint $v_{ij} + c_{subject,i,j} = 1$.

Similarly, $\mathbf{O_t}$ is the set of object pronouns[7] inside the quote marks of q_t. The set of all mentions that refer to the listener of q_t is $\text{LISTENER}_t = \{m_{ot}\} \cup \mathbf{O_t}$. For each pair of mentions $(m_i, m_j) \in \text{LISTENER}_t \times \text{LISTENER}_t$ with $i > j$, we introduce a constraint indicator variable $c_{object,i,j}$, and add the constraint $v_{ij} + c_{object,i,j} = 1$.

If a conversation is detected (a sequence of "question" and "answer" semantic frames), the subject of the "question" is coreferent with the object

[5] We can relax the constraint by replacing $\mathbf{M}_{t-1}$ with $\mathbf{M}_{t-1} \cup \mathbf{M}_{t-2} \cup \mathbf{M}_{t-3}$

[6] ("I", "me", "my", "mine", "myself") if m_{st} is singular or ("we", "us", "our", "ourself") if m_{st} is plural

[7] ("you", "your", "yours", "yourself")

Method	UMIREC (Tales)				N2 (Hadith)			
	MUC	BCUB	CEAFE	AVG	MUC	BCUB	CEAFE	AVG
ILPI with gold mentions	**84.16**	**65.65**	**50.47**	**66.76**	**80.47**	**65.53**	**54.06**	**66.69**
BER with gold mentions	80.58	60.96	42.48	61.34	76.28	62.66	45.48	61.47
ILPI with predicted mentions	**73.32**	**59.18**	**37.54**	**56.68**	**66.13**	**62.55**	**40.51**	**56.40**
BER with predicted mentions	72.71	58.12	35.76	55.53	64.87	59.60	37.96	54.14

Table 1: ILPI and BER inference results on UMIREC (Tales) and N2 (Hadith) data.

of the "answer" and vice versa. For each pair of direct speech acts (q_t, q_{t+1}) that is a ("question", "answer") pair, for each pair of mentions $(m_i, m_j) \in \{\text{LISTENER}_{t+1} \times \text{SPEAKER}_t\} \cup \{\text{SPEAKER}_{t+1} \times \text{LISTENER}_t\}$, we introduce a constraint indicator variable $c_{conversation,i,j}$ and add the ILP constraint $v_{ij} + c_{conversation,i,j} = 1$.

6 Definite noun phrase and exact match constraints

In short narrative stories, characters are frequently named via proper names, pronouns or definite noun phrases (Toolan, 2009). Character names are repeated regularly over the whole stories. A character is often first presented as an indefinite noun phrase (such as "a woman"), then later as a definite noun phrase (such as "the woman"). In this work we introduce the definite noun phrase constraint: For each pair $j < i$, if m_j is the indefinite form and m_i is the definite form of the same noun phrase, to enforce that m_i and m_j are coreferent, we introduce a constraint indicator variable $c_{name,i,j}$, and add the ILP constraint $v_{ij} + c_{name,i,j} = 1$. To boost the identification of characters in the stories, the definite noun phrase constraint is used together with the exact match constraint (See Section 3) applied to noun phrases and proper nouns.

7 Experiment

We test our model on 30 English folktales from the UCM/MIT Indications, Referring Expressions, and Coreference (UMIREC) Corpus v1.1 (Finlayson and Hervs, 2010), and 64 text stories from the Hadith section of the Narrative Networks (N2) Corpus (Finlayson et al., 2014). The texts are preprocessed using the Stanford sentence splitter (Manning et al., 2014)[8] and the Berkeley coreference system's preprocessor. The Berkeley coreference system is trained on OntoNotes (newswire, broadcast news/conversation, and web texts). We use Gurobi[9] to solve our ILP problem, and the Lund semantic role labeler (Björkelund et al., 2009) to detect semantic frames. Note that in our implementation, "subject" and "object" used in Section 4 and Section 5 refer to "subject role" and "object role" of the semantic frames respectively. We use a separate section of the N2 corpus, the Inspire story texts, as the held-out validation set used for parameter tuning, resulting in $window_v = 40$, $\rho_{subject} = \rho_{object} = \rho_{conversation} = \rho_{definite} = \rho_{exact} = \rho_{disjoint} = 1$, $\rho_{protop} = 0.2$, $\rho_{famdef} = 0.2$, $\rho_{align} = 0.1$.

We compare our ILP inference (ILPI) to the standard Berkeley coreference system (BER) with both gold and predicted mentions. Table 1 shows that our inference improves the MUC, BCUB and CEAFE scores on both datasets, especially when using gold mentions[10]. The average ILP running times are 42.37s per UMIREC document and 22.7s per N2 document on a Core I7 2.3 GHz quad-core computer. Table 2 shows the effects of each constraint type when used alone. Surprisingly, the simplest constraint type (definite & exact match constraints) gives us the best improvement especially in terms of CEAFE score. This may be because definite & exact match constraint links mentions in the whole document, while the centering theory and direct speech act constraints are more local. And since short narrative stories often have a small set of characters (usually represented by definite noun phrases or proper nouns), when these characters are linked correctly, the coreference resolution result is improved considerably.

8 Discussion

Our method provides a promising solution when retraining a system is impossible or difficult. However, it may raise a question of the computing cost

[8] If two direct speech acts enclosed in quotation marks are adjacent and one is placed at the end of a sentence, we separate them into two different sentences.

[9] http://www.gurobi.com/

[10] Using gold mentions, our method also improves the score on the CoNLL 2011 test set by +1.11% (AVG: 72.46).

Constraint	MUC	BCUB	CEAFE	AVG
Centering theory	81.15	61.80	43.01	61.99
Direct speech	81.26	62.74	42.93	62.31
Definite & Exact	83.09	62.85	49.60	65.18

Table 2: Effects of different constraints on ILP inference on UMIREC (Tales) with gold mentions.

for tuning penalty scores especially with the large number of constraints. In such these cases, dividing the constraints into different groups where all constraints in the same group have the same penalty score may help to limit the number of scores that need to be tuned. In our case study, the system is not very sensitive to the values of the penalty parameters. If we set all the penalty scores to 1, the final AVG results on UMIREC and N2 corpus are 66.05 and 66.68 respectively[11]. Those scores are a bit less than the scores obtained after tuning parameters but still higher than the results obtained without ILP. Regardless, it's true that the proposed ILP approach is not necessarily less costly in some settings, but it can be applied to any coreference system that provides back pointers, not just the Berkeley one.

Instead of adopting features of the target domain as soft constraints as in our method, one may consider to use them as linguistic features and retrain the model. A simple domain adaptation approach by augmenting the feature space (Daumé et al., 2010) based on a limited set of annotated data in the target domain might be an alternative solution. But note that our approach does not use any annotated data of the target domain. Also, an unsupervised system as (Lee et al., 2011) might encode the target domain features (exact match noun phrases, direct speech act) as sieves (**hard**), but with the **soft** constraints, our system is more flexible when making global decisions.

Our approach can be applied to another target domain, such as bio-medical domain where we have entities and a list of acronyms in texts. Constraining the entities with their acronyms might help to improve the coreference resolution for bio-medical texts.

9 Conclusion

We have proposed a novel approach to adapt a supervised coreference resolution system trained on newswire domain to short narrative stories without retraining the system by modeling the inference as an ILP problem with the features of narratives adopted as soft constraints. Three phenomena that are important for short narrative stories: local discourse coherence, speaker-listener relations, and character-naming are modeled via centering theory, direct speech act and definite noun phrase & exact match constraints. We obtain promising results when transferring the Berkeley coreference resolution trained on OntoNotes to UMIREC (Tales) and N2 (Hadith). We find that the simplest constraints, definite noun phrase & exact match constraints, are the most effective in our case study assuming the gold mentions. We also suggest an approach to compute back pointers in coreference resolution globally.

[11] with gold mentions

Acknowledgment

This work acknowledges the financial support of the Future and Emerging Technologies (FET) programme within the Seventh Framework Programme for Research of the European Commission, under FET-Open grant number 295703 (FET project "Machine Understanding for interactive StorytElling" (MUSE) `http://www.muse-project.eu/`).

References

Regina Barzilay and Mirella Lapata. 2006. Aggregation via set partitioning for natural language generation. In *Proceedings of the Main Conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, HLT-NAACL '06, pages 359–366, Stroudsburg, PA, USA. Association for Computational Linguistics.

David I. Beaver. 2004. The optimization of discourse anaphora. *Linguistics and Philosophy*, 27(1):3–56.

Eric Bengtson and Dan Roth. 2008. Understanding the value of features for coreference resolution. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 294–303, Stroudsburg, PA, USA. Association for Computational Linguistics.

Anders Björkelund, Love Hafdell, and Pierre Nugues. 2009. Multilingual semantic role labeling. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning: Shared Task*, CoNLL '09, pages 43–48, Stroudsburg, PA, USA. ACL.

Hal Daumé, III, Abhishek Kumar, and Avishek Saha. 2010. Frustratingly easy semi-supervised domain

adaptation. In *Proceedings of the 2010 Workshop on Domain Adaptation for Natural Language Processing*, DANLP 2010, pages 53–59, Stroudsburg, PA, USA. Association for Computational Linguistics.

Pascal Denis and Jason Baldridge. 2007. Joint determination of anaphoricity and coreference resolution using integer programming. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 236–243, Rochester, New York, April. Association for Computational Linguistics.

Greg Durrett and Dan Klein. 2013. Easy victories and uphill battles in coreference resolution. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, Seattle, Washington, October. Association for Computational Linguistics.

Jenny Rose Finkel and Christopher D. Manning. 2008. Enforcing transitivity in coreference resolution. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics, Short Papers*, pages 45–48.

M.A. Finlayson and R Hervs. 2010. Ucm/mit indications, referring expressions, and co-reference corpus v1.1 (umirec corpus). MIT CSAIL Work Product.

Mark A. Finlayson, Jeffry R. Halverson, and Steven R. Corman. 2014. The n2 corpus: A semantically annotated collection of islamist extremist stories. The 9th Language Resources and Evaluation Conference (LREC), Reykjavik, Iceland.

B. J. Grosz, A. K. Joshi, and S. Weinstein. 1995. Centering: A framework for modelling the local coherence of discourse. *Computational Linguistics*, 21(2):203–226.

Aria Haghighi and Dan Klein. 2009. Simple coreference resolution with rich syntactic and semantic features. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 3 - Volume 3*, EMNLP '09, pages 1152–1161, Stroudsburg, PA, USA. Association for Computational Linguistics.

Aria Haghighi and Dan Klein. 2010. Coreference resolution in a modular, entity-centered model. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 385–393, Stroudsburg, PA, USA. Association for Computational Linguistics.

Hudson-D'Zmura. 1988. The structure of discourse and anaphore resolution: The discourse center and the roles of nouns and pronouns. Unpublished doctoral dissertation.

Heeyoung Lee, Yves Peirsman, Angel Chang, Nathanael Chambers, Mihai Surdeanu, and Dan Jurafsky. 2011. Stanford's multi-pass sieve coreference resolution system at the conll-2011 shared task. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task*, CONLL Shared Task '11, pages 28–34, Stroudsburg, PA, USA. Association for Computational Linguistics.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60.

Alan Prince and Paul Smolensky. 2004. *Optimality theory: Constraint interaction in generative grammar*. Wiley-Blackwell.

Manfred Stede. 2011. *Discourse processing*. Morgan & Claypool Publishers.

Veselin Stoyanov, Claire Cardie, Nathan Gilbert, Ellen Riloff, David Buttler, and David Hysom. 2010. Coreference resolution with reconcile. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 156–161, Uppsala, Sweden, July. Association for Computational Linguistics.

Michael J. Toolan. 2009. *Narrative Progression in the Short Story: A Corpus Stylistic Approach*. John Benjamins Publishing.

Joint Lemmatization and Morphological Tagging with LEMMING

Thomas Müller[1] **Ryan Cotterell**[1,2] **Alexander Fraser**[1] **Hinrich Schütze**[1]
Center for Information and Language Processing[1]
University of Munich, Germany
muellets@cis.lmu.de
Department of Computer Science[2]
Johns Hopkins University, USA
ryan.cotterell@jhu.edu

Abstract

We present LEMMING, a modular log-linear model that jointly models lemmatization and tagging and supports the integration of arbitrary global features. It is trainable on corpora annotated with gold standard tags and lemmata and does not rely on morphological dictionaries or analyzers. LEMMING sets the new state of the art in token-based statistical lemmatization on six languages; e.g., for Czech lemmatization, we reduce the error by 60%, from 4.05 to 1.58. We also give empirical evidence that jointly modeling morphological tags and lemmata is mutually beneficial.

1 Introduction

Lemmatization is important for many NLP tasks, including parsing (Björkelund et al., 2010; Seddah et al., 2010) and machine translation (Fraser et al., 2012). Lemmata are required whenever we want to map words to lexical resources and establish the relation between inflected forms, particularly critical for morphologically rich languages to address the sparsity of unlemmatized forms. This strongly motivates work on language-independent token-based lemmatization, but until now there has been little work (Chrupała et al., 2008).

Many regular transformations can be described by simple replacement rules, but lemmatization of unknown words requires more than this. For instance the Spanish paradigms for verbs ending in *ir* and *er* share the same 3rd person plural ending *en*; this makes it hard to decide which paradigm a form belongs to.[1] Solving these kinds of problems requires global features on the lemma. Global features of this kind were not supported by previous work (Dreyer et al., 2008; Chrupała, 2006; Toutanova and Cherry, 2009; Cotterell et al., 2014).

There is a strong mutual dependency between (i) lemmatization of a form in context and (ii) disambiguating its part-of-speech (POS) and morphological attributes. Attributes often disambiguate the lemma of a form, which explains why many NLP systems (Manning et al., 2014; Padró and Stanilovsky, 2012) apply a pipeline approach of tagging followed by lemmatization. Conversely, knowing the lemma of a form is often beneficial for tagging, for instance in the presence of syncretism; e.g., since German plural noun phrases do not mark gender, it is important to know the lemma (singular form) to correctly tag gender on the noun.

We make the following contributions. (i) We present the first joint log-linear model of morphological analysis and lemmatization that operates at the *token* level and is also able to lemmatize unknown forms; and release it as open-source (`http://cistern.cis.lmu.de/lemming`). It is trainable on corpora annotated with gold standard tags and lemmata. Unlike other work (e.g., (Smith et al., 2005)) it does not rely on morphological dictionaries or analyzers. (ii) We describe a log-linear model for lemmatization that can easily be incorporated into other models and supports arbitrary global features on the lemma. (iii) We set the new state of the art in token-based statistical lemmatization on six languages (English, German, Czech, Hungarian, Latin and Spanish). (iv) We experimentally show that jointly modeling morphological tags and lemmata is mutually beneficial and yields significant improvements in joint (tag+lemma) accuracy for four out of six languages; e.g., Czech lemma errors are reduced by $>37\%$ and tag+lemma errors by $>6\%$.

[1] Compare *admiten* "they admit" → *admitir* "to admit", but *deben* "they must" → *deber* "to must".

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 2268–2274,
Lisbon, Portugal, 17-21 September 2015.

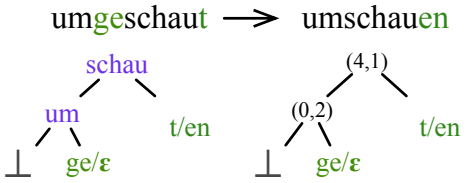


Figure 1: Edit tree for the inflected form *umgeschaut* "looked around" and its lemma *umschauen* "to look around". The right tree is the actual edit tree we use in our model, the left tree visualizes what each node corresponds to. The root node stores the length of the prefix *umge* (4) and the suffix *t* (1).

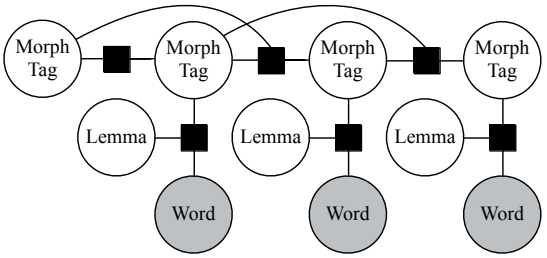


Figure 2: Our model is a 2nd-order linear-chain CRF augmented to predict lemmata. We heavily prune our model and can easily exploit higher-order (>2) tag dependencies.

2 Log-Linear Lemmatization

Chrupała (2006) formalizes lemmatization as a classification task through the deterministic pre-extraction of edit operations transforming forms into lemmata. Our lemmatization model is in this vein, but allows the addition of external lexical information, e.g., whether the candidate lemma is in a dictionary. Formally, lemmatization is a string-to-string transduction task. Given an alphabet Σ, it maps an inflected form $w \in \Sigma^*$ to its lemma $l \in \Sigma^*$ given its morphological attributes m. We model this process by a log-linear model:

$$p(l \mid w, m) \propto h_w(l) \cdot \exp\left(\boldsymbol{f}(l, w, m)^T \boldsymbol{\theta}\right),$$

where $\boldsymbol{f}$ represents hand-crafted feature functions, $\boldsymbol{\theta}$ is a weight vector, and $h_w : \Sigma^* \rightarrow \{0, 1\}$ determines the support of the distribution, i.e., the set of candidates with non-zero probability.

Candidate selection. A proper choice of the support function $h(\cdot)$ is crucial to the success of the model – too permissive a function and the computational cost will build up, too restrictive and the correct lemma may receive no probability mass. Following Chrupała (2008), we define $h(\cdot)$ through a deterministic pre-extraction of *edit trees*. To extract an edit tree e for a pair form-lemma $\langle w, l \rangle$, we first find the longest common substring (LCS) (Gusfield, 1997) between them and then recursively model the prefix and suffix pairs of the LCS. When no LCS can be found the string pair is represented as a substitution operation transforming the first string to the second. The resulting edit tree does not encode the LCSs but only the length of their prefixes and suffixes and the substitution nodes (cf. Figure 1); e.g., the same tree transforms *worked* into *work* and *touched* into *touch*.

As a preprocessing step, we extract all edit trees that can be used for more than one pair $\langle w, l \rangle$. To generate the candidates of a word-form, we apply all edit trees and also add all lemmata this form was seen with in the training set (note that only a small subset of the edit trees is applicable for any given form because most require incompatible substitution operations).[2]

Features. Our novel formalization lets us combine a wide variety of features that have been used in different previous models. All features are extracted given a form-lemma pair $\langle w, l \rangle$ created with an edit tree e.

We use the following three *edit tree features* of Chrupała (2008). (i) The edit tree e. (ii) The pair $\langle e, w \rangle$. This feature is crucial for the model to memorize irregular forms, e.g., the lemma of *was* is *be*. (iii) For each form affix (of maximum length 10): its conjunction with e. These features are useful in learning orthographic and phonological regularities, e.g., the lemma of *signalling* is *signal*, not *signall*.

We define the following *alignment features*. Similar to Toutanova and Cherry (2009) (TC), we define an alignment between w and l. Our alignments can be read from an edit tree by aligning the characters in LCS nodes character by character and characters in substitution nodes block-wise. Thus the alignment of *umgeschaut* - *umschauen* is: u-u, m-m, ge-ϵ, s-s, c-c, h-h, a-a, u-u, t-en. Each alignment pair constitutes a feature in our model. These features allow the model to learn that the substitution *t/en* is likely in German. We also concatenate each alignment pair with its form and lemma character context (of up to length 6) to learn, e.g., that *ge* is often deleted after *um*.

We define two simple *lemma features*. (i) We use the lemma itself as a feature, allowing us to learn which lemmata are common in the language. (ii) Prefixes and suffixes of the lemma (of maxi-

[2]Pseudo-code for edit tree creation and candidate lemma generation with examples can be found in the appendix (`http://cistern.cis.lmu.de/lemming/appendix.pdf`).

mum length 10). This feature allows us to learn that the typical endings of Spanish verbs are *ir*, *er*, *ar*.

We also use two *dictionary features* (on lemmata): Whether l occurs > 5 times in Wikipedia and whether it occurs in the dictionary ASPELL.[3] We use a similar feature for different capitalization variants of the lemma (lowercase, first letter uppercase, all uppercase, mixed). This differentiation is important for German, where nouns are capitalized and *en* is both a noun plural marker and a frequent verb ending. Ignoring capitalization would thus lead to confusion.

POS & morphological attributes. For each feature listed previously, we create a conjunction with the POS and each morphological attribute.[4]

3 Joint Tagging and Lemmatization

We model the sequence of morphological tags using MARMOT (Müller et al., 2013), a pruned higher-order CRF. This model avoids the exponential runtime of higher-order models by employing a pruning strategy. Its feature set consists of standard tagging features: the current word, its affixes and shape (capitalization, digits, hyphens) and the immediate lexical context. We combine lemmatization and higher-order CRF components in a tree-structured CRF. Given a sequence of forms $\boldsymbol{w}$ with lemmata $\boldsymbol{l}$ and morphological+POS tags $\boldsymbol{m}$, we define a globally normalized model:

$$p(\boldsymbol{l}, \boldsymbol{m} \mid \boldsymbol{w}) \propto \prod_i h_{w_i}(l_i) \exp(\boldsymbol{f}(l_i, w_i, m_i)^T \boldsymbol{\theta} + \boldsymbol{g}(m_i, m_{i-1}, m_{i-2}, \boldsymbol{w}, i)^T \boldsymbol{\lambda}),$$

where $\boldsymbol{f}$ and $\boldsymbol{g}$ are the features associated with lemma and tag cliques respectively and $\boldsymbol{\theta}$ and $\boldsymbol{\lambda}$ are weight vectors. The graphical model is shown in Figure 2. We perform inference with belief propagation (Pearl, 1988) and estimate the parameters with SGD (Tsuruoka et al., 2009). We greatly improved the results of the joint model by initializing it with the parameters of a pretrained tagging model.

4 Related Work

In functionality, our system resembles MORFETTE (Chrupała et al., 2008), which generates lemma candidates by extracting edit operation sequences between lemmata and surface forms (Chrupała, 2006), and then trains two maximum entropy Markov models (Ratnaparkhi, 1996) for morphological tagging and lemmatization, which are queried using a beam search decoder.

In our experiments we use the latest version[5] of MORFETTE. This version is based on structured perceptron learning (Collins, 2002) and edit trees (Chrupała, 2008). Models similar to MORFETTE include those of Björkelund et al. (2010) and Gesmundo and Samardzic (2012) and have also been used for generation (Dušek and Jurčíček, 2013). Wicentowski (2002) similarly treats lemmatization as classification over a deterministically chosen candidate set, but uses distributional information extracted from large corpora as a key source of information.

Toutanova and Cherry (2009)'s joint morphological analyzer predicts the set of possible lemmata and *coarse-grained POS* for a word *type*. This is different from our problem of lemmatization and *fine-grained* morphological tagging of *tokens in context*. Despite the superficial similarity of the two problems, *direct comparison is not possible*. TC's model is best thought of as inducing a tagging dictionary for OOV types, mapping them to a set of tag and lemma pairs, whereas LEMMING is a token-level, context-based morphological tagger.

We do, however, use TC's model of lemmatization, a string-to-string transduction model based on Jiampojamarn et al. (2008) (JCK), as a standalone baseline. Our tagging-in-context model is faced with higher complexity of learning and inference since it addresses a more difficult task; thus, while we could in principle use JCK as a replacement for our candidate selection, the edit tree approach – which has high coverage at a low average number of lemma candidates (cf. Section 5) – allows us to train and apply LEMMING efficiently.

Smith et al. (2005) proposed a log-linear model for the context-based disambiguation of a morphological dictionary. This has the effect of joint tagging, morphological segmentation and lemmatization, but, critically, is limited to the entries in the morphological dictionary (without which the approach cannot be used), causing problems of recall. In contrast, LEMMING can analyze any word,

[3] `ftp://ftp.gnu.org/gnu/aspell/dict`

[4] Example: for the Spanish noun *medidas* "measures" with attributes NOUN, COMMON, PLURAL and FEMININE, we conjoin each feature above with NOUN, NOUN+COMMON, NOUN+PLURAL and NOUN+FEMININE.

[5] `https://github.com/gchrupala/morfette/commit/ca886556916b6cc1e808db4d32daf720664d17d6`

				cs		de		en		es		hu		la	
				all	unk	all	unk	all	unk	all	unk	all	unk	all	unk
1	MARMOT		tag	89.75	76.83	82.81	61.60	**96.45**	**90.68**	97.05	90.07	93.64	84.65	82.37	53.73
2	JCK		lemma	95.95	81.28	96.63	85.84	99.08	94.28	97.69	87.19	96.69	88.66	90.79	58.23
3			tag+lemma	87.85	67.00	81.60	55.97	96.17	87.32	95.44	80.62	92.15	78.89	79.51	39.07
4	LEMMING-P	+dict	lemma	97.46	89.14	97.70	91.27	**99.21**	**95.59**	98.48	92.98	97.53	92.10	93.07	69.83
5			tag+lemma	88.86	72.51	82.27	59.42	**96.27**	**88.49**	96.12	85.80	92.59	80.77	80.49	44.26
6		+mrph	lemma	97.29	88.98	97.51	90.85	NA	NA	98.68	94.32	97.53	92.15	92.54	67.81
7			tag+lemma	89.23	74.24	82.49	60.42	NA	NA	96.35	87.25	93.11	82.56	80.67	45.21
8	LEMMING-J	+dict	tag	**90.34**$^{+}$	78.47	**83.10**$^{+}$	62.36	96.32	89.70	97.11	90.13	93.64	84.78	82.89	54.69
9			lemma	98.27	92.67	**98.10**$^{+}$	92.79	**99.21**	95.23	98.67	94.07	98.02	94.15	**95.58**$^{+}$	**81.74**$^{+}$
10			tag+lemma	89.69	75.44	82.64	60.49	96.17	87.87	96.23	86.19	92.84	81.89	81.92	49.97
11		+mrph	tag	90.20	**79.72***	**83.10**$^{+}$	**63.10***	NA	NA	**97.16**	**90.66**	**93.67**	**85.12**	**83.49***	**58.76***
12			lemma	**98.42***	**93.46***	**98.10**$^{+}$	**93.02**$^{+}$	NA	NA	**98.78***	**94.86***	**98.08**$^{+}$	**94.26**$^{+}$	95.36	80.94
13			tag+lemma	**89.90***	**78.34***	**82.84***	**62.10***	NA	NA	**96.41**$^{\times}$	**87.47**$^{\times}$	**93.40***	**84.15***	**82.57**$^{+}$	**54.63**$^{+}$

Table 2: *Test* results for LEMMING-J, the joint model, and pipelines (lines 2–7) of MARMOT and (i) JCK and (ii) LEMMING-P. In each cell, overall token accuracy is left (all), accuracy on unknown forms is right (unk). Standalone MARMOT tagging accuracy (line 1) is not repeated for pipelines (lines 2–7). The best numbers are bold. LEMMING-J models significantly better than LEMMING-P ($+$), or LEMMING models not using morphology (+*dict*) ($\times$) or both ($*$) are marked. More baseline numbers in the appendix (Table A2).

including OOVs, and only requires the same training corpus as a generic tagger (containing tags and lemmata), a resource that is available for many languages.

5 Experiments

Datasets. We present experiments on the joint task of lemmatization and tagging in six diverse languages: English, German, Czech, Hungarian, Latin and Spanish. We use the same data sets as in Müller and Schütze (2015), but do not use the out-of-domain test sets. The English data is from the Penn Treebank (Marcus et al., 1993), Latin from PROIEL (Haug and Jøhndal, 2008), German and Hungarian from SPMRL 2013 (Seddah et al., 2013), and Spanish and Czech from CoNLL 2009 (Hajič et al., 2009). For German, Hungarian, Spanish and Czech we use the splits from the shared tasks; for English the split from SANCL (Petrov and McDonald, 2012); and for Latin a 8/1/1 split into *train/dev/test*. For all languages we limit our training data to the first 100,000 tokens. Dataset statistics can be found in Table A4 of the appendix. The lemma of Spanish *se* is set to be consistent.

Baselines. We compare our model to three baselines. (i) MORFETTE (see Section 4). (ii) SIMPLE, a system that for each form-POS pair, returns the most frequent lemma in the training data or the form if the pair is unknown. (iii) JCK, our reimplementation of Jiampojamarn et al. (2008). Recall that JCK is TC's lemmatization model and that the full TC model is a type-based model that cannot be applied to our task.

As JCK struggles to memorize irregulars, we only use it for unknown form-POS pairs and use SIMPLE otherwise. For aligning the training data we use the edit-tree-based alignment described in the feature section. We only use output alphabet symbols that are used for ≥ 5 form-lemma pairs and also add a special output symbol that indicates that the aligned input should simply be copied. We train the model using a structured averaged perceptron and stop after 10 training iterations. In preliminary experiments we found type-based training to outperform token-based training. This is understandable as we only apply our model to unseen form-POS pairs. The feature set is an exact reimplementation of (Jiampojamarn et al., 2008), it consists of input-output pairs and their character context in a window of 6.

Results. Our candidate selection strategy results in an average number of lemma candidates between 7 (Hungarian) and 91 (Czech) and a coverage of the correct lemma on *dev* of >99.4 (except 98.4 for Latin).[6] We first compare the baselines to LEMMING-P, a pipeline based on Section 2, that lemmatizes a word given a predicted tag and is trained using L-BFGS (Liu and Nocedal, 1989). We use the implementation of MALLET (McCallum, 2002). For these experiments we train all models on gold attributes and test on attributes predicted by MORFETTE. MORFETTE's lemmatizer can only be used with its own tags. We thus use MORFETTE tags to have a uniform setup,

[6]Note that our definition of lemmatization accuracy and unknown forms ignores capitalization.

		cs	de	en	es	hu	la
baselines	SIMPLE	87.22	93.27	97.60	92.92	86.09	85.19
	JCK	96.24	97.67	98.71	97.61	97.48	93.26
	MORFETTE	96.25	97.12	98.43	97.97	97.22	91.89
LEMMING-P	edittree	96.29	97.84^+	98.71	97.91	97.31	93.00
	+align,+lemma	96.74^+	98.17^+	98.76^+	98.05	97.70^+	93.76^+
	+dict	$\mathbf{97.50}^+$	$\mathbf{98.36}^+$	$\mathbf{98.84}^+$	98.39^+	$\mathbf{97.98}^+$	$\mathbf{94.64}^+$
	+mrph	96.59^+	97.43^+	NA	$\mathbf{98.46}^+$	97.77^+	93.60

Table 1: Lemma accuracy on *dev* for the baselines and the different versions of LEMMING-P. POS and morphological attributes are predicted using MORFETTE. The best baseline numbers are underlined, the best numbers are bold. Models significantly better than the best baseline are marked (+).

which isolates the effects of the different taggers. Numbers for MARMOT tags are in the appendix (Table A1). For the initial experiments, we only use POS and ignore additional morphological attributes. We use different feature sets to illustrate the utility of our templates.

The first model uses the *edit tree features* (edittree). Table 1 shows that this version of LEMMING outperforms the baselines on half of the languages.[7] In a second experiment we add the *alignment* (+align) and *lemma features* (+lemma) and show that this consistently outperforms all baselines and edittree. We then add the *dictionary feature* (+dict). The resulting model outperforms all previous models and is significantly better than the best baselines for all languages.[8] These experiments show that LEMMING-P yields state-of-the-art results and that all our features are needed to obtain optimal performance. The improvements over the baselines are >1 for Czech and Latin and $\geq.5$ for German and Hungarian.

The last experiment also uses the additional morphological attributes predicted by MORFETTE (+mrph). This leads to a drop in lemmatization performance in all languages except Spanish (English has no additional attributes). However, preliminary experiments showed that correct morphological attributes would substantially improve lemmatization as they help in cases of ambiguity. As an example, number helps to lemmatize the singular German noun *Raps* "canola", which looks like the plural of *Rap* "rap". Numbers can be found in Table A3 of the appendix. This motivates the necessity of *joint tagging and lemmatization*.

For the final experiments, we run pipeline models on tags predicted by MARMOT (Müller et al., 2013) and compare them to LEMMING-J, the joint model described in Section 3. All LEMMING versions use exactly the same features. Table 2 shows that LEMMING-J outperforms LEMMING-P in three measures (see bold tag, lemma & joint (tag+lemma) accuracies) except for English, where we observe a tie in lemma accuracy and a small drop in tag and tag+lemma accuracy. Coupling morphological attributes and lemmatization (lines 8–10 vs 11–13) improves tag+lemma prediction for five languages. Improvements in lemma accuracy of the joint over the best pipeline systems range from .1 (Spanish), over $>.3$ (German, Hungarian) to $\geq.96$ (Czech, Latin).

Lemma accuracy improvements for our best models (lines 4–13) over the best baseline (lines 2–3) are >1 (German, Spanish, Hungarian), >2 (Czech, Latin) and even more pronounced on unknown forms: >1 (English), >5 (German, Spanish, Hungarian) and >12 (Czech, Latin).

6 Conclusion

LEMMING is a modular lemmatization model that supports arbitrary global lemma features and joint modeling of lemmata and morphological tags. It is trainable on corpora annotated with gold standard tags and lemmata, and does not rely on morphological dictionaries or analyzers. We have shown that modeling lemmatization and tagging jointly benefits both tasks, and we set the new state of the art in token-based lemmatization on six languages. LEMMING is available under an open-source licence (`http://cistern.cis.lmu.de/lemming`).

Acknowledgments

We would like to thank the anonymous reviewers for their comments. The first author is a recipient of the Google Europe Fellowship in Natural Language Processing, and this research is supported by this Google fellowship. The second author is supported by a Fulbright fellowship awarded by the German-American Fulbright Commission and the National Science Foundation under Grant No. 1423276. This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 644402 (HimL) and the DFG grant *Models of Morphosyntax for Statistical Machine Translation*. The fourth author was partially supported by Deutsche Forschungsgemeinschaft (grant SCHU 2246/10-1).

[7] Unknown word accuracies in the appendix (Table A1).

[8] We use the randomization test (Yeh, 2000) and $p = .05$.

References

Anders Björkelund, Bernd Bohnet, Love Hafdell, and Pierre Nugues. 2010. A high-performance syntactic and semantic dependency parser. In *Proceedings of COLING: Demonstrations*.

Grzegorz Chrupała, Georgiana Dinu, and Josef Van Genabith. 2008. Learning morphology with Morfette. In *Proceedings of LREC*.

Grzegorz Chrupała. 2006. Simple data-driven contextsensitive lemmatization. *Procesamiento del Lenguaje Natural*.

Grzegorz Chrupała. 2008. *Towards a machine-learning architecture for lexical functional grammar parsing*. Ph.D. thesis, Dublin City University.

Michael Collins. 2002. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *Proceedings of EMNLP*.

Ryan Cotterell, Nanyun Peng, and Jason Eisner. 2014. Stochastic contextual edit distance and probabilistic FSTs. In *Proceedings of ACL*.

Markus Dreyer, Jason R Smith, and Jason Eisner. 2008. Latent-variable modeling of string transductions with finite-state methods. In *Proceedings of EMNLP*.

Ondřej Dušek and Filip Jurčíček. 2013. Robust multilingual statistical morphological generation models. In *Proceedings of ACL: Student Research Workshop*.

Alexander Fraser, Marion Weller, Aoife Cahill, and Fabienne Cap. 2012. Modeling inflection and word-formation in SMT. In *Proceedings of EACL*.

Andrea Gesmundo and Tanja Samardzic. 2012. Lemmatisation as a tagging task. In *Proceedings of ACL*.

Dan Gusfield. 1997. *Algorithms on Strings, Trees, and Sequences - Computer Science and Computational Biology*. Cambridge University Press.

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, et al. 2009. The CoNLL-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of CoNLL*.

Dag TT Haug and Marius Jøhndal. 2008. Creating a parallel treebank of the old Indo-European bible translations. In *Proceedings of LaTeCH*.

Sittichai Jiampojamarn, Colin Cherry, and Grzegorz Kondrak. 2008. Joint processing and discriminative training for letter-to-phoneme conversion. In *Proceedings of ACL*.

Dong C Liu and Jorge Nocedal. 1989. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of ACL: Demonstrations*.

Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of English: The Penn treebank. *Computational linguistics*.

Andrew K McCallum. 2002. MALLET: A machine learning for language toolkit.

Thomas Müller and Hinrich Schütze. 2015. Robust morphological tagging with word representations. In *Proceedings of NAACL*.

Thomas Müller, Helmut Schmid, and Hinrich Schütze. 2013. Efficient higher-order CRFs for morphological tagging. In *Proceedings of EMNLP*.

Lluís Padró and Evgeny Stanilovsky. 2012. FreeLing 3.0: Towards wider multilinguality. In *Proceedings of LREC*.

Judea Pearl. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann.

Slav Petrov and Ryan McDonald. 2012. Overview of the 2012 shared task on parsing the web. In *Proceedings of SANCL*.

Adwait Ratnaparkhi. 1996. A maximum entropy model for part-of-speech tagging. In *Proceedings of EMNLP*.

Djamé Seddah, Grzegorz Chrupała, Özlem Çetinoğlu, Josef Van Genabith, and Marie Candito. 2010. Lemmatization and lexicalized statistical parsing of morphologically rich languages: the case of French. In *Proceedings of SPMRL*.

Djamé Seddah, Reut Tsarfaty, Sandra Kübler, Marie Candito, Jinho D. Choi, Richárd Farkas, Jennifer Foster, Iakes Goenaga, Koldo Gojenola Galletebeitia, Yoav Goldberg, Spence Green, Nizar Habash, Marco Kuhlmann, Wolfgang Maier, Joakim Nivre, Adam Przepiórkowski, Ryan Roth, Wolfgang Seeker, Yannick Versley, Veronika Vincze, Marcin Woliński, Alina Wróblewska, and Eric Villemonte de la Clergerie. 2013. Overview of the SPMRL 2013 shared task: Cross-Framework evaluation of parsing morphologically rich languages. In *Proceedings of SPMRL*.

Noah A. Smith, David A. Smith, and Roy W. Tromble. 2005. Context-based morphological disambiguation with random fields. In *Proceedings of HLT-EMNLP*.

Kristina Toutanova and Colin Cherry. 2009. A global model for joint lemmatization and part-of-speech prediction. In *Proceedings of ACL-IJCNLP*.

Yoshimasa Tsuruoka, Jun'ichi Tsujii, and Sophia Ananiadou. 2009. Stochastic gradient descent training for l1-regularized log-linear models with cumulative penalty. In *Proceedings of ACL-IJCNLP*.

Richard Wicentowski. 2002. *Modeling and learning multilingual inflectional morphology in a minimally supervised framework*. Ph.D. thesis, Johns Hopkins University.

Alexander Yeh. 2000. More accurate tests for the statistical significance of result differences. In *Proceedings of COLING*.

Transducer Disambiguation with Sparse Topological Features

Gonzalo Iglesias[†] **Adrià de Gispert**[†‡] **Bill Byrne**[†‡]

[†]SDL Research, Cambridge, U.K.
{giglesias|agispert|bbyrne}@sdl.com

[‡]Department of Engineering, University of Cambridge, U.K.

Abstract

We describe a simple and efficient algorithm to disambiguate non-functional weighted finite state transducers (WFSTs), i.e. to generate a new WFST that contains a unique, best-scoring path for each hypothesis in the input labels along with the best output labels. The algorithm uses topological features combined with a tropical sparse tuple vector semiring. We empirically show that our algorithm is more efficient than previous work in a PoS-tagging disambiguation task. We use our method to rescore very large translation lattices with a bilingual neural network language model, obtaining gains in line with the literature.

1 Introduction

Weighted finite-state transducers (WFSTs), or *lattices*, are used in speech and language processing to compactly represent and manipulate a large number of strings. Applying a finite-state operation (eg. PoS tagging) to a lattice via composition produces a WFST that maps input (eg. words) onto output strings (eg. PoS tags) and preserves the arc-level alignment between each input and output symbol (eg. each arc is labeled with a word-tag pair and has a weight). Typically, the result of such operation is a WFST that is *ambiguous* because it contains multiple paths with the same input string, and *non-functional* because it contains multiple output strings for a given input string (Mohri, 2009).

Disambiguating such WFSTs is the task of creating a WFST that encodes only the best-scoring path of each input string, while still maintaining the arc-level mapping between input and output symbols. This is a non-trivial task[1], and so far only one algorithm has been described (Shafran et al., 2011); the main steps are:

(a) Map the WFST into an equivalent weighted finite-state automata (WFSA) using weights that contain *both* the WFST weight *and* output symbols (using a special semiring)

(b) Apply WFSA determinization under this semiring to ensure that only one unique path per input string survives

(c) Expand the result back to an WFST that preserves arc-level alignments

We present a new disambiguation algorithm that can efficiently accomplish this. In Section 2 we describe how the *tropical sparse tuple vector* semiring can keep track of individual arcs in the original WFST as topological features during the mapping step (**a**). This allows us to describe in Section 3 an efficient expansion algorithm for step (**c**). We show in Section 4 empirical evidence that our algorithm is more efficient than Shafran et al. (2011) in their same PoS-tagging task. We also show how our method can be applied in rescoring translation lattices under a bilingual neural-network model (Devlin et al., 2014), obtaining BLEU score gains consistent with the literature. Section 5 reviews related work and concludes.

2 Semiring Definitions

A WFST $T = (\Sigma, \Delta, Q, I, F, E, \rho)$ over a semiring $(\mathbb{K}, \oplus, \otimes, \overline{0}, \overline{1})$ has input and output alphabets Σ and Δ, a set of states Q, the initial state $I \in Q$, a set of final states $F \subset Q$, a set of transitions (edges) $E \subset (Q \times \Sigma \times \Delta \times \mathbb{K} \times Q)$, and a final state function $\rho : F \to \mathbb{K}$. We focus on extensions to the tropical semiring $(\mathbb{R} \pm \infty, \min, +, \infty, 0)$.

[1]Unless one enumerates all the possible input strings in the lattice, searches for the best output string for each input string, and converts the resulting sequences back into a WFST, which is clearly inefficient.

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 2275–2280,
Lisbon, Portugal, 17-21 September 2015.

Let $e = (p[e], i[e], o[e], w[e], n[e])$ be an edge in E. A path $\pi = e_1 ... e_n$ is a sequence of edges such that $n[e_j] = p[e_{j+1}], 1 \leq j < n$. $w[\pi] = \bigotimes_{e_j \in \pi} w[e_j]$; $p[\pi] = p[e_1], n[\pi] = n[e_n]$. A path is accepting if $p[\pi] = I$ and $n[\pi] \in F$. The weight associated by T to a set of paths Π is $T(\Pi) = \bigoplus_{\pi \in \Pi} w[\pi] \otimes \rho(n[\pi])$.

2.1 Tropical Sparse Vector Semiring

Let $\bar{f}[e_i] = f_i \in \mathbb{R}^N$ be the unweighted feature vector associated with edge e_i, and let $\bar{\alpha} \in \mathbb{R}^N$ be a global feature weight vector. The tropical weight is then found as $w_i = w[e_i] = \bar{\alpha} \cdot \bar{f}_i = \sum_k \alpha_k f_{i,k}$.

Given a fixed $\bar{\alpha}$, we define the operators for the *tropical vector* semiring as: $\bar{f}_i \oplus_\alpha \bar{f}_j = \min(\bar{\alpha} \cdot \bar{f}_i, \bar{\alpha} \cdot \bar{f}_j)$ and $\bar{f}_i \otimes_\alpha \bar{f}_j = \sum_k \alpha_k (f_{i,k} + f_{j,k})$. The tropical weights are maintained correctly by the vector semiring as $\bar{f}_i \oplus_\alpha \bar{f}_j = w_i \oplus w_j$ and $\bar{f}_i \otimes_\alpha \bar{f}_j = w_i \otimes w_j$. Finally, we define the element-wise times operator as: $\bar{f}_i * \bar{f}_j = \bar{f}_m$, where $f_{m,k} = f_{i,k} + f_{j,k}, \forall k$. It follows that $w_i \otimes w_j = \alpha \cdot (\bar{f}_i * \bar{f}_j)$.

When dealing with high-dimensional feature vectors which have few non-zero elements, it is convenient in practice (for computational efficiency) to use a sparse representation for vectors: $\bar{f} = [(i, f_i), i : f_i \neq 0]$. That is, $\bar{f}$ is comprised of a sparse set of tuples (i, f_i), where i is a feature index and f_i is its value; e.g. $[(2, f_2)]$ is short for $[0, f_2, 0, 0]$ if $N = 4$.

The semiring that operates on sparse feature vectors, which we call *tropical sparse tuple vector* semiring[2], uses conceptually identical operators as the non-sparse version defined above, so it also maintains the tropical weights w correctly.

3 A Disambiguation Algorithm

We now describe how we use the semiring described in Section 2 for steps (**a**) and (**b**), and describe an expansion algorithm for step (**c**) that efficiently converts the output of determinization into an unambiguous WFST with arc-level alignments.

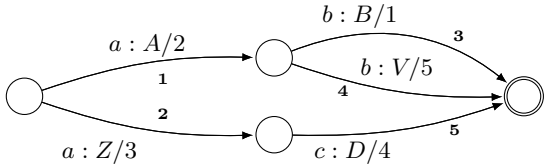
WFSA with Sparse Topological Features

Let T be a tropical-weight WFST with K edges. T is topologically sorted so that if an edge e_k preceeds an edge $e_{k'}$ on a path, then $k < k'$. We now use tropical sparse tuple vector weights to create a WFSA A that maintains (in its weights) pointers to specific edges in T. These 'pointers' are sparse topological features.

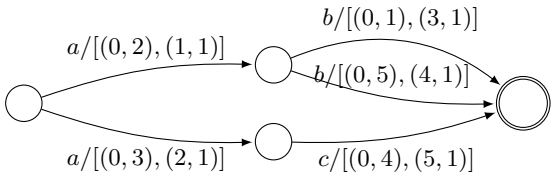
[2]We implement this semiring as an extension to the sparse tuple weights of the OpenFst library (Allauzen et al., 2007).

For each edge $e_k = (p_k, i_k, o_k, w_k, n_k)$ of T, we create an edge $e'_k = (p_k, i_k, i_k, \bar{f}_k, n_k)$ in A, where $\bar{f}_k = [w_k, 0, \ldots, 0, 1, 0, \ldots, 0]$; the 1 is in the k^{th} position. In other words, $f_{k,0}$ is the tropical weight of the k^{th} edge in T and $f_{k,k} = 1$ indicates that this tropical weight belongs to edge k in T. In sparse notation, $\bar{f}_k = [(0, w_k), (k, 1)]$.

For example, this non-deterministic transducer T:



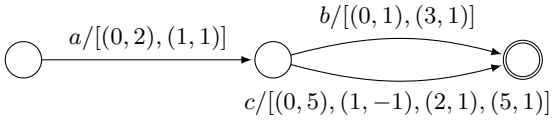
is mapped to acceptor A with topological features:



Given $\alpha = [1, 0, \ldots, 0]$, operations on A yield the same path weights as in the usual tropical semiring.

WFSA determinization

We now apply the standard determinization algorithm to A, which yields A^D:



This now accepts only one best-scoring path for each input string, and the weights 'point' to the sequence of arcs traversed by the relevant path in T. In turn, this reveals the best output string for the given input string. For example, the path-level features associated with '$a\,b$' are $[(0, 3), (1, 1), (3, 1)]$, indicating a path $\pi = e_1 e_3$ with tropical weight 3 through T (and hence output string 'AB').

The topology of A^D is compact because multiple input strings may share arcs while still encoding different output strings in their weights. This is achieved by 'cancelling' topological features in subsequent arcs and 'replacing' them by new ones as one traverses the path. For example, the string '$a\,c$' initially has feature $(1, 1)$, but this gets cancelled later in the path by $(1, -1)$, and replaced by $[(2, 1), (5, 1)]$, indicating a path $\pi = e_2 e_5$ through T with output string 'ZD'.

EXPANDTFEA($A^r = (\Sigma, \Delta, Q, I, F, E)$)

```
1  I' ← (I,0)
2  ENQUEUE(S, I')
3  while |S| do
4    (q, f̄) ← HEAD(S)
5    DEQUEUE(S)
6    if q ∈ F then
7      F' ← F' ∪ {(q, f̄))}
8    for each e ∈ E(q) do
9      (w', t', f̄') ← POPTFEA(f̄, e)
10     q' ← (n[e], f̄')
11     e' ← ((q, f̄), i[e], i[e], [(0, w'), (t', 1)], q')
12     Q' ← Q' ∪ {q'}
13     E' ← E' ∪ {e'}
14     ENQUEUE(S, (n[e], f̄'))
15 return B^r = (Σ, Δ, Q', I', F', E')
```

Figure 1: Expansion and topological feature repositioning algorithm for step (**c**).

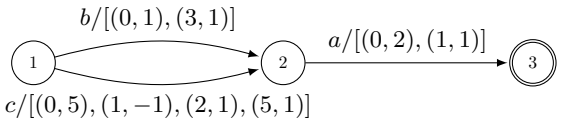
Expansion Algorithm

We now describe an expansion algorithm to convert A^D into an unambiguous WFST T' that maintains the arc-level input-output alignments of the original transducer T. In our example, T' should be identical to T except for edge **4**, which is removed.

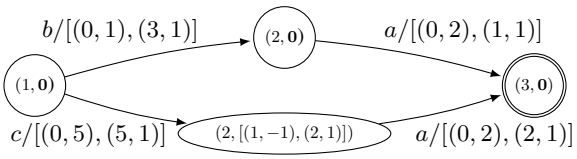
Due to the WFSA determinization algorithm, we observe empirically that the cancelling features in A^D *tend* to appear in a path *after* the feature itself. This allows us to define an algorithm that traverses A^D in reverse (from its final states to its initial state) and creates an equivalent acceptor with the topology of T'.

The algorithm is described in Figure 1. It performs a forward pass through A^r (the reverse of A^D). The intuition is that, for each arc, we create a new arc where we 'pop' the highest topological feature (as it will not be cancelled later) and its tropical weight. The new states encode the original state q and the residual features that have not been 'popped' yet. For each edge $E(q)$, the auxiliary POPTFEA($\bar{f}, e$) returns a $(w', t', \bar{f}')$ tuple, where w' is the tropical weight obtained as $\bar{f} \otimes_\alpha \bar{f}[e]$ (which is equivalent to $f_0 + f[e]_0$ given our $\bar{\alpha}$); t' is the index of the highest topological feature of $\bar{f} * \bar{f}[e]$; and $\bar{f}'$ is the vector of residual topological features after excluding f_0 and $f_{t'}$, that is, $\bar{f} * \bar{f}[e] * [(0, -w'), (t', -1)]$. For example, POPTFEA($[(0,5),(1,-1),(2,1),(6,1)]$) returns $(5, 6, [(1,-1),(2,1)])$; if w has only one topological feature, the residual is $\mathbf{0}$. The residual in all final states of B^r will be $\mathbf{0}$ (no topological features still to be popped).

Graphically, in our running example A^r is:



and the output is B^r:



Reversing B^r yields an acceptor B (still in the sparse tuple vector semiring) which has the same topology as our goal T' and can be trivially mapped to T' in linear time: each arc takes the tropical weight via $\bar{\alpha}$ and has only one topological feature which points to the arc in T containing the required output symbol.

Two-pass Expansion

As mentioned earlier, our algorithm relies on 'cancelling' topological features appearing after the feature they cancel in a given path. In general, consider T a weighted transducer and A it's equivalent automaton with sparse topological features, as described here. A^p is the result of applying standard WFST operations, such as determinization, minimization or shortest path. Assume as a final step that the weights have been pushed towards the final states. It is worth noting the property that: two topological features in a path accepted by A will never get reordered in A^p, although they can appear together on the same edge, as shown in our running example. Indeed, if A^p contains only one single path, all the topological features would appear on the final state.

Let us define a function $d_A(e)$ as the minimum number of edges on any path in A from the start state to $n[e]$ through edge e.

Consider all edges e_i in A and e^p in A^p, with $f[e_i]_i = 1$ and $f[e^p]_i \neq 0$, i.e. we are interested in the topological feature contribution on e^p due to the edge e_i in A. If $d_A(e_i) \leq d_{A^p}(e^p)$ is always satisfied, then EXPANDTFEA will yield the correct answer because the residual at each state, together with the the weight of the current edge, contains all the necessary information to pop the next correct topological feature.

However, many deterministic WFSAs will not exhibit this behaviour (eg. minimised WFSAs), even after pushing the weights towards the final states. For example see this acceptor A:

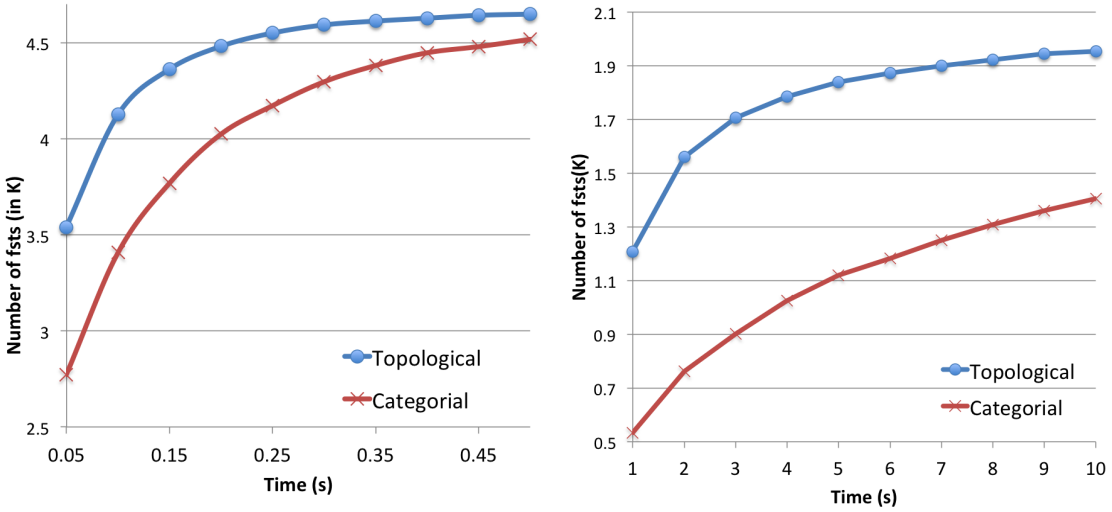
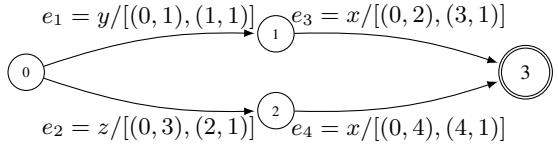
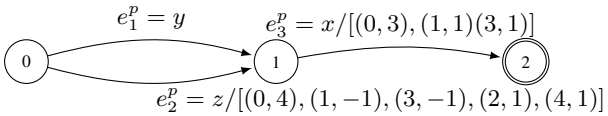


Figure 2: Number of succesfully disambiguated transducers over time for PoS tagged lattices (left) and HiFST lattices (right).



And A^p is a minimised version of A:



As $d_A(e_3) = d_A(e_4) = 2$ and $d_{A^p}(e_2^p) = 1$, the distance test fails for both topological features $(3,-1)(4,1)$. Running B^r=EXPANDTFEA($(A^p)^r$) will not cancel feature $(3,1)$ along the path 'z x' and will pop $(4,1)$ instead, storing the remaining none-$\mathbf{0}$ residual in a final state of B^r.

As mentioned before, two topological features along the same path in A will not reorder in A^p. In this example, as $(4,1)$ appears in edge e_2^p, feature $(2,1)$ must also appear in this edge (or on an earlier edge, in a more complicated machine). In general, any remaining topological features along the path back to the start state of A^p will all be popped *after* their correct edges in B^r. All edges in B^r pass the distance test compared to A^r, the reversed form of A: for all edges e_i with $f[e_i]_i = 1$ in A^r and e^q in B^r such that $f[e^q]_i \neq 0$, $d_{A^r}(e_i) \leq d_{B^r}(e^q)$. Edges in these machines are now reverse sorted, i.e. if an edge e_k precedes an edge $e_{k'}$ on a path, then $k' < k$.

We can perform a second pass with the same algorithm over B, with the only minor modification that t' is now the index of the *lowest* topological feature of $\bar{f} * \bar{f}[e]$. This expands the acceptor correctly. Because correct expansions yield $\mathbf{0}$ residuals on the final states, the algorithm can be trivially modified to trigger the second pass automatically if the residual on any final state is not $\mathbf{0}$.

4 Experiments

We evaluate our algorithm, henceforth called *topological*, in two ways: we empirically contrast disambiguation times against previous work, and then apply it to rescore translation lattices with bilingual neural network models.

4.1 PoS Transducer Disambiguation

We apply our algorithm to the 4,664 NIST English CTS RT Dev04 set PoS tagged lattices used by Sproat el al. (2014); these were generated with a speech recognizer similar to (Soltau et al., 2005) and tagged with a WFST-based HMM tagger. The average number of states is 493. We contrast with the *lexicographic tropical categorial semiring* implementation of Shafran et al. (2011), henceforth referred to as the *categorial* method.

Figure 2 (left) shows the number of disambiguated WFSTs as processing time increases. The topological algorithm proves much faster (and we observe no memory footprint differences). In 50ms it disambiguates 3540 transducers, as opposed to the 2771 completed by the categorial procedure; the slowest WFST to disambiguate takes 230 seconds in the categorial procedure and 60 seconds in our method. Using sparse topological features with our semiring disambiguates all WF-

STs faster in 99.8% of cases.

4.2 Neural Network Bilingual Rescoring

We use the disambiguation algorithm to apply the bilingual neural network language model (BiLM) of Devlin et al. (2014) to the output lattices of the CUED OpenMT12 Arabic-English hierarchical phrase-based translation system[3] using HiFST (de Gispert et al., 2010). We use a development set *mt0205tune* (2075 sentences) and a validation set *mt0205test* (2040 sentences) from the NIST MT02 through MT05 evaluation sets.

The edges in these WFSTs are of the form $t{:}i/w$, where t is the target word, i is the source sentence position t aligns to, and w contains the translation and language model score. HiFST outputs these WFSTs by using a standard hiero grammar (Chiang, 2007) augmented with target side heuristic alignments or *affiliations* to the source (Devlin et al., 2014).

In a rule over source and target words

$$X \rightarrow < s_1\, X\, s_2\, s_3,\, t_1\, X\, t_2 > / 2, 1$$

the feature '$2, 1$' indicates that the target word t_1 is aligned to source word s_2 and that t_2 aligns to s_1. As rules are applied in translation, this information can be used to link target words to absolute positions within the source sentence.

Allowing for admissible pruning, all possible affiliation sequences under the grammar for every translation are available in the WFSTs; disambiguation keeps the best affiliation sequence for each translation hypothesis, which allows the rescoring of very large lattices with the BiLM model.

This disambiguation task involves much bigger lattices than the POS-tagging task: the average number of states of the HiFST lattices is 38,200. Figure 2 (right) shows the number of *mt0205tune* disambiguated WFSTs over time compared to the categorial method. As with the PoS disambiguation task, the topological method is always much faster than the categorial one. After 10 seconds, our method has disambiguated 1953 lattices out of 2075, whereas the categorial method has only finished 1405. The slowest WFST to disambiguate takes 6700 seconds with the categorial procedure, which compares to 1000 seconds in our case.

The BiLM model is trained with NPLM (Vaswani et al., 2013) with a context of 3 source and 4 target words. Lattice rescoring with this model requires a special variation of the standard WFST composition which looks at both input and output labels on a transducer arc; we use KenLM (Heafield, 2011) to retrieve neural network scores for on-the-fly composition. We retune the parameters with Lattice MERT (Macherey et al., 2008) . Results are shown in Table 1. Acknowledging the task differences with respect to (Devlin et al., 2014), we find BLEU gains consistent with rescoring results reported in their Table 5.

[3] See http://www.nist.gov/itl/iad/mig/openmt12results.cfm.

system	mt0205tune	mt0205test
baseline	52.2	51.9
+BiLM	53.0	52.9

Table 1: Translation scores in lower-case BLEU.

5 Conclusions and Related Work

We have described a tagging disambiguation algorithm that supports non-functional WFSTs, which cannot be handled directly by neither WFST determinization (Mohri, 1997) nor WFST disambiguation (Mohri, 2012). We show it is faster than the implementation with a lexicographic-tropical-categorial semiring described by Shafran et al. (2011) and describe a use case in a practical rescoring task of an MT system with bilingual neural networks that yield 1.0 BLEU gain.

Povey et al. (2012) also use a special semiring that allows to map non-functional WFSTs into WFSAs by inserting the tag into a string weight. However, in contrast to our implementation and that of Shafran et al (2011), no expansion into an WFST with aligned input/output is described.

Lexicographic semirings, used for PoS tagging disambiguation (Shafran et al., 2011), have been also shown to be useful in other tasks (Sproat et al., 2014), such as optimized epsilon encoding for backoff language models (Roark et al., 2011), and hierarchical phrase-based decoding with Pushdown Automata (Allauzen et al., 2014).

The tools for disambiguation and WFST composition with bilingual models, along with a tutorial to replicate Section 4.2, are all available at `http://ucam-smt.github.io`.

Acknowledgments

We thank the authors of (Sproat et al., 2014) for generously sharing their PoS-tagging experiments.

References

Cyril Allauzen, Michael Riley, Johan Schalkwyk, Wojciech Skut, and Mehryar Mohri. 2007. OpenFst: A General and Efficient Weighted Finite-State Transducer Library. In *Proceedings of CIAA*.

Cyril Allauzen, Bill Byrne, Adrià de Gispert, Gonzalo Iglesias, and Michael Riley. 2014. Pushdown Automata in Statistical Machine Translation. *Computational Linguistics*, 40(3):687–723.

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.

Adrià de Gispert, Gonzalo Iglesias, Graeme Blackwood, Eduardo R. Banga, and William Byrne. 2010. Hierarchical phrase-based translation with weighted finite-state transducers and shallow-n grammars. *Computational Linguistics*, 36(3):505–533.

Jacob Devlin, Rabih Zbib, Zhongqiang Huang, Thomas Lamar, Richard Schwartz, and John Makhoul. 2014. Fast and Robust Neural Network Joint Models for Statistical Machine Translation. In *Proceedings of ACL*.

Kenneth Heafield. 2011. KenLM: Faster and Smaller Language Model Queries. In *Proceedings of EMNLP*.

Wolfgang Macherey, Franz Och, Ignacio Thayer, and Jakob Uszkoreit. 2008. Lattice-based Minimum Error Rate Training for Statistical Machine Translation. In *Proceedings of EMNLP*.

Mehryar Mohri. 1997. Finite-state transducers in language and speech processing. *Computational Linguistics*, 23:269–311.

Mehryar Mohri. 2009. Weighted automata algorithms. In Manfred Droste, Werner Kuich, and Heiko Vogler, editors, *Handbook of Weighted Automata*, Monographs in Theoretical Computer Science. An EATCS Series, pages 213–254. Springer Berlin Heidelberg.

Mehryar Mohri. 2012. A Disambiguation Algorithm for Finite Automata and Functional Transducers. In *Proceedings of CIAA*, volume 7381, pages 265–277.

Daniel Povey, Mirko Hannemann, Gilles Boulianne, Lukas Burget, Arnab Ghoshal, Milos Janda, Martin Karafiat, Stefan Kombrink, Petr Motlıcek, Yanmin Qian, Korbinian Riedhammer, Karel Vesely, and Ngoc Thang Vu. 2012. Generating Exact Lattices in the WFST Framework. In *Proceedings of ICASSP*.

Brian Roark, Richard Sproat, and Izhak Shafran. 2011. Lexicographic Semirings for Exact Automata Encoding of Sequence Models. In *Proceedings of ACL-HLT*.

Izhak Shafran, Richard Sproat, Mahsa Yarmohammadi, and Brian Roark. 2011. Efficient Determinization of Tagged Word Lattices using Categorial and Lexicographic Semirings. In *Proceedings of ASRU*.

Hagen Soltau, Brian Kingsbury, Lidia Mangu, Daniel Povey, George Saon, and Geoffrey Zweig. 2005. The IBM 2004 Conversational Telephony System for Rich Transcription. In *Proceedings of ICASSP*.

Richard Sproat, Mahsa Yarmohammadi, Izhak Shafran, and Brian Roark. 2014. Applications of Lexicographic Semirings to Problems in Speech and Language Processing. *Computational Linguistics*, 40(4):733–761.

Ashish Vaswani, Yinggong Zhao, Victoria Fossum, and David Chiang. 2013. Decoding with Large-Scale Neural Language Models Improves Translation. In *Proceedings of EMNLP*.

Arabic Diacritization with Recurrent Neural Networks

Yonatan Belinkov and **James Glass**
Massachusetts Institute of Technology
Computer Science and Artificial Intelligence Laboratory
{belinkov, glass}@mit.edu

Abstract

Arabic, Hebrew, and similar languages are typically written without diacritics, leading to ambiguity and posing a major challenge for core language processing tasks like speech recognition. Previous approaches to automatic diacritization employed a variety of machine learning techniques. However, they typically rely on existing tools like morphological analyzers and therefore cannot be easily extended to new genres and languages. We develop a recurrent neural network with long short-term memory layers for predicting diacritics in Arabic text. Our language-independent approach is trained solely from diacritized text without relying on external tools. We show experimentally that our model can rival state-of-the-art methods that have access to additional resources.

1 Introduction

Hebrew, Arabic, and other languages based on the Arabic script usually represent only consonants in writing and do not mark vowels. In such writing systems, diacritics are used for marking short vowels, gemination, and other phonetic units. In practice, diacritics are usually restricted to specific settings such as language teaching or to religious texts. Faced with a non-diacritized word, readers infer missing diacritics based on their prior knowledge and the context of the word in order to resolve ambiguities. For example, Maamouri et al. (2006) mention several types of ambiguity for the Arabic string عـلـم *Elm*, both within and across part-of-speech tags, and at a grammatical level. In practice, a morphological analyzer like MADA (Habash et al., 2009) produces at least 13 different diacritized forms for this word, a subset of which is shown in Table 1.[1]

Word	Gloss
Ealima	he knew
Eulima	it was known
Eal~ama	he taught
Eilomu	knowledge (def.nom)
...	...
EilomK	knowledge (indef.gen)
Ealamu	flag (def.nom)
...	...
EalamK	flag (indef.gen)

Table 1: Possible diacritized forms for عَلَم *Elm*.

The ambiguity in Arabic orthography presents a problem for many language processing tasks, including acoustic modeling for speech recognition, language modeling, text-to-speech, and morphological analysis. Automatic methods for diacritization aim to restore diacritics in a non-diacritized text. While earlier work used rule-based methods, more recent studies attempted to learn a diacritization model from diacritized text. A variety of methods have been used, including hidden Markov models, finite-state transducers, and maximum entropy – see the review in (Zitouni and Sarikaya, 2009) – and more recently, deep neural networks (Al Sallab et al., 2014). In addition to learning from diacritized text, these methods typically rely on external resources such as part-of-speech taggers and morphological analyzers like the MADA tool (Habash and Rambow, 2007). However, building such resources is a labor-intensive task and cannot be easily extended to new languages, dialects, and domains.

[1] Arabic transliteration follows the Buckwalter scheme: http://www.qamus.org/transliteration.htm.

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 2281–2285,
Lisbon, Portugal, 17-21 September 2015.

Diacritic	Transliteration	Transcription
دَ	*a*	/a/
دُ	*u*	/u/
دِ	*i*	/i/
دً	*F*	/an/
دٌ	*N*	/un
دٍ	*K*	/in/
دّ	~	Gemination
دْ	*o*	No vowel

Table 2: Arabic diacritics.

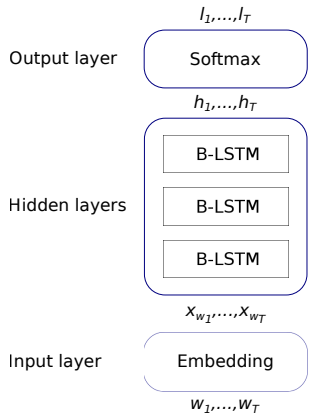


Figure 1: An illustration of our network topology.

In this work, we propose a diacritization method based solely on diacritized text. We treat the problem as a sequence classification task, where each character has a corresponding diacritic label. The sequence is modeled with a recurrent neural network whose input is a sequence of characters and whose output is a probability distribution over the diacritics. Any RNN architecture can be used in this framework; here we focus on long short-term memory (LSTM) networks, which have shown recent success in a number of NLP tasks. We experiment with several architectures and show that we can achieve state-of-the-art results, without relying on external resources. Error analysis demonstrates the benefit of using LSTM over simpler neural networks.

2 Linguistic Background

Languages based on the Arabic script typically employ an *abjad* writing system, where each symbol represents a consonant while vowels and other phonetic units, commonly known as diacritics, are usually omitted in writing. In modern standard and classical Arabic, these include the short vowels *a*, *u*, and *i*, the case endings *F*, *N*, and *K*, the gemination marker ~, and the silence marker *o*.[2] Table 2, modified from (Habash et al., 2007), lists the diacritics. Importantly, the gemination marker ~ can combine with short vowels and case endings (e.g. Table 1, row 3).

3 Approach

We define the following sequence classification task, similarly to (Zitouni and Sarikaya, 2009).

[2]We also include the low-frequency superscript Alif ‘ that is usually ignored due to its limitation to fixed lexical items.

Let $\mathbf{w} = (w_1, ..., w_T)$ denote a sequence of characters, where each character w_t is associated with a label l_t. A label may represent 0, 1, or more diacritics, depending on the language. Assume further that each character w in the alphabet is represented as a real-valued vector x_w. This character embedding may be learned during training or fixed.

Our neural network has the following structure, illustrated in Figure 1:

- Input layer: mapping the letter sequence $\mathbf{w}$ to a vector sequence $\mathbf{x}$.
- Hidden layer(s): mapping the vector sequence $\mathbf{x}$ to a hidden sequence $\mathbf{h}$.
- Output layer: mapping each hidden vector h_t to a probability distribution over labels l.

During training, each sequence is fed into this network to create a prediction for each character. As errors are back-propagated down the network, the weights at each layer are updated. During testing, the learned weights are used in a forward step to compute a prediction over the labels. We always take the best predicted label for evaluation.

Hidden layer Our main system relies on long short-term memory networks (LSTM) (Hochreiter and Schmidhuber, 1997; Graves et al., 2013). Here we describe a single LSTM layer and refer to Graves et al. (2013) for the extension to bidirectional LSTM (B-LSTM) and to multiple layers. The LSTM computes the hidden representation for

	Train	Dev	Test
Words	470K	81K	80K
Letters	2.6M	438K	434K

Table 3: Arabic diacritization corpus statistics.

input x_t with the following iterative process:

$$
\begin{aligned}
i_t &= \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i) \\
f_t &= \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f) \\
c_t &= f_t \odot c_{t-1} + \\
&\quad i_t \odot \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \\
o_t &= \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o) \\
h_t &= o_t \odot \tanh(c_t)
\end{aligned}
$$

where σ is the sigmoid function, $\odot$ is element-wise multiplication, and i, f, o, and c are input, forget, output, and memory cell activation vectors. The crucial element is the memory cell c that is able to store and reuse long term dependencies over the sequence. The W matrices and b bias vectors are learned during training.

Implementation details The input layer maps the character sequence to a sequence of letter vectors, initialized randomly. We also tried initializing with letter vectors trained from raw text with `word2vec` (Mikolov et al., 2013a; Mikolov et al., 2013b), but did not notice any improvement, probably due to the small letter vocabulary size. The input layer also stacks previous and future letter vectors, enabling the model to learn contextual information. We use a letter embedding size of 10 and a window size of 5 characters, so the input size is 110.

We experiment with several types of hidden layers, ranging from one feed-forward layer to multiple B-LSTM layers. We also add a linear projection after the input layer. This has the effect of learning a new representation for the letter embeddings. The output layer is a Softmax over labels:

$$P(l|w_t) = \frac{\exp(y_t[l])}{\sum_{l'} \exp(y_t[l'])}$$

where $y_t = W_{hy}h_t + b_y$ and $y_t[l]$ is the l^{th} element of y_t.

Training is done with stochastic gradient descent with momentum, optimizing the cross-entropy objective function. Layer sizes and other hyper-parameters are tuned on the Dev set. Our implementation is based on Currennt (Weninger et al., 2015).

	DER		
Model	All	End	# params
Feed-forward	11.76	22.90	63K
Feed-forward (large)	11.55	23.40	908K
LSTM	6.98	10.36	838K
B-LSTM	6.16	9.85	518K
2-layer B-LSTM	5.77	9.18	916K
3-layer B-LSTM	5.08	8.14	1,498K

Table 4: Diacrtic error rates (DERs) on the Dev set, over all diacritics and only at word ending.

MaxEnt (only lexical)	8.1
MaxEnt (full)	5.1
3-layer B-LSTM	4.85

Table 5: Results (DER) on the Test set. MaxEnt results from (Zitouni and Sarikaya, 2009)

4 Experiments

Data We extract diacritizied and non-diacritized texts from the Arabic treebank, following the Train/Dev/Test split in (Zitouni and Sarikaya, 2009). Table 3 provides statistics for the corpus.

Every character in our corpus has a label corresponding to 0, 1, or 2 diacritics, in the case of the gemination marker combining with another diacritic. Thus the label set almost doubles. We opted for this formulation due to its simplicity and generalizability to other languages, even though previous work reported improved results by first predicting gemination and then all other diacritics (Zitouni and Sarikaya, 2009).

Results Table 4 shows the results of our models on the Dev set in terms of the diacritic error rate (DER). Clearly, LSTM models perform much better than simple feed-forward networks. To make the comparison fair, we increased the number of parameters in the feed-forward model to match that of the LSTM. In this setting, the LSTM is still much better, indicating that it is far more successful at exploiting the larger parameter set. Interestingly, the bidirectional LSTM works better than a unidirectional one, despite having less parameters. Finally, deeper models achieve the best results.

On the Test set (Table 5), our 3-layer B-LSTM model beats the lexical variant of Zitouni and Sarikaya (2009) by 3.25% DER, a 40% error reduction. Moreover, we outperform their best model, which also used a segmenter and part-of-

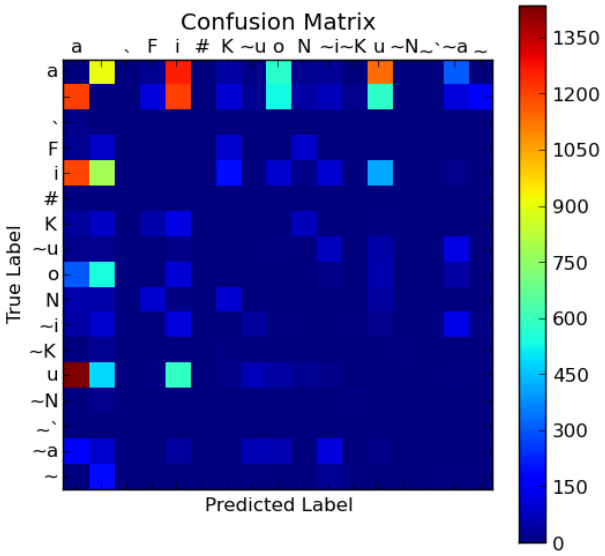


Figure 2: A confusion matrix of errors made by our system. "#" marks word boundary. Best viewed in color.

Model	Diacritization
Gold	AiEotabara Almudiyru AlEAm~u l " Aln~ahAri " juborAn tuwayoniy~ Aan Alt~a$okiylAti AlqaDA}iy~apa jA'at lita-moyiyEi milaf~i maHaT~api Al " Aim . tiy . fiy . "
Feed-forward	AiEotabara Almudiyru AlEAm~u l " Aln~ahAr_ " jaborAn tuwayoniy_ Aan Alt~a$okiylAti AlqaDA}iy~api jA'at litamayiyEi malaf~i maHaT~api Al " A_m . tiy . fiy . "
LSTM	AiEotabara Almudiyru AlEAm~u l_ " Aln~ahAri " juborAn t_w_yoniy_ Ain Alt~a$okiylAti AlqaDA}iy~apa jA'at litamoyiyEa milaf~i maHaT~api Al " Aim . tiy . fiy . "
B-LSTM	AiEotabara Almudiyru AlEAm~u l " Aln~ahAri " juborAn t_wayoniy_ Aan Alt~a$okiylAti AlqaDA}iy~apa jA'at lita-moyiyEi milaf~i maHaT~api Al " Aim . tiy . fiy . "

Figure 3: Sample errors by selected diacritization models. Wrong predicted diacritics are underlined and in red; missing diacritics are noted by underscore. Translation: "The editor of An Nahar, Gebran Tueni, thought that the judicial formations came to dilute the issue of MTV station".

speech tagger. This shows that our model can effectively learn to diacritize without relying on any resources other than diacritized text.

Finally, some studies report work on a Train/Test data split, without a dedicated Dev set (Zitouni et al., 2006; Habash and Rambow, 2007; Rashwan et al., 2011; Al Sallab et al., 2014). We were reluctant to follow this setting so we performed all development on the Dev set of (Zitouni and Sarikaya, 2009). Still, we ran our best model on the Train/Test split and achieved a DER of 5.39% on all diacritics and 8.74% on case endings. The first result is behind the state-of-the-art (Al Sallab et al., 2014) by 2% but the second one is better by 3%. Given that we did not tune the system for this data set, this result is encouraging.

Error Analysis A quantitative analysis of the errors produced by one of our models on the Dev set is shown in Figure 2. The heat map denotes the number of errors produced. The major source of errors comes from confusing the short vowels *a*, *i*, and *u*, among themselves and with no diacritic. This is expected due to the high rate of short vowels in Arabic compared to other diacritics. It also explains why methods that take the confusion matrix into account in their classification algorithm do quite well (Al Sallab et al., 2014).

We also analyzed some errors qualitatively. Figure 3 shows the errors produced by several of our diacritization models on a sample sentence. Interestingly, the simple feed-forward model fails to predict the correct case ending on the word *AlqaDA}iy~ap* ("judicial"), while both LSTM models succeed. This may indicate that LSTM indeed captures the kind of long-distance dependencies that are responsible for case marking. Other errors are more difficult to explain, but note that all models struggle with the proper name *tuwayoniy~* ("Tueini"), which is difficult to solve without external resources.

5 Conclusion

In this work, we develop a recurrent neural network that predicts diacritics in non-diacritized texts. Our model is language agnostic: it is trained solely from diacritized text without relying on additional resources. Using LSTM units, we demonstrate that our model can effectively learn to diacritize Arabic texts and rivals state-of-the-art methods that rely on language-specific tools.

In future work, we intend to incorporate our diacritization system in a speech recognizer. Recent work has shown improvements in Arabic speech recognition by diacritizing with MADA (Al Hanai and Glass, 2014). Since creating such tools is a labor-intensive task, we expect our diacritization approach to promote the development of speech recognizers for other languages and dialects.

Acknowledgments

This research was supported by the Qatar Computing Research Institute (QCRI).

References

Tuka Al Hanai and James Glass. 2014. Lexical Modeling for Arabic ASR: A Systematic Approach. In *Proceedings of INTERSPEECH*.

Ahmad Al Sallab, Mohsen Rashwan, Hazem M. Raafat, and Ahmed Rafea. 2014. Automatic Arabic diacritics restoration based on deep nets. In *Proceedings of the EMNLP 2014 Workshop on Arabic Natural Language Processing (ANLP)*.

Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *Proceedings of ICASSP*.

Nizar Habash and Owen Rambow. 2007. Arabic Diacritization through Full Morphological Tagging. In *Proceedings of HLT-NAACL*.

Nizar Habash, Abdelhadi Soudi, and Timothy Buckwalter. 2007. On Arabic Transliteration. In Abdelhadi Soudi, Günter Neumann, and Antal Van den Bosch, editors, *Arabic Computational Morphology: Knowledge-based and Empirical Methods*, pages 15–22. Springer.

Nizar Habash, Owen Rambow, and Ryan Roth. 2009. MADA+TOKAN: A Toolkit for Arabic Tokenization, Diacritization, Morphological Disambiguation, POS Tagging, Stemming and Lemmatization. In *Proceedings of the Second International Conference on Arabic Language Resources and Tools*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural computation*, 9(8):1735–1780.

Mohamed Maamouri, Ann Bies, and Seth Kulick. 2006. Diacritization: A challenge to Arabic treebank annotation and parsing. In *Proceedings of the British Computer Society Arabic NLP/MT Conference*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient Estimation of Word Representations in Vector Space. In *Proceedings of Workshop at ICLR*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013b. Distributed Representations of Words and Phrases and their Compositionality. In *Proceedings of NIPS*.

M.A.A. Rashwan, M.A.S.A.A. Al-Badrashiny, M. Attia, S.M. Abdou, and A. Rafea. 2011. A Stochastic Arabic Diacritizer Based on a Hybrid of Factorized and Unfactorized Textual Features. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(1):166–175, Jan.

Felix Weninger, Johannes Bergmann, and Björn Schuller. 2015. Introducing CURRENNT: The Munich Open-Source CUDA RecurREnt Neural Network Toolkit. *JMLR*, 16:547–551.

Imed Zitouni and Ruhi Sarikaya. 2009. Arabic Diacritic Restoration Approach Based on Maximum Entropy Models. *Comput. Speech Lang.*, 23(3).

Imed Zitouni, Jeffrey S. Sorensen, and Ruhi Sarikaya. 2006. Maximum Entropy Based Restoration of Arabic Diacritics. In *Proceedings of COLING and ACL*.

Automatic Diacritics Restoration for Hungarian

Attila Novák[1,2] **and Borbála Siklósi**[2]
[1]MTA-PPKE Hungarian Language Technology Research Group,
[2]Pázmány Péter Catholic University, Faculty of Information Technology and Bionics
50/a Práter street, 1083 Budapest, Hungary
{novak.attila, siklosi.borbala}@itk.ppke.hu

Abstract

In this paper, we describe a method based on statistical machine translation (SMT) that is able to restore accents in Hungarian texts with high accuracy. Due to the agglutination in Hungarian, there are always plenty of word forms unknown to a system trained on a fixed vocabulary. In order to be able to handle such words, we integrated a morphological analyzer into the system that can suggest accented word candidates for unknown words. We evaluated the system in different setups, achieving an accuracy above 99% at the highest.

1 Introduction

Due to clumsy mobile device interfaces and reluctance of users to spend too much time entering their message, a great amount of text is generated in a format that lacks the diacritic marks normally used in the orthography of the language the text is written in. Whatever the causes for the missing accents are, NLP applications should be able to restore or generate the accented version of such texts prior to any further syntactic or semantic processing to avoid upstream errors.

In this paper, we aim at solving the problem of restoring accents in Hungarian texts with the combined application of a statistical machine translation system and a morphological analyzer. Our method can be applied to any other languages that have an accurate morphological analyzer.

2 Related work

For Hungarian, there have been some attempts at creating accent restoration systems. Zainkó et al. (2000) and Mihalcea and Nastase (2002) are examples for ML approaches, where the correct places of diacritics are predicted from the immediate grapheme-level context of the unaccented letter with an accuracy of 95%. Thus, unseen words can also be accented, but incorrect forms may also be introduced into the text. Dictionary-based approaches rely on large text corpora and the distribution of the different accented forms. Zainkó et al. (2000) report to have achieved a performance of 98% of accuracy with their dictionary-based method. Nevertheless, their system cannot recognize unseen wordforms quite common in Hungarian. Németh et al. (2000) have implemented a complex text processing system for TTS applications, applying morphological and syntactic analysis. The authors report that the performance of accent restoration depends very much on the performance of the analyzers (achieving 95% accuracy at best). Neither the implementations nor the resources used in these systems have been made publicly available.

A language-independent tool, Charlifter (Scannell, 2011), is based on statistical methods relying on a lexicon, a bigram contextual model and character distributions built from a training corpus. Its performance on Hungarian with its pre-built models is compared to our results in Section 5.

For other languages, similar methods are used. Yarowsky (1994) presents a comprehensive report on corpus-based techniques used for French and Spanish texts. The role of the context is emphasized in this report, however, both word form and accent variations are relatively moderate in the investigated languages compared to Hungarian. The study of Zweigenbaum and Grabar (2002) is also aiming at French, but in the medical domain, which contains a higher ratio of unknown words than general language. In their work, a tagging method is applied in combination with transducers, resulting in a tag sequence corresponding to each letter. The method is successfully (92% precision) applied to single headwords of a medical thesaurus (without exploiting any context). The most similar method to ours is that of Pham et al. (2013), who also applied SMT in order to au-

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 2286–2291,
Lisbon, Portugal, 17-21 September 2015.

tomatically restore accents in Vietnamese texts. In their case, the best results produced an accuracy of 93%. However, their system is augmented with a dictionary, and the distribution of accents and grammatical behaviour are also quite different from Hungarian.

3 Hungarian

Hungarian is an agglutinating language with an orthography that represents compounds as single word forms. These may result in rather complex word forms and words are often composed of long sequences of morphemes. Thus, agglutination and compounding yield a huge number of different word forms.

In Hungarian, umlauts and acute accents are used as diacritics for vowels. Acute accents mark long vowels, while umlauts are used to indicate the frontness of rounded vowels *o*→*ö* [o→ø] and *u*→*ü* [u→y], like in German. A combination of acutes and umlauts is the double acute diacritic to mark long front rounded vowels *ő* [øː] and *ű* [yː]. Long vowels generally have essentially the same quality as their short counterpart (*i-í, ü-ű, u-ú, ö-ő, o-ó*). The long pairs of the low vowels *a* [ɔ] and *e* [ɛ], on the other hand, also differ in quality: *á* [aː] and *é* [eː]. There are a few lexicalized cases where there is a free variation of vowel length without distinguishing meaning, e.g. *hova*∼*hová* 'where to'. In most cases, however, the meaning of differently accented variants of a word is quite different. Table 1 shows all the possible unaccented-accented pairs of vowels in Hungarian together with their distribution in a corpus of 1 804 252 tokens.

a	a: 70.33%; á: 29.66%
e	e: 73.40% é: 26.59%
i	i: 86.04% í: 13.95%
o	o: 55.41% ó: 14.65% ö: 15.82% ő: 14.10%
u	u: 46.96% ú: 12.72% ü: 29.98% ű: 10.32%

Table 1: Possible accent variations in Hungarian

4 Method

In this research, we considered the problem of accent restoration as a translation task, where the source language is the unaccented version, and the target language is accented Hungarian. Since it is easy to come up with a parallel training corpus for this task, methods of SMT can be applied.

In our experiments, we used Moses (Koehn et al., 2007), a widely used SMT toolkit for building the translation models and performing decoding, and SRILM (Stolcke et al., 2011) to build the necessary language models. Moses was used with its default configuration settings and monotone decoding (i.e. reordering was not allowed), and without the alignment step, which was not needed in our case.

4.1 The baseline setup

In the baseline setup, only the translation and language models built from the training corpus were used. The input for the decoder was Hungarian raw texts with all the accents removed. The translation model contained only unigram phrases (larger n-grams were also tried, but did not change the results) and the language model contained phrases up to 5 grams. Thus, the translation model was responsible for predicting the distribution of accented forms and the language model exploited contextual information.

Another baseline was also created in order to monitor the effect of the SMT system. In this second baseline, each unaccented word form was replaced by its most frequent accented form in the training set.

4.2 Incorporating a morphological analyzer

In order to be able to restore accents in unseen words as well, a Hungarian morphological analyzer (Prószéky and Kis, 1999; Novák, 2003) was integrated. A special version of the analyzer was created that directly maps unaccented word forms to their possible accented variants while also marking morpheme boundaries and adding morphosyntactic category tags. The segmentation marks (e.g. compound and derivational suffix boundaries) and the tags are used when we assign a score to the accented candidates. We also reanalyze accented forms to retrieve lemmas not directly returned by the accenting analyzer. In our test set of 1 804 252 tokens, about 1% of the words were not found in the translation model even in the case of the largest, 440 million words, training set. Table 2 shows the ratio of unknown words (OOV) as a function of the size of the training set used for building the phrase table.

For these unknown words, all possible correct accented candidates were generated by the morphological analyzer. These candidates were then fed to the Moses decoder using its `-xml-input`

train	sentences	M words	OOV in test
100K	100 000	1.738	9.63%
1000K	1 000 000	18.078	3.44%
5000K	5 000 000	89.907	1.23%
10M	10 000 000	180.644	1.68%
ALL	24 048 302	437.559	0.81%

Table 2: Ratio of OOV after building a translation model from a training set of a certain size

parameter. In order to be able to use this feature of the decoder, a probability for each candidate form had to be estimated. First, we assumed uniform distribution among the candidates. However, this approach assigned the same probability to the most common and the most nonsensical (although grammatical) candidates as well. Thus, in some cases these forms showed up in the results. In order to avoid the system to make such errors, a more sophisticated distribution was estimated for the candidate set. For this, we applied a linear regression model based on corpus frequency data determined for the lemma and other features of the candidate word (since the actual wordform was not present in the corpus). Thus, for each candidate, its lemma frequency (LEM), the number of productively applied compounding (CMP), the number of productively applied derivational affixes (DER), and the frequency of the inflectional suffix sequence returned by the analysis were determined. Compounding and derivation were penalized (i.e. they were given a negative sign), because the morphological analyzer could suggest some nonsensical, though grammatical compound or derived forms. Sometimes such forms could be the correct ones, but the more productive compounding and derivation there is in a word, the lower score it should get. On the other hand, the frequencies of the lemma and the inflectional pattern should increase the score of a candidate, thus these components were given positive weights. Based on these components, a score was assigned to each candidate based on Formula (1).

$$score = -\lambda_c \#CMP - \lambda_d \#DER + log_{10}LEM + \lambda_i log_{10}INF + MS \quad (1)$$

, where

$$MS = \begin{cases} |minscore| + 1 & if \quad minscore \leq 0 \\ 0 & otherwise \end{cases} \quad (2)$$

The MS component was used to scale up the scores by adding $|minscore| + 1$, i.e. the lowest score received for any candidate in the actual candidate set in order to evade negative scores. The λ weights were set by the `mert` tuning of the Moses system. We used a separate development set for this, on which we observed the distribution of compounds, derivational and inflectional suffixes in OOV words analyzed by the morphology and from which we sampled 1000 words approximating the observed distributions. The target of the optimization in the `mert` tuning was the accuracy of the system on these words, resulting in the optimal values for each λ. Even though, in linear regression, it is standard to use an additional bias weight, we did not find it necessary, because we did not need to bring our estimates in sync with estimates from other sources. And assuming one factor to have a fixed unit value was just another simplification that would not affect the overall ranking, just its scaling.

Even though, following an appropriate scaling of the scores, the ranked candidates could be used the same way as the entries in the translation table, the system would never select any accented form other than the most probable one, since the language model does not include any of these forms. Thus, only the candidate with the highest relative score was made available to the system.

5 Experiments and results

In our experiments, the Hungarian webcorpus (Halácsy et al., 2004) was used for training and testing purposes. A set of 100 000 sentences were separated from the corpus as the test set, and another 100 000 sentences were used as a development set. The rest were used for training in different settings. The size of each training set is shown in Table 2.

We evaluated the performance on all the 1 804 252 tokens of the test set (56.84% correct without accent) and on a subset of 1 472 200 words that included any vowels (47.09% correct without accent). The experiments were then performed for the baseline system using the most frequent form (BL-FREQ), for the baseline SMT system (BL-SMT) and for the one augmented by the morphology with the first-ranked candidate (RANK). Table 3 shows the detailed results for the smallest and largest training sets for all words (ALL) and for words that include vowels (VOWEL). It can be seen that the precision of the system is only

	100K			ALL		
system	prec	rec	acc	prec	rec	acc
BL-FREQ-ALL	98.25	82.82	92.34	98.37	96.26	98.13
BL-FREQ-VOW	98.25	82.82	90.62	98.37	96.26	97.71
BL-SMT-ALL	99.03	83.88	92.91	99.09	97.36	98.72
BL-SMT-VOW	99.03	83.88	**91.31**	99.09	97.36	**98.44**
RANK-ALL	98.81	98.08	98.99	99.01	98.56	99.23
RANK-VOW	98.82	98.08	**98.77**	99.02	98.56	**99.06**

Table 3: Performance results for each experimental settings and training size

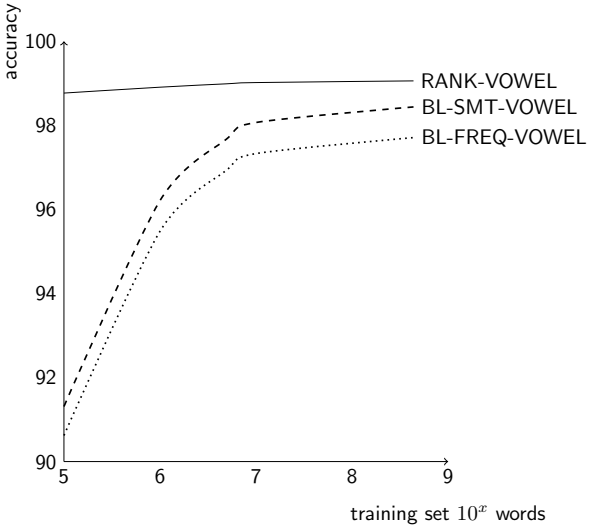


Figure 1: Accuracy as a function of the size of the training set for each system, measured on words containing vowels.

slightly improved when increasing the size of the training corpus, but the values of recall and accuracy do dramatically improve in the case of the baseline system. However, the integration of the suggestions of the morphology can make up for the lack of information due to the small training set improving recall a great deal while only slightly affecting precision. Even for the biggest 437.6M-word training corpus, incorporating the morphological analyzer with ranking yielded a relative error rate reduction of 39.74%, reducing the word error rate from 1.56% to 0.94%. For the smallest 1.74M-word training corpus tested, the relative error rate reduction was 85.85%. The system including the morphological analyzer performs better even with the smallest training corpus in terms of word accuracy than the baseline Moses system with the biggest corpus. Figure 1 shows the learning curves for each system with accuracy as a function of training set size.

Comparing our results to those we obtained using Charlifter (89.75% with most frequent accented form baseline, 90.00% with the lexicon-lookup+bigram contextual model and 93.31% with lookup+bigram context+character-n-gram-based model), the results reveal that both the contextual model in the SMT system improves accuracy better than the bigram context model of Charlifter, and the performance boost we get by incorporating morphology vastly exceeds the accuracy improvement yielded by the incorporation of the character-n-gram-based model used in Charlifter.

6 Error analysis

We performed a detailed error analysis on a 5000-sentence (87786-token) fragment of the test set. The results of the error analysis are presented in Table 4.

The detailed analysis showed that 14.7% of mismatches between the original and the system output is in fact not due to the latter being erroneous. 3.55% are equivalent forms , while the rest is correct in the output and erroneous in the reference, i.e. the system corrected errors in the original.

Another part of the reference (17.91%) is likewise erroneous, however, since the error in these cases was not in the accents, the system was not able to correct it. Missing or substituted letters are the most common mistakes (10.81%), and further 6.42% of the errors is due to punctuation errors in the original.

About $2/3$ of the mismatches are real errors. 5.57% of these could be attributed to the stem of the word missing from the database of the morphological analyzer. In 3.55% of the cases, the system transforms a name to a more frequent word: sometimes to another name, but more often to some common frequent word. A similar case is when some common noun is transformed to a more frequent name (another 1.35%). The number of these errors could be reduced to some extent by making the system rely on case information (in the case of some proper name-common noun ambiguities), however this could make the system perform worse elsewhere due to increased data sparseness. 2.20% of the errors is due to errors in the training corpus. Since rare word forms are quite frequent in Hungarian, the chances are high that a specific form is more often mistyped

Mismatch type	Ratio	Examples
Output correct	**14.70%**	
Equivalent forms	3.55%	lévő→levő fele→felé áhá→aha periférikus→periferikus
Corrected erroneous name	1.01%	USÁ-ban→USA-ban Szóladon→Szóládon
Other corrected erroneous	10.14%	un.→ún. kollegánk→kollégánk lejto→lejtő lathato→látható
Real errors	**67.40%**	
Missing from MA	5.57%	hemokromatózis-gén→hemokromatozis-gen
Correct name to erroneous output	3.55%	MIG→míg Bösz→Bősz Ladd→Ládd Márton→Marton
Other correct original to some erroneous form	2.20%	megőrzést→megorzést routeréhez→routerehez
Other correct original to contextually inadequate name	1.35%	logó→logo eperjeskein→eperjeskéin
Other correct original to some contextually inadequate form	51.01%	még→meg termek→termék gépét→gépet címét→címet vágyók→vagyok érméket→érmeket képé→képe
Original is a filename or a url containing accents	3.72%	latok→látok viz→víz szantok→szántók telepok→telepók felhasználó@profinter.hu→felhasznalo@profinter.hu www.valamicég.hu→www.valamiceg.hu
Uncorrected error in original	**17.91%**	
Punctuation error in original	6.42%	közalk.tan→kozalk.tan 1922.évi→1922.evi
Hyphenation error in original	0.68%	bemuta-tásra→bemuta-tasra
Other error in original	10.81%	véri→veri ra→rá gonolkozásában→gonolkozasaban imátkoztok→imatkoztok hirújsághoz→hirujsasghoz változaban→változabán környezetkíméli→kornyezetkimeli

Table 4: Analysis of mismatches between the system output and the input on a 5000-sentence test sample

than not (this is especially true for word forms that occur only once in the training data). 3.72% of the errors in the analyzed test data was due to either transforming arbitrary unaccented letter sequences used as file names in the text being transformed to some meaningful words or to accented words being used in an url in the original text.

The most common error (51.01% of all mismatches) is the case where the system is simply unable to correctly disambiguate the word in context, and this is not due to some other error or information loss. Interestingly, more than half (51%) of these errors belong to a single type where the system is unable to distinguish a possessive and a non-possessive form of the same nominal lemma: *gyereket∼gyerekét* 'the child (accusative)' vs.'his/her child (accusative)', *gyereken∼gyerekén* 'on/about the child' vs. 'on/about his/her child', and *gyereke∼gyereké* 'his/her child' vs.'(belongs to a) child' (anaphoric possessive).

Another 26% of the mismatches is due to a similar problem concerning verbs. In Hungarian, transitive verbs agree with their object in definiteness. Certain past, present and conditional verb forms differing in definiteness are only distinguished by an accent: *hajtottak∼hajtották* 'they drove' vs. 'they drove it'; *hajtanak∼hajtanák* 'they drive' vs. 'they would drive it'; *hajtana∼hajtaná* 'he/she would drive' vs. 'he/she would drive it'.

A factored model could in theory improve the recognition of these structures. It is questionable however, whether the improvement would justify the costs.

7 Conclusion

We have described a method to restore accents in Hungarian texts. The baseline method using only a fixed training corpus to build translation and language models for a statistical machine translation system, which is limited to handling word forms present in the training corpus achieved an accuracy of 98.44% at best. In order to process unknown words, a morphological analyzer was integrated to produce accented candidates for these unknown words as well, resulting in an improved accuracy of 99.06%. This performance could only be achieved by a system that is able to produce correct word forms and takes context into account. Our method can be applied to any other languages for which a training corpus and a morphological analyzer are available.

References

Péter Halácsy, András Kornai, László Németh, András Rung, István Szakadát, and Viktor Trón. 2004. Creating open language resources for hungarian. In *LREC*. European Language Resources Association.

P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin, and E. Herbst. 2007. Moses: Open Source Toolkit for Statistical Machine Translation. In *Proceedings of the ACL 2007 Demo and Poster Sessions*, pages 177–180, Prague. Association for Computational Linguistics.

Rada Mihalcea and Vivi Nastase. 2002. Letter level learning for language independent diacritics restoration. In *Proceedings of the 6th Conference on Natural Language Learning - Volume 20*, COLING-02, pages 1–7, Stroudsburg, PA, USA. Association for Computational Linguistics.

Géza Németh, Csaba Zainkó, László Fekete, Gábor Olaszy, Gábor Endrédi, Péter Olaszi, Géza Kiss, and Péter Kis. 2000. The design, implementation, and operation of a Hungarian e-mail reader. *International Journal of Speech Technology*, 3(3-4):217–236.

Attila Novák. 2003. What is good Humor like? [Milyen a jó Humor?]. In *I. Magyar Számítógépes Nyelvészeti Konferencia*, pages 138–144, Szeged. SZTE.

Luan-Nghia Pham, Viet-Hong Tran, and Vinh-Van Nguyen. 2013. Vietnamese text accent restoration with statistical machine translation. In *Proceedings of the 27th Pacific Asia Conference on Language, Information, and Computation (PACLIC 27)*, pages 423–429. Department of English, National Chengchi University.

Gábor Prószéky and Balázs Kis. 1999. A unification-based approach to morpho-syntactic parsing of agglutinative and other (highly) inflectional languages. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, ACL '99, pages 261–268, Stroudsburg, PA, USA. Association for Computational Linguistics.

Kevin P. Scannell. 2011. Statistical unicodification of african languages. *Language Resources and Evaluation*, 45(3):375–386.

Andreas Stolcke, Jing Zheng, Wen Wang, and Victor Abrash. 2011. SRILM at sixteen: Update and outlook. In *Proc. IEEE Automatic Speech Recognition and Understanding Workshop*, Waikoloa, Hawaii, December.

D. Yarowsky. 1994. A comparison of corpus-based techniques for restoring accents in spanish and french text. In *Proceedings of the 2nd Annual Workshop on Very Large Text Corpora*, pages 19—-32, Las Cruces.

Cs. Zainkó, G. Németh, G. Olaszy, and G. Gordos. 2000. Eljárás adott nyelven ékezetes betűk használata nélkül készített szövegek ékezetes betűinek visszaállítására.

Pierre Zweigenbaum and Natalia Grabar. 2002. Accenting unknown words in a specialized language. In Stephen Johnson, editor, *ACL Workshop on Natural Language Processing in the Biomedical Domain*, pages 21–28. ACL.

Morphological Analysis for Unsegmented Languages using Recurrent Neural Network Language Model

Hajime Morita[1,2] **Daisuke Kawahara**[1] **Sadao Kurohashi**[1,2]
[1] Kyoto University [2] CREST, Japan Science and Technology Agency
{hmorita, dk, kuro}@i.kyoto-u.ac.jp

Abstract

We present a new morphological analysis model that considers semantic plausibility of word sequences by using a recurrent neural network language model (RNNLM). In unsegmented languages, since language models are learned from automatically segmented texts and inevitably contain errors, it is not apparent that conventional language models contribute to morphological analysis. To solve this problem, we do not use language models based on raw word sequences but use a semantically generalized language model, RNNLM, in morphological analysis. In our experiments on two Japanese corpora, our proposed model significantly outperformed baseline models. This result indicates the effectiveness of RNNLM in morphological analysis.

1 Introduction

In contrast to space-delimited languages like English, word segmentation is the first and most crucial step for natural language processing (NLP) in unsegmented languages like Japanese, Chinese, and Thai (Kudo et al., 2004; Kaji and Kitsuregawa, 2014; Shen et al., 2014; Kruengkrai et al., 2006). Word segmentation is usually performed jointly with related analysis: POS tagging for Chinese, and POS tagging and lemmatization (analysis of inflected words) for Japanese. Morphological analysis including word segmentation has been widely and actively studied, and for example, Japanese word segmentation accuracy is in the high 90s. However, we often observe that strange outputs of downstream NLP applications such as machine translation and question answering come from incorrect word segmentations.

For example, the state-of-the-art and popular Japanese morphological analyzers, JUMAN (Kurohashi and Kawahara, 2009) and MeCab (Kudo et al., 2004) both analyze "外国人参政権 (foreigner's right to vote)" not into the correct segmentation of (1a), but into the incorrect and awkward segmentation of (1b).

(1) a. 外国 / 人 / 参政 / 権
foreigner right to vote
b. 外国 / 人参 / 政権
foreign carrot regime

JUMAN is a rule-based morphological analyzer, defining word-to-word (including inflection) connectivities and their scores. MeCab is a supervised morphological analyzer, learning the probabilities of word/POS/inflection sequence from an annotated corpus of tens of thousands of sentences. Both systems, however, cannot realize semantically appropriate analysis, and often produce totally strange outputs like the above.

This paper proposes a semantically appropriate morphological analysis method for unsegmented languages using a language model. For unsegmented languages, morphological analysis and language modeling form a chicken-and-egg problem. That is, if high-quality morphological analysis is available, we can learn a high-quality language model from a morphologically analyzed large corpus. On the other hand, if a high-quality language model is available, we can achieve high-quality morphological analysis by looking for a segmented word sequence with a large language model score. However, even if we learn a language model from a corpus analyzed by a certain level of morphological analyzer, the language model is affected by the analysis errors of the morphological analyzer and it is no practical use for the improvement of the morphological analyzer. A language model trained by incorrectly segmented "外国 (foreign)/人参 (carrot)/政権 (regime)" just supports that incorrect segmentation.

The point of the paper is that we have tackled the chicken-and-egg problem, not by using a lan-

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 2292–2297,
Lisbon, Portugal, 17-21 September 2015.

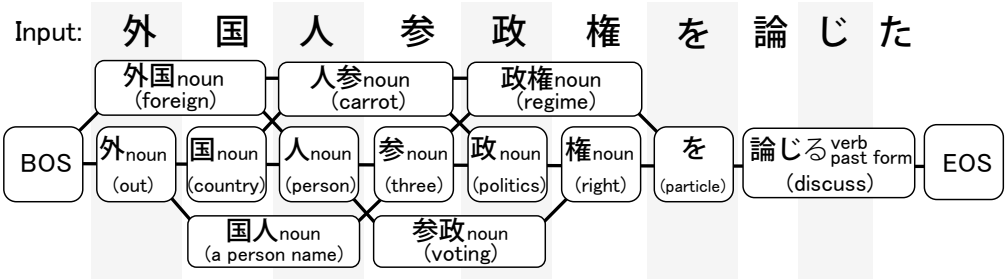


Figure 1: An example of a word lattice.

guage model of raw word sequences, but by using a semantically generalized language model based on word embeddings, RNNLM (Recurrent Neural Network Language Model) (Mikolov et al., 2010; Mikolov et al., 2011). The RNNLM is trained on an automatically analyzed corpus of ten million sentences, which possibly includes incorrect segmentations such as "外国 (foreign)/人参 (carrot)/政権 (regime)." However, on semantically generalized level, it is an unnatural semantic sequence like *nation vegetable politics*. Since the state-of-the-art morphological analyzer achieves the high accuracy, it does not often produce incorrect analyses which support such a semantically strange sequence. This would prefer analysis toward semantically appropriate word sequences. When a morphological analyzer utilizes such a generalized and reasonable language model, it can penalize strange segmentations like "外国 (foreign)/人参 (carrot)/政権 (regime)," leading to better accuracy.

We furthermore retrain RNNLM using an annotated corpus of manually segmented 45k sentences, which further improves morphological analysis.

2 Related Work

There have been several studies that have integrated language models into morphological analysis. Wang et al. (2011) improved Chinese word segmentation and POS tagging by using N-gram features learned from an automatically segmented corpus. However, since the auto-segmented corpus inevitably contains segmentation errors, frequent N-grams are not always correct and thus this problem might affect the performance of morphological analysis. They also divided N-gram frequencies into three binned features: high-frequency, middle-frequency and low-frequency. Such coarse features cannot express slight differences in the likelihood of language models.

Kaji and Kitsuregawa (2014) used a bigram language model feature for Japanese word segmentation and POS tagging. Their objective of using a language model is to normalize informally spelled words in microblogs. Therefore, their objective is different from ours.

Some studies have used character-based language models for Chinese word segmentation and POS tagging (Zheng et al., 2013; Liu et al., 2014). Although their approaches have no drawbacks of learning incorrect segmentations, they only capture more local information than word-based language models.

Word embeddings have been also used for morphological analysis. Neural network based models have been proposed for Chinese word segmentation and POS tagging (Pei et al., 2014) or word segmentation (Mansur et al., 2013). These methods acquire word embeddings from a corpus, and then use them as the input of the neural networks. Our proposed model learns word embeddings via RNNLM, and these embeddings are used for scoring word transitions in morphological analysis. Our usage of word embeddings is different from the previous studies.

3 Proposed Method

We propose a new morphological analysis model that considers semantic plausibility of word sequences by using RNNLM. We integrate RNNLM into morphological analysis (Figure 2). We train the RNNLM using both an automatically analyzed corpus and a manually labeled corpus.

3.1 Recurrent Neural Network Language Model

RNNLM is a recurrent neural network language model (Mikolov et al., 2010), which outputs a probability distribution of the next word, given the embedding of the last word and its context. We

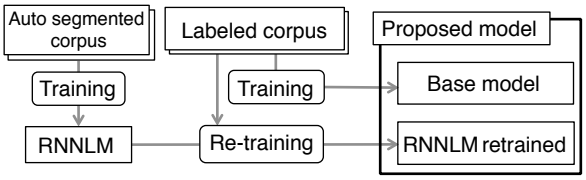


Figure 2: Workflow for training RNNLM and base model.

employ the RNNME language model[1] proposed by (Mikolov et al., 2011; Mikolov, 2012) as the implementation of RNNLM. The RNNME language model has direct connections from the input layer of the recurrent neural network to the output layer, which act as a maximum entropy model and avoid to waste a lot of parameters to describe simple patterns. Hereafter, we refer to the RNNME language model simply as RNNLM.

To train RNNLM, we use a raw corpus of 10 million sentences from the web corpus (Kawahara and Kurohashi, 2006). These sentences are automatically segmented by JUMAN (Kurohashi and Kawahara, 2009). The training of RNNLM is based on lemmatized word sequences without POS tags.

The trained model contains errors caused by an automatically analyzed corpus. We retrain RNNLM using a manually labeled corpus after training RNNLM using the automatically analyzed corpus as shown in Figure 2. The retraining aims to cope with errors related to function word sequences.

3.2 Base Model

For our base model, we adopt a model for supervised morphological analysis, which performs segmentation, lemmatization and POS tagging jointly. We train this model using a tagged corpus of tens of thousands of sentences that contain gold segmentations, lemmas, inflection forms and POS tags. To predict the most probable sequence of words with lemmas and POS tags given an input sentence, we execute the following procedure:

1. Look up the string of the input sentence using a dictionary.
2. Make a word lattice.
3. Search for the path with the highest score from the lattice.

[1]RNNME is the abbreviation of Recurrent Neural Network trained jointly with Maximum Entropy model.

Figure 1 illustrates the constructed lattice during the procedure. At the dictionary lookup step, we use the basic dictionary of JUMAN and an additional dictionary comprising 0.8 million words, both of which have lemma, POS and inflection information. The additional dictionary mainly consists of itemizations in articles and article titles in Japanese Wikipedia.

We define the scoring function as follows:

$$\text{score}_B(\mathbf{y}) = \Phi(\mathbf{y}) \cdot \vec{w}, \quad (1)$$

where $\mathbf{y}$ is a tagged word sequence, $\Phi(\mathbf{y})$ is a feature vector for $\mathbf{y}$, and $\vec{w}$ is a weight vector. Each element in $\vec{w}$ gives a weight to its corresponding feature in $\Phi(\mathbf{y})$. We use the unigram and the bigram features composed from word base form, POS and inflection described in Kudo et al. (2004). We also use additional lexical features such as character type, and trigram features used in Zhang and Clark (2008). To learn the weight vector, we adopt exact soft confidence-weighted learning (Wang et al., 2012).

To consider out-of-vocabulary (OOV) words that are not found in the dictionary, we automatically generate words at the lookup step by segmenting the input string by character types[2]. For training, we regard words that are not found in the dictionary but found in the training corpus as OOV words to learn their weights.

3.3 RNNLM Integrated Model

Based on retrained RNNLM, we calculate an RNNLM score ($\text{score}_R(\mathbf{y})$) to be integrated into the base model. The RNNLM score is defined as the log probability of the next word given its context (path). Here, the score for an OOV word is given by the following formula:

$$-C_p - L_p \cdot \text{length}(n), \quad (2)$$

where C_p is a constant penalty for OOV words, L_p is a factor for the character length penalty, and $\text{length}(n)$ returns the character length of the next word n. This formula is defined to penalize longer words, which are likely to produce segmentation errors.

We then integrate the RNNLM score into the base model using the following equation:

$$\text{score}_I(\mathbf{y}) = (1-\alpha)\text{score}_B(\mathbf{y}) + \alpha\ \text{score}_R(\mathbf{y}), \quad (3)$$

[2]Japanese has three types of characters: Kanji, Hiragana and Katakana.

where α is an interpolation parameter that is tuned on development data.

For decoding, we employ beam search as used in Zhang and Clark (2008). Since the possible context (paths in the word lattice) considered in RNNLM falls into combinatorial explosion in morphological analysis, we keep only probable context candidates inside the beam. That is, each node keeps candidates inside the beam width. Each candidate has a vector representing context, and two words of history. The recurrent model makes decoding harder than non-recurrent neural network language models. However, we use RNNLM because the model outperforms other NNLMs (Mikolov, 2012) and the result suggests that the model is more likely to capture semantic plausibility. Since a sentence rarely contains ambiguous and semantically appropriate word sequences, we think that beam search with enough beam size is able to keep the ambiguous candidates of word sequences. In the case of non-recurrent NNLMs and the base model, which uses trigram features, we can conduct exact decoding using the second-order Viterbi algorithm (Thede and Harper, 1999).

4 Experiments

4.1 Experimental Settings

In our experiments, we used the Kyoto University Text Corpus (Kawahara et al., 2002) and Kyoto University Web Document Leads Corpus (Hangyo et al., 2012) as manually tagged corpora. We randomly chose 2,000 sentences from each corpus for test data, and 500 sentences for development data. We used the remaining part of the corpora as training data to train our base model and retrain RNNLM. In total, we used 45,000 sentences for training.

For comparative purposes, we used the following four baselines: the Japanese morphological analyzer JUMAN, the supervised morphological analyzer MeCab, the base model, and a model using a conventional language model. For this language model, we built a trigram language model with Kneser-Ney smoothing using SRILM (Stolcke, 2002) from the same automatically segmented corpus. The language model is modified to have an interpolation parameter α and length penalty for OOV, L_p.

We set the beam width to 5 by preliminary experiments. We also set a constant penalty for OOV words (C_p) as 5, which is the default value in the implementation of Mikolov et al. (2011). We tuned the parameters of our proposed model and the baseline model (α and L_p) and the parameters of language models using grid search on the development data. We set α = 0.3, L_p =1.5 for the proposed model (" Base + $\text{RNNLM}_{\text{retrain}}$").[3]

We measured the performance of the baseline models and the proposed model by F-value of word segmentation and F-value of joint evaluation of word segmentation and POS tagging. We calculated F-value for the two corpora (news and web) and the merged corpus (all).

We used the bootstrapping method (Zhang et al., 2004) to test statistical significance between proposed models and other models. Suppose we have a test set T that includes N sentences. The method repeatedly creates M new test sets by resampling N sentences with replacement from T. We calculate the F-value of each model on $M + 1$ test sets including T, and then we have $M + 1$ score differences. From the scores, we calculate the 95% confidence interval. If the interval does not overlap with zero, the two models are considered as statistically significantly different. In our evaluation, M is set to 2,000.

4.2 Results and Discussions

Table 1 lists the results of our proposed model and the baseline models. Our proposed model ("Base + $\text{RNNLM}_{\text{retrain}}$") significantly outperforms all the baseline models and "Base + RNNLM," which does not use retraining. In particular, we achieved a large improvement for segmentation. This can be attributed to the use of RNNLM that was learned based on lemmatized word sequence without POS tags.

"Base + SRILM" segmented the example described in Section 1 ("外国人参政権") into the incorrect segmentation "外国/人参/政権" in the same way as JUMAN. This segmentation error was caused by errors in the automatically segmented corpus that was used to train the language model. Our proposed model can correctly segment this example if a proper context is available by semantically capturing word transitions using RNNLM.

The base model, JUMAN and "Base + SRILM" incorrectly segmented "健康 (healthy)/など (etc.)/

[3] We set α = 0.1, L_p = 2.0 for "Base + RNNLM", and α = 0.3, L_p = 0.5 for "Base + SRILM."

	Segmentation (news)	Seg + POS (news)	Segmentation (web)	Seg + POS (web)	Segmentation (all)	Seg + POS (all)
JUMAN	98.92	98.47	98.20	97.64	98.64	98.14
MeCab	99.07	98.58	98.22	97.51	98.74	98.16
Base model	98.94	98.46	97.71	96.90	98.46	97.85
Base + SRILM	98.94	98.40	98.13	97.33	98.62	97.98
Base + RNNLM	99.06	98.59	98.17	97.45	98.71	98.14
Base + $\text{RNNLM}_{\text{retrain}}$	$\mathbf{99.15}^*$	$\mathbf{98.70}^*$	$\mathbf{98.37}^*$	$\mathbf{97.68}^*$	$\mathbf{98.84}^*$	$\mathbf{98.30}^*$

Table 1: Results for test datasets. $*$ means the score of "Base + $\text{RNNLM}_{\text{retrain}}$" is significantly improved from that of all other models.

の (of)/点 (point)/で (in)/……" (in terms of health and so on) into "健康 な(healthy)/どの(any)/点 (point)/で (in)/……." Although this segmentation can be grammatically accepted, it is difficult to semantically interpret this word sequence. Our proposed model can correctly segment this example because RNNLM learns semantically plausible word sequences.

5 Conclusion

In this paper, we proposed a new model for morphological analysis that is integrated with RNNLM. We trained RNNLM on an automatically segmented corpus and tuned on a manually tagged corpus. The proposed model was able to significantly reduce errors in the base model by capturing semantic plausibility of word sequences using RNNLM. In the future, we will design features derived from RNNLM models, and integrate them into a unified learning framework. We also intend to apply our method to unsegmented languages other than Japanese, such as Chinese and Thai.

References

Masatsugu Hangyo, Daisuke Kawahara, and Sadao Kurohashi. 2012. Building a diverse document leads corpus annotated with semantic relations. In *Proceedings of the 26th Pacific Asia Conference on Language, Information, and Computation*, pages 535–544.

Nobuhiro Kaji and Masaru Kitsuregawa. 2014. Accurate word segmentation and POS tagging for japanese microblogs: Corpus annotation and joint modeling with lexical normalization. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 99–109, Doha, Qatar. Association for Computational Linguistics.

Daisuke Kawahara and Sadao Kurohashi. 2006. Case frame compilation from the web using high-performance computing. In *Proceedings of the 5th International Conference on Language Resources and Evaluation*, pages 1344–1347.

Daisuke Kawahara, Sadao Kurohashi, and Kôiti Hasida. 2002. Construction of a Japanese relevance-tagged corpus. In *Proceedings of the Third International Conference on Language Resources and Evaluation (LREC-2002)*, Las Palmas, Canary Islands - Spain, May. European Language Resources Association (ELRA). ACL Anthology Identifier: L02-1302.

Canasai Kruengkrai, Virach Sornlertlamvanich, and Hitoshi Isahara. 2006. A conditional random field framework for Thai morphological analysis. In *Proceedings of LREC*, pages 2419–2424.

Taku Kudo, Kaoru Yamamoto, and Yuji Matsumoto. 2004. Applying conditional random fields to Japanese morphological analysis. In *Proceedings of the Conference on Emprical Methods in Natural Language Processing (EMNLP 2004)*, volume 2004.

Sadao Kurohashi and Daisuke Kawahara, 2009. *Japanese Morphological Analysis System JUMAN 6.0 Users Manual*. `http://nlp.ist.i.kyoto-u.ac.jp/EN/index.php?JUMAN`.

Xiaodong Liu, Kevin Duh, Yuji Matsumoto, and Tomoya Iwakura. 2014. Learning character representations for Chinese word segmentation. In *NIPS 2014 Workshop on Modern Machine Learning and Natural Language Processing*.

Mairgup Mansur, Wenzhe Pei, and Baobao Chang. 2013. Feature-based neural language model and Chinese word segmentation. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 1271–1277, Nagoya, Japan, October. Asian Federation of Natural Language Processing.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010*, pages 1045–1048.

Tomas Mikolov, Anoop Deoras, Dan Povey, Lukar Burget, and Jan Honza Cernocky. 2011. Strategies for training large scale neural network language

models. In *Proceedings of ASRU 2011*, pages 196–201. IEEE Automatic Speech Recognition and Understanding Workshop.

Tomas Mikolov. 2012. *Statistical language models based on neural networks*. Ph.D. thesis, Brno university of technology.

Wenzhe Pei, Tao Ge, and Baobao Chang. 2014. Max-margin tensor neural network for Chinese word segmentation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 1: Long Papers*, pages 293–303.

Mo Shen, Hongxiao Liu, Daisuke Kawahara, and Sadao Kurohashi. 2014. Chinese Morphological Analysis with Character-level POS Tagging. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 253–258, Baltimore, Maryland. Association for Computational Linguistics.

Andreas Stolcke. 2002. SRILM - an extensible language modeling toolkit. In John H L Hansen and Bryan L Pellom, editors, *7th International Conference on Spoken Language Processing, ICSLP2002 - INTERSPEECH 2002, Denver, Colorado, USA, September 16-20, 2002*, pages 901–904. ISCA.

Scott M. Thede and Mary P. Harper. 1999. A second-order Hidden Markov Model for part-of-speech tagging. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, pages 175–182, Morristown, NJ, USA, June. Association for Computational Linguistics.

Yiou Wang, Jun'ichi Kazama, Wenliang Chen, Yujie Zhang, Kentaro Torisawa, and Yoshimasa Tsuruoka. 2011. Improving chinese word segmentation and POS tagging with semi-supervised methods using large auto-analyzed data. In *Proceedings of the Fifth International Joint Conference on Natural Language Processing (IJCNLP-2011)*, pages 309–317, Chiang Mai, Thailand. Asian Federation of Natural Language Processing.

Jialei Wang, Peilin Zhao, and Steven C.H. Hoi. 2012. Exact soft confidence-weighted learning. In *29th International Conference on Machine Learning (ICML 2012)*, pages 121–128.

Yue Zhang and Stephen Clark. 2008. Joint word segmentation and pos tagging using a single perceptron. In *Proceedings of ACL-08: HLT*, pages 888–896, Columbus, Ohio, June. Association for Computational Linguistics.

Ying Zhang, Stephan Vogel, and Alex Waibel. 2004. Interpreting bleu/nist scores: How much improvement do we need to have a better system? In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC-2004)*, Lisbon, Portugal, May. European Language Resources Association (ELRA). ACL Anthology Identifier: L04-1489.

Xiaoqing Zheng, Hanyang Chen, and Tianyu Xu. 2013. Deep learning for Chinese word segmentation and POS tagging. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 647–657.

Can Symbol Grounding Improve Low-Level NLP? Word Segmentation as a Case Study

Hirotaka Kameko†, Shinsuke Mori‡, and Yoshimasa Tsuruoka†
†Graduate School of Engineering, The University of Tokyo
Hongo, Bunkyo-ku, Tokyo, Japan
{kameko, tsuruoka}@logos.t.u-tokyo.ac.jp
‡Academic Center for Computing and Media Studies, Kyoto University
Yoshida Honmachi, Sakyo-ku, Kyoto, Japan
forest@i.kyoto-u.ac.jp

Abstract

We propose a novel framework for improving a word segmenter using information acquired from symbol grounding. We generate a term dictionary in three steps: generating a pseudo-stochastically segmented corpus, building a symbol grounding model to enumerate word candidates, and filtering them according to the grounding scores. We applied our method to game records of Japanese chess with commentaries. The experimental results show that the accuracy of a word segmenter can be improved by incorporating the generated dictionary.

1 Introduction

Today we can easily obtain a large amount of text associated with multi-modal information, and there is a growing interest in the use of non-textual information in the natural language processing (NLP) community. Many of these studies aim to output natural language sentences from a nonlinguistic modality, such as image (Farhadi et al., 2010; Yang et al., 2011; Rohrbach et al., 2013). Kiros et al. (2014) showed that multi-modal information improves the performance of a language model.

Inspired by these studies, we explore a method for improving the performance of a low-level NLP task using multi-modal information. In this work, we focus on the task of word segmentation (WS) in Japanese. WS is often performed as the first processing step for languages without clear word boundaries, and it is as important as part-of-speech (POS) tagging in English. We assume that a large set of pairs of non-textual data and sentences describing them is available as the information source. In our experiments, the pairs consist of game states in *Shogi* (Japanese chess) and textual comments on them, which were made by Shogi experts. We enumerate substrings (character sequences) in the sentences and match them with *Shogi* states by a neural network model. The rationale here is that substrings which match with non-language data well tend to be real words.

Our method consists of three steps (see Figure 1). First, we segment commentary sentences for a game state in various ways to produce word candidates. Then, we match them with game states of a Shogi playing program. Finally, we compile the symbol grounding results at all states and incorporate them to an automatic WS. To the best of our knowledge, this is the first result reporting a performance improvement in an NLP task by symbol grounding.

2 Stochastically Segmented Corpus

Before symbol grounding, we need to segment the text into words that include probable candidate words. For this purpose, we use a stochastically segmented corpus (SSC) (Mori and Takuma, 2004). Then we propose to simulate it by a normal (deterministically) segmented corpus to avoid the problem of computational cost.

2.1 Stochastically Segmented Corpora

An SSC is defined as a combination of a raw corpus C_r (hereafter referred to as the character sequence $\boldsymbol{x}_1^{n_r}$) and word boundary probabilities of the form P_i, which is the probability that a word boundary exists between two characters x_i and x_{i+1}. These probabilities are estimated by a model based on logistic regression (LR) (Fan et al., 2008) trained on a manually segmented corpus by referring to the surrounding characters[1]. Since there are word boundaries before the first character and after the last character of the corpus, $P_0 = P_{n_r} = 1$. The expected frequency of a word

[1]In the experiment we used the same features as those used in Neubig et al., (2011). .

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 2298–2303,
Lisbon, Portugal, 17-21 September 2015.

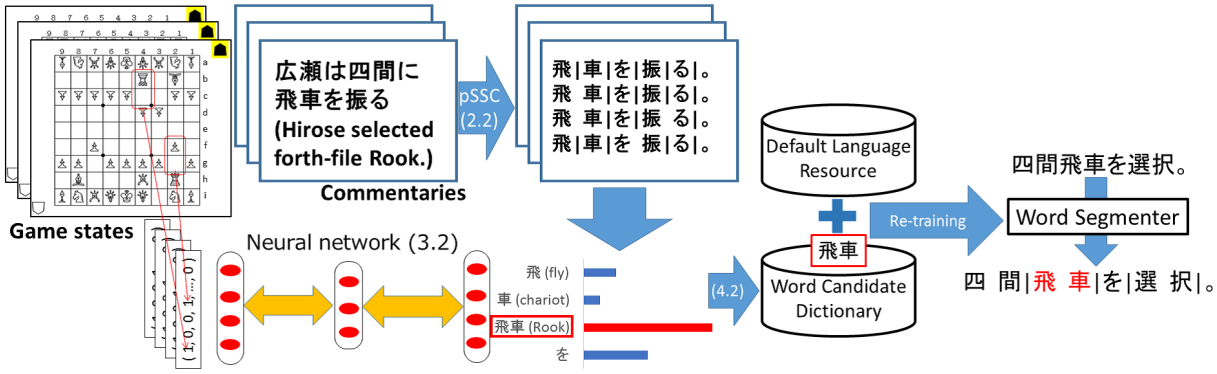


Figure 1: Overview of our method.

$\boldsymbol{w}$ in an SSC is calculated as follows: $f_r(\boldsymbol{w}) = \sum_{i \in O} P_i \left\{ \prod_{j=1}^{k-1} (1 - P_{i+j}) \right\} P_{i+k}$, where $O = \{i \mid x_{i+1}^{i+k} = \boldsymbol{w}\}$ is the set of all the occurrences of the string matching with $\boldsymbol{w}$[2].

2.2 Pseudo-Stochastically Segmented Corpora

The computational cost (in terms of both time and space) for calculating the expected frequencies in an SSC is very high[3], so it is not a practical approach for symbol grounding. In this work, we approximate an SSC using a deterministically segmented corpus, which we call a pseudo-stochastically segmented corpus (pSSC). The following is the process we use to produce a pSSC from an SSC.

- For i = 1 to $n_r - 1$
  1. output a character x_i,
  2. generate a random number $0 \leq p < 1$,
  3. output a word boundary if $p < P_i$ or output nothing otherwise.

Now we have a corpus in the same format as a standard segmented corpus with variable (non-constant) segmentation, where x_i and x_{i+1} are segmented with the probability of P_i. We execute the above procedure m times and divide the counts by m. The law of large numbers guarantees that the approximation errors decrease to 0 when $m \to \infty$.

3 Symbol Grounding

As the target of symbol grounding, we use states (piece positions) of a Shogi game and commentaries associated with them. We should note, however, that our framework is general and applicable to different types of combinations such as image/description pairs (Regneri et al., 2013).

3.1 Game Commentary

The Japanese language is one of the languages without clear word boundaries and we need an automatic WS as the first step of NLP. In Shogi, there are many professional players and many commentaries about game states are available.

3.2 Grounding Words

We build a symbol grounding model using a Shogi commentary dataset. We use a set of pairs of a Shogi state S_i and a commentary sentence C_i as the training set. A Shogi state S_i is converted into a feature vector f(S_i). We generate m (in our experiment, $m = 4$) pSSC C'_i from C_i. C'_i contains m corpora of the same text body but with different word segmentation, C'_{ij} ($j = 1, \ldots, m$). We treat these as m pairs of a feature vector of Shogi state f(S_i) and a sequence of words C'_{ij}. We train a model which predicts words in C'_{ij} using f(S_i) as input.

We use a multi-layer perceptron as the prediction model. The input is a vector of the features of a state. The hidden layer is a 100-dimensional vector and is activated by a bipolar sigmoid function. Its output is a d-dimensional real-valued vector, each of whose elements indicates whether a word in the vocabulary of d words appears in the commentary or not. The output layer is activated by a binary sigmoid function.

We use features of Shogi states which a computer Shogi program called Gekisashi (Tsuruoka et al., 2002) uses to evaluate the states in game tree search as input. The features of Shogi states used in this experiment are below:

a) Positions of pieces (e.g. my rook is at 2h).

[2]For a detailed explanation and a mathematical proof of this method, please refer to Mori and Takuma (2004) .

[3]This is because an SSC has many words and word fragments. Additionally, word 1-gram frequencies must be calculated using floating point numbers instead of integers.

b) Pieces captured (e.g. the opponent has a bishop).
c) Combinations of a) and b) (e.g. my king is at 7h and the opponent's rook is at 7b).
d) Other heuristic features.

Among them, a), b) and c) occupy the majority.

Unlike normal symbol grounding, the vocabulary contains many word candidates appearing in the pSSC generated from the commentaries. Some are real words and some are wrong fragments. These wrong fragments will appear more or less randomly in the commentaries than real words. The perceptron therefore cannot acquire strong relation between states and fragments and the output values of the perceptron will be smaller than those of real words.

4 Word Segmentation Using Symbol Grounding Result

This section describes a baseline automatic word segmenter and a method for incorporating the symbol grounding result to it.

4.1 Baseline Word Segmenter

Among many Japanese WS and morphological analyzers (word segmentation and POS tagging), we adopt pointwise WS (Neubig et al., 2011), because it is the only word segmenter which is capable of adding new words without POS information.

The input of the pointwise WS is an unsegmented character sequence $\boldsymbol{x} = x_1 x_2 \cdots x_k$. The word segmenter decides if there is a word boundary $t_i = 1$ or not $t_i = 0$ by using support vector machines (SVMs) (Fan et al., 2008). The features are character n-grams and character type n-grams ($n = 1, 2, 3$) around the decision points in a window with a width of 6 characters. Additional features are triggered if character n-grams in the window match with character sequences in the dictionary.

4.2 Training a Word Segmenter with Grounded Words

As a first trial for incorporating symbol grounding results to an NLP task, we propose to generate a dictionary based on the symbol grounding result. We can expect that the word candidates that are given high scores by the perceptron in the symbol grounding result have strong relationship to the positions. In other words, we can make a good dictionary by selecting word candidates in descending order of the scores. As a method for taking all the occurrences into account, we test the following three functions:

sum: the summation of the scores of all the output vectors,
ave: the average of them,
max: the maximum in them.

First, we acquire a V-dimensional real-valued vector for each Shogi state S_i as the result of symbol grounding. Then, for each candidate in C'_{ij}, we get the element of the vector which corresponds to the candidate as the score of the candidate. After that, we get the summation of, the average of, or the maximum in the scores of the same candidate over the whole dataset.

Finally we select the top R percent of word candidates in descending order of the value of **sum**, **ave**, or **max** and add them to the WS dictionary and retrain the model.

Table 1: Corpus specifications.

	#sent.	#words	#char.
Training			
BCCWJ	56,753	1,324,951	1,911,660
Newspaper	8,164	240,097	361,843
Conversation	11,700	147,809	197,941
Develepment			
Shogi-dev.	170	2,501	3,340
Test			
BCCWJ-test	6,025	148,929	212,261
Shogi-test	3,299	24,966	32,481

5 Evaluation

We conducted word segmentation experiments in the following settings.

5.1 Corpora

The annotated corpus we used to build the baseline word segmenter is the manually annotated part (core data) of the Balanced Corpus of Contemporary Written Japanese (BCCWJ) (Maekawa, 2008), plus newspaper articles and daily conversation sentences. We also used a 234,652-word dictionary (UniDic) provided with the BCCWJ. A small portion of the BCCWJ core data is reserved for testing. In addition, we manually segmented sentences randomly obtained from Shogi commentaries. We divided these sentences into two parts: a development set and a test set. Table 1 shows the details of these corpora.

To make a pSSC, we prepared 33,151 pairs of a Shogi position and a commentary sentence. The

Table 2: WS accuracy on BCCWJ.

	Recall	Prec.	F-meas.
Baseline	98.99	99.06	99.03
+ Sym.Gro.	99.03	99.01	99.02

Table 3: WS accuracy on Shogi commentaries.

	Recall	Prec.	F-meas.
Baseline	90.12	91.43	90.77
+ Sym.Gro.	90.60	91.66	91.13

sentences are converted into pSSC $m = 4$ times by an LR word segmentation model trained from the training data in Table 1 and sent to the symbol grounding module.

5.2 Word Segmentation Systems

We built the following two word segmentation models (Neubig et al., 2011) to evaluate our framework.

Baseline: The model is trained from training data shown in Table 1 and UniDic.

+Sym.Gro.: The model is trained from the language resources for the **Baseline** and the symbol grounding result.

To decide the function and the value of R for **+Sym.Gro.** (see Section 4.2), we measured the accuracies on the development set of all the combinations. The best combination was **sum** and $R = 0.011$[4]. In this case, 127 words were added to the dictionary.

5.3 Results and Discussion

Following the standard in word segmentation experiments, the evaluation criteria are recall, precision, and F-measure (their harmonic mean).

Table 2 and 3 show WS accuracies on BCCWJ-test and Shogi-test, respectively. The difference in accuracy of the baseline method on BCCWJ-test and Shogi-test shows that WS of Shogi commentaries is very difficult. Like many other domains, Shogi commentaries contain many special words and expressions, which decrease the accuracy.

When we compare the F-measures on Shogi-test (Table 3), **+Sym.Gro.** outperforms **Baseline**. The improvement is statistically significant (at 5% level). The error reduction ratio is comparable to a natural annotation case (Liu et al., 2014), despite the fact that our method is unsupervised except for a hyperparameter. Thus we can say that WS improvement by symbol grounding is as valuable as the annotation additions.

From a close look at the comparison of the recall and the precision, we see that the improvement in the recall is higher than that of the precision. This result shows that the symbol grounding successfully acquired new words with a few erroneous words. As the final remark, the result on the general domain (Table 2) shows that our framework does not cause a severe performance degradation in the general domain.

6 Related Work

The NLP task we focus on in this paper is word segmentation. One of the first empirical methods was based on a hidden Markov model (Nagata, 1994). In parallel, there were attempts at solving Chinese word segmentation in a similar way (Sproat and Chang, 1996). These methods take words as the modeling unit.

Recently, Neubig et al. (2011) have presented a method for directly deciding whether there is a word boundary or not at each point between characters. For Chinese word segmentation, there are some attempts at tagging characters with BIES tags (Xue, 2003) by a sequence labeller such as CRFs (Lafferty et al., 2001), where B, I, E, and S means the beginning of a word, intermediate of a word, the end of a word, and a single character word, respectively. The pointwise WS can be seen as character tagging with the BI tag system, in which there is no constraint between neighboring tags. For Japanese WS, our preliminary experiments showed that the combination of the BI tag system with SVMs is slightly better than the BIES tag system with CRFs. This is another reason why we used the former in this paper. Our extension of word segmentation is, however, applicable to the BIES/CRFs combination as well.

The method we describe in this paper is unsupervised and requires a small amount of annotated data to tune the hyperparameter. From this viewpoint, the approach based on natural annotation (Yang and Vozila, 2014; Jiang et al., 2013; Liu et al., 2014) may come to readers' mind. In these studies, tags in hyper-texts were regarded as partial annotations and used to improve WS performance using CRFs trainable from such data (Tsuboi et al., 2008). Mori and Nagao (1996) proposed a method for extracting new words from a large amount of raw text. Murawaki and Kuro-

[4] In addition we measured the accuracies on the test set of all the combinations and found that the same function and the value of the parameter are the best. This indicates the stability of the function and the parameter.

hashi (2008) proposed an online method in a similar setting. In contrast to these studies, this paper proposes to use other modalities, game states as the first trial, than languages.

7 Conclusion

We have described an unsupervised method for improving word segmentation based on symbol grounding results. To extract word candidates from raw sentences, we first segment sentences stochastically, and then match the word candidate sequences with game states that are described by the sentences. Finally, we selected word candidates referring to the grounding scores. The experimental results showed that we can improve word segmentation by using symbol grounding results. Our framework is general and it is worth testing on other NLP tasks. As future work, we will apply other deep neural network models to our approach. It is interesting to apply the symbol grounding results to an embedding model-based word segmentation approach (Ma and Hinrichs, 2015). It is also interesting to extend our method to deal with other types of non-textual information such as images and economic indices.

Acknowledgment

We thank the anonymous reviewers for their helpful comments and suggestions. This work was supported by JSPS Grants-in-Aid for Scientific Research Grant Number 26540190.

References

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874.

Ali Farhadi, Mohsen Hejrati, Mohammad Amin Sadeghi, Peter Young, Cyrus Rashtchian, Julia Hockenmaier, and David Forsyth. 2010. Every picture tells a story: Generating sentences from images. In *Proceedings of the 11th European Conference on Computer Vision*, pages 15–29.

Wenbin Jiang, Meng Sun, Yajuan Lu, Yating Yang, and Qun Liu. 2013. Discriminative learning with natural annotations: Word segmentation as a case study. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 761–769.

Ryan Kiros, Ruslan Salakhutdinov, and Rich Zemel. 2014. Multimodal neural language models. In *Proceedings of the 31st International Conference on Machine Learning*, pages 595–603.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 282–289.

Yijia Liu, Yue Zhang, Wangxiang Che, Ting Liu, and Fan Wu. 2014. Domain adaptation for CRF-based Chinese word segmentation using free annotations. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 864–874.

Jianqiang Ma and Erhard Hinrichs. 2015. Accurate linear-time Chinese word segmentation via embedding matching. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*, pages 1733–1743.

Kikuo Maekawa. 2008. Balanced corpus of contemporary written Japanese. In *Proceedings of the 6th Workshop on Asian Language Resources*, pages 101–102.

Shinsuke Mori and Makoto Nagao. 1996. Word extraction from corpora and its part-of-speech estimation using distributional analysis. In *Proceedings of the 16th International Conference on Computational Linguistics*, pages 1119–1122.

Shinsuke Mori and Daisuke Takuma. 2004. Word *n*-gram probability estimation from a Japanese raw corpus. In *Proceedings of the Eighth International Conference on Speech and Language Processing*, pages 1037–1040.

Yugo Murawaki and Sadao Kurohashi. 2008. Online acquisition of Japanese unknown morphemes using morphological constraints. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 429–437.

Masaaki Nagata. 1994. A stochastic Japanese morphological analyzer using a forward-DP backward-A* N-best search algorithm. In *Proceedings of the 15th International Conference on Computational Linguistics*, pages 201–207.

Graham Neubig, Yosuke Nakata, and Shinsuke Mori. 2011. Pointwise prediction for robust, adaptable Japanese morphological analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, pages 529–533.

Michaela Regneri, Marcus Rohrbach, Dominikus Wetzel, Stefan Thater, Bernt Schiele, and Manfred Pinkal. 2013. Grounding action descriptions in videos. *Transactions of the Association for Computational Linguistics*, 1:25–36.

Marcus Rohrbach, Wei Qiu, Ivan Titov, Stefan Thater, Manfred Pinkal, and Bernt Schiele. 2013. Translating video content to natural language descriptions. In *Proceedings of the 14th International Conference on Computer Vision*, pages 433–440.

Richard Sproat and Chilin Shih William Gale Nancy Chang. 1996. A stochastic finite-state word-segmentation algorithm for Chinese. *Computational Linguistics*, 22(3):377–404.

Yuta Tsuboi, Hisashi Kashima, Shinsuke Mori, Hiroki Oda, and Yuji Matsumoto. 2008. Training conditional random fields using incomplete annotations. In *Proceedings of the 22nd International Conference on Computational Linguistics*, pages 897–904.

Yoshimasa Tsuruoka, Daisaku Yokoyama, and Takashi Chikayama. 2002. Game-tree search algorithm based on realization probability. *ICGA Journal*, 25(3):145–152.

N. Xue. 2003. Chinese word segmentation as character tagging. *International Journal of Computational Linguistics and Chinese*, 8(1):29–48.

Fan Yang and Paul Vozila. 2014. Semi-supervised Chinese word segmentation using partial-label learning with conditional random fields. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 90–98.

Yezhou Yang, Ching Lik Teo, Hal Daumé III, and Yiannis Aloimonos. 2011. Corpus-guided sentence generation of natural images. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 444–454.

When Are Tree Structures Necessary for Deep Learning of Representations?

Jiwei Li[1], **Minh-Thang Luong**[1], **Dan Jurafsky**[1] **and Eduard Hovy**[2]
[1]Computer Science Department, Stanford University, Stanford, CA 94305
[2]Language Technology Institute, Carnegie Mellon University, Pittsburgh, PA 15213
jiweil,lmthang,jurafsky@stanford.edu ehovy@andrew.cmu.edu

Abstract

Recursive neural models, which use syntactic parse trees to recursively generate representations bottom-up, are a popular architecture. However there have not been rigorous evaluations showing for exactly which tasks this syntax-based method is appropriate. In this paper, we benchmark *recursive* neural models against sequential *recurrent* neural models, enforcing apples-to-apples comparison as much as possible. We investigate 4 tasks: (1) sentiment classification at the sentence level and phrase level; (2) matching questions to answer-phrases; (3) discourse parsing; (4) semantic relation extraction.

Our goal is to understand better when, and why, recursive models can outperform simpler models. We find that recursive models help mainly on tasks (like semantic relation extraction) that require long-distance connection modeling, particularly on very long sequences. We then introduce a method for allowing recurrent models to achieve similar performance: breaking long sentences into clause-like units at punctuation and processing them separately before combining. Our results thus help understand the limitations of both classes of models, and suggest directions for improving recurrent models.

1 Introduction

Deep learning based methods learn low-dimensional, real-valued vectors for word tokens, mostly from large-scale data corpus (e.g., (Mikolov et al., 2013; Le and Mikolov, 2014; Collobert et al., 2011)), successfully capturing syntactic and semantic aspects of text.

For tasks where the inputs are larger text units (e.g., phrases, sentences or documents), a compositional model is first needed to aggregate tokens into a vector with fixed dimensionality that can be used as a feature for other NLP tasks. Models for achieving this usually fall into two categories: *recurrent* models and *recursive* models:

Recurrent models (also referred to as *sequence* models) deal successfully with time-series data (Pearlmutter, 1989; Dorffner, 1996) like speech (Robinson et al., 1996; Lippmann, 1989; Graves et al., 2013) or handwriting recognition (Graves and Schmidhuber, 2009; Graves, 2012). They were applied early on to NLP (Elman, 1990), by modeling a sentence as tokens processed sequentially and at each step combining the current token with previously built embeddings. Recurrent models can be extended to bidirectional ones from both left-to-right and right-to-left. These models generally consider no linguistic structure aside from word order.

Recursive neural models (also referred to as *tree* models), by contrast, are structured by syntactic parse trees. Instead of considering tokens sequentially, recursive models combine neighbors based on the recursive structure of parse trees, starting from the leaves and proceeding recursively in a bottom-up fashion until the root of the parse tree is reached. For example, for the phrase *the food is delicious*, following the operation sequence *((the food) (is delicious))* rather than the sequential order *(((the food) is) delicious)*. Many recursive models have been proposed (e.g., (Paulus et al., 2014; Irsoy and Cardie, 2014)), and applied to various NLP tasks, among them entailment (Bowman, 2013; Bowman et al., 2014), sentiment analysis (Socher et al., 2013; Irsoy and Cardie, 2013; Dong et al., 2014), question-answering (Iyyer et al., 2014), relation classification (Socher et al., 2012; Hashimoto et al., 2013), and discourse (Li and Hovy, 2014).

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 2304–2314,
Lisbon, Portugal, 17-21 September 2015.

One possible advantage of recursive models is their potential for capturing long-distance dependencies: two tokens may be structurally close to each other, even though they are far away in word sequence. For example, a verb and its corresponding direct object can be far away in terms of tokens if many adjectives lies in between, but they are adjacent in the parse tree (Irsoy and Cardie, 2013). However we do not know if this advantage is truly important, and if so for which tasks, or whether other issues are at play. Indeed, the reliance of recursive models on parsing is also a potential disadvantage, given that parsing is relatively slow, domain-dependent, and can be errorful.

On the other hand, recent progress in multiple subfields of neural NLP has suggested that recurrent nets may be sufficient to deal with many of the tasks for which recursive models have been proposed. Recurrent models without parse structures have shown good results in sequence-to-sequence generation (Sutskever et al., 2014) for machine translation (e.g., (Kalchbrenner and Blunsom, 2013; 3; Luong et al., 2014)), parsing (Vinyals et al., 2014), and sentiment, where for example recurrent-based paragraph vectors (Le and Mikolov, 2014) outperform recursive models (Socher et al., 2013) on the Stanford sentimentbank dataset.

Our goal in this paper is thus to investigate a number of tasks with the goal of understanding for which kinds of problems recurrent models may be sufficient, and for which kinds recursive models offer specific advantages. We investigate four tasks with different properties.

- Binary **sentiment classification** at the sentence level (Pang et al., 2002) and phrase level (Socher et al., 2013) that focus on understanding the role of recursive models in dealing with semantic compositionally in various scenarios such as different lengths of inputs and whether or not supervision is comprehensive.

- **Phrase Matching** on the UMD-QA dataset (Iyyer et al., 2014) can help see the difference between outputs from intermediate components from different models, i.e., representations for intermediate parse tree nodes and outputs from recurrent models at different time steps. It also helps see whether parsing is useful for finding similarities between question sentences and target phrases.

- **Semantic Relation Classification** on the SemEval-2010 (Hendrickx et al., 2009) data can help understand whether parsing is helpful in dealing with long-term dependencies, such as relations between two words that are far apart in the sequence.

- **Discourse parsing** (RST dataset) is useful for measuring the extent to which parsing improves discourse tasks that need to combine meanings of larger text units. Discourse parsing treats elementary discourse units (EDUs) as basic units to operate on, which are usually short clauses. The task also sheds light on the extent to which syntactic structures help acquire shot text representations.

The principal motivation for this paper is to understand better when, and why, recursive models are needed to outperform simpler models by enforcing apples-to-apples comparison as much as possible. This paper applies existing models to existing tasks, barely offering novel algorithms or tasks. Our goal is rather an analytic one, to investigate different versions of recursive and recurrent models. This work helps understand the limitations of both classes of models, and suggest directions for improving recurrent models.

The rest of this paper organized as follows: We detail versions of recursive/recurrent models in Section 2, present the tasks and results in Section 3, and conclude with discussions in Section 4.

2 Recursive and Recurrent Models

2.1 Notations

We assume that the text unit S, which could be a phrase, a sentence or a document, is comprised of a sequence of tokens/words: $S = \{w_1, w_2, ..., w_{N_S}\}$, where N_s denotes the number of tokens in S. Each word w is associated with a K-dimensional vector embedding $e_w = \{e_w^1, e_w^2, ..., e_w^K\}$. The goal of recursive and recurrent models is to map the sequence to a K-dimensional e_S, based on its tokens and their correspondent embeddings.

Standard Recurrent/Sequence Models successively take word w_t at step t, combines its vector representation e_t with the previously built hidden vector h_{t-1} from time $t-1$, calculates the re-

sulting current embedding h_t, and passes it to the next step. The embedding h_t for the current time t is thus:

$$h_t = f(W \cdot h_{t-1} + V \cdot e_t) \tag{1}$$

where W and V denote compositional matrices. If N_s denotes the length of the sequence, h_{N_s} represents the whole sequence S.

Standard recursive/Tree models work in a similar way, but processing neighboring words by parse tree order rather than sequence order. It computes a representation for each parent node based on its immediate children recursively in a bottom-up fashion until reaching the root of the tree. For a given node η in the tree and its left child η_{left} (with representation e_{left}) and right child η_{right} (with representation e_{right}), the standard recursive network calculates e_η as follows:

$$e_\eta = f(W \cdot e_{\eta_{\text{left}}} + V \cdot e_{\eta_{\text{right}}}) \tag{2}$$

Bidirectional Models (Schuster and Paliwal, 1997) add bidirectionality to the recurrent framework where embeddings for each time are calculated both forwardly and backwardly:

$$\begin{aligned} h_t^{\rightarrow} &= f(W^{\rightarrow} \cdot h_{t-1}^{\rightarrow} + V^{\rightarrow} \cdot e_t) \\ h_t^{\leftarrow} &= f(W^{\leftarrow} \cdot h_{t+1}^{\leftarrow} + V^{\leftarrow} \cdot e_t) \end{aligned} \tag{3}$$

Normally, final representations for sentences can be achieved either by concatenating vectors calculated from both directions $[e_1^{\leftarrow}, e_{N_S}^{\rightarrow}]$ or using further compositional operation to preserve vector dimensionality

$$h_t = f(W_L \cdot [h_t^{\leftarrow}, h_t^{\rightarrow}]) \tag{4}$$

where W_L denotes a $K \times 2K$ dimensional matrix.

Long Short Term Memory (LSTM) LSTM models (Hochreiter and Schmidhuber, 1997) are defined as follows: given a sequence of inputs $X = \{x_1, x_2, ..., x_{n_X}\}$, an LSTM associates each timestep with an input, memory and output gate, respectively denoted as i_t, f_t and o_t. We notationally disambiguate e and h: e_t denotes the vector for individual text units (e.g., word or sentence) at time step t, while h_t denotes the vector computed by the LSTM model at time t by combining e_t and h_{t-1}. σ denotes the sigmoid function. The vector representation h_t for each time-step t is given by:

$$\begin{bmatrix} i_t \\ f_t \\ o_t \\ l_t \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{bmatrix} W \cdot \begin{bmatrix} h_{t-1} \\ e_t \end{bmatrix} \tag{5}$$

$$c_t = f_t \cdot c_{t-1} + i_t \cdot l_t \tag{6}$$

$$h_t^s = o_t \cdot c_t \tag{7}$$

where $W \in \mathbb{R}^{4K \times 2K}$. Labels at the phrase/sentence level are predicted representations outputted from the last time step.

Tree LSTMs Recent research has extended the LSTM idea to tree-based structures (Zhu et al., 2015; Tai et al., 2015) that associate memory and forget gates to nodes of the parse trees.

Bi-directional LSTMs These combine bi-directional models and LSTMs.

3 Experiments

In this section, we detail our experimental settings and results. We consider the following tasks, each representative of a different class of NLP tasks.

- **Binary sentiment classification** on the Pang et al. (2002) dataset. This addresses the issues where supervision only appears globally after a long sequence of operations.
- **Sentiment Classification on the Stanford Sentiment Treebank** (Socher et al., 2013): comprehensive labels are found for words and phrases where local compositionally (such as from negation, mood, or others cued by phrase-structure) is to be learned.
- **Sentence-Target Matching** on the UMD-QA dataset (Iyyer et al., 2014): Learns matches between target and components in the source sentences, which are parse tree nodes for recursive models and different time-steps for recurrent models.
- **Semantic Relation Classification** on the SemEval-2010 task (Hendrickx et al., 2009). Learns long-distance relationships between two words that may be far apart sequentially.
- **Discourse Parsing** (Li et al., 2014; Hernault et al., 2010): Learns sentence-to-sentence relations based on calculated representations.

In each case we followed the protocols described in the original papers. We first group the algorithm variants into two groups as follows:

- Standard tree models vs standard sequence models vs standard bi-directional sequence models
- LSTM tree models, LSTM sequence models vs LSTM bi-directional sequence models.

We employed standard training frameworks for neural models: for each task, we used stochastic gradient decent using AdaGrad (Duchi et al., 2011) with minibatches (Cotter et al., 2011). Parameters are tuned using the development dataset if available in the original datasets or from cross-validation if not. Derivatives are calculated from standard back-propagation (Goller and Kuchler, 1996). Parameters to tune include size of mini batches, learning rate, and parameters for L2 penalizations. The number of running iterations is treated as a parameter to tune and the model achieving best performance on the development set is used as the final model to be evaluated.

For settings where no repeated experiments are performed, the bootstrap test is adopted for statistical significance testing (Efron and Tibshirani, 1994). Test scores that achieve significance level of 0.05 are marked by an asterisk (*).

3.1 Stanford Sentiment TreeBank

Task Description We start with the Stanford Sentiment TreeBank (Socher et al., 2013). This dataset contains gold-standard labels for every parse tree constituent, from the sentence to phrases to individual words.

Of course, any conclusions drawn from implementing sequence models on a dataset that was based on parse trees may have to be weakened, since sequence models may still benefit from the way that the dataset was collected. Nevertheless we add an evaluation on this dataset because it has been a widely used benchmark dataset for neural model evaluations.

For recursive models, we followed the protocols in Socher et al. (2013) where node embeddings in the parse trees are obtained from recursive models and then fed to a softmax classifier. We transformed the dataset for recurrent model use as illustrated in Figure 1. Each phrase is reconstructed from parse tree nodes and treated as a separate data point. As the treebank contains 11,855 sentences with 215,154 phrases, the reconstructed dataset for recurrent models comprises 215,154 examples. Models are evaluated at both the phrase level (82,600 instances) and the sentence root level (2,210 instances).

	Fine-Grained	Binary
Tree	0.433	0.815
Sequence	0.420 (-0.013)	0.807 (-0.007)
P-value	0.042*	0.098
Bi-Sequence	0.435 (+0.08)	0.816 (+0.002)
P-value	0.078	0.210

Table 1: Test set accuracies on the Stanford Sentiment Treebank at root level.

	Fine-Grained	Binary
Tree	0.820	0.860
Sequence	0.818 (-0.002)	0.864 (+0.004)
P-value	0.486	0.305
Bi-Sequence	0.826 (+0.06)	0.862 (+0.002)
P-value	0.148	0.450

Table 2: Test set accuracies on the Stanford Sentiment Treebank at phrase level.

Results are shown in Table 1 and 2[1]. When comparing the standard version of tree models to sequence models, we find it helps a bit at root level identification (for sequences but not bi-sequences), but yields no significant improvement at the phrase level.

LSTM Tai et al. (2015) discovered that LSTM tree models generate better performances in terms of sentence **root** level evaluation than sequence models. We explore this task a bit more by training deeper and more sophisticated models. We examine the following three models:

1. Tree-structured LSTM models (Tai et al., 2015)[2].
2. Deep Bi-LSTM sequence models (denoted as **Sequence**) that treat the whole sentence as just one sequence.
3. Deep Bi-LSTM hierarchical sequence models (denoted as **Hierarchical Sequence**) that first slice the sentence into a sequence of subsentences by using a look-up table of punctuations (i.e., comma, period, question mark

[1]The performance of our implementations of recursive models is not exactly identical to that reported in Socher et al. (2013), but the relative difference is around 1% to 2%.

[2]Tai et al.. achieved 0.510 accuracy in terms of fine-grained evaluation at the root level as reported in (Tai et al., 2015), similar to results from our implementations (0.504).

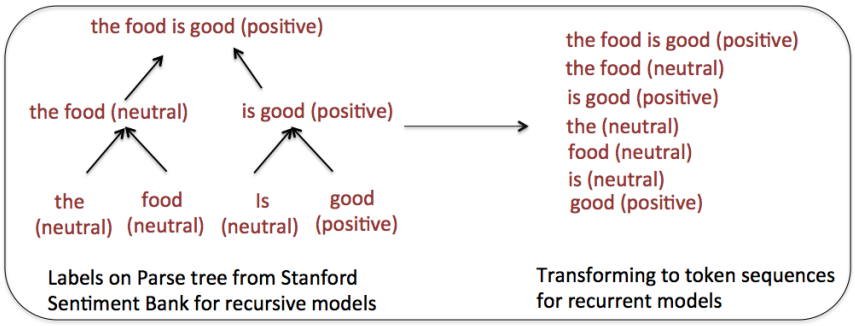


Figure 1: Transforming Stanford Sentiment Treebank to Sequences for Sequence Models.

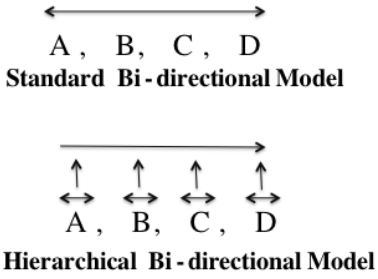


Figure 2: Illustration of two sequence models. A, B, C, D denote clauses or sub sentences separated by punctuation.

and exclamation mark). The representation for each sub-sentence is first computed separately, and another level of sequence LSTM (one-directional) is then used to join the sub-sentences. Illustrations are shown in Figure2.

We consider the third model because the dataset used in Tai et al. (2015) contains long sentences and the evaluation is performed only at the sentence root level. Since a parsing algorithm will naturally break long sentences into sub-sentences, we would like to know whether any performance boost is introduced by the intra-clause parse tree structure or just by this broader segmentation of a sentence into clause-like units; this latter advantage could be approximated by using punctuation-based approximations to clause boundaries.

We run 15 iterations for each algorithm. Parameters are harvested at the end of each iteration; those performing best on the development set are used on the test set. The whole process takes roughly 15-20 minutes on a single GPU machine[3]. For a more convincing comparison, we did not use the bootstrap test where parallel examples are generated from one same dataset. Instead, we repeated the aforementioned procedure for each algorithm 20 times and report accuracies with standard deviation in Table 3.

[3]Tesla K40m, 2880 Cuda cores.

Model	all-fine	root-fine	root-coarse
Tree LSTM	**83.4** (0.3)	50.4 (0.9)	86.7 (0.5)
Bi-Sequence	83.3 (0.4)	49.8 (0.9)	86.7 (0.5)
Hier-Sequence	82.9 (0.3)	**50.7** (0.8)	**86.9** (0.6)

Table 3: Test set accuracies on the Stanford Sentiment Treebank with deviations. For our experiments, we report accuracies over 20 runs with standard deviation.

Tree LSTMs are equivalent or marginally better than standard bi-directional sequence model (two-tailed p-value equals 0.041*, and only at the root level, with p-value for the phrase level at 0.376). The hierarchical sequence model achieves the same performance with a p-value of 0.198.

Discussion The results above suggest that clausal segmentation of long sentences offers a slight performance boost, a result also supported by the fact that very little difference exists between the three models for phrase-level sentiment evaluation. Clausal segmentation of long sentences thus provides a simple approximation to parse-tree based models.

We suggest a few reasons for this slightly better performances introduced by clausal segmentation:

1. Treating clauses as basic units (to the extent that punctuation approximates clauses) preserves the semantic structure of text.

2. Semantic compositions such as negations or conjunctions usually appear at the clause level. Working on clauses individually and then combining them model inter-clause compositions.

3. Errors are back-propagated to individual tokens using fewer steps in hierarchical models than in standard models. Consider a movie

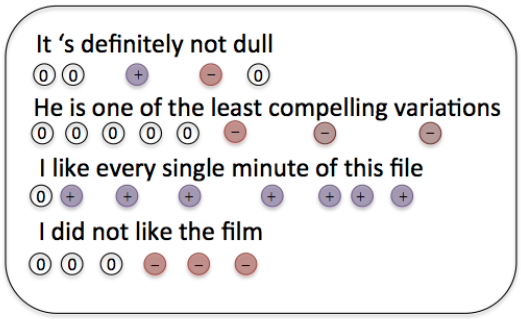


Figure 3: Sentiment prediction using a one-directional (left to right) LSTM. Decisions at each time step are made by feeding embeddings calculated from the LSTM into a softmax classifier.

review "simple as the plot was , i still like it a lot". With standard recurrent models it takes 12 steps before the prediction error gets back to the first token "simple":

error→lot→a→it→like→still→i→,→was →plot→ the→as→simple

In a hierarchical model, the second clause is compacted into one component, and the error propagation is thus given by:

error→ second-clause → first-clause → was→plot→the→as→simple.

Propagation with clause segmentation consists of only 8 operations. Such a procedure thus tends to attenuate the gradient vanishing problem, potentially yielding better performance.

3.2 Binary Sentiment Classification (Pang)

Task Description: The sentiment dataset of Pang et al. (2002) consists of sentences with a sentiment label for each sentence. We divide the original dataset into training(8101)/dev(500)/testing(2000). No pre-training procedure as described in Socher et al. (2011b) is employed. Word embeddings are initialized using skip-grams and kept fixed in the learning procedure. We trained skip-gram embeddings on the Wikipedia+Gigaword dataset using the word2vec package[4]. Sentence level embeddings are fed into a sigmoid classifier. Performances for 50 dimensional vectors are given in the table below:

Discussion Why don't parse trees help on this task? One possible explanation is the distance

[4]https://code.google.com/p/word2vec/

	Standard	LSTM
Tree	0.745	0.774
Sequence	0.733 (-0.012)	0.783 (+0.008)
P-value	0.060	0.136
Bi-Sequence	0.754 (+0.09)	0.790 (+0.016)
P-value	0.058	0.024*

Table 4: Test set accuracies on the Pang's sentiment dataset using Standard model settings.

of the supervision signal from the local compositional structure. The Pang et al. dataset has an average sentence length of 22.5 words, which means it takes multiple steps before sentiment related evidence comes up to the surface. It is therefore unclear whether local compositional operators (such as negation) can be learned; there is only a small amount of training data (around 8,000 examples) and the sentiment supervision only at the level of the sentence may not be easy to propagate down to deeply buried local phrases.

3.3 Question-Answer Matching

Task Description: In the question-answering dataset QANTA[5], each answer is a token or short phrase. The task is different from standard generation focused QA task but formalized as a multi-class classification task that matches a source question with a candidates phrase from a predefined pool of candidate phrases We give an illustrative example here:

Question: *He left unfinished a novel whose title character forges his father's signature to get out of school and avoids the draft by feigning desire to join. Name this German author of The Magic Mountain and Death in Venice.*

Answer: *Thomas Mann* from the pool of phrases. Other candidates might include George Washington, Charlie Chaplin, etc.

The model of Iyyer et al. (2014) minimizes the distances between answer embeddings and node embeddings along the parse tree of the question. Concretely, let c denote the correct answer to question S, with embedding $\vec{c}$, and z denoting any random wrong answer. The objective function sums over the dot product between representation for every node η along the question parse trees and the answer representations:

$$L = \sum_{\eta \in [\text{parse tree}]} \sum_{z} max(0, 1 - \vec{c} \cdot e_\eta + \vec{z} \cdot e_\eta) \quad (8)$$

[5]http://cs.umd.edu/~miyyer/qblearn/. Because the publicly released dataset is smaller than the version used in (Iyyer et al., 2014) due to privacy issues, our numbers are not comparable to those in (Iyyer et al., 2014).

where e_η denotes the embedding for parse tree node calculated from the recursive neural model. Here the parse trees are dependency parses following (Iyyer et al., 2014).

By adjusting the framework to recurrent models, we minimize the distance between the answer embedding and the embeddings calculated from each timestep t of the sequence:

$$L = \sum_{t \in [1, N_s]} \sum_{z} max(0, 1 - \vec{c} \cdot e_t + \vec{z} \cdot e_t) \quad (9)$$

At test time, the model chooses the answer (from the set of candidates) that gives the lowest loss score. As can be seen from results presented in Table 5, the difference is only significant for the LSTM setting between the tree model and the sequence model; no significant difference is observed for other settings.

	Standard	LSTM
Tree	0.523	0.558
Sequence	0.525 (+0.002)	0.546 (-0.012)
P-value	0.490	0.046*
Bi-Sequence	0.530 (+0.007)	0.564 (+0.006)
P-value	0.075	0.120

Table 5: Test set accuracies for UMD-QA dataset.

Discussion The UMD-QA task represents a group of situations where because we have insufficient supervision about matching (it's hard to know which node in the parse tree or which timestep provides the most direct evidence for the answer), decisions have to be made by looking at and iterating over all subunits (all nodes in parse trees or timesteps). Similar ideas can be found in pooling structures (e.g. Socher et al. (2011a)).

The results above illustrate that for tasks where we try to align the target with different source components (i.e., parse tree nodes for tree models and different time steps for sequence models), components from sequence models are able to embed important information, despite the fact that sequence model components are just sentence fragments and hence usually not linguistically meaningful components in the way that parse tree constituents are.

3.4 Semantic Relationship Classification

Task Description: SemEval-2010 Task 8 (Hendrickx et al., 2009) is to find semantic relationships between pairs of nominals, e.g., in "My [apartment]$_{e1}$ has a pretty large [kitchen]$_{e2}$" classifying the relation between [apartment] and [kitchen] as *component-whole*. The dataset contains 9 ordered relationships, so the task is formalized as a 19-class classification problem, with directed relations treated as separate labels; see Hendrickx et al. (2009; Socher et al. (2012) for details.

For the recursive implementations, we follow the neural framework defined in Socher et al. (2012). The path in the parse tree between the two nominals is retrieved, and the embedding is calculated based on recursive models and fed to a softmax classifier[6]. Retrieved paths are transformed for the recurrent models as shown in Figure 5.

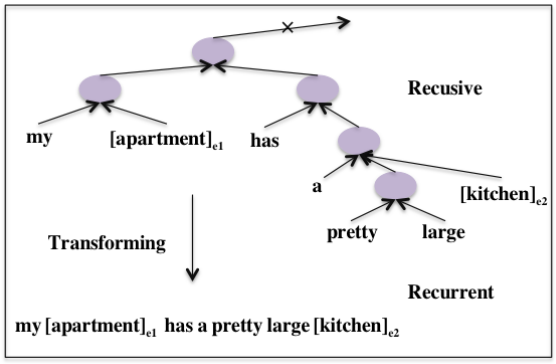


Figure 4: Illustration of Models for Semantic Relationship Classification.

Discussion Unlike for earlier tasks, here recursive models yield much better performance than the corresponding recurrent versions for all versions (e.g., standard tree vs. standard sequence, $p = 0.004$). These results suggest that it is the need to integrate structures far apart in the sentence that characterizes the tasks where recursive models surpass recurrent models. In parse-based models, the two target words are drawn together much earlier in the decision process than in recurrent models, which must remember one target until the other one appears.

3.5 Discourse Parsing

Task Description: Our final task, discourse parsing based on the RST-DT corpus (Carlson et

[6](Socher et al., 2012) achieve state-of-art performance by combining a sophisticated model, MV-RNN, in which each word is presented with both a matrix and a vector with human-feature engineering. Again, because MV-RNN is difficult to adapt to a recurrent version, we do not employ this state-of-the-art model, adhering only to the general versions of recursive models described in Section 2, since our main goal is to compare equivalent recursive and recurrent models rather than implement the state of the art.

	Standard	LSTM
Tree	0.748	0.767
Sequence	0.712 (-0.036)	0.740 (-0.027)
P-value	0.004*	0.020*
Bi-Sequence	0.730 (-0.018)	0.752 (-0.014)
P-value	0.017*	0.041*

Table 6: Test set accuracies on the SemEval-2010 Semantic Relationship Classification task.

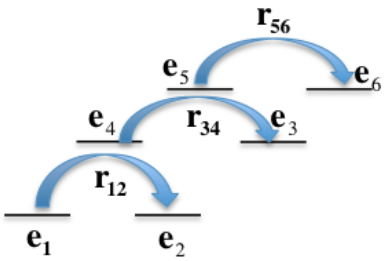


Figure 5: An illustration of discourse parsing. $[e_1, e_2, ...]$ denote EDUs (elementary discourse units), each consisting of a sequence of tokens. $[r_{12}, r_{34}, r_{56}]$ denote relationships to be classified. A binary classification model is first used to decide whether two EDUs should be merged and a multi-class classifier is then used to decide the relation type.

al., 2003), is to build a discourse tree for a document, based on assigning Rhetorical Structure Theory (RST) relations between elementary discourse units (EDUs). Because discourse relations express the coherence structure of discourse, they presumably express different aspects of compositional meaning than sentiment or nominal relations. See Hernault et al. (2010) for more details on discourse parsing and the RST-DT corpus.

Representations for adjacent EDUs are fed into binary classification (whether two EDUs are related) and multi-class relation classification models, as defined in Li et al. (2014). Related EDUs are then merged into a new EDU, the representation of which is obtained through an operation of neural composition based on the previous two related EDUs. This step is repeated until all units are merged.

Discourse parsing takes EDUs as the basic units to operate on; EDUs are short clauses, not full sentences, with an average length of 7.2 words. Recursive and recurrent models are applied on EDUs to create embeddings to be used as inputs for discourse parsing. We use this task for two reasons: (1) to illustrate whether syntactic parse trees are useful for acquiring representations for short clauses. (2) to measure the extent to which parsing improves discourse tasks that need to combine the meanings of larger text units.

Models are traditionally evaluated in terms of three metrics, i.e., spans[7], nuclearity[8], and identifying the rhetorical relation between two clauses. Due to space limits, we only focus the last one, rhetorical relation identification, because (1) relation labels are treated as correct only if spans and nuclearity are correctly labeled (2) relation identification between clauses offer more insights about model's abilities to represent sentence semantics. In order to perform a plain comparison, no additional human-developed features are added.

	Standard	LSTM
Tree	0.568	0.564
Sequence	0.572 (+0.004)	0.563 (-0.002)
P-value	0.160	0.422
Bi-Sequence	0.578 (+0.01)	0.575 (+0.012)
P-value	0.054	0.040*

Table 7: Test set accuracies for relation identification on RST discourse parsing data set.

Discussion We see no large differences between equivalent recurrent and recursive models. We suggest two possible explanations. (1) EDUs tend to be short; thus for some clauses, parsing might not change the order of operations on words. Even for those whose orders are changed by parse trees, the influence of short phrases on the final representation may not be great enough. (2) Unlike earlier tasks, where text representations are immediately used as inputs into classifiers, the algorithm presented here adopts additional levels of neural composition during the process of EDU merging. We suspect that neural layers may act as information filters, separating the informational chaff from the wheat, which in turn makes the model a bit more immune to the initial inputs.

4 Discussions and Conclusions

We compared recursive and recurrent neural models for representation learning on 5 distinct NLP tasks in 4 areas for which recursive neural models are known to achieve good performance (Socher et al., 2012; Socher et al., 2013; Li et al., 2014; Iyyer et al., 2014).

As with any comparison between models, our results come with some caveats: First, we explore the most general or basic forms of recur-

[7] on blank tree structures.

[8] on tree structures with nuclearity indication.

sive/recurrent models rather than various sophisticated algorithm variants. This is because fair comparison becomes more and more difficult as models get complex (e.g., the number of layers, number of hidden units within each layer, etc.). Thus most neural models employed in this work are comprised of only one layer of neural compositions—despite the fact that deep neural models with multiple layers give better results. Our conclusions might thus be limited to the algorithms employed in this paper, and it is unclear whether they can be extended to other variants or to the latest state-of-the-art. Second, in order to compare models "fairly", we force every model to be trained exactly in the same way: AdaGrad with minibatches, same set of initializations, etc. However, this may not necessarily be the optimal way to train every model; different training strategies tailored for specific models may improve their performances. In that sense, our attempts to be "fair" in this paper may nevertheless be unfair.

Pace these caveats, our conclusions can be summarized as follows:

- In tasks like semantic relation extraction, in which single headwords need to be associated across a long distance, recursive models shine. This suggests that for the many other kinds of tasks in which long-distance semantic dependencies play a role (e.g., translation between languages with significant reordering like Chinese-English translation), syntactic structures from recursive models may offer useful power.

- Tree models tend to help more on long sequences than shorter ones with sufficient supervision: tree models slightly help root level identification on the Stanford Sentiment Treebank, but do not help much at the phrase level. Adopting bi-directional versions of recurrent models seem to largely bridge this gap, producing equivalent or sometimes better results.

- On long sequences where supervision is not sufficient, e.g., in Pang at al.,'s dataset (supervision only exists on top of long sequences), no significant difference is observed between tree based and sequence based models.

- In cases where tree-based models do well, a simple approximation to tree-based models seems to improve recurrent models to equivalent or almost equivalent performance: (1) break long sentences (on punctuation) into a series of clause-like units, (2) work on these clauses separately, and (3) join them together. This model sometimes works as well as tree models for the sentiment task, suggesting that one of the reasons tree models help is by breaking down long sentences into more manageable units.

- Despite that the fact that components (outputs from different time steps) in recurrent models are not linguistically meaningful, they may do as well as linguistically meaningful phrases (represented by parse tree nodes) in embedding informative evidence, as demonstrated in UMD-QA task. Indeed, recent work in parallel with ours (Bowman et al., 2015) has shown that recurrent models like LSTMs can discover implicit recursive compositional structure.

5 Acknowledgments

We would especially like to thank Richard Socher and Kai-Sheng Tai for insightful comments, advice, and suggestions. We would also like to thank Sam Bowman, Ignacio Cases, Jon Gauthier, Kevin Gu, Gabor Angeli, Sida Wang, Percy Liang and other members of the Stanford NLP group, as well as the anonymous reviewers for their helpful advice on various aspects of this work. We acknowledge the support of NVIDIA Corporation with the donation of Tesla K40 GPUs We gratefully acknowledge support from an Enlight Foundation Graduate Fellowship, a gift from Bloomberg L.P., the Defense Advanced Research Projects Agency (DARPA) Deep Exploration and Filtering of Text (DEFT) Program under Air Force Research Laboratory (AFRL) contract no. FA8750-13-2-0040, and the NSF via award IIS-1514268. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of Bloomberg L.P., DARPA, AFRL, NSF, or the US government.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly

learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Samuel R Bowman, Christopher Potts, and Christopher D Manning. 2014. Recursive neural networks for learning logical semantics. *arXiv preprint arXiv:1406.1827*.

Samuel R Bowman, Christopher D Manning, and Christopher Potts. 2015. Tree-structured composition in neural networks without tree-structured architectures. *arXiv preprint arXiv:1506.04834*.

Samuel R Bowman. 2013. Can recursive neural tensor networks learn logical reasoning? *arXiv preprint arXiv:1312.6192*.

Lynn Carlson, Daniel Marcu, and Mary Ellen Okurowski. 2003. Building a discourse-tagged corpus in the framework of rhetorical structure theory. In *Current and New Directions in Discourse and Dialogue Text, Speech and Language Technology.* volume 22. Springer.

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

Andrew Cotter, Ohad Shamir, Nati Srebro, and Karthik Sridharan. 2011. Better mini-batch algorithms via accelerated gradient methods. In *Advances in Neural Information Processing Systems*, pages 1647–1655.

Li Dong, Furu Wei, Chuanqi Tan, Duyu Tang, Ming Zhou, and Ke Xu. 2014. Adaptive recursive neural network for target-dependent twitter sentiment classification. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 49–54.

Georg Dorffner. 1996. Neural networks for time series processing. In *Neural Network World*.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159.

Bradley Efron and Robert J Tibshirani. 1994. *An introduction to the bootstrap*. CRC press.

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

Christoph Goller and Andreas Kuchler. 1996. Learning task-dependent distributed representations by backpropagation through structure. In *Neural Networks, 1996., IEEE International Conference on*, volume 1, pages 347–352. IEEE.

Alex Graves and Juergen Schmidhuber. 2009. Offline handwriting recognition with multidimensional recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 545–552.

Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 6645–6649. IEEE.

Alex Graves. 2012. Supervised sequence labeling with recurrent neural networks, In *Studies in Computational Intelligence.* volume 385. Springer.

Kazuma Hashimoto, Makoto Miwa, Yoshimasa Tsuruoka, and Takashi Chikayama. 2013. Simple customization of recursive neural networks for semantic relation classification. In *EMNLP*, pages 1372–1376.

Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. 2009. Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In *Proceedings of the Workshop on Semantic Evaluations: Recent Achievements and Future Directions*, pages 94–99. Association for Computational Linguistics.

Hugo Hernault, Helmut Prendinger, Mitsuru Ishizuka. 2010. Hilda: a discourse parser using support vector machine classification. *Dialogue & Discourse*, 1(3).

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

Ozan Irsoy and Claire Cardie. 2013. Bidirectional recursive neural networks for token-level labeling with structure. *arXiv preprint arXiv:1312.0493*.

Ozan Irsoy and Claire Cardie. 2014. Deep recursive neural networks for compositionality in language. In *Advances in Neural Information Processing Systems*, pages 2096–2104.

Mohit Iyyer, Jordan Boyd-Graber, Leonardo Claudino, Richard Socher, and Hal Daumé III. 2014. A neural network for factoid question answering over paragraphs. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 633–644.

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

Quoc V Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. *arXiv preprint arXiv:1405.4053*.

Jiwei Li and Eduard Hovy. 2014. A model of coherence based on distributed sentence representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*

Jiwei Li, Rumeng Li, and Eduard Hovy. 2014. Recursive deep models for discourse parsing. In *Proceedings of the 2014 Conference on Empirical Methods*

in Natural Language Processing (EMNLP), pages 2061–2069.

Richard P Lippmann. 1989. Review of neural networks for speech recognition. *Neural computation*, 1(1):1–38.

Thang Luong, Ilya Sutskever, Quoc V Le, Oriol Vinyals, and Wojciech Zaremba. 2014. Addressing the rare word problem in neural machine translation. *Proceedings of ACL*. 2015.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013. Linguistic regularities in continuous space word representations. In *HLT-NAACL*, pages 746–751.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 79–86. Association for Computational Linguistics.

Romain Paulus, Richard Socher, and Christopher D Manning. 2014. Global belief recursive neural networks. In *Advances in Neural Information Processing Systems*, pages 2888–2896.

Barak A Pearlmutter. 1989. Learning state space trajectories in recurrent neural networks. *Neural Computation*, 1(2):263–269.

Tony Robinson, Mike Hochberg, and Steve Renals. 1996. The use of recurrent neural networks in continuous speech recognition. In *Automatic speech and speaker recognition*, pages 233–258. Springer.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *Signal Processing, IEEE Transactions on*, 45(11):2673–2681.

Richard Socher, Eric H Huang, Jeffrey Pennin, Christopher D Manning, and Andrew Y Ng. 2011a. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Advances in Neural Information Processing Systems*, pages 801–809.

Richard Socher, Jeffrey Pennington, Eric H Huang, Andrew Y Ng, and Christopher D Manning. 2011b. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 151–161. Association for Computational Linguistics.

Richard Socher, Brody Huval, Christopher D Manning, and Andrew Y Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1201–1211. Association for Computational Linguistics.

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

Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. Improved semantic representations from tree-structured long short-term memory networks. *ACL*. 2015.

Oriol Vinyals, Lukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2014. Grammar as a foreign language. *arXiv preprint arXiv:1412.7449*.

Xiaodan Zhu, Parinaz Sobihani, and Hongyu Guo. 2015. Long short-term memory over recursive structures. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 1604–1612.

Discriminative Neural Sentence Modeling by Tree-Based Convolution

Lili Mou[*], Hao Peng[*], Ge Li[†], Yan Xu, Lu Zhang, Zhi Jin
{doublepower.mou, penghao.pku}@gmail.com
{lige, xuyan14, zhanglu, zhijin}@sei.pku.edu.cn
Software Institute, Peking University, 100871, P. R. China

Abstract

This paper proposes a tree-based convolutional neural network (TBCNN) for discriminative sentence modeling. Our model leverages either constituency trees or dependency trees of sentences. The tree-based convolution process extracts sentences structural features, which are then aggregated by max pooling. Such architecture allows short propagation paths between the output layer and underlying feature detectors, enabling effective structural feature learning and extraction. We evaluate our models on two tasks: sentiment analysis and question classification. In both experiments, TBCNN outperforms previous state-of-the-art results, including existing neural networks and dedicated feature/rule engineering. We also make efforts to visualize the tree-based convolution process, shedding light on how our models work.

1 Introduction

Discriminative sentence modeling aims to capture sentence meanings, and classify sentences according to certain criteria (e.g., sentiment). It is related to various tasks of interest, and has attracted much attention in the NLP community (Allan et al., 2003; Su and Markert, 2008; Zhao et al., 2015). Feature engineering—for example, n-gram features (Cui et al., 2006), dependency subtree features (Nakagawa et al., 2010), or more dedicated ones (Silva et al., 2011)—can play an important role in modeling sentences. Kernel machines, e.g., SVM, are exploited in Moschitti (2006) and Reichartz et al. (2010) by specifying a certain measure of similarity between sentences, without explicit feature representation.

Recent advances of neural networks bring new techniques in understanding natural languages, and have exhibited considerable potential. Bengio et al. (2003) and Mikolov et al. (2013) propose unsupervised approaches to learn word embeddings, mapping discrete words to real-valued vectors in a meaning space. Le and Mikolov (2014) extend such approaches to learn sentences' and paragraphs' representations. Compared with human engineering, neural networks serve as a way of automatic feature learning (Bengio et al., 2013).

Two widely used neural sentence models are convolutional neural networks (CNNs) and recursive neural networks (RNNs). CNNs can extract words' neighboring features effectively with short propagation paths, but they do not capture inherent sentence structures (e.g., parse trees). RNNs encode, to some extent, structural information by recursive semantic composition along a parse tree. However, they may have difficulties in learning deep dependencies because of long propagation paths (Erhan et al., 2009). (CNNs/RNNs and a variant, recurrent networks, will be reviewed in Section 2.)

A curious question is whether we can combine the advantages of CNNs and RNNs, i.e., whether we can exploit sentence structures (like RNNs) effectively with short propagation paths (like CNNs).

In this paper, we propose a novel neural architecture for discriminative sentence modeling, called the *Tree-Based Convolutional Neural Network* (TBCNN).[1] Our models can leverage different sentence parse trees, e.g., constituency trees and dependency trees. The model variants are denoted as c-TBCNN and d-TBCNN, respectively. The idea of tree-based convolution is to apply a set of subtree feature detectors, sliding over the entire

[*]These authors contribute equally to this paper.
[†]To whom correspondence should be addressed.

[1]The model of tree-based convolution was firstly proposed to process program source code in our (unpublished) previous work (Mou et al., 2014).

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 2315–2325,
Lisbon, Portugal, 17-21 September 2015.

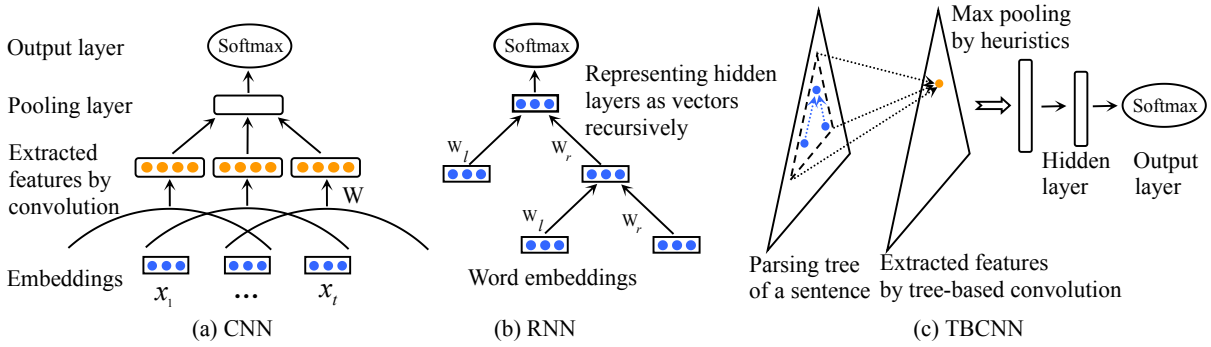


Figure 1: A comparison of information flow in the convolutional neural network (CNN), the recursive neural network (RNN), and the tree-based convolutional neural network (TBCNN).

parse tree of a sentence; then pooling aggregates these extracted feature vectors by taking the maximum value in each dimension. One merit of such architecture is that all features, along the tree, have short propagation paths to the output layer, and hence structural information can be learned effectively.

TBCNNs are evaluated on two tasks, sentiment analysis and question classification; our models have outperformed previous state-of-the-art results in both experiments. To understand how TBCNNs work, we also visualize the network by plotting the convolution process. We make our code and results available on our project website.[2]

2 Background and Related Work

In this section, we present the background and related work regarding two prevailing neural architectures for discriminative sentence modeling.

2.1 Convolutional Neural Networks

Convolutional neural networks (CNNs), early used for image processing (LeCun, 1995), turn out to be effective with natural languages as well. Figure 1a depicts a classic convolution process on a sentence (Collobert and Weston, 2008). A set of fixed-width-window feature detectors slide over the sentence, and output the extracted features. Let t be the window size, and $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_t \in \mathbb{R}^{n_e}$ be n_e-dimensional word embeddings. The output of convolution, evaluated at the current position, is

$$\boldsymbol{y} = f\left(W \cdot [\boldsymbol{x}_1; \cdots; \boldsymbol{x}_t] + \boldsymbol{b}\right)$$

where $\boldsymbol{y} \in \mathbb{R}^{n_c}$ (n_c is the number of feature detectors). $W \in \mathbb{R}^{n_c \times (t \cdot n_e)}$ and $\boldsymbol{b} \in \mathbb{R}^{n_c}$ are parameters; f is the activation function. Semicolons represent column vector concatenation. After convolution, the extracted features are pooled to a fixed-size vector for classification.

Convolution can extract neighboring information effectively. However, the features are "local"—words that are not in a same convolution window do not interact with each other, even though they may be semantically related. Blunsom et al. (2014) build deep convolutional networks so that local features can mix at high-level layers. Similar CNNs include Kim (2014) and Hu et al. (2014). All these models are "flat," by which we mean no structural information is used explicitly.

2.2 Recursive Neural Networks

Recursive neural networks (RNNs), proposed in Socher et al. (2011b), utilize sentence parse trees. In the original version, RNN is built upon a binarized constituency tree. Leaf nodes correspond to words in a sentence, represented by n_e-dimensional embeddings. Non-leaf nodes are sentence constituents, coded by child nodes recursively. Let node p be the parent of c_1 and c_2, vector representations denoted as $\boldsymbol{p}$, $\boldsymbol{c}_1$, and $\boldsymbol{c}_2$. The parent's representation is composited by

$$\boldsymbol{p} = f(W \cdot [\boldsymbol{c}_1; \boldsymbol{c}_2] + \boldsymbol{b}) \quad (1)$$

where W and $\boldsymbol{b}$ are parameters. This process is done recursively along the tree; the root vector is then used for supervised classification (Figure 1b).

Dependency parse and the combinatory categorical grammar can also be exploited as RNNs' skeletons (Hermann and Blunsom, 2013; Iyyer et al., 2014). Irsoy and Cardie (2014) build deep RNNs to enhance information interaction. Im-

[2] https://sites.google.com/site/tbcnnsentence/

provements for semantic compositionality include matrix-vector interaction (Socher et al., 2012), tensor interaction (Socher et al., 2013). They are more suitable for capturing logical information in sentences, such as negation and exclamation.

One potential problem of RNNs is that the long propagation paths—through which leaf nodes are connected to the output layer—may lead to information loss. Thus, RNNs bury illuminating information under a complicated neural architecture. Further, during back-propagation over a long path, gradients tend to vanish (or blow up), which makes training difficult (Erhan et al., 2009). Long short term memory (LSTM), first proposed for modeling time-series data (Hochreiter and Schmidhuber, 1997), is integrated to RNNs to alleviate this problem (Tai et al., 2015; Le and Zuidema, 2015; Zhu et al., 2015).

Recurrent networks. A variant class of RNNs is the recurrent neural network (Bengio et al., 1994; Shang et al., 2015), whose architecture is a rightmost tree. In such models, meaningful tree structures are also lost, similar to CNNs.

3 Tree-based Convolution

This section introduces the proposed tree-based convolutional neural networks (TBCNNs). Figure 1c depicts the convolution process on a tree.

First, a sentence is converted to a parse tree, either a constituency or dependency tree. The corresponding model variants are denoted as c-TBCNN and d-TBCNN. Each node in the tree is represented as a distributed, real-valued vector.

Then, we design a set of fixed-depth subtree feature detectors, called the *tree-based convolution window*. The window slides over the entire tree to extract structural information of the sentence, illustrated by a dashed triangle in Figure 1c. Formally, let us assume we have t nodes in the convolution window, $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_t$, each represented as an n_e-dimensional vector. Let n_c be the number of feature detectors. The output of the tree-based convolution window, evaluated at the current subtree, is given by the following generic equation.

$$\boldsymbol{y} = f\left(\sum_{i=1}^{t} W_i \cdot \boldsymbol{x}_i + \boldsymbol{b}\right) \tag{2}$$

where $W_i \in \mathbb{R}^{n_c \times n_e}$ is the weight parameter associated with node x_i; $\boldsymbol{b} \in \mathbb{R}^{n_c}$ is the bias term.

Extracted features are thereafter packed into one or more fixed-size vectors by max pooling, that is, the maximum value in each dimension is taken. Finally, we add a fully connected hidden layer, and a $\mathrm{softmax}$ output layer.

From the designed architecture (Figure 1c), we see that our TBCNN models allow short propagation paths between the output layer and any position in the tree. Therefore structural feature learning becomes effective.

Several main technical points in tree-based convolution include: (1) How can we represent hidden nodes as vectors in constituency trees? (2) How can we determine weights, W_i, for dependency trees, where nodes may have different numbers of children? (3) How can we pool varying sized and shaped features to fixed-size vectors?

In the rest of this section, we explain model variants in detail. Particularly, Subsections 3.1 and 3.2 address the first and second problems; Subsection 3.3 deals with the third problem by introducing several pooling heuristics. Subsection 3.4 presents our training objective.

3.1 c-TBCNN

Figure 2a illustrates an example of the constituency tree, where leaf nodes are words in the sentence, and non-leaf nodes represent a grammatical constituent, e.g., a noun phrase. Sentences are parsed by the Stanford parser;[3] further, constituency trees are binarized for simplicity.

One problem of constituency trees is that non-leaf nodes do not have such vector representations as word embeddings. Our strategy is to pretrain the constituency tree with an RNN by Equation 1 (Socher et al., 2011b). After pretraining, vector representations of nodes are fixed.

We now consider the tree-based convolution process in c-TBCNN with a two-layer-subtree convolution window, which operates on a parent node p and its direct children c_l and c_r, their vector representations denoted as $\boldsymbol{p}$, $\boldsymbol{c}_l$, and $\boldsymbol{c}_r$. The convolution equation, specific for c-TBCNN, is

$$\boldsymbol{y} = f\left(W_p^{(c)} \cdot \boldsymbol{p} + W_l^{(c)} \cdot \boldsymbol{c}_l + W_r^{(c)} \cdot \boldsymbol{c}_r + \boldsymbol{b}^{(c)}\right)$$

where $W_p^{(c)}$, $W_l^{(c)}$, and $W_r^{(c)}$ are weights associated with the parent and its child nodes. Superscript (c) indicates that the weights are for c-TBCNN. For leaf nodes, which do not have children, we set $\boldsymbol{c}_l$ and $\boldsymbol{c}_r$ to be $\mathbf{0}$.

[3] http://nlp.stanford.edu/software/lex-parser.shtml

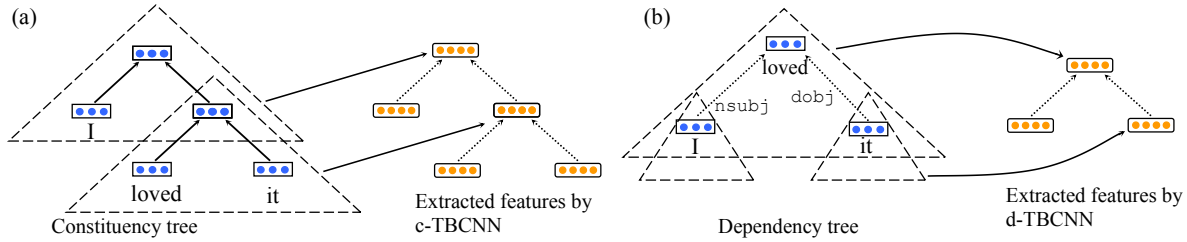


Figure 2: Tree-based convolution in (a) c-TBCNN, and (b) d-TBCNN. The parse trees correspond to the sentence "I loved it." The dashed triangles illustrate a shared-weight convolution window sliding over the tree. For clarity, only two positions are drawn in c-TBCNN. Notice that dotted arrows are not part of neural connections; they merely indicate the topologies of tree structures. Specially, an edge $a \xrightarrow{r} b$ in the dependency tree refers to a being governed by b with dependency type r.

Tree-based convolution windows can be extended to arbitrary depths straightforwardly. The complexity is exponential to the depth of the window, but linear to the number of nodes. Hence, tree-based convolution, compared with "flat" CNNs, does not add to computational cost, provided the same amount of information to process at a time. In our experiments, we use convolution windows of depth 2.

3.2 d-TBCNN

Dependency trees are another representation of sentence structures. The nature of dependency representation leads to d-TBCNN's major difference from traditional convolution: there exist nodes with different numbers of child nodes. This causes trouble if we associate weight parameters according to positions in the window, which is standard for traditional convolution, e.g., Collobert and Weston (2008) or c-TBCNN.

To overcome the problem, we extend the notion of convolution by assigning weights according to dependency types (e.g, `nsubj`) rather than positions. We believe this strategy makes much sense because dependency types (de Marneffe et al., 2006) reflect the relationship between a governing word and its child words. To be concrete, the generic convolution formula (Equation 2) for d-TBCNN becomes

$$\boldsymbol{y} = f\left(W_p^{(d)} \cdot \boldsymbol{p} + \sum_{i=1}^{n} W_{r[c_i]}^{(d)} \cdot \boldsymbol{c}_i + \boldsymbol{b}^{(d)}\right)$$

where $W_p^{(d)}$ is the weight parameter for the parent p (governing word); $W_{r[c_i]}^{(d)}$ is the weight for child c_i, who has grammatical relationship $r[c_i]$ to its parent, p. Superscript (d) indicates the parameters are for d-TBCNN. Note that we keep 15 most frequently occurred dependency types; others appearing rarely in the corpus are mapped to one shared weight matrix.

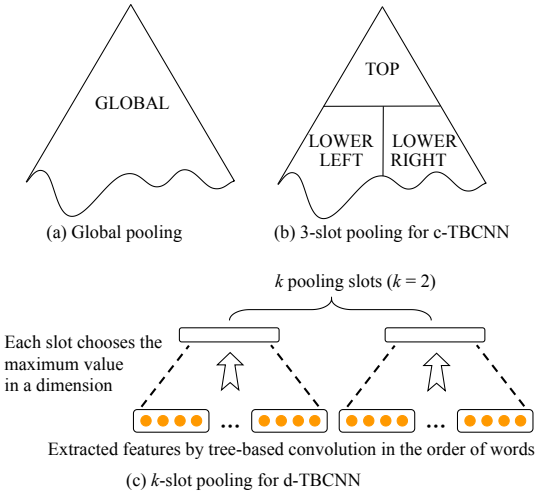


Figure 3: Pooling heuristics. (a) Global pooling. (b) 3-slot pooling for c-TBCNN. (c) k-slot pooling for d-TBCNN.

Both c-TBCNN and d-TBCNN have their own advantages: d-TBCNN exploits structural features more efficiently because of the compact expressiveness of dependency trees; c-TBCNN may be more effective in integrating global features due to the underneath pretrained RNN.

3.3 Pooling Heuristics

As different sentences may have different lengths and tree structures, the extracted features by tree-based convolution also have topologies varying in size and shape. Dynamic pooling (Socher et al., 2011a) is a common technique for dealing with

Task	Data samples	Label
Sentiment Analysis	Offers that rare combination of entertainment and education.	++
	An idealistic love story that brings out the latent 15-year-old romantic in everyone.	+
	Its mysteries are transparently obvious, and it's too slowly paced to be a thriller.	−
Question Classification	What is the temperature at the center of the earth?	`number`
	What state did the Battle of Bighorn take place in?	`location`

Table 1: Data samples in sentiment analysis and question classification. In the first task, "++" refers to strongly positive; "+" and "−" refer to positive and negative, respectively.

this problem. We propose several heuristics for pooling along a tree structure. Our generic design criteria for pooling include: (1) Nodes that are pooled to one slot should be "neighboring" from some viewpoint. (2) Each slot should have similar numbers of nodes, in expectation, that are pooled to it. Thus, (approximately) equal amount of information is aggregated along different parts of the tree. Following the above intuition, we propose pooling heuristics as follows.

- Global pooling. All features are pooled to one vector, shown in Figure 3a. We take the maximum value in each dimension. This simple heuristic is applicable to any structure, including c-TBCNN and d-TBCNN.
- 3-slot pooling for c-TBCNN. To preserve more information over different parts of constituency trees, we propose 3-slot pooling (Figure 3b). If a tree has maximum depth d, we pool nodes of less than $\alpha \cdot d$ layers to a TOP slot (α is set to 0.6); lower nodes are pooled to slots LOWER_LEFT or LOWER_RIGHT according to their relative position with respect to the root node.
  For a constituency tree, it is not completely obvious how to pool features to more than 3 slots and comply with the aforementioned criteria at the same time. Therefore, we regard 3-slot pooling for c-TBCNN is a "hard mechanism" temporarily. Further improvement can be addressed in future work.
- k-slot pooling for d-TBCNN. Different from constituency trees, nodes in dependency trees are one-one corresponding to words in a sentence. Thus, a total order on features (after convolution) can be defined according to their corresponding word orders. For k-slot pooling, we can adopt an "equal allocation" strategy, shown in Figure 3c. Let i be the position of a word in a sentence ($i = 1, 2, \cdots, n$). Its extracted feature vector is pooled to the j-th slot, if

$$(j-1)\frac{n}{k} \leq i \leq j\frac{n}{k}$$

We assess the efficacy of pooling quantitatively in Section 4.3.1. As we shall see by the experimental results, complicated pooling methods do preserve more information along tree structures to some extent, but the effect is not large. TBCNNs are not very sensitive to pooling methods.

3.4 Training Objective

After pooling, information is packed into one or more fixed-size vectors (slots). We add a hidden layer, and then a $\mathrm{softmax}$ layer to predict the probability of each target label in a classification task. The error function of a sample is the standard cross entropy loss, i.e., $J = -\sum_{i=1}^{c} t_i \log y_i$, where $\boldsymbol{t}$ is the ground truth (one-hot represented), $\boldsymbol{y}$ the output by $\mathrm{softmax}$, and c the number of classes. To regularize our model, we apply both ℓ_2 penalty and dropout (Srivastava et al., 2014). Training details are further presented in Section 4.1 and 4.2.

4 Experimental Results

In this section, we evaluate our models with two tasks, sentiment analysis and question classification. We also conduct quantitative and qualitative model analysis in Subsection 4.3.

4.1 Sentiment Analysis

4.1.1 The Task and Dataset

Sentiment analysis is a widely studied task for discriminative sentence modeling. The Stanford sentiment treebank[4] consists of more than 10,000 movie reviews. Two settings are considered for sentiment prediction: (1) fine-grained classification with 5 labels (`strongly positive`, `positive`, `neutral`, `negative`, and `strongly negative`), and (2) coarse-gained polarity classification with 2 labels (`positive` versus `negative`). Some examples are shown in

[4] http://nlp.stanford.edu/sentiment/

Table 1. We use the standard split for training, validating, and testing, containing 8544/1101/2210 sentences for 5-class prediction. Binary classification does not contain the `neutral` class.

In the dataset, phrases (sub-sentences) are also tagged with sentiment labels. RNNs deal with them naturally during the recursive process. We regard sub-sentences as individual samples during training, like Blunsom et al. (2014) and Le and Mikolov (2014). The training set therefore has more than 150,000 entries in total. For validating and testing, only whole sentences (root labels) are considered in our experiments.

Both c-TBCNN and d-TBCNN use the Stanford parser for data preprocessing.

4.1.2 Training Details

This subsection describes training details for d-TBCNN, where hyperparameters are chosen by validation. c-TBCNN is mostly tuned synchronously (e.g., optimization algorithm, activation function) with some changes in hyperparameters. c-TBCNN's settings can be found on our website.

In our d-TBCNN model, the number of units is 300 for convolution and 200 for the last hidden layer. Word embeddings are 300 dimensional, pretrained ourselves using `word2vec` (Mikolov et al., 2013) on the English Wikipedia corpus. 2-slot pooling is applied for d-TBCNN. (c-TBCNN uses 3-slot pooling.)

To train our model, we compute gradient by back-propagation and apply stochastic gradient descent with mini-batch 200. We use ReLU (Nair and Hinton, 2010) as the activation function .

For regularization, we add ℓ_2 penalty for weights with a coefficient of 10^{-5}. Dropout (Srivastava et al., 2014) is further applied to both weights and embeddings. All hidden layers are dropped out by 50%, and embeddings 40%.

4.1.3 Performance

Table 2 compares our models to state-of-the-art results in the task of sentiment analysis. For 5-class prediction, d-TBCNN yields 51.4% accuracy, outperforming the previous state-of-the-art result, achieved by the RNN based on long-short term memory (Tai et al., 2015). c-TBCNN is slightly worse. It achieves 50.4% accuracy, ranking third in the state-of-the-art list (including our d-TBCNN model).

Regarding 2-class prediction, we adopted a simple strategy in Irsoy and Cardie (2014),[5] where the 5-class network is "transferred" directly for binary classification, with estimated target probabilities (by 5-way $\mathrm{softmax}$) reinterpreted for 2 classes. (The `neutral` class is discarded as in other studies.) This strategy enables us to take a glance at the stability of our TBCNN models, but places itself in a difficult position. Nonetheless, our d-TBCNN model achieves 87.9% accuracy, ranking forth in the list.

In a more controlled comparison—with shallow architectures and the basic interaction (linearly transformed and non-linearly squashed)—TBCNNs, of both variants, consistently outperform RNNs (Socher et al., 2011b) to a large extent (50.4–51.4% versus 43.2%); they also consistently outperform "flat" CNNs by more than 10%. Such results show that structures are important when modeling sentences; tree-based convolution can capture these structural information more effectively than RNNs.

We also observe d-TBCNN achieves higher performance than c-TBCNN. This suggests that compact tree expressiveness is more important than integrating global information in this task.

4.2 Question Classification

We further evaluate TBCNN models on a question classification task.[6] The dataset contains 5452 annotated sentences plus 500 test samples in TREC 10. We also use the standard split, like Silva et al. (2011). Target labels contain 6 classes, namely `abbreviation`, `entity`, `description`, `human`, `location`, and `numeric`. Some examples are also shown in Table 1.

We chose this task to evaluate our models because the number of training samples is rather small, so that we can know TBCNNs' performance when applied to datasets of different sizes. To alleviate the problem of data sparseness, we set the dimensions of convolutional layer and the last hidden layer to 30 and 25, respectively. We do not back-propagate gradient to embeddings in this

[5] Richard Socher, who first applies neural networks to this task, thinks direct transfer is fine for binary classification. We followed this strategy for simplicity as it is non-trivial to deal with the neutral sub-sentences in the training set if we train a separate model. Our website reviews some related work and provides more discussions.

[6] http://cogcomp.cs.illinois.edu/Data/QA/QC/

Group	Method	5-class accuracy	2-class accuracy	Reported in
Baseline	SVM	40.7	79.4	Socher et al. (2013)
	Naïve Bayes	41.0	81.8	Socher et al. (2013)
CNNs	1-layer convolution	37.4	77.1	Blunsom et al. (2014)
	Deep CNN	48.5	86.8	Blunsom et al. (2014)
	Non-static	48.0	87.2	Kim (2014)
	Multichannel	47.4	**88.1**	Kim (2014)
RNNs	Basic	43.2	82.4	Socher et al. (2013)
	Matrix-vector	44.4	82.9	Socher et al. (2013)
	Tensor	45.7	85.4	Socher et al. (2013)
	Tree LSTM (variant 1)	48.0	–	Zhu et al. (2015)
	Tree LSTM (variant 2)	51.0	88.0	Tai et al. (2015)
	Tree LSTM (variant 3)	49.9	88.0	Le and Zuidema (2015)
	Deep RNN	49.8	86.6†	Irsoy and Cardie (2014)
Recurrent	LSTM	45.8	86.7	Tai et al. (2015)
	bi-LSTM	49.1	86.8	Tai et al. (2015)
Vector	Word vector avg.	32.7	80.1	Socher et al. (2013)
	Paragraph vector	48.7	87.8	Le and Mikolov (2014)
TBCNNs	c-TBCNN	50.4	86.8†	Our implementation
	d-TBCNN	**51.4**	87.9†	Our implementation

Table 2: Accuracy of sentiment prediction (in percentage). For 2-class prediction, "†" remarks indicate that the network is transferred directly from that of 5-class.

Method	Acc. (%)	Reported in
SVM 10k features + 60 rules	95.0	Silva et al. (2011)
CNN-non-static	93.6	Kim (2014)
CNN-mutlichannel	92.2	Kim (2014)
RNN	90.2	Zhao et al. (2015)
Deep-CNN	93.0	Blunsom et al. (2014)
Ada-CNN	92.4	Zhao et al. (2015)
c-TBCNN	94.8	Our implementation
d-TBCNN	**96.0**	Our implementation

Table 3: Accuracy of 6-way question classification.

task. Dropout rate for embeddings is 30%; hidden layers are dropped out by 5%.

Table 3 compares our models to various other methods. The first entry presents the previous state-of-the-art result, achieved by traditional feature/rule engineering (Silva et al., 2011). Their method utilizes more than 10k features and 60 hand-coded rules. On the contrary, our TBCNN models do not use a single human-engineered feature or rule. Despite this, c-TBCNN achieves similar accuracy compared with feature engineering; d-TBCNN pushes the state-of-the-art result to 96%. To the best of our knowledge, this is the first time that neural networks beat dedicated human engineering in this question classification task.

The result also shows that both c-TBCNN and d-TBCNN reduce the error rate to a large extent, compared with other neural architectures in this task.

4.3 Model Analysis

In this part, we analyze our models quantitatively and qualitatively in several aspects, shedding some light on the mechanism of TBCNNs.

4.3.1 The Effect of Pooling

The extracted features by tree-based convolution have topologies varying in size and shape. We propose in Section 3.3 several heuristics for pooling. This subsection aims to provide a fair comparison among these pooling methods.

One reasonable protocol for comparison is to tune all hyperparameters for each setting and compare the highest accuracy. This methodology, however, is too time-consuming, and depends largely on the quality of hyperparameter tuning. An alternative is to predefine a set of sensible hyperparameters and report the accuracy under the same setting. In this experiment, we chose the latter protocol, where hidden layers are all 300-dimensional; no ℓ_2 penalty is added. Each configuration was run five times with different random initializations. We summarize the mean and standard deviation in Table 4.

As the results imply, complicated pooling is better than global pooling to some degree for both model variants. But the effect is not strong; our models are not that sensitive to pooling methods, which mainly serve as a necessity for dealing with varying-structure data. In our experiments, we apply 3-slot pooling for c-TBCNN and 2-slot pooling for d-TBCNN.

Model	Pooling method	5-class accuracy (%)
c-TBCNN	Global	48.48 ± 0.54
	3-slot	48.69 ± 0.40
d-TBCNN	Global	49.39 ± 0.24
	2-slot	49.94 ± 0.63

Table 4: Accuracies of different pooling methods, averaged over 5 random initializations. We chose sensible hyperparameters manually in advance to make a fair comparison. This leads to performance degradation (1–2%) vis-a-vis Table 2.

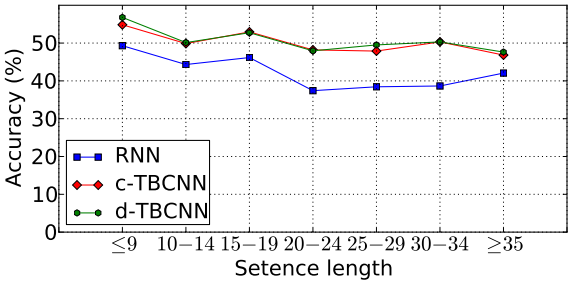


Figure 4: Accuracies versus sentence lengths.

Comparing with other studies in the literature, we also notice that pooling is very effective and efficient in information gathering. Irsoy and Cardie (2014) report 200 epochs for training a deep RNN, which achieves 49.8% accuracy in the 5-class sentiment classification. Our TBCNNs are typically trained within 25 epochs.

4.3.2 The Effect of Sentence Lengths

We analyze how sentence lengths affect our models. Sentences are split into 7 groups by length, with granularity 5. A few too long or too short sentences are grouped together for smoothing; the numbers of sentences in each group vary from 126 to 457. Figure 4 presents accuracies versus lengths in TBCNNs. For comparison, we also reimplemented RNN, achieving 42.7% overall accuracy, slightly worse than 43.2% reported in Socher et al. (2011b). Thus, we think our reimplementation is fair and that the comparison is sensible.

We observe that c-TBCNN and d-TBCNN yield very similar behaviors. They consistently outperform the RNN in all scenarios. We also notice the gap, between TBCNNs and RNN, increases when sentences contain more than 20 words. This result confirms our theoretical analysis in Section 2—for long sentences, the propagation paths in RNNs are deep, causing RNNs' difficulty in information processing. By contrast, our models explore structural information more effectively with tree-based convolution. As information from any part of the tree can propagate to the output layer with short paths, TBCNNs are more capable for sentence modeling, especially for long sentences.

4.3.3 Visualization

Visualization is important to understanding the mechanism of neural networks. For TBCNNs, we would like to see how the extracted features (after convolution) are further processed by the max pooling layer, and ultimately related to the supervised task.

To show this, we trace back where the max pooling layer's features come from. For each dimension, the pooling layer chooses the maximum value from the nodes that are pooled to it. Thus, we can count the fraction in which a node's features are gathered by pooling. Intuitively, if a node's features are more related to the task, the fraction tends to be larger, and vice versa.

Figure 5 illustrates an example processed by d-TBCNN in the task of sentiment analysis.[7] Here, we applied global pooling because information tracing is more sensible with one pooling slot. As shown in the figure, tree-based convolution can effectively extract information relevant to the task of interest. The 2-layer windows corresponding to "*visual will impress viewers*," "*the stunning dreamlike visual*," say, are discriminative to the sentence's sentiment. Hence, large fractions (0.24 and 0.19) of their features, after convolution, are gathered by pooling. On the other hand, words like *the*, *will*, *even* are known as stop words (Fox, 1989). They are mostly noninformative for sentiment; hence, no (or minimal) features are gathered. Such results are consistent with human intuition.

We further observe that tree-based convolution does integrate information of different words in the window. For example, the word *stunning* appears in two windows: (a) the window "*stunning*" itself, and (b) the window of "*the stunning dreamlike visual*," with root node *visual*, *stunning* acting as a child. We see that Window b is more relevant to the ultimate sentiment than Window a, with fractions 0.19 versus 0.07, even though the root *visual* itself is neutral in sentiment. In fact,

[7] We only have space to present one example in the paper. This example was not chosen deliberately. Similar traits can be found through out the entire gallery, available on our website. Also, we only present d-TBCNN, noticing that dependency trees are intrinsically more suitable for visualization since we know the "meaning" of every node.

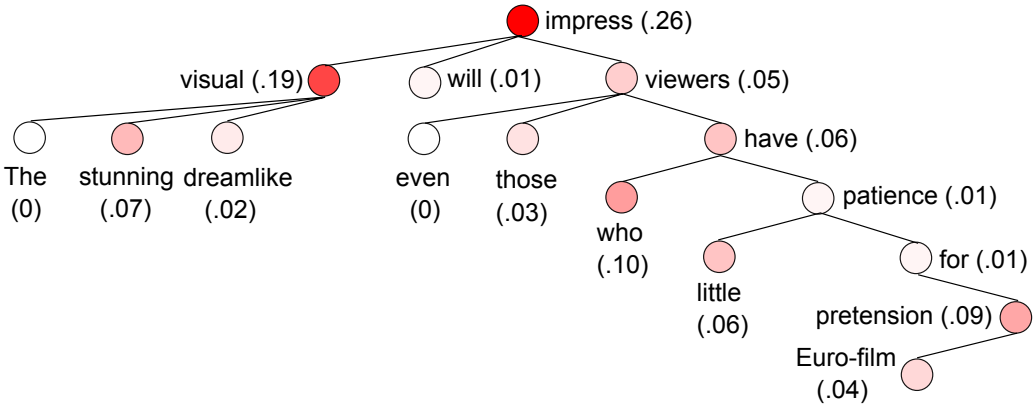


Figure 5: Visualizing how features (after convolution) are related to the sentiment of a sentence. The sample corresponds a sentence in the dataset, "The stunning dreamlike visual will impress even those viewers who have little patience for Euro-film pretension." The numbers in brackets denote the fraction of a node's features that are gathered by the max pooling layer (also indicated by colors).

Window a has a larger fraction than the sum of its children's (the windows of "*the*," "*stunning*," and "*dreamlike*").

5 Conclusion

In this paper, we proposed a novel neural discriminative sentence model based on sentence parsing structures. Our model can be built upon either constituency trees (denoted as c-TBCNN) or dependency trees (d-TBCNN).

Both variants have achieved high performance in sentiment analysis and question classification. d-TBCNN is slightly better than c-TBCNN in our experiments, and has outperformed previous state-of-the-art results in both tasks. The results show that tree-based convolution can capture sentences' structural information effectively, which is useful for sentence modeling.

Acknowledgments

This research is supported by the National Basic Research Program of China (the 973 Program) under Grant No. 2015CB352201 and the National Natural Science Foundation of China under Grant Nos. 61232015 and 91318301.

References

James Allan, Courtney Wade, and Alvaro Bolivar. 2003. Retrieval and novelty detection at the sentence level. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval*, pages 314–321. ACM.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166.

Yoshua Bengio, Rejean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155.

Yoshua Bengio, Aaron Courville, and Pierre Vincent. 2013. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828.

Phil Blunsom, Edward Grefenstette, and Nal Kalchbrenner. 2014. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine learning*, pages 160–167.

Hang Cui, Vibhu Mittal, and Mayur Datar. 2006. Comparative experiments on sentiment classification for online product reviews. In *Proceedings 21st AAAI Conference on Artificial Intelligence*, volume 6, pages 1265–1270.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of Language Resource and Evaluation Conference*, volume 6, pages 449–454.

Dumitru Erhan, Pierre-Antoine Manzagol, Yoshua Bengio, Samy Bengio, and Pascal Vincent. 2009. The difficulty of training deep architectures and the

effect of unsupervised pre-training. In *Proceedings of International Conference on Artificial Intelligence and Statistics*, pages 153–160.

Christopher Fox. 1989. A stop list for general text. In *ACM SIGIR Forum*, volume 24, pages 19–21.

Karl M. Hermann and Phil Blunsom. 2013. The role of syntax in vector space models of compositional semantics. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 894–904.

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

Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. In *Advances in Neural Information Processing Systems*, pages 2042–2050.

Ozan Irsoy and Claire Cardie. 2014. Deep recursive neural networks for compositionality in language. In *Advances in Neural Information Processing Systems*, pages 2096–2104.

Mohit Iyyer, Jordan Boyd-Graber, Leonardo Claudino, Richard Socher, and Hal Daumé III. 2014. A neural network for factoid question answering over paragraphs. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 633–644.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*.

Quoc V. Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning*.

Phong Le and Willem Zuidema. 2015. Compositional distributional semantics with long short term memory. *arXiv preprint arXiv:1503.02510*.

Yann LeCun. 1995. Comparison of learning algorithms for handwritten digit recognition. In *Proceedings of International Conference on Artificial Neural Networks*, volume 60, pages 53–60.

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

Alessandro Moschitti. 2006. Efficient convolution kernels for dependency and constituent syntactic trees. In *Proceedings of European Conference of Machine Learning*, pages 318–329. Springer.

Lili Mou, Ge Li, Zhi Jin, Lu Zhang, and Tao Wang. 2014. TBCNN: A tree-based convolutional neural network for programming language processing. *arXiv preprint arXiv:1409.5718*.

Vinod Nair and Geoffrey E. Hinton. 2010. Rectified linear units improve restricted Boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning*, pages 807–814.

Tetsuji Nakagawa, Kentaro Inui, and Sadao Kurohashi. 2010. Dependency tree-based sentiment classification using CRFs with hidden variables. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 786–794.

Frank Reichartz, Hannes Korte, and Gerhard Paass. 2010. Semantic relation extraction with kernels over typed dependency trees. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 773–782.

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 1577–1586.

Joao Silva, Luísa Coheur, Ana C. Mendes, and Andreas Wichert. 2011. From symbolic to sub-symbolic information in question classification. *Artificial Intelligence Review*, 35(2):137–154.

Richard Socher, Eric H. Huang, Jeffrey Pennin, Christopher D. Manning, and Andrew Y. Ng. 2011a. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Advances in Neural Information Processing Systems*, pages 801–809.

Richard Socher, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher D. Manning. 2011b. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 151–161.

Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1201–1211.

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

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

Fangzhong Su and Katja Markert. 2008. From words to senses: a case study of subjectivity recognition. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, pages 825–832. Association for Computational Linguistics.

Kaisheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 1556–1566.

Han Zhao, Zhengdong Lu, and Pascal Poupart. 2015. Self-adaptive hierarchical sentence model. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence*, pages 4069–4076.

Xiaodan Zhu, Parinaz Sobhani, and Hongyu Guo. 2015. Long short-term memory over tree structures. In *Proceedings of The 32nd International Conference on Machine Learning*, pages 1604–1612.

Multi-Timescale Long Short-Term Memory Neural Network for Modelling Sentences and Documents

Pengfei Liu, Xipeng Qiu*, Xinchi Chen, Shiyu Wu, Xuanjing Huang
Shanghai Key Laboratory of Intelligent Information Processing, Fudan University
School of Computer Science, Fudan University
825 Zhangheng Road, Shanghai, China
{pfliu14,xpqiu,xinchichen13,syu13,xjhuang}@fudan.edu.cn

Abstract

Neural network based methods have obtained great progress on a variety of natural language processing tasks. However, it is still a challenge task to model long texts, such as sentences and documents. In this paper, we propose a multi-timescale long short-term memory (MT-LSTM) neural network to model long texts. MT-LSTM partitions the hidden states of the standard LSTM into several groups. Each group is activated at different time periods. Thus, MT-LSTM can model very long documents as well as short sentences. Experiments on four benchmark datasets show that our model outperforms the other neural models in text classification task.

1 Introduction

Distributed representations of words have been widely used in many natural language processing (NLP) tasks (Collobert et al., 2011; Turian et al., 2010; Mikolov et al., 2013b; Bengio et al., 2003). Following this success, it is rising a substantial interest to learn the distributed representations of the continuous words, such as phrases, sentences, paragraphs and documents (Mitchell and Lapata, 2010; Socher et al., 2013; Mikolov et al., 2013b; Le and Mikolov, 2014; Kalchbrenner et al., 2014). The primary role of these models is to represent the variable-length sentence or document as a fixed-length vector. A good representation of the variable-length text should fully capture the semantics of natural language.

Recently, the long short-term memory neural network (LSTM) (Hochreiter and Schmidhuber, 1997) has been applied successfully in many NLP tasks, such as spoken language understanding (Yao et al., 2014), sequence labeling (Chen et al., 2015) and machine translation (Sutskever et al., 2014). LSTM is an extension of the recurrent neural network (RNN) (Elman, 1990), which can capture the long-term and short-term dependencies and is very suitable to model the variable-length texts. Besides, LSTM is also sensitive to word order and does not rely on the external syntactic structure as recursive neural network (Socher et al., 2013). However, when modeling long texts, such as documents, LSTM need to keep the useful features for a quite long period of time. The long-term dependencies need to be transmitted one-by-one along the sequence. Some important features could be lost in transmission process. Besides, the error signal is also back-propagated one-by-one through multiple time steps in the training phase with back-propagation through time (BPTT) (Werbos, 1990) algorithm. The learning efficiency could also be decreased for the long texts. For example, if a valuable feature occurs at the begin of a long document, we need to back-propagate the error through the whole document.

In this paper, we propose a multi-timescale long short-term memory (MT-LSTM) to capture the valuable information with different timescales. Inspired by the works of (El Hihi and Bengio, 1995) and (Koutnik et al., 2014), we partition the hidden states of the standard LSTM into several groups. Each group is activated and updated at different time periods. The fast-speed groups keep the short-term memories, while the slow-speed groups keep the long-term memories. We evaluate our model on four benchmark datasets of text classification. Experimental results show that our model can not only handle short texts, but can model long texts.

Our contributions can be summarized as follows.

- With the multiple different timescale memories, MT-LSTM easily carries the crucial information over a long distance. MT-LSTM

*Corresponding author

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 2326–2335,
Lisbon, Portugal, 17-21 September 2015.

can well model both short and long texts.

- MT-LSTM has faster convergence speed than the standard LSTM since the error signal can be back-propagated through multiple timescales in the training phase.

2 Neural Models for Sentences and Documents

The primary role of the neural models is to represent the variable-length sentence or document as a fixed-length vector. These models generally consist of a projection layer that maps words, subword units or n-grams to vector representations (often trained beforehand with unsupervised methods), and then combine them with the different architectures of neural networks. Most of these models for distributed representations of sentences or documents can be classified into four categories.

Bag-of-words models A simple and intuitive method is the Neural Bag-of-Words (NBOW) model, in which the representation of sentences or documents can be generated by averaging constituent word representations. However, the main drawback of NBOW is that the word order is lost. Although NBOW is effective for general document classification, it is not suitable for short sentences.

Sequence models Sequence models construct the representation of sentences or documents based on the recurrent neural network (RNN) (Mikolov et al., 2010) or the gated versions of RNN (Sutskever et al., 2014; Chung et al., 2014). Sequence models are sensitive to word order, but they have a bias towards the latest input words. This gives the RNN excellent performance at language modelling, but it is suboptimal for modeling the whole sentence, especially for the long texts. Le and Mikolov (2014) proposed a Paragraph Vector (PV) to learn continuous distributed vector representations for pieces of texts, which can be regarded as a long-term memory of sentences as opposed to the short-memory in RNN.

Topological models Topological models compose the sentence representation following a given topological structure over the words (Socher et al., 2011a; Socher et al., 2012; Socher et al., 2013). Recursive neural network (RecNN) adopts a more general structure to encode sentence (Pollack, 1990; Socher et al., 2013). At every node in the tree the contexts at the left and right children of the node are combined by a classical layer. The weights of the layer are shared across all nodes in the tree. The layer computed at the top node gives a representation for the sentence. However, RecNN depends on external constituency parse trees provided by an external topological structure, such as parse tree.

Convolutional models Convolutional neural network (CNN) is also used to model sentences (Collobert et al., 2011; Kalchbrenner et al., 2014; Hu et al., 2014). It takes as input the embeddings of words in the sentence aligned sequentially, and summarizes the meaning of a sentence through layers of convolution and pooling, until reaching a fixed length vectorial representation in the final layer. CNN can maintain the word order information and learn more abstract characteristics.

3 Long Short-Term Memory Networks

A recurrent neural network (RNN) (Elman, 1990) is able to process a sequence of arbitrary length by recursively applying a transition function to its *internal hidden state vector* h_t of the input sequence. The activation of the hidden state h_t at time-step t is computed as a function f of the current input symbol $\mathbf{x}_t$ and the previous hidden state $\mathbf{h}_{t-1}$

$$\mathbf{h}_t = \begin{cases} 0 & t = 0 \\ f(\mathbf{h}_{t-1}, \mathbf{x}_t) & \text{otherwise} \end{cases} \quad (1)$$

It is common to use the state-to-state transition function f as the composition of an element-wise nonlinearity with an affine transformation of both $\mathbf{x}_t$ and $\mathbf{h}_{t-1}$.

Traditionally, a simple strategy for modeling sequence is to map the input sequence to a fixed-sized vector using one RNN, and then to feed the vector to a softmax layer for classification or other tasks (Sutskever et al., 2014; Cho et al., 2014).

Unfortunately, a problem with RNNs with transition functions of this form is that during training, components of the gradient vector can grow or decay exponentially over long sequences (Bengio et al., 1994; Hochreiter et al., 2001; Hochreiter and Schmidhuber, 1997). This problem with *exploding* or *vanishing gradients* makes it difficult for the RNN model to learn long-distance correlations in a sequence.

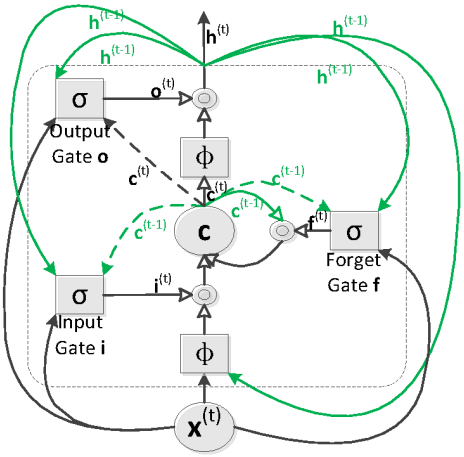


Figure 1: A LSTM unit. The dashed line is the recurrent connection, and the solid link is the connection at the current time.

Long short-term memory network (LSTM) was proposed by (Hochreiter and Schmidhuber, 1997) to specifically address this issue of learning long-term dependencies. The LSTM maintains a separate memory cell inside it that updates and exposes its content only when deemed necessary. A number of minor modifications to the standard LSTM unit have been made. While there are numerous LSTM variants, here we describe the implementation used by Graves (2013).

We define the LSTM *units* at each time step t to be a collection of vectors in $\mathbb{R}^d$: an *input gate* $\mathbf{i}_t$, a *forget gate* $\mathbf{f}_t$, an *output gate* $\mathbf{o}_t$, a *memory cell* $\mathbf{c}_t$ and a hidden state $\mathbf{h}_t$. d is the number of the LSTM units. The entries of the gating vectors $\mathbf{i}_t$, $\mathbf{f}_t$ and $\mathbf{o}_t$ are in $[0, 1]$. The LSTM transition equations are the following:

$$\mathbf{i}_t = \sigma(\mathbf{W}_i\mathbf{x}_t + \mathbf{U}_i\mathbf{h}_{t-1} + \mathbf{V}_i\mathbf{c}_{t-1}) \quad (2)$$

$$\mathbf{f}_t = \sigma(\mathbf{W}_f\mathbf{x}_t + \mathbf{U}_f\mathbf{h}_{t-1} + \mathbf{V}_f\mathbf{c}_{t-1}), \quad (3)$$

$$\mathbf{o}_t = \sigma(\mathbf{W}_o\mathbf{x}_t + \mathbf{U}_o\mathbf{h}_{t-1} + \mathbf{V}_o\mathbf{c}_t), \quad (4)$$

$$\tilde{\mathbf{c}}_t = \tanh(\mathbf{W}_c\mathbf{x}_t + \mathbf{U}_c\mathbf{h}_{t-1}), \quad (5)$$

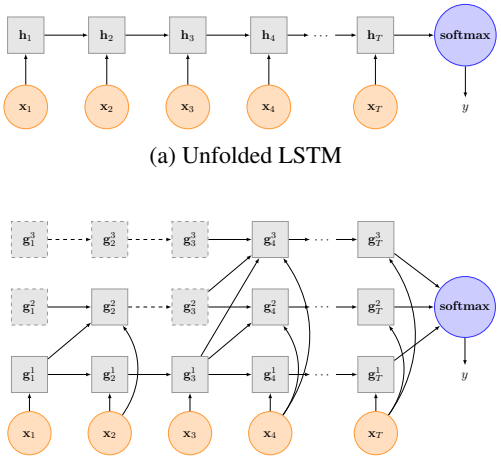
$$\mathbf{c}_t = \mathbf{f}_t^i \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t, \quad (6)$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t), \quad (7)$$

where x_t is the input at the current time step, σ denotes the logistic sigmoid function and $\odot$ denotes elementwise multiplication. Intuitively, the forget gate controls the amount of which each unit of the memory cell is erased, the input gate controls how much each unit is updated, and the output gate controls the exposure of the internal memory state.

Figure 1 shows the structure of a LSTM unit. In particular, these gates and the memory cell allow a LSTM unit to adaptively forget, memorize and expose the memory content. If the detected feature, i.e., the memory content, is deemed important, the forget gate will be closed and carry the memory content across many time-steps, which is equivalent to capturing a long-term dependency. On the other hand, the unit may decide to reset the memory content by opening the forget gate.

4 Multi-Timescale Long Short-Term Memory Neural Network



(a) Unfolded LSTM

(b) Unfolded MT-LSTM with Fast-to-Slow Feedback Strategy

Figure 2: Illustration of the unfolded LSTM and unfolded MT-LSTM. The dotted node indicates the unit which is inactivated at current time, while the solid node indicates the unit which is activated. The dotted lines indicate the units which kept unchanged, while the solid lines indicate the units which will be updated at the next time step.

LSTM can capture the long-term and short-term dependencies in a sequence. But the long-term dependencies need to be transmitted one-by-one along the sequence. Some important information could be lost in transmission process for long texts, such as documents. Besides, the error signal is back-propagated through multiple time steps when we use the back-propagation through time (BPTT) (Werbos, 1990) algorithm. The training efficiency could also be low for the long texts. For example, if a valuable feature occurs at the begin of a long document, we need to back-propagate the error through the whole document.

Inspired by the works of (El Hihi and Bengio, 1995) and (Koutnik et al., 2014), which use de-

layed connections and units operating at different timescales to improve the simple RNN, we separate the LSTM units into several groups. Different groups capture different timescales dependencies.

More formally, the LSTM units are partitioned into g groups $\{G_1, \cdots, G_g\}$. Each group $G_k, (1 \leq k \leq g)$ is activated at different time periods T_k. Accordingly, the gates and weight matrices are also partitioned to maintain the corresponding LSTM groups. The MT-LSTM with just one group is the same to the standard LSTM.

At each time step t, only the groups G_k that satisfy $(t \,\mathrm{MOD}\, T_k) = 0$ are executed. The choice of the set of periods $T_k \in \{T_1, \cdots, T_g\}$ is arbitrary. Here, we use the exponential series of periods: group G_k has the period of $T_k = 2^{k-1}$. The group G_1 is the fastest one and can be executed at every time step, which works like the standard LSTM. The group G_k is the slowest one.

At time step t, the memory cell vector and hidden state vector of group G_k are calculate in two cases:

(1) When group G_k is activated at time step t, the LSMT units of this group are calculated by the following equations:

$$\mathbf{i}_t^k = \sigma(\mathbf{W}_i^k \mathbf{x}_t + \sum_{j=1}^{g} \mathbf{U}_i^{j\to k} \mathbf{h}_{t-1}^j + \sum_{j=1}^{g} \mathbf{V}_i^{j\to k} \mathbf{c}_{t-1}^j), \quad (8)$$

$$\mathbf{f}_t^k = \sigma(\mathbf{W}_f^k \mathbf{x}_t + \sum_{j=1}^{g} \mathbf{U}_f^{j\to k} \mathbf{h}_{t-1}^j + \sum_{j=1}^{g} \mathbf{V}_f^{j\to k} \mathbf{c}_{t-1}^j), \quad (9)$$

$$\mathbf{o}_t^k = \sigma(\mathbf{W}_o^k \mathbf{x}_t + \sum_{j=1}^{g} \mathbf{U}_o^{j\to k} \mathbf{h}_{t-1}^j + \sum_{j=1}^{g} \mathbf{V}_o^{j\to k} \mathbf{c}_{t}^j), \quad (10)$$

$$\tilde{\mathbf{c}}_t^k = \tanh(\mathbf{W}_c^k \mathbf{x}_t + \sum_{j=1}^{g} \mathbf{U}_c^{j\to k} \mathbf{h}_{t-1}^j), \quad (11)$$

$$\mathbf{c}_t^k = \mathbf{f}_t^k \odot \mathbf{c}_{t-1}^k + \mathbf{i}_t^k \odot \tilde{\mathbf{c}}_t^k, \quad (12)$$

$$\mathbf{h}_t^k = \mathbf{o}_t^k \odot \tanh(\mathbf{c}_t^k), \quad (13)$$

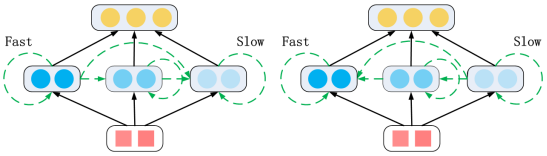
where $\mathbf{i}_t^k$, $\mathbf{f}_t^k$ and $\mathbf{o}_t^k$ are the vectors of input gates, forget gates, and output gates of group G_k at time step t respectively; $\mathbf{c}_t^k$ and $\mathbf{h}_t^k$ are the memory cell vector and hidden state vector of group G_k at time step t respectively.

(2) When group G_k is non-activated at time step t, its LSMT units keep unchanged.

$$\mathbf{c}_t^k = \mathbf{c}_{t-1}^k, \quad (14)$$

$$\mathbf{h}_t^k = \mathbf{h}_{t-1}^k. \quad (15)$$

Figure 3 shows the different between the standard LSTM and MT-LSTM.



(a) Fast-to-Slow Strategy (b) Slow-to-Fast Strategy

Figure 3: Two feedback strategies of our model. The dashed line shows the feedback connection, and the solid link shows the connection at current time.

4.1 Two Feedback Strategies

The feedback mechanism of LSTM is implemented by the recurrent connections from time step $t-1$ to t. Since the MT-LSTM groups are updated with the different frequencies, we can regard the different group as the human memory. The fast-speed groups are short-term memories, while the slow-speed groups are long-term memories. Therefore, an important consideration is what feedback mechanism is between the short-term and long-term memories.

For the proposed MT-LSTM, we consider two feedback strategies to define the connectivity patterns among the different groups.

Fast-to-Slow (F2S) Strategy Intuitively, when we accumulate the short-term memory to a certain degree, we store some valuable information from the short-term memory into the long-term memory. Therefore, we firstly define a fast to slow strategy, which updates the slower group using the faster group. The connections from group j to group k exist if and only if $T_j \leq T_k$. The weight matrices $\mathbf{U}_i^{j\to k}$, $\mathbf{U}_f^{j\to k}$, $\mathbf{U}_o^{j\to k}$, $\mathbf{U}_c^{j\to k}$, $\mathbf{V}_i^{j\to k}$, $\mathbf{V}_f^{j\to k}$, $\mathbf{V}_o^{j\to k}$ are set to zero when $T_j > T_k$.

The F2S updating strategy is shown in Figure 3a.

Slow-to-Fast (S2F) Strategy Following the work of (Koutnik et al., 2014), we also investigate another update scheme from slow-speed group to fast-speed group. The motivation is that a long term memory can be "distilled" into a short-term memory. The connections from group j to group i exist only if $T_j \geq T_i$. The weight matrices $\mathbf{U}_i^{j\to k}$, $\mathbf{U}_f^{j\to k}$, $\mathbf{U}_o^{j\to k}$, $\mathbf{U}_c^{j\to k}$, $\mathbf{V}_i^{j\to k}$, $\mathbf{V}_f^{j\to k}$, $\mathbf{V}_o^{j\to k}$ are set to zero when $T_j < T_k$.

The S2F update strategy is shown in Figure 3b.

Dataset	Type	Train Size	Dev. Size	Test Size	Class	Averaged Length	Vocabulary Size
SST-1	Sentence	8544	1101	2210	5	19	18K
SST-2	Sentence	6920	872	1821	2	18	15K
QC	Sentence	5452	-	500	6	10	9.4K
IMDB	Document	25,000	-	25,000	2	294	392K

Table 1: Statistics of the four datasets used in this paper.

4.2 Dynamic Selection of the Number of the MT-LSTM Unit Groups

Another consideration is how many groups need to be used. An intuitive way is that we need more groups for long texts than short texts. The number of the group depends the length of the texts.

Here, we use a simple dynamic strategy to choose the maximum number of groups, and then the best g is chosen as a hyperparameter according to different tasks. The upper bound of the number of groups is calculated by

$$g = \log_2 L - 1, \tag{16}$$

where L is the average length of the corpus. Thus, the slowest group is activated at least twice.

5 Training

In each of the experiments, the hidden layer at the last moment has a fully connected layer followed by a softmax non-linear layer that predicts the probability distribution over classes given the input sentence. The network is trained to minimise the cross-entropy of the predicted and true distributions; the objective includes an L_2 regularization term over the parameters. The network is trained with backpropagation and the gradient-based optimization is performed using the Adagrad update rule (Duchi et al., 2011).

The back propagation of the error propagation is similar to LSTM as well. The only difference is that the error propagates only from groups that were executed at time step t. The error of non-activated groups gets copied back in time (similarly to copying the activations of nodes not activated at the time step t during the corresponding forward pass), where it is added to the back-propagated error.

6 Experiments

In this section, we investigate the empirical performances of our proposed MT-LSTM model on four benchmark datasets for sentence and document classification and then compare it to other competitor models.

6.1 Datasets

We evaluate our model on four different datasets. The first three datasets are sentence-level, and the last dataset is document-level. The detailed statistics about the four datasets are listed in Table 1. Each dataset is briefly described as follows.

- **SST-1** The movie reviews with five classes (negative, somewhat negative, neutral, somewhat positive, positive) in the Stanford Sentiment Treebank[1] (Socher et al., 2013).
- **SST-2** The movie reviews with binary classes. It is also from the Stanford Sentiment Treebank.
- **QC** The TREC questions dataset[2] involves six different question types, e.g. whether the question is about a location, about a person or about some numeric information (Li and Roth, 2002).
- **IMDB** The IMDB dataset[3] consists of 100,000 movie reviews with binary classes (Maas et al., 2011). One key aspect of this dataset is that each movie review has several sentences.

6.2 Competitor Models

We compare our model with the following models:

- **NB-SVM** and **MNB**. Naive Bayes SVM and Multinomial Naive Bayes with uni and bigram features (Wang and Manning, 2012).
- **NBOW** The NBOW sums the word vectors and applies a non-linearity followed by a softmax classification layer.
- **RAE** Recursive Autoencoders with pre-trained word vectors from Wikipedia (Socher et al., 2011b).
- **MV-RNN** Matrix-Vector Recursive Neural Network with parse trees (Socher et al., 2012).

[1] http://nlp.stanford.edu/sentiment.

[2] http://cogcomp.cs.illinois.edu/Data/QA/QC/.

[3] http://ai.stanford.edu/~amaas/data/sentiment/

	SST-1	SST-2	QC	IMDB
Embedding size	100	100	100	100
hidden layer size	60	60	55	100
Initial learning rate	0.1	0.1	0.1	0.1
Regularization	10^{-5}	10^{-5}	10^{-5}	10^{-5}
Number of Groups	3	3	3	5

Table 2: Hyper-parameter settings for the LSTM and MT-LSTM.

- **RNTN** Recursive Neural Tensor Network with tensor-based feature function and parse trees (Socher et al., 2013).
- **AdaSent** Self-adaptive hierarchical sentence model with gated mechanism (Zhao et al., 2015).
- **DCNN** Dynamic Convolutional Neural Network with dynamic k-max pooling (Kalchbrenner et al., 2014).
- **CNN-non-static** and **CNN-multichannel** Convolutional Neural Network (Kim, 2014).
- **PV** Logistic regression on top of paragraph vectors (Le and Mikolov, 2014). Here, we use the popular open source implementation of PV in Gensim[4].
- **LSTM** The standard LSTM for text classification. We use the implementation of Graves (2013). The unfolded illustration is shown in Figure 2a.

6.3 Hyperparameters and Training

In all of our experiments, the word embeddings are trained using word2vec (Mikolov et al., 2013a) on the Wikipedia corpus (1B words). The vocabulary size is about 500,000. The the word embeddings are fine-tuned during training to improve the performance (Collobert et al., 2011). The other parameters are initialized by randomly sampling from uniform distribution in [-0.1, 0.1]. The hyperparameters which achieve the best performance on the development set will be chosen for the final evaluation. For datasets without development set, we use 10-fold cross-validation (CV) instead. The final hyper-parameters for the LSTM and MT-LSTM are set as Figure 2.

6.4 Results

Table 3 shows the classification accuracies of the standard LSTM, MT-LSTM compared with the competitor models.

Firstly, we compare two feedback strategies of MT-LSTM. The fast-to-slow feedback strategy (MT-LSTM (F2S)) is better than the slow-to-fast strategy (MT-LSTM (S2F)), which indicates that MT-LSTM benefits from periodically storing some valuable information "purified" from the short-term memory into the long-term memory. In the following discussion, we use fast-to-slow feedback strategy as the default setting of MT-LSTM.

[4] https://github.com/piskvorky/gensim/

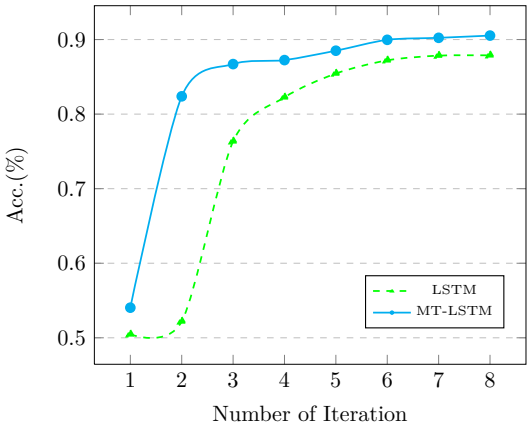


Figure 4: Convergence Speeds on IMDB dataset.

Compared with the standard LSTM, MT-LSTM results in significantly improvements with the same size of hidden layers.

MT-LSTM outperforms the competitor models on the SST-1, QC and IMDB datasets, and is close to the two best CNN based models on the SST-2 dataset. But MT-LSTM uses much fewer parameters than the CNN based models. The number of parameters of LSTM range from 10K to 40K while the number of parameters is about 400K in CNN.

Moreover, MT-LSTM can not only handle short texts, but can model long texts in classification task.

Documents Modeling Most of the competitor models cannot deal with the texts of with several sentences (paragraphs, documents). For instance, MV-RNN and RNTN (Socher et al., 2013) are based on the parsing over each sentence and it is unclear how to combine the representations over many sentences. The convolutional models, such as CNN (Kim, 2014) and AdaSent (Zhao et al., 2015), need more hidden layers or nodes for long texts and result in a very complicated model. These models therefore are restricted to working on sentences instead of paragraphs or documents. Denil et al. (2014) used two-level version of DCNN (Kalchbrenner et al., 2014) to model documents. The first level uses a DCNN to trans-

Model	SST-1	SST-2	QC	IMDB
NBOW (Kalchbrenner et al., 2014)	42.4	80.5	88.2	-
RAE (Socher et al., 2011b)	43.2	82.4	-	-
MV-RNN (Socher et al., 2012)	44.4	82.9	-	-
RNTN (Socher et al., 2013)	45.7	85.4	-	-
DCNN (Kalchbrenner et al., 2014)	48.5	86.8	93.0	-
CNN-non-static (Kim, 2014)	48.0	87.2	93.6	-
CNN-multichannel (Kim, 2014)	47.4	**88.1**	92.2	-
AdaSent (Zhao et al., 2015)	-	-	92.4	-
NBSVM (Wang and Manning, 2012)	-	-	-	91.2
MNB (Wang and Manning, 2012)	-	-	-	86.6
Two-level DCNN (Denil et al., 2014)	-	-	-	89.4
PV (Le and Mikolov, 2014)	44.6*	82.7*	91.8*	91.7*
LSTM	47.9	85.8	91.3	88.5
MT-LSTM (S2F)	48.9	86.7	93.3	90.2
MT-LSTM (F2S)	**49.1**	87.2	**94.4**	**92.1**

Table 3: Results of our MT-LSTM model against state-of-the-art neural models. All the results without marks are reported in the corresponding paper.

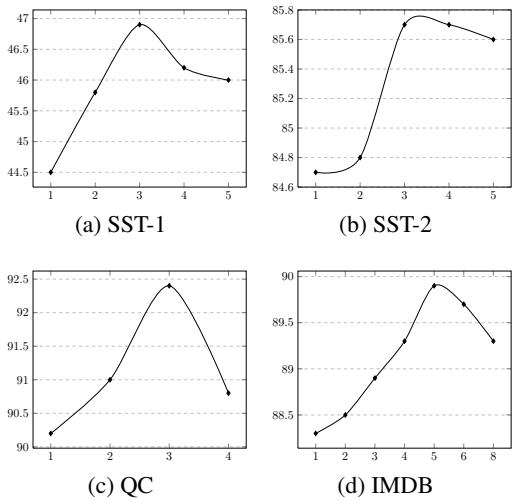


Figure 5: Performances of our model with the different numbers of memory groups g on four development datasets: SST-1,SST-2, QC, and IMDB. Y-axis represents the accuracy(%), and X-axis represents different numbers of memory groups. All memory groups share a fixed-size memory layer h, and here we set h=120.

form embeddings for the words in each sentence into an embedding for the entire sentence. The second level uses another DCNN to transform sentence embeddings from the first level into a single embedding vector that represents the entire document. However, their result is unsatisfactory and they reported that the IMDB dataset is too small to train a CNN model.

The standard LSTM has an advantage to model documents due to its simplification. However, it is also difficult to train LSTM since the error signals need to be back-propagated over a long distance with the BPTT algorithm.

Our MT-LSTM can alleviate this problem with multiple timescale memories. The experiment on IMDB dataset demonstrates this advantage. MT-LSTM achieves the accuracy of 92.1% , which are better than the other models.

Moreover, MT-LSTM converges at a faster rate than the standard LSTM. Figure 4 plots the convergence on the IMDB dataset. In practice, MT-LSTM is approximately three times faster than the standard LSTM since the hidden states of low-speed group often keep unchanged and need not to be re-calculated at each time step.

Impact of the Different Number of Memory Groups In our model, the number of memory groups is a hyperparameter. Here we plotted the accuracy curves of our model with the different numbers of memory groups in Figure 5 to show its impacts on the four datasets.

When the length of text (SST-1, SST-2 and QC) is small, not all memory groups can be activated if we set too many groups, which may harm the performance. When dealing with the long texts (IMBD), more groups lead to a better performance. The performance can be improved with the increase of the number of memory groups.

According to our dynamic strategy, the maximum numbers of groups is 3, 3, 2, 7 for the four datasets. The best numbers of groups from experiments are 3, 3, 3, 5 respectively. Therefor, our dynamic strategy is reasonable. All the datasets except QC, the best number of groups is equal to or smaller than our calculated upper bound. MT-LSMT suffers underfitting when the number of groups is larger than the upper bound.

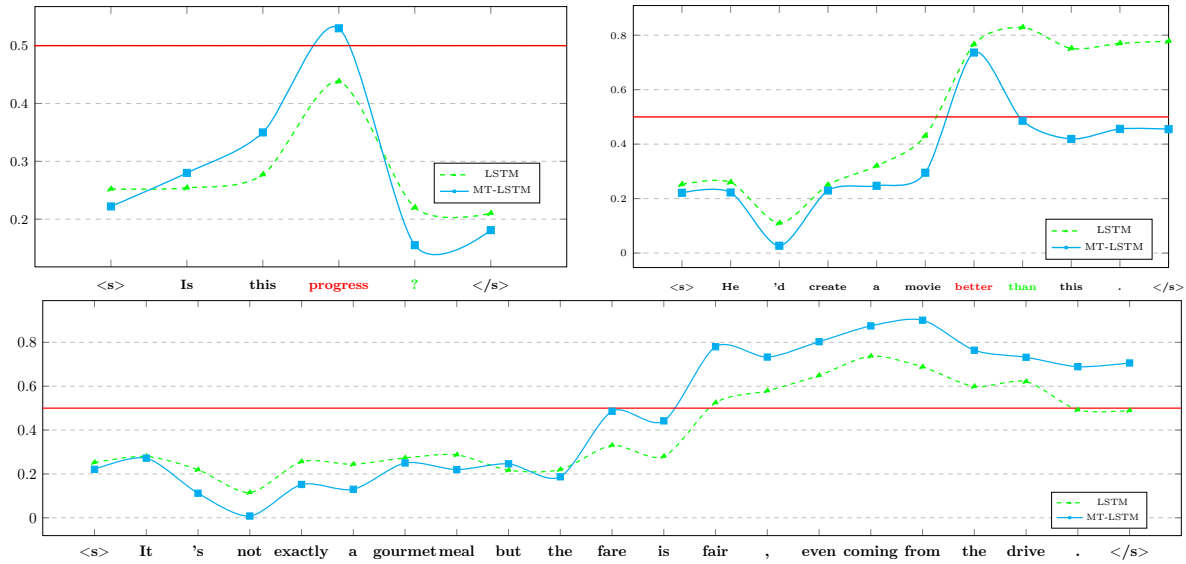


Figure 6: The dynamical changes of the predicted sentiment score over time. Y-axis represents the sentiment score, while X-axis represents the input words in chronological order. The red horizontal line gives a border between the positive and negative sentiments.

6.5 Case Study

To get an intuitive understanding of what is happening when we use LSTM or MT-LSTM to predict the class of text, we design an experiment to analyze the output of LSTM and MT-LSTM at each time step.

We sample three sentences from the SST-2 test dataset, and the dynamical changes of the predicted sentiment score over time are shown in Figure 6. It is intriguing to notice that our model can handle the rhetorical question well.

The first sentence "`Is this progress ?`" has a negative sentiment. Although the word "`progress`" is positive, our model can adjust the sentiment correctly after seeing the question mark "`?`", and finally gets a correct prediction.

The second sentence "`He 'd create a movie better than this .`" also has a negative sentiment. The word "`better`" is positive. Our model finally gets a correct negative prediction after seeing "than this", while LSTM gets a wrong prediction.

The third sentence "`It 's not exactly a gourmet meal but fare is fair , even coming from the drive .`" is positive and has more complicated semantic composition. Our model can still capture the useful long-term features and gets the correct prediction, while LSTM does not work well.

7 Related Work

There are many previous works to model the variable-length text as a fixed-length vector. Specific to text classification task, most of the models cannot deal with the texts of several sentences (paragraphs, documents), such as MV-RNN (Socher et al., 2012), RNTN (Socher et al., 2013), CNN (Kim, 2014), AdaSent (Zhao et al., 2015), and so on. The simple neural bag-of-words model can deal with long texts, but it loses the word order information. PV (Le and Mikolov, 2014) works in unsupervised way, and the learned vector cannot be fine-tuned on the specific task.

Our proposed MT-LSTM can handle short texts as well as long texts in classification task.

8 Conclusion

In this paper, we introduce the MT-LSTM, a generalization of LSTMs to capture the information with different timescales. MT-LSTM can well model both short and long texts. With the multiple different timescale memories. Intuitively, MT-LSTM easily carries the crucial information over a long distance. Another advantage of MT-LSTM is that the training speed is faster than the standard LSTM (approximately three times faster in practice).

In future work, we would like to investigate the other feedback mechanism between the short-term and long-term memories.

Acknowledgments

We would like to thank the anonymous reviewers for their valuable comments. This work was partially funded by the National Natural Science Foundation of China (61472088, 61473092), National High Technology Research and Development Program of China (2015AA015408), Shanghai Science and Technology Development Funds (14ZR1403200).

References

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *Neural Networks, IEEE Transactions on*, 5(2):157–166.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *The Journal of Machine Learning Research*, 3:1137–1155.

Xinchi Chen, Xipeng Qiu, Chenxi Zhu, Pengfei Liu, and Xuanjing Huang. 2015. Long short-term memory neural networks for chinese word segmentation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Proceedings of EMNLP*.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

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

Misha Denil, Alban Demiraj, Nal Kalchbrenner, Phil Blunsom, and Nando de Freitas. 2014. Modelling, visualising and summarising documents with a single convolutional neural network. *arXiv preprint arXiv:1406.3830*.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159.

Salah El Hihi and Yoshua Bengio. 1995. Hierarchical recurrent neural networks for long-term dependencies. In *NIPS*, pages 493–499. Citeseer.

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

Alex Graves. 2013. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.

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

Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, and Jürgen Schmidhuber. 2001. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies.

Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. In *Advances in Neural Information Processing Systems*.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of ACL*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Jan Koutnik, Klaus Greff, Faustino Gomez, and Juergen Schmidhuber. 2014. A clockwork rnn. In *Proceedings of The 31st International Conference on Machine Learning*, pages 1863–1871.

Quoc V. Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of ICML*.

Xin Li and Dan Roth. 2002. Learning question classifiers. In *Proceedings of the 19th International Conference on Computational Linguistics*, pages 556–562.

Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 142–150. Association for Computational Linguistics.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH*.

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

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

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

Jordan B Pollack. 1990. Recursive distributed representations. *Artificial Intelligence*, 46(1):77–105.

Richard Socher, Cliff C Lin, Chris Manning, and Andrew Y Ng. 2011a. Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 129–136.

Richard Socher, Jeffrey Pennington, Eric H Huang, Andrew Y Ng, and Christopher D Manning. 2011b. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 151–161.

Richard Socher, Brody Huval, Christopher D Manning, and Andrew Y Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1201–1211.

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*.

Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 384–394. Association for Computational Linguistics.

Sida Wang and Christopher D Manning. 2012. Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pages 90–94. Association for Computational Linguistics.

Paul J Werbos. 1990. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560.

Kaisheng Yao, Baolin Peng, Yu Zhang, Dong Yu, Geoffrey Zweig, and Yangyang Shi. 2014. Spoken language understanding using long short-term memory neural networks. *IEEE SLT*.

Han Zhao, Zhengdong Lu, and Pascal Poupart. 2015. Self-adaptive hierarchical sentence model. *arXiv preprint arXiv:1504.05070*.

Verbal and Nonverbal Clues for Real-life Deception Detection

Verónica Pérez-Rosas, Mohamed Abouelenien, Rada Mihalcea,
Yao Xiao, CJ Linton, Mihai Burzo
University of Michigan
(vrncapr,zmohamed,mihalcea,xyaoinum,cjlint,mburzo)@umich.edu

Abstract

Deception detection has been receiving an increasing amount of attention from the computational linguistics, speech, and multimodal processing communities. One of the major challenges encountered in this task is the availability of data, and most of the research work to date has been conducted on acted or artificially collected data. The generated deception models are thus lacking real-world evidence. In this paper, we explore the use of multimodal real-life data for the task of deception detection. We develop a new deception dataset consisting of videos from real-life scenarios, and build deception tools relying on verbal and nonverbal features. We achieve classification accuracies in the range of 77-82% when using a model that extracts and fuses features from the linguistic and visual modalities. We show that these results outperform the human capability of identifying deceit.

1 Introduction

As deceptive behavior occurs on a daily basis in different areas of life (Meyer, 2010; Smith et al., 2014), the need arises for automated methodologies to detect deception in an efficient, yet reliable manner. There are many applications that can benefit from automatic deception identification, such as airport security screening, crime investigation and interrogation, interviews, advertisement, and others. In many of these settings, the polygraph test has been used as the main method to identify deceptive behavior. However, this method requires the use of skin-contact devices and human expertise, making it infeasible for large-scale applications. Moreover, polygraph tests were shown to be misleading in multiple cases (Vrij, 2001; Gannon et al., 2009), as human judgment is often biased.

Given the difficulties associated with the use of polygraph-like methods, learning-based approaches have been proposed to address the deception detection task using a number of modalities, including text (Feng et al., 2012) and speech (Hirschberg et al., 2005; Newman et al., 2003). Unlike the polygraph methods, learning-based methods for deception detection rely mainly on data collected from deceivers and truth-tellers. The data is usually elicited from human contributors, in a lab setting or via crowdsourcing. An important problem identified in this data-driven research is the lack of real data. Because of the artificial setting, the subjects may not be emotionally aroused, as they may not take the experiments seriously given the lack of motivation and/or penalty.

In this paper, we describe what we believe is a first attempt at building a multimodal system that detects deception in real-life settings. We collect a dataset consisting of 118 deceptive and truthful video clips, from real trials and live street interviews aired in television shows. We use the transcription of these videos to extract several linguistic features, and we manually annotate the videos for the presence of several gestures that are used to extract nonverbal features. We then build a system that jointly uses the verbal and nonverbal modalities to automatically detect the presence of deception. Our experiments show that the multimodal system can identify deception with an accuracy in the range of 77-82%, significantly improving over the baseline. In addition, we present a study on the human ability to detect deception in single or multimodal data streams, and show that our system outperforms humans on this task.

2 Dataset

Our goal is to build a multimodal collection of occurrences of real deception, which will allow us to analyze both verbal and nonverbal behaviors in relation to deception.

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 2336–2346,
Lisbon, Portugal, 17-21 September 2015.

Figure 1: Sample screenshots showing facial displays and hand gestures from real-life deception and truthful clips. Starting at the top left-hand corner: deceptive interview with up gaze (*Up*), deceptive interview with side gaze (*Side*), deceptive trial with both hands (*Both-H*), truthful trial with forward head (*Forward*), truthful interview with side turn (*Side-Turn*), and truthful interview with single hand (*Single-H*).

Truthful	Deceptive
I was sentenced to forty to sixty years in prison for this crime that I didn't commit. At the trial the judge had exceeded the sentence guidelines because he said I failed to show remorse. And I told him, you know, I felt terrible for what happen to this woman, shouldn't happen to anyone, but I can't show remorse for something I didn't do.	We had some drinks at the bar, maybe one ... two. um I got onto the dance floor myself as I explained, um I have been a trained dancer for some time, going to be able to dance freely is like a ... release. I'm very much in my own space when I do that and so I got up, and I was dancing alone on the dance floor.
It's difficult to pick just one but um I think Tender Mercies uh is ... really captured my imagination um when I was in junior high. Had a lot to do with Robert Duval's performance certainly and that got me excited about the possibility of um .... pulling off an acting career for myself.	Yeah, yeah he was convincing as a wolf. Ahhh actually you know ahhh this is like crazy I'm terrified from wolves, it's my worst fear even though they don't exist but thats my worst fear, sharks and stuff like that. Yeah its my worst fear, I am being honest with you.

Table 1: Sample transcripts for deceptive and truthful clips. The first row presents transcripts from the *Trials* domain while the second shows transcripts corresponding to the *Interviews* domain.

2.1 Data Collection

To collect real deception data, we start by identifying online multimedia sources where deceptive behavior can be observed and verified. We specifically target videos of people, on which we enforce some of the constraints imposed by current data processing technologies: the person in the video should be in front of the camera; her face should be clearly visible; visual quality should be clear enough to identify the facial expressions; and finally, audio quality should be clear enough to hear the voices and understand what the person is saying. We collect video clips from public real trials and interviews aired during television shows, where the truth or falsehood of the participant's statements ends up being known. Video clips from trials consist of statements from witnesses and defendants in the same trial. In order to have a clear distinction between deceptive and truthful trial videos portraying defendants, the process of labeling the trial relies on the verdict. Thus, clips with a guilty verdict are considered deceptive whereas clips with a non-guilty verdict or exoneration are labeled as truthful. Clips containing witness testimonies are labeled as truthful if their statements are verified by police investigations. Examples of trials included in our dataset are Jodi Arias, Andrea Sneiderman, and Amanda Hayes. Exoneree's statements were taken from "The Innocence Project" (`http://www.innocenceproject.org`).

Deceptive and truthful responses are also collected from TV shows and interviews. Examples of such shows are "Lie Witness," "Golden Balls," and the "American Film Institute" and "RevYOU" You-Tube channels. Deceptive videos portray scenarios where interviewees' responses were known to be a lie. For example, the interviewer asks a random individual on his opinion on a non-existing film where the interviewee fabricates a story. On the other hand, truthful videos are collected from individuals asked on their opinions on real movies.

Given our goals and constraints, data collection ended up being a lengthy and laborious process consisting of several iterations of Web mining, data processing and analysis, and content validation.

The final dataset includes 118 videos, including 59 that are labeled as deceptive and 59 labeled as truthful. Among them, 62 belong to the TV street interviews and shows category (*Interviews*) with 28 deceptive and 34 truthful video clips, and 56 belong to the trials category (*Trials*) with 31 deceptive and 25 truthful clips. The average length of the videos in the dataset is 27.28 seconds, with an average length of 33.02 seconds for the truthful clips and 21.54 seconds for the deceptive clips. Collected trial samples cover famous murder cases, while street interviews cover several topics such as movies, music, politics, and religion. The dataset contains 23 unique female and 39 unique male speakers, with their ages ranging approximately between 16 and 60 years.

2.2 Transcriptions and Nonverbal Behavior Annotations

Our goal is to analyze both verbal and nonverbal behavior to understand their relation to deception.

First, all the video clips were manually transcribed. The transcription was performed by two transcribers using the Elan software (Wittenburg et al., 2006). We asked transcribers to include word repetitions and fillers such as um, ah, and uh, as well as long pauses that were marked using three consecutive dots. The final set of transcriptions contain 7835 words, with an average of 66 words per transcript. Table 1 shows transcriptions of sample deceptive and truthful statements from both trials and reality shows.

Second, we annotate the gestures[1] observed during the interactions in the video clips. We specifically focus on the annotation of facial displays and hand movements, as they have been previously found to correlate with deceptive behavior (Depaulo et al., 2003). The gesture annotation is performed using the MUMIN coding scheme (Allwood et al., 2007).

Gesture Category	Agreement	Kappa
Facial Expressions	72.88%	0.576
Eyebrows	80.51%	0.656
Eyes	68.64%	0.517
Gaze	61.40%	0.432
Mouth Openness	77.97%	0.361
Mouth Lips	82.20%	0.684
Head Movements	55.08%	0.420
Hand Movements	91.53%	0.858
Hand Trajectory	84.75%	0.753
Average	75.00%	0.584

Table 2: Gesture annotation agreement

In the MUMIN scheme, facial displays consist of several different facial expressions associated with eyebrows, eyes, gaze, and mouth. Smile, laughter, and scowl are also included, as well as general head and hand movements.

The multimodal annotation was performed by two annotators using the Elan software (Wittenburg et al., 2006). We decided to perform the gesture annotations at video level, rather than at utterance level, because the overall judgment of truthfulness and deceitfulness is based on the whole video content. During the annotation process, annotators were allowed to watch each video clip as many times as they needed. They were asked to identify the facial displays and hand gestures that were most frequently observed or dominating during the entire clip duration. For each video clip, the annotators had to choose one label for each of the nine gestures listed in Table 3.

Table 3 shows the frequency counts associated with the nine gestures considered during the annotation. Note that the counts under each gesture add up to 118, reflecting the fact that for every gesture, the annotators had to choose one label for every video clip. When none of the labels applied, the "Other" category was selected. In the case of gestures associated with hand movements, the "Other" label also accounted for those cases where the speaker's hands were not moving or were not visible.

After all the video clips were annotated for gestures, the inter-annotator agreement was mea-

[1]As done in the Human-Computer Interaction community, we use the term "gesture" to broadly refer to body movements, including facial expressions and hand gestures.

Label	Count
Eyebrows	
Frown (Frowning)	17
Raise (Raising)	71
Other	30
Eyes	
X-open (Exaggerated opening)	17
Close-BE (Closing both)	7
Closing-E (Closing one)	1
Close-R (Closing repeated)	20
Other	73
Gaze	
Interlocutor	69
Up	7
Down	14
Side	24
Other	4

Label	Count
General Facial Expressions	
Smile	41
Scowl	13
Laugh (Laughter)	1
Other	63
Mouth Openness	
Close-M (Closed mouth)	26
Open-M (Open mouth)	92
Mouth Lips	
Up-C (Corners up)	61
Down-C (Corners down)	51
Protruded	1
Retracted	5
Hand Movements	
Both hands (Both-H)	31
Single hands (Single-H)	26
Other	61

Label	Count
Hand Trajectory	
Up (Upwards)	13
Down (Downwards)	5
Sideways	5
Complex	33
Other	62
Head Movements	
Down (Single nod)	3
Down-R (Repeated nods)	48
Forward (Move forward)	3
Back (Move backward)	3
Side-tilt (Single tilt)	8
Side-Tilt-R (Repeated tilts)	9
Side-Turn	9
Side-Turn-R (Shake repeated)	26
Waggle	3
Other	6

Table 3: Frequency counts for nine facial displays and hand gestures

sured. Table 2 shows the observed annotation agreement between the two annotators, along with the Kappa statistic. The agreement measure represents the percentage of times the two annotators agreed on the same label for each gesture category. For instance, 72.88% of the time the annotators agreed on the label assigned to the *General Face* category. On average, the observed agreement was measured at 75%, with a Kappa of 0.58 (macro-averaged over the nine categories), which reflects substantial agreement. Observed agreement for Head Movements and Gaze is noticeably lower than other categories, which can be attributed to a higher number of available gesture choices, as seen in Table 3.

3 Features of Verbal and Nonverbal Behaviors

Given the multimodal nature of our dataset, we decided to focus on the linguistic and gesture components. In this section, we describe the sets of features extracted for each modality, which will then be used to build classifiers of deception.

3.1 Verbal Features

We implement three types of features, consisting of unigrams, psycholinguistic features, and syntactic complexity features.

Unigrams. We extract unigrams derived from the bag-of-words representation of the video transcripts. The unigram features are encoded as word frequencies and include all the words present in the transcripts.

Psycholinguistic Features. The Linguistic Word Count (LIWC) is a psycholinguistics lexicon that has been frequently used to incorporate semantic and psychological information into linguistic analysis (Pennebaker and Francis, 1999). It has been successfully used in previous work on deception detection (Newman et al., 2003; Mihalcea and Strapparava, 2009; Ott et al., 2011). We obtain features for each of the 80 psycholinguistic classes present in the lexicon by calculating the percentage of words in the transcription belonging to each class.

Syntactic Complexity. We also extract features to measure the syntactic complexity of the speech produced by the speakers in truthful and deceptive clips. This set of features is motivated by previous research that has suggested that deceivers' speech has lower complexity (Depaulo et al., 2003). We use the tool described in (Lu, 2010), which generates indexes of syntactic complexity, including general complexity metrics, length of production, and amount of coordination. The set of features consists of fourteen indexes including statistics related to T-units, which are linguistic units that include a main clause in addition to attached subordinate clauses. T-unit analysis is extensively used to analyze syntactic complexity in speech and written content. The set of features includes the mean length of sentence, mean length of T-

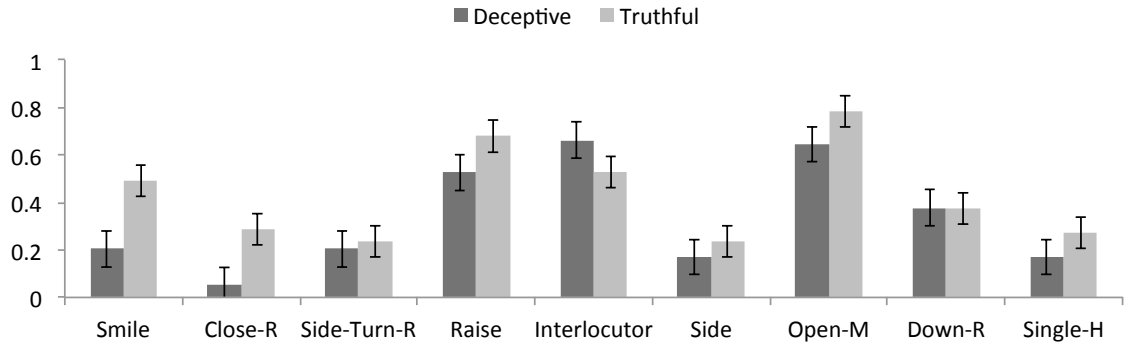


Figure 2: Distribution of nonverbal features for deceptive and truthful groups

unit, mean length of clause, clauses per sentence, verb phrases per T-unit, clauses per T-unit, dependent clauses per clause, dependent clauses per T-unit, T-units per sentence, complex T-unit ratio, coordinate phrases per T-unit, coordinate phrases per clause, complex nominals per T-unit, and complex nominals per clause.

3.2 Nonverbal Features

The nonverbal features are derived from the annotations performed using the MUMIN coding scheme as described in section 2.2. We create a binary feature for each of the 40 available gesture labels. Each feature indicates the presence of a gesture only if it is observed during the majority of the interaction duration. The generated features represent nine different gesture categories covering facial displays and hand movements.

Facial Displays. These are facial expressions or head movements displayed by the speaker during the deceptive or truthful interaction. They include all the behaviors listed in Table 3 under the General Facial Expressions, Eyebrows, Eyes, Mouth Openness, Mouth Lips, and Head Movements.

Hand Gestures. The second broad category covers gestures made with the hands, and it includes the Hand Movements and Hand Trajectories listed in Table 3.

4 Experiments

We start our experiments with an analysis of the nonverbal behaviors occurring in deceptive and truthful videos. We compare the percentage of each behavior as observed in each class. For instance, there is a total of 41 videos in the dataset that include the Smile feature (as shown in Table 3), out of which 12 are part of the deceptive set of 59 videos, and 29 are part of the truthful set (again, of 59 videos). Hence, the percentages for this feature are 20.33% in the deceptive class, and 49.13% in the truthful class. Figure 2 shows the percentages of all the nonverbal features for which we observe noticeable differences for the deceptive and truthful groups. As the figure suggests, facial displays seem to help differentiate between the deceptive and truthful conditions. For instance, we can observe that truth-tellers smile (Smile) and blink more (Close-R). Interestingly deceivers seem to make more eye contact (Interlocutor gaze) and nod (Side-Turn-R) more frequently than truth-tellers. This agrees with the findings in (Depaulo et al., 2003) that liars who are more motivated to get away with their lies (i.e., trials) are likely to increase their eye-contact behavior.

Feature Set	SVM	DT	RF
Unigrams	69.49%	76.27%	67.79%
Psycholinguistic	53.38%	50.00%	66.10%
Syntactic Complexity	52.54%	62.71%	53.38%
Facial Displays	78.81%	74.57%	67.79%
Hand Gestures	59.32%	57.62%	57.62%
Unigr.+Facial Disp.	71.18%	70.33%	68.64%
All Verbal	65.25%	63.55 %	57.62 %
All Nonverbal	75.42%	68.64%	72.03%
All Features	77.11%	69.49%	73.72%

Table 4: Deception classifiers using individual and combined sets of verbal and nonverbal features.

Motivated by these results, we proceed to conduct further experiments to evaluate the performance of the extracted features using a machine learning approach.

Feature Set	SVM
All	**77.11%**
– Hand gestures	74.57%
– Facial displays	64.40%
– Syntactic	76.27%
– Semantic	72.03%
– Unigrams	73.72%

Table 5: Feature ablation study.

We run our learning experiments on the real-deception dataset introduced earlier. Given the even distribution between deceptive and truthful clips, the baseline on this dataset is 50%. For each video clip, we create feature vectors formed by combinations of the verbal and nonverbal features described in the previous section. We build deception classifiers using three classification algorithms: Support Vector Machines (SVM), Decision Trees (DT), and Random Forest (RF).[2] We run several comparative experiments using leave-one-out cross-validation. Table 4 shows the accuracy figures obtained by the three classifiers on the major feature groups described in Section 3. As shown in this table, the facial displays classifier achieves the highest accuracy among the individual classifiers, followed by the unigrams classifier.

We also evaluate classifiers that rely on combined sets of features. The nonverbal features clearly outperform the verbal features, and the classifier that includes all the features improves over the classifiers that rely on all the verbal features or all the nonverbal features. Importantly, several of the classifiers improve significantly over the baseline.

4.1 Analysis of Feature Contribution

To better understand the contribution of the different feature sets to the overall classifier performance, we conduct an ablation study where we remove one group of features at a time. Given that SVM had the best performance in our initial set of experiments, we run all our analysis experiments only using this classifier. Table 5 shows the accuracies obtained when one feature group is removed and the deception classifier is built using the remaining features. From this table, we can again observe that Facial Displays contribute the most to the classifier performance, while Syntactic Features show the lowest contribution.

[2]We use the implementation available in the Weka toolkit with the default parameters.

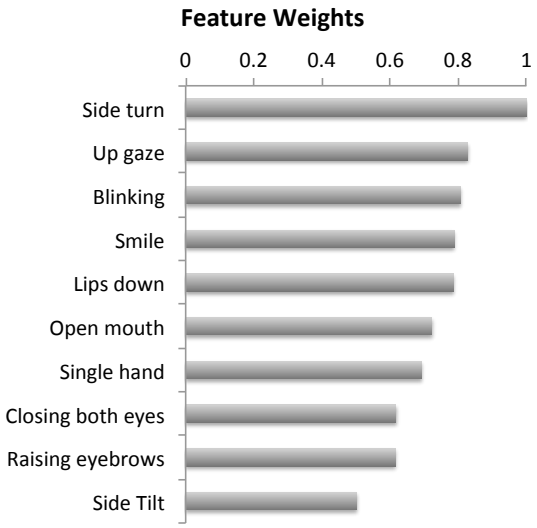


Figure 3: Weights of top nonverbal features

For a closer look at the contribution of individual features included in the group of *Facial Displays*, we analyzed the absolute values of the weights assigned by the learning algorithm to the features in this group. Figure 3 shows the features normalized with respect to the largest feature weight. The five most predictive features are the presence of side turns, up gazes, blinking, and smiling, which we previously identified as possible indicators of deception. This further confirms our initial hypothesis that gestures associated with human interaction are an important component of human deception.

We also analyze the contribution of the linguistic features. Using the linguistic ethnography method (Mihalcea and Pulman, 2009), we obtain the most dominant LIWC word classes associated with deceptive and truthful transcripts extracted from trials and interviews clips. Results are shown in Table 6. Interestingly, the most dominant classes in truthful clips, regardless of being from interviews or trials, correspond to words related to Family, Home, and Humans. This suggests that truth-tellers show similar word usage when interviewed on a real scenario. On the other hand, dominant classes associated to deceivers are less consistent as they discuss aspects related to the topic being discussed. For instance, while being interviewed about a non-existing movie, deceivers talk about their Past, Assent, and use Motion words in order to support their lies. In contrast, while being on trial stating their (false) innocence, they use Anxiety, Anger, and negative emo-

Truthful			
Interviews		Trials	
Class	Score	Class	Score
Metaphor	2.98	You	3.99
Money	2.74	Family	3.07
Inhibition	2.74	Home	2.45
Home	2.13	Humans	1.87
Humans	2.02	Posemo	1.81
Family	1.96	Insight	1.64
Deceptive			
Interviews		Trials	
Class	Score	Class	Score
Assent	4.81	Anger	2.61
Past	2.59	Anxiety	2.61
Sexual	2.00	Certain	2.28
Other	1.87	Death	1.96
Motion	1.68	Physical	1.77
Negemo	1.44	Negemo	1.52

Table 6: LIWC word classes most strongly associated with deception and truth.

tion words (class Negemo). In line with earlier observations (Mihalcea and Strapparava, 2009), deceptive texts include more words that reflect certainty (class Certain, with words such as *completely, truly, always*) and more references to others (class Other, with words such as *she, day, him*).

4.2 Domain Experiments

We perform three sets of experiments to determine the role played by the domain. The first set of experiments uses only the *Interviews* video clips (62 in total), and the results are shown in the left column of Table 7. The second set uses only the *Trials* instances (56 in total), with results shown in the right column of Table 7. Finally, we also perform cross-domain experiments, with the training data drawn from one domain and the test data from the other. The results of these experiments are shown in Table 8. Given the uneven distribution of the truthful and deceptive video clips in two domains, the baselines are 54.83% for the *Interviews* domain (34 truthful, 28 deceptive), and 55.35% for the *Trials* domain (25 truthful, 31 deceptive).

What we learn from these experiments is that the domain does matter. Despite the smaller dataset, the experiments run on one domain at a time lead to results that are higher than the ones obtained with more data but with a mix of domains. The cross-domain experiments also support this argument, as the performance drops significantly when there is no overlap in domain between the training and the test instances. Overall, in all our machine learning experiments, the combined classifier that makes use of all the verbal and nonverbal features achieves the best trade-off between performance and robustness, as it always leads to the best or second best performance across all the experiments using individual or combined feature sets. While a classifier based on an individual feature set can sometime lead to a better performance (e.g., the Facial Displays classifier has better performance when all the video clips are used), that same classifier may not perform well in another setting (e.g., the Facial Displays classifier is significantly below the All Features classifier in the domain experiments).

Feature Set	Interviews	Trials
Baseline	54.83%	55.35%
Unigrams	75.80 %	82.14%
Psycholinguistics	59.67%	50.00%
Syntactic Complexity	54.83%	60.71%
Facial Displays	70.96%	80.35%
Hand Gestures	56.45%	48.21%
Unigr.+Facial Disp.	70.96%	76.78%
All Verbal	70.96%	64.28%
All Nonverbal	67.14%	83.92%
All features	79.03%	82.14%

Table 7: Deception classifiers for the *Interviews* and *Trials* domains, using a SVM classifier trained on individual and combined sets of verbal and nonverbal features.

Training	Test	SVM
Trials	Interviews	58.06%
Interviews	Trials	58.92%

Table 8: Cross-domain classification results using a SVM classifier trained on all the features

5 Human Performance

An important remaining question is concerned with the human performance on the task of deception detection. An answer to this question can shed light on the difficulty of the task, and can also place our results in perspective.

We conduct a study where we evaluate the human ability to identify deceit when exposed to four different modalities: *Text*, consisting of the language transcript; *Audio*, consisting of the audio track of the clip; *Silent video*, consisting of only the video with muted audio; and *Full video*, where

Modality	Agreement	Kappa
Text	58.80%	0.047
Audio	66.70%	0.288
Silent video	52.00%	0.065
Full Video	61.60%	0.191

Table 9: Agreement among three human annotators on text, audio, silent video, and full video modalities.

	Text	Audio	Silent video	Full video
A1	54.24%	58.47%	50.85%	63.00%
A2	55.93%	67.80%	45.76%	68.00%
A3	65.25%	70.34%	55.93%	71.00%
Sys.	65.75%	NA	75.42%	77.11%

Table 10: Performance of three annotators and the developed automatic system (Sys) on the real-deception dataset over four modalities.

audio and video are played simultaneously. We create an annotation interface that shows an annotator instances for each modality in random order, and ask him or her to select a label of either "Deception" or "Truth" according to his or her perception of truthfulness or falsehood.

To avoid annotation bias, we show the modalities in the following order: first we show either *Text* or *Silent video*, then we show *Audio*, followed by *Full video*. Note that apart from this constraint which is enforced over the four modalities belonging to each video clip, the order in which instances are presented to an annotator is random. Furthermore, the annotators did not have access to any information that would reveal the true label of an instance. The only exception to this could have been the annotators' previous knowledge of some of the public trials in our dataset. A discussion with the annotators after the annotation took place indicated however that this was not the case.

Three annotators labeled all the 118 video clips in the dataset. Since four modalities were extracted from each video, each annotator annotated a total of 412 instances. Annotators were not offered a monetary reward and we considered their judgments to be honest as they participated voluntarily in this experiment. Table 9 shows the observed agreement and Kappa statistics among the three annotators for each modality.[3] The agreement for most modalities is rather low and the Kappa scores range between slight to fair agreement. As noted before (Ott et al., 2011), this low agreement can be interpreted as an indication that people are poor judges of deception.

We also determine each annotator's performance for each modality. The results, shown in Table 10, additionally support the argument that human judges have difficulty performing the deception detection task. An interesting, yet perhaps unsurprising observation is that the human performance increases with the availability of modalities. The poorest accuracy is obtained in *Silent video*, followed by *Text*, *Audio*, and *Full Video* where the judges have the highest performance.

Overall, our study indicates that detecting deception is indeed a difficult task for humans and further verifies previous findings where human ability to spot liars was found to be slightly better than chance (Aamodt and Custer, 2006). Moreover, the performance of the human annotators appears to be significantly below that of our system.

6 Related Work

Verbal Deception Detection. To date, several research publications on verbal-based deception detection have explored the identification of deceptive content in a variety of domains, including online dating websites (Toma and Hancock, 2010; Guadagno et al., 2012), forums (Warkentin et al., 2010; Joinson and Dietz-Uhler, 2002), social networks (Ho and Hollister, 2013), and consumer report websites (Ott et al., 2011; Li et al., 2014). Research findings have shown the effectiveness of features derived from text analysis, which frequently includes basic linguistic representations such as n-grams and sentence count statistics (Mihalcea and Strapparava, 2009), and also more complex linguistic features derived from syntactic CFG trees and part of speech tags (Feng et al., 2012; Xu and Zhao, 2012). Research work has also relied on the LIWC lexicon to build deception models using machine learning approaches (Mihalcea and Strapparava, 2009; Ángela Almela et al., 2012) and showed that the use of psycholinguistic information is helpful for the automatic identification of deceit. Following the hypothesis that deceivers might create less complex sentences in an effort to conceal the truth and being able to recall their lies more easily, several researchers have also studied the relation between text syntactic complexity and deception (Yancheva and Rudzicz, 2013).

Nonverbal Deception Detection. Earlier approaches to nonverbal deception detection relied

[3] Inter-rater agreement with multiple raters and variables. https://mlnl.net/jg/software/ira/

on polygraph tests to detect deceptive behavior. These tests are mainly based on such physiological features such as heart rate, respiration rate, skin temperature. Several studies (Vrij, 2001; Gannon et al., 2009; Derksen, 2012) indicated that relying solely on physiological measurements can be biased and misleading. Chittaranjan et al. (Chittaranjan and Hung, 2010) created an audio visual recording of the "Are you a Werewolf?" game in order to detect deceptive behaviour using non-verbal audio cues and to predict the subjects' decisions in the game. For hand gestures, blob analysis was used to detect deceit by tracking the hand movements of the subjects (Lu et al., 2005; Tsechpenakis et al., 2005), or using geometric features related to the hand and head motion (Meservy et al., 2005). Caso et al. (Caso et al., 2006) identified particular hand gestures that can be related to the act of deception using data from simulated interviews. Cohen et al. (2010) found that fewer iconic hand gestures were a sign of a deceptive narration, and Hillman et al. (2012) determined that increased speech prompting gestures were associated with deception while increased rhythmic pulsing gestures were associated with truthful behavior. Also related is the taxonomy of hand gestures developed by (Maricchiolo et al.,) for deception and social behavior. Facial expressions also played a critical role in the identification of deception. (Ekman, 2001) defined micro-expressions as relatively short involuntary expressions, which can be indicative of deceptive behavior. Moreover, these expressions were analyzed using smoothness and asymmetry measurements to further relate them to an act of deceit (Ekman, 2003). Tian et al. (Tian et al., 2005) considered features such as face orientation and facial expression intensity. Owayjan et al. (Owayjan et al., 2012) extracted geometric-based features from facial expressions, and Pfister and Pietikainen (Pfister and Pietikäinen, 2012) developed a micro-expression dataset to identify expressions that are clues for deception. Recently, features from different modalities were integrated in order to find a combination of multimodal features with superior performance (Burgoon et al., 2009; Jensen et al., 2010). A multimodal deception dataset consisting of linguistic, thermal, and physiological features was introduced in (Pérez-Rosas et al., 2014), which was then used to develop a multimodal deception detection system (Abouelenien et al., 2014). An extensive review of approaches for evaluating human credibility using physiological, visual, acoustic, and linguistic features is available in (Nunamaker et al., 2012).

7 Conclusions

In this paper we presented a study of multimodal deception detection using real-life occurrences of deceit. We introduced a novel dataset covering recordings from public real trials and street interviews, and used this dataset to perform both qualitative and quantitative experiments. Our analysis of nonverbal behaviors occurring in deceptive and truthful videos brought insight into the gestures that play a role in deception. We also built classifiers relying on individual or combined sets of verbal and nonverbal features, and showed that we can achieve accuracies in the range of 77-82%.

Additional analyses showed the role played by the various feature sets used in the experiments, and the importance of the domain. To place our results in perspective and better understand the difficulty of the task, we performed a study of human ability to detect deception, which revealed high disagreement among the annotators. Our automatic system outperforms the human detection of deceit by 6-15%.

To our knowledge this is the first work to automatically detect instances of deceit using both verbal and nonverbal features extracted from real deception data. In order to develop a fully automated deception deception system, our future work will address the use of automatic gesture and facial expression identification and automated speech transcription. Our goal is to move forward towards a real-time deception detection system.

The dataset introduced in this paper is publicly available from http://lit.eecs.umich.edu.

Acknowledgments

This material is based in part upon work supported by National Science Foundation awards #1344257 and #1355633, by grant #48503 from the John Templeton Foundation, and by DARPA-BAA-12-47 DEFT grant #12475008. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation, the John Templeton Foundation, or the Defense Advanced Research Projects Agency.

References

Michael G. Aamodt and Heather Custer. 2006. Who can best catch a liar? a meta-analysis of individual differences in detecting deception. *Forensic Examiner*, 15(1):6–11.

Mohamed Abouelenien, Veronica Pérez-Rosas, Rada Mihalcea, and Mihai Burzo. 2014. Deception detection using a multimodal approach. In *Proceedings of the 16th International Conference on Multimodal Interaction*, pages 58–65, New York, NY, USA. ACM.

Jens Allwood, Loredana Cerrato, Kristiina Jokinen, Costanza Navarretta, and Patrizia Paggio. 2007. The mumin coding scheme for the annotation of feedback, turn management and sequencing phenomena. *Language Resources and Evaluation*, 41(3-4):273–287.

Ángela Almela, Rafael Valencia-García, and Pascual Cantos. 2012. Seeing through deception: A computational approach to deceit detection in written communication. In *Proceedings of the Workshop on Computational Approaches to Deception Detection*, pages 15–22, Avignon, France, April.

Judee K. Burgoon, Douglas P. Twitchell, Matthew L. Jensen, Thomas O. Meservy, Mark Adkins, John Kruse, Amit V. Deokar, Gabriel Tsechpenakis, Shan Lu, Dimitris N. Metaxas, et al. 2009. Detecting concealment of intent in transportation screening: A proof of concept. *IEEE Transactions on Intelligent Transportation Systems*, 10(1):103–112, March.

Letizia Caso, Fridanna Maricchiolo, Marino Bonaiuto, Aldert Vrij, and Samantha Mann. 2006. The impact of deception and suspicion on different hand movements. *Journal of Nonverbal Behavior*, 30(1):1–19.

Gokul Chittaranjan and Hayley Hung. 2010. Are you awerewolf? detecting deceptive roles and outcomes in a conversational role-playing game. In *2010 IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP)*, pages 5334–5337, March.

Doron Cohen, Geoffrey Beattie, and Heather Shovelton. 2010. Nonverbal indicators of deception: How iconic gestures reveal thoughts that cannot be suppressed. *Semiotica*, 2010(182):133–174.

Bella Depaulo, Brian Malone, James Lindsay, Laura Muhlenbruck, Kelly Charlton, and Harris Cooper. 2003. Cues to deception. *Psychological Bulletin*, pages 74–118.

Maarten Derksen. 2012. Control and resistance in the psychology of lying. *Theory and Psychology*, 22(2):196–212.

Paul Ekman, 2001. *Telling Lies: Clues to Deceit in the Marketplace, Politics and Marriage*. Norton, W.W. and Company.

Paul Ekman. 2003. Darwin, deception, and facial expression. *Annals of the New York Academy of Sciences*, 1000(EMOTIONS INSIDE OUT: 130 Years after Darwin's The Expression of the Emotions in Man and Animals):205–221.

Song Feng, Ritwik Banerjee, and Yejin Choi. 2012. Syntactic stylometry for deception detection. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 171–175, Jeju Island, Korea, July. Association for Computational Linguistics.

Theresa Gannon, Anthony Beech, and Tony Ward, 2009. *Risk Assessment and the Polygraph*, pages 129–154. John Wiley and Sons Ltd.

Rosanna E. Guadagno, Bradley M. Okdie, and Sara A. Kruse. 2012. Dating deception: Gender, online dating, and exaggerated self-presentation. *Comput. Hum. Behav.*, 28(2):642–647, March.

Jackie Hillman, Aldert Vrij, and Samantha Mann. 2012. Um they were wearing : The effect of deception on specific hand gestures. *Legal and Criminological Psychology*, 17(2):336–345.

Julia Hirschberg, Stefan Benus, Jason M. Brenier, Frank Enos, Sarah Friedman, Sarah Gilman, Cynthia Girand, Martin Graciarena, Andreas Kathol, Laura Michaelis, et al. 2005. Distinguishing deceptive from non-deceptive speech. In *In Proceedings of Interspeech 2005 - Eurospeech*, pages 1833–1836.

Shuyuan Mary Ho and Jonathan M Hollister. 2013. Guess who? an empirical study of gender deception and detection in computer-mediated communication. *Proceedings of the American Society for Information Science and Technology*, 50(1):1–4.

Matthew Jensen, Thomas Meservy, Judee Burgoon, and Jay Nunamaker. 2010. Automatic, multimodal evaluation of human interaction. *Group Decision and Negotiation*, 19(4):367–389.

Adam N. Joinson and Beth Dietz-Uhler. 2002. Explanations for the perpetration of and reactions to deception in a virtual community. *Social Science Computer Review*, 20(3):275–289.

Jiwei Li, Myle Ott, Claire Cardie, and Eduard Hovy. 2014. Towards a general rule for identifying deceptive opinion spam. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, Baltimore, Maryland, June.

Shan Lu, Gabriel Tsechpenakis, Dimitris Metaxas, Matthew Jensen, and John Kruse. 2005. Blob analysis of the head and hands: A method for deception detection. In *Proceedings of the 38th Annual Hawaii International Conference on System Sciences (HICSS'05)*, HICSS '05, pages 20–29, Washington, DC, USA.

Xiaofei Lu. 2010. Automatic analysis of syntactic complexity in second language writing. *International Journal of Corpus Linguistics*, 15(4):474–496.

Fridanna Maricchiolo, Augusto Gnisci, and Marino Bonaiuto. In Anna Esposito, AntoniettaM. Esposito, Alessandro Vinciarelli, Rdiger Hoffmann, and VincentC. Mller, editors, *Cognitive Behavioural Systems*, pages 405–416. Springer Berlin Heidelberg.

Thomas Meservy, Matthew Jensen, John Kruse, Douglas Twitchell, Gabriel Tsechpenakis, Judee Burgoon, Dimitris Metaxas, and Jay Nunamaker. 2005. Deception detection through automatic, unobtrusive analysis of nonverbal behavior. *IEEE Intelligent Systems*, 20(5):36–43, September.

Pamela Meyer. 2010. *Liespotting: Proven Techniques to Detect Deception*. New York: St. Martin's.

Rada Mihalcea and Stephen Pulman. 2009. Linguistic ethnography: Identifying dominant word classes in text. In *Computational Linguistics and Intelligent Text Processing*, pages 594–602. Springer.

Rada Mihalcea and Carlo Strapparava. 2009. The lie detector: Explorations in the automatic recognition of deceptive language. In *Proceedings of the Association for Computational Linguistics (ACL 2009)*, Singapore. Association for Computational Linguistics.

Matthew L. Newman, James W. Pennebaker, Diane S. Berry, and Jane M. Richards. 2003. Lying words: Predicting deception from linguistic styles. *Personality and Social Psychology Bulletin*, 29.

Jay F. Nunamaker, Judee K. Burgoon, Nathan W. Twyman, Jeffrey Gainer Proudfoot, Ryan M. Schuetzler, and Justin Scott Giboney. 2012. Establishing a foundation for automated human credibility screening. In *2012 IEEE International Conference on Intelligence and Security Informatics (ISI)*, pages 202–211, June.

Myle Ott, Yejin Choi, Claire Cardie, and Jeffrey Hancock. 2011. Finding deceptive opinion spam by any stretch of the imagination. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 309–319, Stroudsburg, PA, USA. Association for Computational Linguistics.

Michel Owayjan, Ahmad Kashour, Nancy Al Haddad, Maurice Fadel, and Ghinwa Al Souki. 2012. The design and development of a lie detection system using facial micro-expressions. In *2012 2nd International Conference on Advances in Computational Tools for Engineering Applications (ACTEA)*, pages 33–38, Dec.

James W. Pennebaker and Martha E. Francis. 1999. Linguistic inquiry and word count: LIWC. Erlbaum Publishers.

Verónica Pérez-Rosas, Rada Mihalcea, Alexis Narvaez, and Mihai Burzo. 2014. A multimodal dataset for deception detection. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014), Reykjavik, Iceland, May 26-31, 2014.*, pages 3118–3122.

Tomas Pfister and Matti Pietikäinen. 2012. Electronic imaging & signal processing automatic identification of facial clues to lies. *SPIE Newsroom*, January.

Madeline Smith, Jeffrey Hancock, Lindsay Reynolds, and Jeremy Birnholtz. 2014. Everyday deception or a few prolific liars? the prevalence of lies in text messaging. *Computers in Human Behavior*, 41(0):220–227.

Ying-Li Tian, Takeo Kanade, and Jeffrey F. Cohn. 2005. Facial expression analysis. In *Handbook of Face Recognition*, pages 247–275. Springer New York.

Catalina L. Toma and Jeffrey T. Hancock. 2010. Reading between the lines: linguistic cues to deception in online dating profiles. In *Proceedings of the 2010 ACM conference on Computer supported cooperative work*, CSCW '10, pages 5–8.

Gabriel Tsechpenakis, Dimitris Metaxas, Mark Adkins, John Kruse, Judee K. Burgoon, Matthew L. Jensen, Thomas Meservy, Douglas P. Twitchell, Amit Deokar, and Jay F. Nunamaker. 2005. Hmm-based deception recognition from visual cues. In *IEEE International Conference on Multimedia and Expo, 2005. ICME 2005*, pages 824–827, July.

Aldert Vrij. 2001. *Detecting Lies and Deceit: The Psychology of Lying and the Implications for Professional Practice*. Wiley series in the psychology of crime, policing and law. Wiley.

Darcy Warkentin, Michael Woodworth, Jeffrey T Hancock, and Nicole Cormier. 2010. Warrants and deception in computer mediated communication. In *Proceedings of the 2010 ACM conference on Computer supported cooperative work*, pages 9–12. ACM.

Peter Wittenburg, Hennie Brugman, Albert Russel, Alex Klassmann, and Han Sloetjes. 2006. Elan: a professional framework for multimodality research. In *Language Resources and Evaluation*, volume 2006.

Qiongkai Xu and Hai Zhao. 2012. Using deep linguistic features for finding deceptive opinion spam. In *Proceedings of COLING 2012: Posters*, Mumbai, India, December.

Maria Yancheva and Frank Rudzicz. 2013. Automatic detection of deception in child-produced speech using syntactic complexity features. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 944–953, Sofia, Bulgaria, August. Association for Computational Linguistics.

Social Media Text Classification under Negative Covariate Shift

Geli Fei and Bing Liu
University of Illinois at Chicago
Chicago, IL 60607, USA
gfei2@uic.edu, liub@cs.uic.edu

Abstract

In a typical social media content analysis task, the user is interested in analyzing posts of a particular topic. Identifying such posts is often formulated as a classification problem. However, this problem is challenging. One key issue is *covariate shift*. That is, the training data is not fully representative of the test data. We observed that the covariate shift mainly occurs in the negative data because topics discussed in social media are highly diverse and numerous, but the user-labeled negative training data may cover only a small number of topics. This paper proposes a novel technique to solve the problem. The key novelty of the technique is the transformation of document representation from the traditional *n-gram feature* space to a *center-based similarity* (CBS) space. In the CBS space, the covariate shift problem is significantly mitigated, which enables us to build much better classifiers. Experiment results show that the proposed approach markedly improves classification.

1 Introduction

Applications using social media data, such as reviews, discussion posts, and (micro) blogs are becoming increasingly popular. We observed from our collaborations with social science and health science researchers that in a typical application, the researcher first need to obtain a set of posts of a particular topic that he/she wants to study, e.g., a political issue. Keyword search is often used as the first step. However, that is not sufficient due to low precision and low recall. A post containing the keyword "politics" may not be a political post while a post that does not contain the keyword may be a political post. Thus, text classification is needed to make more sophisticated decisions to improve accuracy.

For classification, the user first manually labels a set of relevant posts (positive data) about the political issue and irrelevant posts (negative data) not about the political issue and then builds a classifier by running a learning algorithm, e.g. SVM or naïve Bayes. However, the resulting classifier may not be satisfactory. There may be many reasons. One key reason we observed is that the labeled negative training data is not fully representative of the negative test data.

Let the user-interested topic be P (positive), and the set of all other irrelevant topics discussed in a social media source be $T = \{T_1, T_2, \ldots, T_n\}$, which forms the negative data. n is usually large. However, due to the labor-intensive effort of manual labeling, the user can label only a certain number of training posts. Then the labeled negative training posts may cover only a small number of irrelevant topics S of T ($S \subseteq T$) as negative. Further, due to the highly dynamic nature of social media, it is probably impossible to label all possible negative topics. In testing, when posts of other negative topics in $T–S$ show up, their classification can be unpredictable. For example, in an application, the training data has no negative examples about *sports*. However, in testing, some sports posts show up. These unexpected sports posts may be classified arbitrarily, which results in low classification accuracy. In this paper, we aim to solve this problem.

In machine learning, this problem is called *covariate shift*, a type of *sample selection bias*. In classic machine learning, it is assumed that the training and testing data are drawn from the same distribution. However, this assumption may not hold in practice such as in our case above, i.e., the training and the test distributions are different (Heckman 1979; Shimodaira 2000; Zadrozny 2004; Huang et al. 2007; Sugiyama et al. 2008; Bickel et al. 2009). In general, the sample selection bias problem is not solvable because the two

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 2347–2356,
Lisbon, Portugal, 17-21 September 2015.

distributions can be arbitrarily far apart from each other. Various assumptions were made to solve special cases of the problem. One main assumption was that the conditional distribution of the class given a data instance is the same in the training and test data sets (Shimodaira 2000; Huang et al. 2007; Bickel et al. 2009). This gives the *covariate shift* problem.

In this paper, we focus on a special case of the covariate shift problem. We assume that the covariate shift problem occurs mainly in the negative training and test data, and no or minimum covariate shift exists in the positive training and test data. This assumption is reasonable because the user knows the type of posts/documents that s/he is looking for and can label many of them.

Following the notations in (Bickel et al. 2009), our special case of the covariate shift problem can be stated formally as follows: let the set of training examples be $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_k, y_k)\}$, where $\mathbf{x}_i$ is the data/feature vector and y_i is the class label of $\mathbf{x}_i$. Let the set of test cases be $\{\mathbf{x}_{k+1}, \mathbf{x}_{k+2}, \ldots, \mathbf{x}_n\}$, which have no class labels. Since we are interested in binary classification, y_i is either 1 (positive class) or -1 (negative class). The labeled training data and the unseen test data have the same target conditional distribution $p(y|\mathbf{x})$ and the marginal distributions of the positive data in both the training and testing are also the same. But the marginal distributions of the negative data in the training and testing are different, i.e., $p_L(\mathbf{x}^-) \neq p_T(\mathbf{x}^-)$, where L, T, and $-$ represent the labeled training data, test data, and the negative class respectively.

Existing methods for addressing the covariate shift problem basically work as follows (see the Related Work section). First, they estimate the bias of the training data based on the given test data using some statistical techniques. Then, a classifier is trained on a weighted version of the original training set based on the estimated bias. Requiring the test data to be available in training is, however, a major weakness. In the social media post classification setting, the system needs to constantly classify the incoming data. It is infeasible to perform training constantly.

In this paper, we propose a novel learning technique that does not need the test data to be available during training due to the specific nature of our problem, i.e., the positive training data does not have the covariate shift issue.

One obvious solution to this problem is one-class classification (Schölkopf et al. 1999; Tax and Duin, 1999a), i.e., one-class SVM. We simply discard the negative training posts/documents completely because they have the covariate shift problem. Although this is a valid solution, as we will see in the evaluation section, the models built based on one-class SVM perform poorly. Although it is conceivable to use an unsupervised method such clustering, SVD (Alter et al., 2000) or LDA (Blei et al., 2003), supervised learning usually give much higher accuracy.

In our proposed method, instead of performing supervised learning in the original document space based on n-grams, we perform learning in a similarity space. Thus, the key novelty of the method is the transformation from the original *document space* (DS) to a *center-based similarity space* (CBS). In the new space, the covariate shift problem is significantly mitigated, which enables us to build more accurate classifiers. The reason for this is that in CBS based learning the vectors in the similarity space enable SVM (which is the learning algorithm that we use) to find a good boundary of the positive class data based on similarity and to separate it from all possible negative class data, including those negative data that is not represented in training. We will explain this in greater detail in Section 3.5 after we present the proposed algorithm, which we call CBS-L (for CBS Learning).

This paper makes three contributions: First, it formulates a special case of the covariate shift problem. This case occurs frequently in social media data classification as we discussed above. Second, it proposes a novel CBS space based learning method, CBS-L, which avoids the covariate shift problem to a large extent because it is able to find a good similarity boundary of the positive data. Third, it experimentally demonstrates the effectiveness of the proposed method.

2 Related Work

Traditional supervised learning assumes that the training and test examples are drawn from the same distribution. However, this assumption can be violated in many applications. This is especially the case for social media data because of the high topic diversity and constant changes of topics. This problem is known as *covariate shift*, which is a form of sample selection bias.

Sample selection bias was first introduced in econometrics by Heckman (1979). It came into the field of machine learning through the work of Zadrozny (2004). The main approach in machine learning is to first estimate the distribution bias of the training data based on the test data, and then learn using weighted training examples to

compensate for the bias (Bickel et al. 2009).

For example, Shimodaira (2000) and Sugiyama and Muller (2005) proposed to estimate the training and test data distributions using kernel density estimation. The estimated density ratio is then used to generate weighted training examples. Dudik et al. (2005) and Bickel and Scheffer (2007) used maximum entropy density estimation, while Huang et al. (2007) proposed kernel mean matching. Sugiyama et al. (2008) and Tsuboi et al. (2008) estimated the weights for the training instances by minimizing the Kullback-Leibler divergence between the test and the weighted training distributions. Bickel et al. (2009) proposed an integrated model. As we discussed in the introduction, the need for the test data at the training time is a major weakness for social media data classification. The proposed technique CBS-L doesn't have this restriction.

As mentioned in the introduction, one-class classification is a suitable approach to solve the problem. Tax and Duin (1999a and 1999b) proposed a model for one-class classification called Support Vector Data Description (SVDD) to seek a hyper-sphere around the positive data that encompasses points in the data with the minimum radius. In order to balance between model over-fitting and under-fitting, Tax and Duin (2001) proposed a method that tries to use artificially generated outliers to optimize the model parameters. However, their experiments suggest that the procedure to generate artificial outliers in a hyper-sphere is only feasible for up to 30 dimensions. Also, as pointed out by (Khan and Madden, 2010; 2014), one drawback of their methods is that they often require a large dataset and the methods become very inefficient in high dimensional feature spaces. Since text documents are usually represented in a much higher dimensional space, these methods are less suitable for text applications. Manevitz and Yousef (2001) performed one-class text classification using one-class SVM as proposed by Schölkopf et al. (1999). The method is based on identifying outlier data that are representative of the second class. Instead of assuming the origin is the only member of the outlier class, it assumes those data points with few non-zero entries are also outliers. However, as reported in the paper, their methods produce quite weak results (Schölkopf et al., 1999; 2000). Li et al. (2003) presented an improved version of one-class SVM for detecting anomalies. Their idea is to consider all data points that are close to the origin as outliers. Both (Yang and Madden, 2007) and (Tian and Gu, 2010) tried to refine Schölkopf's models by searching optimal parameters. Luo et al., (2007) proposed a cost-sensitive one-class SVM algorithm for intrusion detection. We will see in the experiment section that one-class classification is far inferior to our proposed CBS-L method.

In this work, we propose to represent documents in the similarity space and thus it is related to works on document representation. Alternative document representations have been proposed in the past and have been shown to perform well in many applications (Radev et al., 2000; He et al., 2004; Lebanon 2006; Ranzato and Szummer, 2008, Wang and Domeniconi, 2008). In (Radev et al., 2000), although the centroid sentence/document vector was computed, it was not transformed to a similarity space vector representation. Wang and Domeniconi (2008) proposed to use external knowledge to build semantic kernels for documents in order to improve text classification. In our problem, the main difficulty is that testing negative documents cannot be well covered in training. It is not clear how the enriched document representations could help solve our problem.

Our work is also related to learning from positive and unlabeled examples, also known as PU learning (Denis, 1998; Yu et al. 2002; Liu et al. 2003; Lee and Liu, 2003; Elkan and Noto, 2008; Li et al. 2010). In this learning model, there is a set of labeled positive training data and a set of unlabeled data, but there is no labeled negative training data. Clearly, their setting is different from ours too. There is also no guarantee that the unlabeled data has the same distribution as the future test data.

Our problem is also very different from domain adaption as we work in the same domain. Due to the use of document similarity, our method has some resemblance to learning to rank (Li, 2011; Liu, 2011). However, CBS-L is very different because we perform supervised classification. Our similarity is also center-based rather than pair-wise document similarity, which is also used in (Qian and Liu 2013) for spam detection.

3 The Proposed CBS Learning

We now formulate the proposed supervised learning in the CBS space, called CSB-L. The key difference between CBS learning and the classic document space (DS) learning is in the document representation, which applies to both training and testing documents or posts. In the next subsection, we first give the intuitive idea

and a simple example. The detailed algorithm follows. In Section 3.5, we explain why CBS-L is better than DS-based learning when unexpected negative data appear in the test set.

3.1 Basic Idea

In the proposed CBS-L formulation, each document d is still represented as a feature vector, but the vector no longer represents the document d itself based on n-grams. Instead, it represents a set of similarity values between document d and the center of the positive documents. Specifically, the learning consists of the following steps:

1. Each document d (in the positive or negative class) is first represented with a set of document representations, i.e., *document space vectors* (*ds-vectors*) based on the document itself as in traditional text classification. Each vector denotes one representation of the document. For example, one representation may be based on only unigrams, and another representation may be based on only bigrams. For simplicity, we use only one representation/vector **x** (e.g., unigrams) here to represent d. Note that we use bold lower case letters to represent vectors. Each feature in a *ds*-vector is called a *ds-feature*.
2. A center vector **c** is then computed for each document representation for the positive class documents using the *ds-vectors* of all positive and negative documents of that representation. **c** is thus also a *ds-vector*.
3. Each document d in the positive and negative class is then transformed to a *center-based similarity space vector* $\mathbf{s}_d$ (called a *cbs-vector*). $\mathbf{s}_d$ consists of a set of similarity values between document d's set of ds-vectors, i.e., {**x**} in our case here (since we use only one representation), and the set of corresponding positive class center vectors, i.e., {**c**} in our case:

   $$\mathbf{s}_d = Sim(\{\mathbf{x}\}, \{\mathbf{c}\}),$$

   where *Sim* is a similarity function consisting of a set of similarity measures. Each feature in $\mathbf{s}_d$ is called an *cbs-feature*. $\mathbf{s}_d$ still has the same original class label as d. Let us see an actual example. We assume that our single center vector for the positive class has been computed (see Section 3.2) based on the unigram representation of documents:

   c: 1:1 2:1 6:2

   where *y*:*z* represents a *ds*-feature *y* (e.g., a word) and its feature value (e.g., term frequency, *tf*). We want to transform the following positive document d_1 and negative document d_2 (*ds-vectors*) to their *cbs*-vectors (the first number is the class):

   d_1: 1 1:2 2:1 3:1 d_2: -1 2:2 3:1 5:2

   If we use *cosine* as the first similarity measure in *Sim*, we can generate a *cbs*-feature 1:0.50 for d_1 (as cosine(c, d_1) = 0.50) and a *cbs*-feature 1:0.27 for d_2 (as cosine(c, d_2) = 0.27). If we have more similarity measures, more *cbs*-features will be produced. The resulting *cbs*-vectors for d_1 and d_2 with their class labels, 1 and -1, are:

   d_1: 1 1:0.50 … d_2: -1 1:0.27 …
4. We now have a binary classification problem in the CBS space. This step simply runs a classification algorithm, e.g., SVM, to build a classifier. We use SVM in our work.

3.2 CBS Based Learning

We are given a binary text classification problem. Let $D = \{(d_1, y_1), (d_2, y_2), \ldots, (d_n, y_n)\}$ be the set of training examples, where d_i is a document and $y_i \in \{1, -1\}$ is its class label. Traditional classification directly uses D to build a binary classifier. However, in the CBS space, we learn a classifier that returns 1 for documents that are "close enough" to the center of the training positive documents and -1 for documents elsewhere.

We now detail the proposed technique. As we mentioned above, instead of using one single *ds*-vector to represent a document $d_i \in D$, we use a set R_d of p *ds*-vectors $R_d = \{\mathbf{x}_1^d, \mathbf{x}_2^d, \ldots, \mathbf{x}_p^d\}$. Each vector $\mathbf{x}_i^d$ denotes one document space representation of the document, e.g., unigram representation. We then compute the center of positive training documents, which is represented as a set of p centroids $C = \{\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_p\}$, each of which corresponds to one document space representation in R_d. The way to compute each center $\mathbf{c}_i$ is similar to that in the Rocchio relevance feedback method in information retrieval (Rocchio, 1971; Manning et al. 2008), which uses the corresponding *ds*-vectors of all training positive and negative documents. The detail will be given below. Based on R_d for document d and the center C, we can transform a document d from its document space representations R_d to one center-based similarity vector **cbs-v** by applying a similarity function *Sim* on each element $\mathbf{x}_i^d$ of R_d and its corresponding center $\mathbf{c}_i$. We now detail document transformation.

Training document transformation: The train-

ing data transformation from *ds*-vectors to *cbs*-vectors performs the following two steps:

Step 1: Compute the set C of centroids for the positive class. Each centroid vector $\mathbf{c}_i \in C$ is for one document representation $\mathbf{x}_i^d$. And it is computed by applying the Rocchio method to the corresponding *ds*-vectors of all documents in both positive and negative training data.

$$\mathbf{c}_i = \frac{\alpha}{|D_+|} \sum_{\mathbf{ds}_i^d \in D_+} \frac{\mathbf{x}_i^d}{\|\mathbf{x}_i^d\|} - \frac{\beta}{|D - D_+|} \sum_{\mathbf{x}_j^d \in D - D_+} \frac{\mathbf{x}_j^d}{\|\mathbf{x}_j^d\|}$$

where D_+ is the set of documents in the positive class and |.| is the size function. α and β are parameters, which are usually set empirically. It is reported that using *tf-idf* representation, $\alpha = 16$ and $\beta = 4$ usually work quite well (Buckley et al. 1994). The subtraction is used to reduce the influence of those terms that are not discriminative (i.e., terms appearing in both positive and negative documents).

Step 2: Compute the similarity vector **cbs-v**$_d$ (center-based similarity space vector) for each document $d \in D$ based on its set of document space vectors R_d and the corresponding centroids C of the positive documents.

$$\textbf{cbs-v}_d = Sim(R_d, C)$$

Sim has a set of similarity measures, and each measure m_j is applied to p document representations $\mathbf{x}_i^d$ in R_d and their corresponding centers $\mathbf{c}_i$ in C to generate p similarity features (*cbs*-features) in **cbs-v**$_d$. We discuss the *ds*-features and similarity measures for computing *cbs*-features in the next two subsections.

Complexity: The data transformation step is clearly linear in the number of examples, i.e., n.

Test document transformation: For each test document d, we can use step 2 above to produce a *cbs*-vector for d.

3.3 DS-Features

In order to compute *cbs*-features (center-based similarity space features) for each document, we need to have the *ds*-features of a document and the center of the positive class. We discuss *ds*-features first, which are extracted from each document itself.

Since our task is document classification, we use the popular unigram, bigram and trigram with *tf-idf* weighting as the *ds*-features for a document. These three types of *ds*-features also give us three different document representations.

3.4 CBS-Features

Ds-vectors are transformed into *cbs*-vectors by applying a set of similarity measures on each document space vector and the corresponding center vector. In this work, we employed five similarity measures from (Cha, 2007) to gauge the similarity of two vectors. Based on these measures, we produce 15 CBS features using the unigram, bigram, and trigrams representations of each document. The similarity measures we used are listed in Table 1, where P and Q are two vectors and d represents the dimension of P and Q.

$s_{Cos} = \frac{\sum_{i=1}^{d} P_i Q_i}{\sqrt{\sum_{i=1}^{d} {P_i}^2} \sqrt{\sum_{i=1}^{d} {Q_i}^2}}$
$s_{gow} = 1 - \frac{1}{d} \sum_{i=1}^{d} \left\| \frac{P_i}{\sqrt{\sum_{i=1}^{d} {P_i}^2}} - \frac{Q_i}{\sqrt{\sum_{i=1}^{d} {Q_i}^2}} \right\|$
$s_{Lor} = 1 - \sum_{i=1}^{d} ln(1 + \|P_i - Q_i\|)$
$s_{Dice} = \frac{2 \sum_{i=1}^{d} P_i Q_i}{\sum_{i=1}^{d} {P_i}^2 + \sum_{i=1}^{d} {Q_i}^2}$
$s_{jac} = \frac{\sum_{i=1}^{d} P_i Q_i}{\sum_{i=1}^{d} {P_i}^2 + \sum_{i=1}^{d} {Q_i}^2 - \sum_{i=1}^{d} P_i Q_i}$

Table 1: similarity measures for CBS-Features

3.5 Why Does CBS Space Learning Work?

We now try to explain why CBS learning (CBS-L) can deal with the covariate shift problem, and thus can perform better than document space learning. The reason is that due to the use of similarity features, CBS-L is essentially trying to generate a boundary for the positive training data because similarity is not directional and thus covers all directions in a spherical shape in the space. In classification, the negative data from anywhere or direction outside the spherical shape can be detected. The covariate shift problem will not affect the classification much. Many types of documents that are not represented in the negative training data will still be detected due to their low similarity. For example, in Figure 1, we want to build a SVM classifier to separate positive data represented as black squares and negative data represented as empty circles. The constructed CBS-L classifier would look like a circle (in dashed line) in the original document space

covering the positive data. The size of this (boundary) circle depends on the separation margin between the two classes. Although data points represented by empty triangles are not represented in the negative training data (which has only empty circles) in building the classifier, our classifier is able to identify them as *not positive* at the test time because they are outside the boundary circle.

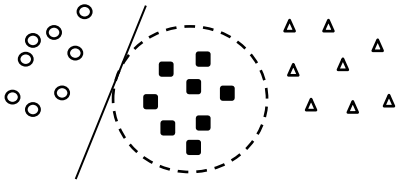


Figure 1: CBS learning vs. DS learning.

If we had used the document space (DS) features to build a SVM classifier, the classifier would be a line (see Figure 1) between the positive data (black squares) and the negative data (empty circles). This line unfortunately will not be able to identify data points represented as empty triangles as not positive because the triangles actually lie on the positive side and would be classified as positive, which is clearly wrong.

4 Experiments

In this section, we evaluate the proposed learning in the center-based similarity space (CBS-L) and compare it with baselines.

4.1 Experimental Dataset

As stated at the beginning of the paper, this work was motivated by the real-life problem of identifying the right social media posts or documents for specific applications. For an effective evaluation, we need a large number of classes in the data to reflect the topic richness and diversity of the social media. The whole data also has to be labeled for evaluation. Using online reviews of a large number of products is a natural choice because there are many types of products and services and there is no need to do manual labeling, which is very labor intensive, time consuming, and error prone. We obtained the Amazon review database from the authors of (Jindal and Liu 2008), and constructed a dataset with reviews of 50 types of products, which we also call 50 topics. Each topic (a type of products) have 1000 reviews. For each topic, we randomly sampled 700 reviews/documents for training and the remaining 300 reviews for testing. Note that although we use this product review collection, we do not perform sentiment classification. Instead, we still perform the traditional topic based classification. That is, given a review, the system decides what type of product the review is about. In our experiments, we use every topic as the positive class. This gives us 50 classification results.

4.2 Baselines

We use three baselines in our evaluation.

Document space one-class SVM (ds-osvm): As we discussed earlier, due to the covariate shift problem in the negative training data, one solution is to drop the negative training data completely to build a one-class classifier. One-class SVM is the state-of-the-art one-class classification algorithm. We apply one-class SVM to the documents in the document space as one of the baselines. One-class SVM was first introduced by Schölkopf et al. (1999; 2000), which is based on the assumption that the origin is the only member of the second class. The data is first mapped into a transformed feature space via a kernel and then standard two-class SVM is employed to construct a hyper-plane that separates the data and the original with maximum margin. As mentioned earlier, there is also the support vector data description (SVDD) formulation for one-class classification proposed by Tax and Duin (1999a; 1999b). SVDD seeks to distinguish the positive class from all other possible data in space. It basically finds a hyper-sphere around the positive class data that contains almost all points in the data set with the minimum radius. It has been shown that the use of Gaussian kernel makes SVDD and One-class SVM equivalent, and the results reported in (Khan and Madden, 2014) demonstrate that SVDD and One-class SVM are comparable when the Gaussian kernel is applied. Thus in this paper, we just use one-class SVM, which is one of the SVM-based classification tools in the LIBSVM[1] library (version 3.20) (Chang and Lin, 2011).

Center-based similarity space one-class SVM (cbs-osvm): Instead of applying one-class SVM to documents in the original document space, this baseline applies it to the CBS space after the documents are transformed to CBS vectors.

SVM: This baseline is the SVM applied in the original document space. Although in this case, there is covariate shift problem, we want to see how serious the problem might be, and how the proposed CBS-L technique can deal with the

[1] http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/

	In-training			Out-of-training			Combined		
	precision	recall	F1-score	precision	recall	F1-score	precision	recall	F1-score
	10 topics are used in the training negative class								
ds-osvm							0.154	0.498	0.205
cbs-osvm	0.664	0.453	0.514	0.357	0.442	0.339	0.343	0.452	0.330
SVM	0.678	0.811	0.736	0.176	0.803	0.282	0.160	0.819	0.262
CBS-L	0.796	0.766	0.776	0.384	0.768	0.491	0.368	0.754	0.481
	20 topics are used in the training negative class								
ds-osvm							0.154	0.498	0.205
cbs-osvm	0.561	0.477	0.466	0.430	0.445	0.390	0.364	0.457	0.344
SVM	0.566	0.753	0.643	0.304	0.753	0.422	0.254	0.758	0.371
CBS-L	0.761	0.700	0.723	0.557	0.702	0.608	0.485	0.693	0.558
	30 topics are used in the training negative class								
ds-osvm							0.154	0.498	0.205
cbs-osvm	0.451	0.491	0.393	0.488	0.524	0.407	0.378	0.487	0.355
SVM	0.508	0.721	0.591	0.450	0.722	0.547	0.323	0.726	0.439
CBS-L	0.723	0.650	0.678	0.721	0.644	0.667	0.569	0.649	0.598
	40 topics are used in the training negative class								
ds-osvm							0.154	0.498	0.205
cbs-osvm	0.423	0.482	0.379	0.590	0.511	0.444	0.372	0.486	0.347
SVM	0.456	0.689	0.544	0.641	0.685	0.658	0.374	0.695	0.481
CBS-L	0.697	0.613	0.644	0.848	0.616	0.699	0.639	0.613	0.619

Table 2: Summary results of the 50 topics

problem. We use the SVM tool in LIBSVM.

4.3 Kernels and Parameters

As Khan and Madden (2014) pointed out that one-class SVM performs the best when Gaussian kernel is used, we use Gaussian kernel as well. Manevitz and Yousef (2001) applied one-class SVM to text classification, and the authors reported that one-class SVM works the best with binary feature weighting scheme compared to *tf* or *tf-idf* weighting schemes. Also, they reported that a small number of features (10) with highest document frequency performed the best with Gaussian kernel. We also use binary representation, but found that 10 features are already too many in our case. In fact, 5 features give the best results. Using a small number of features is intuitive because to find the boundary of a very high dimensional space is very difficult. We also tried more features but they were poorer.

For SVM classification in the document space, we use the linear kernel as it has been shown by many researchers that the linear kernel performs the best (e.g., Joachims, 1998; Colas and Brazdil, 2006). We experimented with RBF kernels extensively, but they did not perform well with the traditional document representation. The term weighting scheme is *tf-idf* (Colas and Brazdil, 2006) with no feature selection.

For our proposed method CBS-L, we use *tf-idf* values of unigram, bigram and trigram to represent a document in three ways in the document space. As mentioned earlier, five document similarity functions are used to transform document space vectors to CBS space vectors. And in order to filter out less useful features for the center vector of the positive class, we performed feature selection in the document space using the classic information gain method (Yang and Pedersen, 1997) to empirically choose the most effective 100 features for the positive class.

For all the kernels, we use the default parameter settings in the LIBSVM systems. We tried to tune the parameters, but did not get better results.

4.4 Results

We now present the experiment results. As mentioned above, we treat each topic as the positive class. This gives 50 tests. To test the effect of covariate shift, we also vary the number of topics in the negative class. We used 10, 20, 30, and 40 topics in the training negative class. The test set always has 49 topics of negative data.

For each setting, we give three sets of results for the positive class, which is the target topic data that we are interested in obtaining through classification. Each set of results includes the standard measures of precision, recall, and F1-score for the positive class. The three sets are:

1. *In-training*: In this case, the test negative data

contains only data from those topics used in training. This is the classical supervised learning setting where the training and test data are randomly drawn from the same distribution.

2. *Not-in-training*: In this case, the test negative set contains only data from the other topics not used in training. The classical setting of supervised learning does not deal with this problem. This represents covariate shift.
3. Combined: In this case, the test data contains both in-training and not-in-training negative topics. Due to the use of not-in-training test data, this is also not the classical setting.

Due to a large number of experiment results, we cannot report all the details. Table 2 summarizes the results. Notice that for ds-osvm, it does not make sense to have in-training and not-in-training results because it does not use any training negative data. Thus, there is only one set of results for "Combined," which is duplicated in the table for easy comparison. However, note that cbs-osvm uses negative data for training in order to compute the center for the positive class.

From the table, we can make the following observations (since there are many numbers, we only focus on F1-scores).

1. The proposed CBS-L method performs markedly better than all baselines. For the results of in-training, not-in-training, and combined, CBS-L is consistently better in all cases than all baselines. Even for in-training, CBS-L perform better than SVM. This clearly shows the superiority of the proposed CBS-L method.
2. ds-osvm performs poorly. cbs-osvm is much better because it uses the negative data in feature selection and center computation.
3. SVM in the document space performed poorly (Combined) when only a small number of negative topics are used in training. It gets better than both one-class SVM baselines when more negative topics are used in training (see the reason in the next point).
4. Finally, we can also see that with the number of training negative topics increases, the results of the combined case of both SVM and CBS-L improve. This is expected because with the increased number of negative topics for training, the number of not-in-training negative topics for testing decreases and the covariate shift problem gets smaller. We can also see that cbs-osvm, SVM and CBS-L's F1-scores for not-in-training improve with the increased training negative topics due to the same reason. However, their F1-scores drop for in-training because with more negative

topic	ds-osvm	cbs-osvm	SVM	CBS-L
Amplifier	0.125	0.360	0.406	0.597
Automotive	0.041	0.031	0.240	0.383
Battery	0.266	0.425	0.433	0.656
Beauty	0.079	0.401	0.470	0.618
Cable	0.131	0.028	0.231	0.500
Camera	0.376	0.361	0.433	0.523
CD Player	0.154	0.274	0.344	0.585
Clothing	0.046	0.234	0.292	0.486
Computer	0.117	0.225	0.328	0.455
Conditioner	0.075	0.195	0.381	0.519
Fan	0.408	0.581	0.581	0.724
Flashlight	0.273	0.487	0.528	0.744
Graphics Card	0.419	0.473	0.552	0.631
Headphone	0.298	0.338	0.432	0.533
Home Improvement	0.039	0.032	0.178	0.233
Jewelry	0.362	0.579	0.632	0.800
Kindle	0.107	0.387	0.416	0.685
Kitchen	0.042	0.118	0.197	0.261
Lamp	0.091	0.249	0.374	0.487
Luggage	0.105	0.482	0.506	0.482
Magazine Subscriptions	0.406	0.597	0.796	0.858
Mattress	0.435	0.562	0.603	0.702
Memory Card	0.134	0.256	0.367	0.508
Microphone	0.103	0.223	0.25	0.417
Microwave	0.378	0.577	0.637	0.735
Monitor	0.136	0.345	0.312	0.513
Mouse	0.493	0.580	0.552	0.779
Movies TV	0.146	0.507	0.641	0.682
Musical Instruments	0.073	0.241	0.446	0.575
Network Adapter	0.164	0.483	0.481	0.596
Office Products	0.040	0.193	0.327	0.346
Patio Lawn Garden	0.043	0.226	0.295	0.483
Pet Supplies	0.098	0.447	0.524	0.584
Pillow	0.491	0.640	0.781	0.888
Printer	0.549	0.557	0.624	0.859
Projector	0.230	0.459	0.482	0.805
Rice Cooker	0.571	0.616	0.692	0.942
Shoes	0.224	0.524	0.585	0.793
Speaker	0.241	0.251	0.253	0.410
Subwoofer	0.147	0.268	0.346	0.401
Table Chair	0.141	0.496	0.571	0.703
Tablet	0.069	0.234	0.142	0.424
Telephone	0.099	0.034	0.144	0.167
Tent	0.289	0.465	0.428	0.764
Toys	0.088	0.029	0.331	0.449
Video Games	0.424	0.387	0.508	0.705
Vitamin Supplement	0.052	0.026	0.341	0.527
Wall Clock	0.401	0.582	0.607	0.777
Watch	0.362	0.553	0.543	0.775
Webcam	0.155	0.304	0.372	0.645
Average	0.205	0.355	0.439	0.598

Table 3: F1-score for each positive topic or class in the combined case

topics, the data becomes more skewed, which hurts in-training classification.

To give a flavor of the detailed results for each topic (product), we give the full results for one setting with 30 randomly selected topics as the training negative data (Table 3). The results in the table are F1-scores of the combined case.

5 Conclusion

The ability to get relevant posts accurately about a topic from social media is a challenging problem. This paper attempted to solve this problem by identifying and dealing with the technical issue of covariate shift. The key idea of our technique is to transform document representation from the traditional n-gram feature space to a similarity based space. Our experimental results show that the proposed method CBS-L outperformed strong baselines by large margins.

Acknowledgments

This research was partially supported by the NSF grants IIS-1111092 and IIS-1407927, and a Google faculty award.

Reference

Alter, O., Brown, P.O. and Bostein, D. 2000. Singular Value Decomposition for Genome-Wide Expression Data Processing and Modeling. *Proc. Nat',l Academy of Science,* vol. 97, no. 18, pp. 10101-10106, Aug.

Blei, D. Ng, A. and Jordan, M., 2003. Latent dirichlet allocation, *The Journal of Machine Learning Research*, 3, p.993-1022, 3/1/2003

Buckley, C., Salton, G., Allan, J. 1994. The Effect of Adding Relevance Information in a Relevance Feedback Environment, *Proceedings of SIGIR Conference*.

Bickel, S., Bruckner, M., and Scheffer. 2009. T. Discriminative learning under covariate shift. *Journal of Machine Learning Research*.

Bickel S. and Scheffer T. 2007. Dirichlet-enhanced spam filtering based on biased samples. *Advances in Neural Information Processing Systems*.

Cha, S.-H. 2007. Comprehensive Survey on Distance/Similarity Measures between Probability Density Functions. *International Journal of Mathematical Models and Methods in Applied Sciences*, 1(4):300--307.

Chang, C-C. and Lin, C-J. 2011. LIBSVM: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1--27:27, http://www.csie.ntu.edu. tw/~cjlin/libsvm

Colas, F. and Brazdil. P. 2006. Comparison of SVM and some older classification algorithms in text classification tasks. *Artificial Intelligence in Theory and Practice*. IFIP International Federation for Information Processing, pp. 169-178.

Denis, F., PAC learning from positive statistical queries. *ALT*, 1998.

Radev, D., Jing, H. and Budzikowska, M. 2000. Centroid-based summarization of multiple documents: Sentence extraction, utility-based evaluation, and user studies. In *ANLP/NAACL Workshop on Summarization*, Seattle, April.

Dudik, M., Schapire, R., and Phillips, S. 2005. Correcting sample selection bias in maximum entropy density estimation. *Advances in Neural Information Processing Systems*.

Elkan, C. and Noto, K. 2008. Learning classifiers from only positive and unlabeled data. *KDD*, 213-220.

He, X., Cai, D., Liu, H. and Ma, W.-Y. 2004. Locality Preserving Indexing for Document Representation. *Proc. Of SIGIR*.

Heckman, J. 1979. Sample selection bias as a specification error. *Econometrica*, 47:153–161.

Huang, J., Smola, A. and Gretton, A., Borgwardt K., and Scholkopf B. 2007. Correcting sample selection bias by unlabeled data. *Advances in Neural Information Processing Systems*.

Joachims, T. 1998. Text categorization with support vector machines: Learning with many relevant features. *ECML*.

Jindal, N. and Liu, B. 2008. Opinion Spam and Analysis. *Proceedings of the ACM Conference on Web Search and Data Mining*.

Khan, S., and Madden, M. 2010. A survey of recent trends in one class classification. *Artificial Intelligence and Cognitive Science*, volume 6206 of Lecture Notes in Computer Science. 188–197.

Khan, S. and Madden, M. 2014. One-Class Classification: Taxonomy of Study and Review of Techniques. *The Knowledge Engineering Review*, 1-30.

Lebanon, G. 2006. Sequential document repre-

sentations and simplicial curves. *UAI*.

Lee, W. S. and Liu, B. 2003. Learning with Positive and Unlabeled Examples Using Weighted Logistic Regression. *ICML*.

Li, H. 2011. *Learning to Rank for Information Retrieval and Natural Language Processing*. Morgan & Claypool publishers.

Li, K., Huang, H., Tian, S. and Xu, W. 2003. Improving One-class SVM for anomaly detection. *Proc. of the Second International conference on Machine Learning and Cybernetics*, volume 5, pages 3077–3081.

Li, X., Liu, B. and Ng. S.-K. 2010. Negative Training Data can be Harmful to Text Classification. *EMNLP*.

Liu, B, Dai, Y., Li, X., Lee, W-S. and Yu. P. 2003. Building text classifiers using positive and unlabeled examples. *ICDM*.

Liu. T. 2011. *Learning to Rank for Information Retrieval*. Springer.

Luo, J., Ding, L., Pan, Z., Ni, G. and Hu, G. 2007. Research on cost-sensitive learning in one-class anomaly detection algorithms. *Autonomic and Trusted Computing*, volume 4610 of Lecture Notes in Computer Science.

Manevitz, L. and Yousef. M. 2001. One-class SVMs for document classification. *Journal of Machine Learning research*.

Manning, C. D., Prabhakar R., and Hinrich, S. 2008. *Introduction to Information Retrieval*. Cambridge University Press.

Qian, T. and Liu, B. 2013. Identifying Multiple Userids of the Same Author. *EMNLP*.

Ranzato, M. and Szummer, M. 2008. Semi-supervised learning of compact document representations with deep networks. *ICML*.

Rocchio, J. 1971. Relevant feedback in information retrieval. In G. Salton (ed.). *The smart retrieval system: experiments in automatic document processing*.

Schölkopf, B., Williamson, R., Smola, A., Taylor, J. and Platt, J. 2000. Support vector method for novelty detection. *Neural Information Processing Systems*, pages 582–588.

Schölkopf, B., Platt, J., Shawe-Taylor, J., Smola, A. and Williamson, R. 1999. Estimating the support of a high-dimensional distribution. *Technical Report*, Microsoft Research, MSR-TR-99-87.

Shimodaira, H. 2000. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of Statistical Planning and Inference*, 90:227–244.

Sugiyama, M. and Muller, K.-R. 2005. Input-dependent estimation of generalization error under covariate shift. *Statistics and Decision*, 23(4):249–279.

Sugiyama, M., Nakajima, S., Kashima, H., von Bunau P., and Kawanabe M. 2008. Direct importance estimation with model selection and its application to covariate shift adaptation. *Advances in Neural Information Processing Systems*.

Tax, D. and Duin, R. 1999a. Data domain description using support vectors. *Proceedings ESAN99*, Brussels. 251-256

Tax, D. and Duin, R. 1999b. Support vector domain description. *Pattern Recognition Letters* 20. 1191-1199

Tax, D. and Duin, R. 2001. Uniform object generation for optimizing one-class classifiers. *J. of Machine Learning Research*, 2:155–173.

Tian, J. and Gu, H. 2010. Anomaly detection combining one-class SVMs and particle swarm optimization algorithms. *Nonlinear Dynamics*, 61(1-2): 303–310.

Tsuboi, J., Kashima, H., Hido, S., Bickel, S., and Sugiyama, M. 2008. Direct density ratio estimation for large-scale covariate shift adaptation. *Proceedings of the SIAM International Conference on Data Mining* (*SDM*).

Wang, P. and Domeniconi, C. 2008. Building semantic kernels for text classification using Wikipedia, *KDD*.

Yang, Y. and Pedersen, J. O. 1997. A comparative study on feature selection in text categorization. *ICML*.

Yang, L., and Madden, M. 2007. One-class support vector machine calibration using particle swarm optimization. *AICS*, Dublin.

Yu, H., Han, J. and Chang, K. 2002. PEBL: Positive example based learning for Web page classification using SVM. *KDD*, 239-248.

Zadrozny, B. 2004. Learning and evaluating classifiers under s ample selection bias, *ICML*.

Co-Training for Topic Classification of Scholarly Data

Cornelia Caragea[1], **Florin Bulgarov**[1], **Rada Mihalcea**[2]
[1]Computer Science and Engineering, University of North Texas, TX, USA
[2]Computer Science and Engineering, University of Michigan, MI, USA
ccaragea@unt.edu, FlorinBulgarov@my.unt.edu,mihalcea@umich.edu

Abstract

With the exponential growth of scholarly data during the past few years, effective methods for topic classification are greatly needed. Current approaches usually require large amounts of expensive labeled data in order to make accurate predictions. In this paper, we posit that, in addition to a research article's textual content, its citation network also contains valuable information. We describe a co-training approach that uses the text and citation information of a research article as two different views to predict the topic of an article. We show that this method improves significantly over the individual classifiers, while also bringing a substantial reduction in the amount of labeled data required for training accurate classifiers.

1 Introduction

As science advances, scientists around the world continue to produce a large number of research articles, which provide the technological basis for worldwide dissemination of scientific discoveries. Online digital libraries such as Google Scholar, CiteSeer^{x}, and PubMed store and index millions of such research articles and their metadata, and make it easier for researchers to search for scientific information. These libraries require *effective* and *efficient* methods for topic classification of research articles in order to facilitate the retrieval of content that is tailored to the interests of specific individuals or groups. Supervised approaches for topic classification of research articles have been developed, which generally use either the content of the articles (Caragea et al., 2011), or take into account the citation relation between research articles (Lu and Getoor, 2003).

To be successful, these supervised approaches assume the availability of large amounts of labeled data, which require intensive human labeling effort. In this paper, we explore a semi-supervised approach that can exploit large amounts of unlabeled data together with small amounts of labeled data for accurate topic classification of research articles, while minimizing the human effort required for data labeling. In the scholarly domain, research articles (or papers) are highly interconnected in giant citation networks, in which papers cite or are cited by other papers. We posit that, in addition to a document's textual content and its local neighborhood in the citation network, other information exists that has the potential to improve topic classification. For example, in a citation network, information flows from one paper to another via the citation relation (Shi et al., 2010). This information flow and the topical influence of one paper on another are specifically captured by means of citation contexts, i.e., short text segments surrounding a citation's mention.

These contexts are not arbitrary, but they often serve as brief summaries of a cited paper. We therefore hypothesize that these *micro-summaries* can be successfully used as an independent view of a research article in a co-training framework to reduce the amount of labeled data needed for the task of topic classification.

The idea of using terms from citation contexts stems from the analysis of hyperlinks and the graph structure of the Web, which are instrumental in Web search (Manning et al., 2008). Many search engines follow the intuition that the anchor text pointing to a page is a good descriptor of its content, and thus anchor text terms are used as additional index terms for a target webpage. The use of links and anchor text was thoroughly researched for information retrieval (Koolen and Kamps, 2010), broadening a user's search (Chakrabarti et al., 1998), query refinement (Kraft and Zien, 2004), and enriching document representations (Metzler et al., 2009). Blum and

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 2357–2366,
Lisbon, Portugal, 17-21 September 2015.

Mitchell (1998) introduced the co-training algorithm using hyperlinks and anchor text as a second, independent view of the data for classifying webpages, in addition to a webpage content.

Contributions and Organization. We present a co-training approach to topic classification of research papers that effectively incorporates information from a citation network, in addition to the information contained in each paper. The result of this classification task will aid indexing of documents in digital libraries, and hence, will lead to improved organization, search, retrieval, and recommendation of scientific documents. Our contributions are as follows:

- We propose the use of citation contexts as an additional view in a co-training approach, which results in high accuracy classifiers. To our knowledge, this has not been addressed in the literature.
- We show experimentally that our co-training classifiers significantly outperform: (1) supervised classifiers trained using either content or citation contexts independently, for the same fraction of labeled data; and (2) several other semi-supervised classifiers, trained on the same fractions of labeled and unlabeled data as co-training.
- We also show that using the citation context information available in citation networks, the human effort involved in data labeling for training accurate classifiers can be largely reduced. Our co-training classifiers trained on a very small sample of labeled data and a large sample of unlabeled data yield accurate topic classification of research articles.

The rest of the paper is organized as follows. In Section 2, we discuss related work. Section 3 describes our data and its characteristics, followed by the presentation of our proposed co-training approach in Section 4. We present experiments and results in Section 5, and conclude the paper and present future directions of our work in Section 6.

2 Related Work

We discuss here the most relevant works to our study. A large variety of methods have been proposed in the literature with regard to automatic text classification and topic prediction. Different classifiers have been applied on the Vector Space Model (VSM), in which a document is represented as a vector of words or phrases associated with their TF-IDF score, i.e. *term frequency - inverse document frequency* (Zhang et al., 2011; Kansheng et al., 2011). VSM is the most used method due to its simple, efficient and easy to understand implementation. Another widely used model is the Latent Semantic Indexing (LSI) where co-occurrences are analyzed to find semantic relationships between words or phrases (Zhang et al., 2011; Ganiz et al., 2011). Moreover, a great range of classifiers were used for this task, including: Naïve Bayes (Lewis and Ringuette, 1994), K-nearest neighbors (Yang, 1999) and Support Vector Machines (Joachims, 1998). These techniques, however, all require a large number of labeled documents in order to build accurate classifiers. In contrast, we propose a co-training algorithm that only requires a small amount of labeled data in order to make accurate topic classification.

Semi-supervised methods essentially involve different means of transferring labels from *labeled* to *unlabeled* samples in the process of learning a classifier that can generalize well on new unseen data. Co-training was originally introduced in (Blum and Mitchell, 1998) where it was used to classify web pages into *academic course home page* or not. This approach has two views of the data as follows: the content of a web page, and the words found in the anchor text of the hyperlinks that point to the web page. Wan (2009) used co-training for cross-lingual sentiment classification of product reviews, where English and Chinese features were considered as two independent views of the data. Furthermore, Gollapalli et al. (2013) used co-training to identify authors' homepages from the current-day university websites. The paper presents novel features, extracted from the URL of a page, that were used in conjunction with content features, forming two complementary views of the data.

Citation networks have been used before in other problems. Caragea et al. (2014) used citation contexts to extract informative features for keyphrase extraction. Lu and Getoor (2003) proposed an approach for document classification that used only citation links, without any textual data from the citation contexts. Ritchie et al. (2006) used a combination of terms from citation contexts and existing index terms of a paper to improve indexing of cited papers. Citation contexts were also used to improve the performance of citation recommendation systems (Kataria et al., 2010) and to study author influence in document networks

(Kataria et al., 2011). Moreover, citation contexts were used for scientific paper summarization (Abu-Jbara and Radev, 2011; Qazvinian et al., 2010; Qazvinian and Radev, 2008; Mei and Zhai, 2008; Lehnert et al., 1990) For example, in Qazvinian et al. (2010), a set of important keyphrases is extracted first from the citation contexts in which the paper to be summarized is cited by other papers and then the "best" subset of sentences that contain such keyphrases is returned as the summary. Mei and Zhai (2008) used information from citation contexts to determine what sentences of a paper are of high impact (as measured by the influence of a target paper on further studies of similar or related topics). These sentences constitute the impact-based summary of the paper.

Despite the use of citation contexts and anchor text in many information retrieval and natural language processing tasks, to our knowledge, we are the first to propose the incorporation of citation context information available in citation networks in a co-training framework for topic classification of research papers.

3 Data

The dataset used in our experiments is a subset sampled from the CiteSeer^x digital library[1] and labeled by Dr. Lise Getoor's research group at the University of Maryland. This subset was previously used in several studies including (Lu and Getoor, 2003) and (Kataria et al., 2010). The dataset consists of 3186 labeled papers, with each paper being categorized into one of six classes: Agents, Artificial Intelligence (AI), Information Retrieval (IR), Machine Learning (ML), Human-Computer Interaction (HCI) and Databases (DB). For each paper, we acquire the citation contexts directly from CiteSeer^x. A citation context is defined as a window of n words surrounding a citation mention. We differentiate between cited and citing contexts for a paper as follows: let d be a target paper and $\mathcal{C}$ be a citation network such that $d \in \mathcal{C}$. A *cited context* for d is a context in which d is cited by some paper d_i in $\mathcal{C}$. A *citing context* for d is a context in which d is citing some paper d_j in $\mathcal{C}$. If a paper is cited in multiple contexts within another paper, the contexts are aggregated into a single context. For each paper in the dataset, we have at least one cited or one citing context. A summary of the dataset is provided in Table 1.

[1] http://citeseerx.ist.psu.edu/

Number of papers in each class						
Agents	AI	IR	ML	HCI	DB	Total
562	239	641	569	490	685	3186

Avg. Cited Contexts	Avg. Citing Contexts
45.59	20.77

Table 1: Dataset summary.

As expected, we have a higher number of cited contexts than citing contexts. This is due to the page restrictions often imposed to research articles that can limit the number of papers each article can cite. On the other hand, a good research paper can accumulate hundreds of citations, and hence, cited contexts over the years.

Context lengths. In CiteSeer^x, citation contexts have about 50 words on each side of a citation mention. A previous study by Ritchie et al. (2008) shows that a fixed window length of about 100 words around a citation mention is generally effective for information retrieval tasks. For this reason, we use the contexts provided by CiteSeer^x directly. In future, it would be interesting to study more sophisticated approaches to identifying the text that is relevant to a target citation (Abu-Jbara and Radev, 2012; Teufel, 1999) and study the influence of context lengths on our task.

For all experiments, our labeled dataset is split in train, validation and test sets. The validation and test sets have about 200 papers each. We sampled another set of papers from the labeled dataset in order to simulate the existence of unlabeled data, with a fixed size of around 2000 papers. The remaining 786 papers are used as labeled training data. Each experiment was repeated 10 times with 10 different random seeds and the results were averaged.

4 Co-Training for Topic Classification

Blum and Mitchell (1998) proposed the co-training algorithm in the context of webpage classification. In co-training, the idea is that two classifiers trained on two different views of the data teach one another by re-training each classifier on the data enriched with predicted examples that the other classifier is most confident about. In Blum and Mitchell (1998), webpages are represented using two different views: (1) using terms from webpages' content and (2) using terms from the anchor text of hyperlinks pointing to these pages.

Algorithm 1 Co-Training

Input: L, U, 's'
  $L_1 \leftarrow L$, $L_2 \leftarrow L$
  while $U \neq \varnothing$ **do**
    Train classifier C_1 on L_1
    Train classifier C_2 on L_2
    $S \leftarrow \varnothing$
    Move 's' examples from U to S
    $U \leftarrow U \backslash S$
    $S_1,\ S_2 \leftarrow$ *GetMostConfidentExamples*(S, C_1, C_2)
    $L_1 \leftarrow L_1 \cup S_1$, $L_2 \leftarrow L_2 \cup S_2$
    $U \leftarrow U \cup [S \backslash (S_1 \cup S_2)]$
  end while
Ouput: The combined classifier C of C_1 and C_2

In this paper, we study the applicability and extension of the co-training algorithm to the task of topic classification of research papers, which are embedded in large citation networks. Here, in addition to the information contained in a paper itself, *citing* and *cited* papers capture different aspects (e.g., topicality, domain of study, algorithms used) about the target paper (Teufel et al., 2006), with *citation contexts* playing an instrumental role. We conjecture that citation contexts, which act as brief summaries about a cited paper, provide important clues in predicting the topicality of a target paper. These clues give rise to the design of our co-training based model for topic classification of research papers. In our model, we use the content of a paper as one view and the citation contexts as another view of our data. In particular, for the content of a paper, we use its title and abstract as it is commonly used in the literature (Lu and Getoor, 2003); for the citation contexts, we use both the cited and citing contexts, as described in the previous section.

Our co-training procedure is described in Algorithm 1. L and U represent the labeled and unlabeled datasets and contain instances from both views. The fractions of the training set are obtained from the 786 papers by selecting $k\%$ random examples from each class. For a round of co-training, we train classifiers C_1 and C_2 on the two views. Next, s examples are sampled from the unlabeled data into S, and C_1, C_2 are used to obtain predictions for these s examples. The *GetMostConfidentExamples* method is a generic placeholder that stands for a function that determines what examples from S are chosen to be added into training. Finally, at the end of an iteration, the examples left into S are moved back to U, and the algorithm iterates until there are no more unlabeled examples in U. The final classifier C is obtained by combining C_1 and C_2 using the product of their class probability distributions. The class with the highest posterior probability (of the product of the two distributions) is chosen as the predicted class.

Unlike the original co-training algorithm described by Blum and Mitchell (1998), which tackled a binary classification task (*course* vs. *non-course* page classification), we address a multi-class classification problem, where each example (i.e., research paper) is classified into one of six different classes. Moreover, in Blum and Mitchell (1998), the co-training algorithm moves p highest confidence positive examples and n highest confidence negative examples from S to L, where $p : n$ represents the class distribution in the original labeled training set (i.e., if there are 10 positive examples and 90 negative examples in the labeled set L, then $p = 1$ positive and $n = 9$ negative examples are moved to the labeled set at each iteration of co-training). Unlike, this approach that preserves the class distribution of the original labeled training set, we move into L all examples that are classified with a confidence above a certain threshold.

5 Results and Discussion

First, the proposed method is evaluated on the validation set. We first compare it against various supervised and semi-supervised baselines. Next, we report the performance of our co-training algorithm under different scenarios, where either cited or citing contexts are used. We also show the most informative words for each classifier. Finally, with the best parameters obtained on the validation set, we report the precision, recall and F1-score, obtained by each method, on the test set.

In experiments, the sample size 's' from Algorithm 1 is set to 300, i.e. the number of documents sampled from the unlabeled pool at each iteration; the confidence threshold is set to 0.95, i.e. if both classifiers agree on the class label and have a confidence ≥ 0.95, the instance is labeled and moved into the labeled training set. These parameters are estimated on the validation set, but the results are not shown due to space limitation.

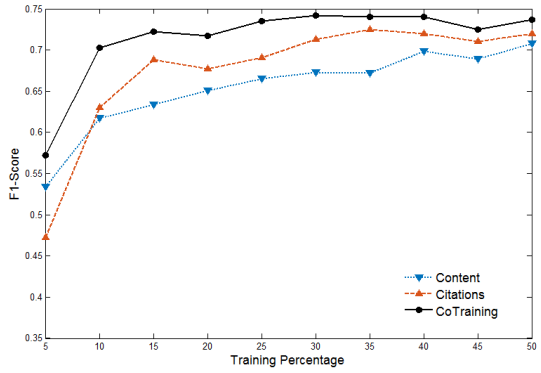


Figure 1: Co-Training vs. Supervised Learning.

Evaluation Measures. We report results averaged over ten different runs with random splits. For each random split, we return the weighted average precision, recall and F1-score. In all the experiments, we use the Naïve Bayes Multinomial classifier and its Weka implementation[2], with term-frequencies as feature values. We experimented with both TF and TF-IDF scores, using different classifiers (Support Vector Machine, Naïve Bayes Multinomial, and simple Naïve Bayes classifiers), but Naive Bayes Multinomial with TF performed best.

5.1 Baseline Comparisons

How does co-training compare with supervised learning techniques? In this experiment, we compare our co-training method with two supervised baselines: (1) when only document content is used and (2) when only citation contexts are used.

Figure 1 shows the F1-scores achieved using different initial training sizes. We can see that overall, the citation contexts are better at predicting the topic of a document compared with the content, outperforming them in 9 out of 10 experimental settings. The only exception to this trend is when a small number (5%) of training instances is available, in which case the supervised content view performs better, reaching an F1-score of 0.534. Regardless, the co-training method shows significant improvement over both baselines, in all experiments. Starting with an F1-score of 0.572, it continues to improve its performance as the training percentage is increasing. The maximum F1-score, i.e. 0.742, is reached when 30% of the labeled training set is used. Note that the difference in performance between co-training and the two supervised baselines is statistically significant for a p value of 0.05.

A fully supervised baseline that uses 100% of the training set achieves an F1-score of 0.720 (using content) and 0.738 (using citation contexts). In contrast, co-training requires only 15% of the labeled training set to outperform the fully supervised content baseline and 30% of the training set to outperform the fully supervised citation contexts baseline. Consequently, using a co-training approach that includes citation contexts as well as the document content can not only increase the performance, but will also significantly reduce the need of expensive labeled instances.

Figure 2 illustrates the confusion matrices of three experiments: (a) supervised content view, i.e. the title and abstract, (b) supervised citation contexts view, and (c) co-training that uses both views. These experiments use 10% of the training set. Each of the matrices are represented by a heat map, i.e. the redder the color, the higher the value assigned to that position. An accuracy of 1 will be represented by a matrix with red blocks on the main diagonal and white blocks everywhere else. This experiment was performed 10 times with 10 different seeds and the results have been averaged.

As can be seen, the matrix that uses only titles and abstracts, i.e. left side, is showing the highest percentage of misclassified documents, classifying correctly about 58.8% instances, on average. Using only citation contexts in a supervised framework, i.e. center matrix, we reach a higher accuracy of 60.7%. The co-training method, which uses the content of the paper and citations as two independent views, significantly increases the average accuracy to 67.3%. This experiment shows that citation contexts are better than titles and abstracts at predicting the topic of a document. Furthermore, our proposed approach, which uses the content of the paper as well as citation contexts, achieves higher results than each view used separately. The difference in accuracy is statistically significant across all three experiments for a p value of 0.05.

Overall, the *Agents* class seem to be the easiest to classify, reaching an accuracy value of 91.6% when using co-training. On the other hand, the *AI* class is the hardest to classify. One reason for this is that the *AI* class contains the lowest number of instances in the dataset. Another can be that the *AI* class is the most general among all classes and therefore, classifying documents with this la-

[2]http://www.cs.waikato.ac.nz/ml/weka/

Figure 2: The accuracy of our method, against two supervised baselines.
Left: using titles and abstracts; Center: using citation contexts; Right: using co-training.

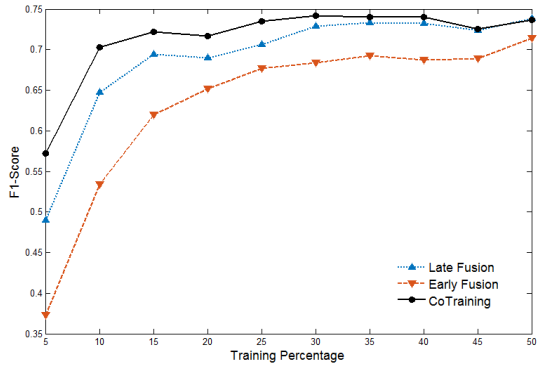


Figure 3: Co-Training vs. Early and Late Fusion.

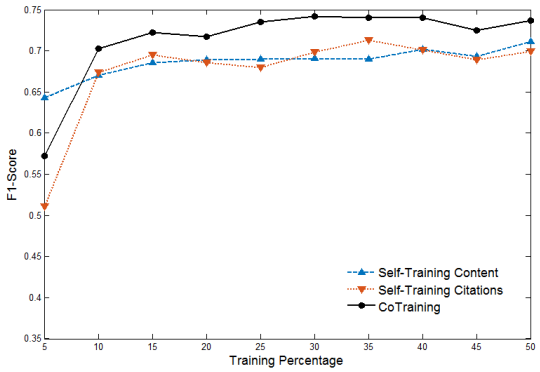


Figure 4: Co-Training vs. Self-Training.

bel can be a difficult task even for a human. Other common misclassifications occur between classes like *HCI* and *Agents*, *ML* and *IR* or *AI* and *ML*, due to their similarity.

How does our co-training method compare with other supervised approaches? In this experiment, we compare the performance of co-training against two other methods: early and late fusion. In early fusion, the feature vectors of the two views are concatenated, creating a single representation of the data. In contrast, late fusion trains two separate classifiers and then combines them by taking the label with the highest confidence.

Figure 3 shows this comparison over different training sizes. The results show that the co-training method is more accurate than all others, performing best in all 10 experimental settings. Late fusion has an overall lower performance compared with co-training, but is in a tight correlation with it. On the other hand, early fusion achieves the lowest F1-score across the experiments. The reported results are statistically significant at p value of 0.05, when the training percentage is between 5 and 35. Therefore, we can say that training two separate classifiers, one of each view, yields higher performance compared with training a single classifier that incorporates both views. Moreover, using a co-training approach that incorporates information from unlabeled data into the model, will help the two classifiers increase their confidences and minimize the error rate.

How does co-training compare with semi-supervised methods? Here, we present results comparing co-training with two other well-known semi-supervised techniques: self-training and Naïve Bayes with Expectation Maximization.

Self-Training. First, we show results of the comparison of co-training with two variations of self-training: (1) self-training using only document content, and (2) self-training using only citation contexts. Figure 4 shows the results of this experiment. Self-training is similar to co-training, except that it uses only one view of the data (Zhu, 2005). Self-training parameters, e.g., sample size 's' or number of iterations, are estimated as in co-training.

Although the document content version of self-training outperforms co-training when using 5%

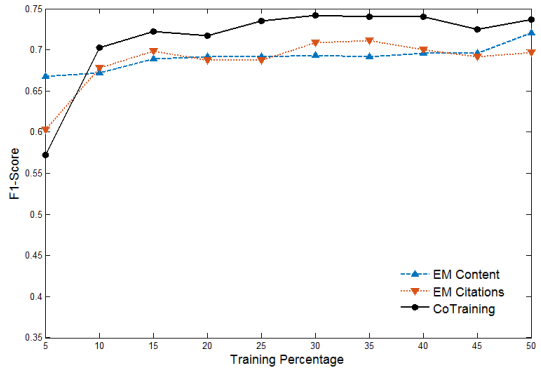


Figure 5: Co-Training vs. EM.

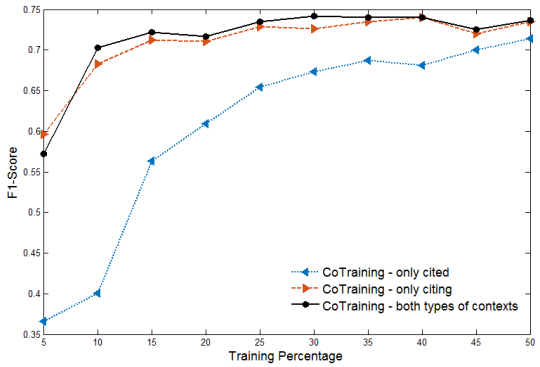


Figure 6: Performance when using only cited / only citing / or both citation contexts.

of the training instances, we can see that overall, there is a significant difference in terms of F1-score values in the favor of co-training. In 9 out of 10 experiments, our co-training approach is superior to both self-training methods. The results are statistically significant across all experimental setups for a p value of 0.05.

Expectation Maximization. Figure 5 shows the F1-score values obtained after running NBM with EM with the same training, unlabeled and test sets. The EM algorithm uses the same classifier, i.e. NBM, and the weight for each unlabeled instance is set to 1, as this setting achieved the highest results. Two different experiments were performed using EM: (1) using only document content, and (2) using only citation contexts. As can be seen in the figure, overall, the co-training approach significantly outperforms both variations of EM. However, the co-training method falls short when using 5% of the training instances, where *EM Content* and *EM Citations* methods are achieving higher F1-score values. Nonetheless, both EM variations tend to achieve an F1-score value below or equal to 0.710, whereas co-training reaches performance values of 0.74 or higher. Again, the comparison results between co-training and both variations of EM are statistically significant for training sizes between 10% and 50%, for a p value of 0.05.

5.2 Using Different Citation Context Types

Which of the two types of citation contexts (cited or citing) help the task of topic classification more and how does co-training perform in the absence of either one? The answer to this question is important as there are cases in which citation contexts are not readily available. One frequently encountered example includes newly published research papers that have no cited contexts. In this case, it is important to know how our method performs when we only have one type of citation contexts. Figure 6 shows the difference in performance when using: (1) only cited contexts, (2) only citing contexts, and (3) both context types. Note that the content view remains the same across all three experiments.

The plot is showing that citing contexts are bringing in a significantly higher margin of knowledge compared with cited contexts. This is consistent over different training set sizes, as shown in the figure, with a more prominent impact when a small training size is used, i.e. 5-30%. The fact that the citing contexts achieve higher F1-score than cited contexts is consistent with the intuition that when citing a paper y, an author generally summarizes the main ideas from y using important words from a target paper x, making the citing contexts to have higher overlap with words from x. In turn, a paper z that cites x may use paraphrasing to summarize ideas from x with words more similar to those from the content of z.

When the two types of contexts are used, co-training achieves higher results compared with cases when only one context type is used. This experiment shows that our method can be applied for both old and new research articles. Citing contexts will be available in the text of the target paper and are independent of the existence of the cited contexts.

5.3 Informative Features

What are the most informative words from each view: document content and citation contexts? Figure 7 shows the words from each view that are most useful for our topic classification task. The larger the word, the more informative is for our

Figure 7: Most informative words from document content (left) and citation contexts (right).

Method	Labeled docs. (%)	Precision	Recall	F1-Score
Co-Training	**30**	**0.749**	**0.743**	**0.742**
Co-Training - only citing	40	0.747	0.740	0.740
Co-Training - only cited	50	0.724	0.717	0.714
Self-Training - Content	50	0.723	0.711	0.711
Self-Training - Citations	35	0.730	0.710	0.713
EM - Content	50	0.738	0.714	0.721
EM - Citations	35	0.729	0.707	0.711
Early Fusion	50	0.718	0.710	0.714
Late Fusion	50	0.748	0.734	0.738
Content - Fully Supervised	100	0.730	0.728	0.720
Citations - Fully Supervised	100	0.745	0.740	0.738

Table 2: A comparison of all methods on the test set.

task. To determine the informativeness of a word, we used its Information Gain score. For these experiments, we used training sets consisting of 30% of the instances, setting in which we achieved the best results on the validation and test sets using our proposed co-training approach.

As can be seen, the two word clouds have a high word overlap. Words such as *agent*, *database* or *query* are almost equally important in the two views, dominating both clouds. However, differences can be observed. For example, words like *learning*, *multi-agent* or *interface* are more important in the content view. On the other hand, words such as *document* or *text* achieve a higher information gain score for the citation contexts view.

5.4 Co-Training vs. All Other Approaches

Table 2 summarizes the results obtained by all the baselines used so far, in comparison with our proposed co-training method. For this experiment, we show the training percentage used, the precision, recall and F1-score for each method, in the setting in which it returned the best results. All measures were averaged after 10 runs with 10 different seeds.

The results in Table 2 show that the proposed co-training method outperforms all compared models, reaching the highest F1-score of 0.742, while using the smallest amount of labeled documents, i.e. 30%. Using only the citing contexts, the performance is similar to that of co-training when both context types are used. However, using only the cited contexts, the performance decreases compared to that of the full model that uses both context types. We see that the citing contexts perform better, reaching an F1-score value of 0.740 compared against 0.714 when only cited contexts are used. Moreover, the method that uses only the citing contexts is using 10% less labeled data.

Self-training and EM show decreased performance compared with co-training. Late Fusion outperforms Early Fusion, i.e., 0.738 vs. 0.714, both obtaining lower results than co-training, while using significantly more labeled data.

The last two lines of the table show the results when all documents (except those in the validation and test), are used for training, in a supervised framework. As can be seen, a supervised method that uses only citations will achieve a higher performance, compared against a method that uses titles and abstracts. Nonetheless, co-training obtains higher results than both fully supervised approaches, while using only 30% of the labeled data.

6 Conclusion and Future Work

In this paper, we studied the problem of using citation contexts in order to predict more accurately the topic of a research article. We showed that a co-training technique, which uses the paper content and its citation contexts as two *conditionally independent* and *sufficient* views of the data, can effectively incorporate cheap, unlabeled data to improve the classification performance and to reduce the need of labeled examples to only a fraction. The results of the experiments showed that the proposed approach performs better than other semi-supervised and supervised methods.

This study also shows that citation contexts are rich sources of information that can be successfully used in various IR and NLP tasks. We showed that document content and citation contexts unified under the same algorithm can dramatically decrease the annotation costs as well. In the future, we plan to extend co-training to include active learning for more robust classification. Moreover, it would be interesting to extend the co-training approach to multi-views that could potentially handle more than two feature spaces, e.g., it could include topics by Latent Dirichlet Allocation (Blei et al., 2003) as an additional view.

Acknowledgments

We are thankful to Dr. Lise Getoor for making the Citeseer^x labeled subset publicly available. We are also grateful to Dr. C. Lee Giles for the CiteSeer^x data, which helped extract the citation contexts of the research papers in the collection. We very much thank our anonymous reviewers for their constructive feedback. This research is supported in part by the NSF award #1423337 to Cornelia Caragea. Any opinions, findings, and conclusions expressed here are those of the authors and do not necessarily reflect the views of NSF.

References

Amjad Abu-Jbara and Dragomir Radev. 2011. Coherent citation-based summarization of scientific papers. In *Proc. of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, HLT '11, pages 500–509.

Amjad Abu-Jbara and Dragomir Radev. 2012. Reference scope identification in citing sentences. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL HLT '12, pages 80–90, Stroudsburg, PA, USA. Association for Computational Linguistics.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *The Journal of Machine Learning Research*, 3:993–1022.

Avrim Blum and Tom Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, COLT' 98, pages 92–100, New York, NY, USA. ACM.

Cornelia Caragea, Adrian Silvescu, Saurabh Kataria, Doina Caragea, and Prasenjit Mitra. 2011. Classifying scientific publications using abstract features. In *The Symposium on Abstraction, Reformulation, and Approximation (SARA)*.

Cornelia Caragea, Adrian Florin Bulgarov, Andreea Godea, and Sujatha Das Gollapalli. 2014. Citation-enhanced keyphrase extraction from research papers: A supervised approach. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1435–1446. Association for Computational Linguistics.

Soumen Chakrabarti, Byron Dom, Prabhakar Raghavan, Sridhar Rajagopalan, David Gibson, and Jon Kleinberg. 1998. Automatic resource compilation by analyzing hyperlink structure and associated text. *Comput. Netw. ISDN Syst.*, 30(1-7):65–74, April.

Murat Can Ganiz, Cibin George, and William M Pottenger. 2011. Higher order naive bayes: a novel non-iid approach to text classification. *Knowledge and Data Engineering, IEEE Transactions on*, 23(7):1022–1034.

Sujatha Das Gollapalli, Cornelia Caragea, Prasenjit Mitra, and C. Lee Giles. 2013. Researcher homepage classification using unlabeled data. In *Proceedings of the 22Nd International Conference on World Wide Web*, WWW '13, pages 471–482, Republic and Canton of Geneva, Switzerland. International World Wide Web Conferences Steering Committee.

Thorsten Joachims. 1998. Text categorization with suport vector machines: Learning with many relevant features. In *Proceedings of the 10th European Conference on Machine Learning*, ECML '98, pages 137–142, London, UK, UK. Springer-Verlag.

SHI Kansheng, HE Jie, Hai-tao LIU, Nai-tong ZHANG, and Wen-tao SONG. 2011. Efficient text classification method based on improved term reduction and term weighting. *The Journal of China Universities of Posts and Telecommunications*, 18:131–135.

Saurabh Kataria, Prasenjit Mitra, and Sumit Bhatia. 2010. Utilizing context in generative bayesian models for linked corpus. In *AAAI*, volume 10, page 1.

Saurabh Kataria, Prasenjit Mitra, Cornelia Caragea, and C. Lee Giles. 2011. Context sensitive topic models for author influence in document networks. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Three*, IJCAI'11, pages 2274–2280.

Marijn Koolen and Jaap Kamps. 2010. The importance of anchor text for ad hoc search revisited. In *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '10, pages 122–129, New York, NY, USA. ACM.

Reiner Kraft and Jason Zien. 2004. Mining anchor text for query refinement. In *Proceedings of the 13th International Conference on World Wide Web*, WWW '04, pages 666–674, New York, NY, USA. ACM.

Wendy Lehnert, Claire Cardie, and Ellen Rilofl. 1990. Analyzing research papers using citation sentences. In *Proceedings of the 12th Annual Conference of the Cognitive Science Society*, pages 511–518.

David D. Lewis and Marc Ringuette. 1994. A comparison of two learning algorithms for text categorization. In *In Third Annual Symposium on Document Analysis and Information Retrieval*, pages 81–93.

Qing Lu and Lise Getoor. 2003. Link-based classification. In *International Conference on Machine Learning*, pages 496–503.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA.

Qiaozhu Mei and ChengXiang Zhai. 2008. Generating impact-based summaries for scientific literature. In *Proceedings of ACL-08: HLT*, pages 816–824, Columbus, Ohio.

Donald Metzler, Jasmine Novak, Hang Cui, and Srihari Reddy. 2009. Building enriched document representations using aggregated anchor text. In *Proceedings of the 32Nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '09, pages 219–226, New York, NY, USA. ACM.

Vahed Qazvinian and Dragomir R. Radev. 2008. Scientific paper summarization using citation summary networks. In *Proc. of the 22nd Intl. Conference on Computational Linguistics*, COLING '08, pages 689–696, Manchester, United Kingdom.

Vahed Qazvinian, Dragomir R. Radev, and Arzucan Özgür. 2010. Citation summarization through keyphrase extraction. In *Proceedings of the 23rd International Conference on Computational Linguistics*, COLING '10, pages 895–903.

Anna Ritchie, Simone Teufel, and Stephen Robertson. 2006. How to find better index terms through citations. In *Proc. of the Workshop on How Can Computational Linguistics Improve Information Retrieval?*, CLIIR '06, pages 25–32, Sydney, Australia.

Anna Ritchie, Stephen Robertson, and Simone Teufel. 2008. Comparing citation contexts for information retrieval. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management*, CIKM '08, pages 213–222, New York, NY, USA. ACM.

Xiaolin Shi, Jure Leskovec, and Daniel A. McFarland. 2010. Citing for high impact. In *Proceedings of the 10th Annual Joint Conference on Digital Libraries*, JCDL '10, pages 49–58.

Simone Teufel, Advaith Siddharthan, and Dan Tidhar. 2006. Automatic classification of citation function. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, EMNLP '06, pages 103–110.

S. Teufel. 1999. *Argumentative Zoning: Information Extraction from Scientific Text*. Ph.D. thesis, University of Edinburgh,.

Xiaojun Wan. 2009. Co-training for cross-lingual sentiment classification. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1 - Volume 1*, ACL '09, pages 235–243, Stroudsburg, PA, USA. Association for Computational Linguistics.

Yiming Yang. 1999. An evaluation of statistical approaches to text categorization. *Journal of Information Retrieval*, 1:67–88.

Wen Zhang, Taketoshi Yoshida, and Xijin Tang. 2011. A comparative study of tf* idf, lsi and multi-words for text classification. *Expert Systems with Applications*, 38(3):2758–2765.

Xiaojin Zhu. 2005. Semi-Supervised learning literature survey. Technical report, Computer Sciences, University of Wisconsin-Madison.

Humor Recognition and Humor Anchor Extraction

Diyi Yang, Alon Lavie, Chris Dyer, Eduard Hovy
Language Technologies Institute, School of Computer Science
Carnegie Mellon University. Pittsburgh, PA, 15213, USA
{diyiy, alavie, cdyer}@cs.cmu.edu, hovy@cmu.edu

Abstract

Humor is an essential component in personal communication. How to create computational models to discover the structures behind humor, recognize humor and even extract humor anchors remains a challenge. In this work, we first identify several semantic structures behind humor and design sets of features for each structure, and next employ a computational approach to recognize humor. Furthermore, we develop a simple and effective method to extract anchors that enable humor in a sentence. Experiments conducted on two datasets demonstrate that our humor recognizer is effective in automatically distinguishing between humorous and non-humorous texts and our extracted humor anchors correlate quite well with human annotations.

1 Introduction

Humor is one of the most interesting and puzzling research areas in the field of natural language understanding. Recently, computers have changed their roles from automatons that can only perform assigned tasks to intelligent agents that dynamically interact with people and learn to understand their users. When a computer converses with a human being, if it can figure out the humor in human's language, it can better understand the true meaning of human language, and thereby make better decisions that improve the user experience. Developing techniques that enable computers to understand humor in human conversations and adapt behavior accordingly deserves particular attention.

The task of *Humor Recognition* refers to determining whether a sentence in a given context expresses a certain degree of humor. Humor recognition is a challenging natural language problem (Attardo, 1994). First, a universal definition of humor is hard to achieve, because different people hold different understandings of even the same sentence. Second, humor is always situated in a broader context that sometimes requires a lot of external knowledge to fully understand it. For example, consider the sentence, "*The one who invented the door knocker got a No Bell prize*" and "*Veni, Vidi, Visa: I came, I saw, I did a little shopping*". One needs a larger cultural context to figure out the subtle humorous meaning expressed in these two sentences. Last but not least, there are different types of humor (Raz, 2012), such as wordplay, irony and sarcasm, but there exist few formal taxonomies of humor characteristics. Thus it is almost impossible to design a general algorithm that can classify all the different types of humor, since even human cannot perfectly classify all of them.

Although it is impossible to understand universal humor characteristics, one can still capture the possible latent structures behind humor (Bucaria, 2004; Binsted and Ritchie, 1997). In this work, we uncover several latent semantic structures behind humor, in terms of meaning incongruity, ambiguity, phonetic style and personal affect. In addition to humor recognition, identifying anchors, or which words prompt humor in a sentence, is essential in understanding the phenomenon of humor in language. Here, *Anchor Extraction* refers to extracting the semantic units (keywords or phrases) that enable the humor in a given sentence. The presence of such anchors plays an important role in generating humor within a sentence or phrase.

In this work, we formulate humor recognition as a classification task in which we distinguish between humorous and non-humorous instances. Then we explore the semantic structure behind humor from four perspectives: incongruity, am-

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 2367–2376,
Lisbon, Portugal, 17-21 September 2015.

biguity, interpersonal effect and phonetic style. For each latent structure, we design a set of features to capture the potential indicators of humor. With high classification accuracy, we then extract humor anchors in sentences via a simple and effective method. Both quantitative and qualitative experimental results are provided to validate the classification and anchor extraction performance.

2 Related Work

Most existing studies on humor recognition are formulated as a binary classification problem and try to recognize jokes via a set of linguistic features (Purandare and Litman, 2006; Kiddon and Brun, 2011). For example, Mihalcea and Strapparava (2005) defined three types of humor-specific stylistic features: Alliteration, Antonym and Adult Slang, and trained a classifier based on these feature representations. Similarly, Zhang and Liu (2014) designed several categories of humor-related features, derived from influential humor theories, linguistic norms, and affective dimensions, and input around fifty features into the Gradient Boosting Regression Tree model for humor recognition. Taylor and Mazlack (2004) recognized wordplay jokes based on statistical language recognition techniques, where they learned statistical patterns of text in N-grams and provided a heuristic focus for a location of where wordplay may or may not occur. Similar work can also be found in (Taylor, 2009), which described humor detection process through Ontological Semantics by automatically transposing the text into the formatted text-meaning representation to detect humor. In addition to language features, some other studies also utilize spoken or multimodal signals. For example, Purandare and Litman (2006) analyzed acoustic-prosodic and linguistic features to automatically recognize humor during spoken conversations. However, the humor related features in most of those works are not systematically derived or explained.

One essential component in humor recognition is the construction of negative data instances. Classifiers based on negative samples that lie in a different domain than humor positive instances will have high classification performance, but are not necessarily good classifiers. There are few existing benchmark datasets for humor recognition and most studies select negative instances specifically. For example, Mihalcea and Strapparava (2005) constructed the set of negative examples by using news title from Reuters news, proverbs and British National Corpus. (Zhang, el. al 2014) randomly sampled 1500 tweets and then asked annotators to filter out humorous tweets.

Compared to humor recognition, humor generation has received quite a lot attention in the past decades(Stock and Strapparava, 2005; Ritchie, 2005; Hong and Ong, 2009). Most generation work draws on humor theories to account for humor factors, such as the Script-based Semantic Theory of Humor (Raskin, 1985; Labutov and Lipson, 2012) and employs templates to generate jokes. For example, Ozbal and Strapparava (2012) created humorous neologism using WordNet and ConceptNet. In detail, their system combined several linguistic resources to generate creative names, more specifically neologisms based on homophonic puns and metaphors. Stock and Strapparava (2005) introduced HAHACRONYM, a system (an acronym ironic re-analyzer and generator) devoted to produce humorous acronyms mainly by exploiting incongruity theories (Stock and Strapparava, 2003).

In contrast to research on humor recognition and generation, there are few studies that identify the humor anchors that trigger humorous effects in general sentences. A certain type of jokes might have specific structures or characteristics that provide pointers to humor anchors. For example, in the problem of "That's what she said" (Kiddon and Brun, 2011), characteristics that involves the using of nouns that are euphemisms for sexually explicit nouns or structures common in the erotic domain might probably give clues to potential humor anchors. Similarly, in the Knock Knock jokes (Taylor and Mazlack, 2004), wordplay is what leads to the humor. However, the wordplay by itself is not enough to trigger the comic effect, thus not equivalent to the humor anchors for a joke. To address these issues, we introduce a formal definition of humor anchors and design an effective method to extract such anchors in this work. To the best of our knowledge, this is the first study on extracting humor anchors that trigger humor in general sentences.

3 Data Preparation

To perform automatic recognition of humor and humor anchor extraction, a data set consisting of both humorous (positive) and non-humorous (negative) examples is needed. The dataset we

use to conduct our humor recognition experiments includes two parts: Pun of the Day [1] and the 16000 One-Liner dataset (Mihalcea and Strapparava, 2005). The two data sets only contain humorous text. In order to acquire negative samples for the humor classification task, we sample negative samples from four resources, including AP News[2], New York Times, Yahoo! Answer[3] and Proverb[4]. Such datasets not only enable us to automatically learn computational models for humor recognition, but also provide us with the chances to evaluate the performance of our model.

However, directly applying sentences extracted from those four resources and simply treating them as negative instances of humor recognition could result in deceptively high performance of classification, due to the domain differences between positive and negative datasets. For example, the humor sentences in our positive datasets often relate to daily lives, such as "*My wife tells me I'm a skeptic, but I don't believe a word she says.*". Meanwhile, sentences in news websites sometimes describe scenes related to wars or politics, such as "*Judge Thomas P. Griesa of Federal District Court in Manhattan stopped short of issuing sanctions*". Such domain differences between descriptive words might make a naive bag of words model perform quite well, without taking into account the deeper semantic structures behind humor. To deal with this issue, we extract our negative instances in a way that tries to minimize such domain differences by (1) selecting negative instances whose words are all contained in our positive instance word dictionary and (2) forcing the text length of non-humorous instances to follow the similar length restriction as humorous examples, i.e. one sentence with an average length of 10-30 words. Here, we assume sentences come from the aforementioned four resources are all non-humorous in nature. Table 1 provides a detailed statistical description to our datasets.

Dataset	#Positive	#Negative
Pun of the Day	2423	2403
16000 One Liners	16000	16002

Table 1: Statistics on Two Datasets

4 Latent Structures behind Humor

In this section, we explore the latent semantic structures behind humor in four aspects: (a) Incongruity; (b) Ambiguity; (c) Interpersonal Effect and (d) Phonetic Style. For each latent structure, a set of features is designed to capture the corresponding indicators of humor.

4.1 Incongruity Structure

"Laughter arises from the view of two or more inconsistent, unsuitable, or incongruous parts or circumstances, considered as united in complex object or assemblage, or as acquiring a sort of mutual relation from the peculiar manner in which the mind takes notice of them" (Lefcourt, 2001). The essence of the laughable is the incongruous, the disconnecting of one idea from another (Paulos, 2008). Humor sometimes relies on a certain type of incongruity, such as opposition or contradiction. For example, the following 'clean desk' and 'cluttered desk drawer' example (Mihalcea and Strapparava, 2005) presents an incongruous/contrast structure, resulting in a comic effect.

A clean desk is a sign of a cluttered desk drawer.

Direct identification of incongruity is hard to achieve, however, it is relatively easier to measure the semantic disconnection in a sentence. Taking advantage of Word2Vec[5], we extract two types of features to evaluate the meaning distance[6] between content word pairs in a sentence (Mikolov et al., 2013):

- Disconnection: the maximum meaning distance of word pairs in a sentence.
- Repetition: the minimum meaning distance of word pairs in a sentence.

4.2 Ambiguity Theory

Ambiguity (Bucaria, 2004), the disambiguation of words with multiple meanings (Bekinschtein et al., 2011), is a crucial component of many humor jokes (Miller and Gurevych, 2015). Humor and ambiguity often come together when a listener expects one meaning, but is forced to use another

[1]Pun of the Day: `http://www.punoftheday.com/` This constructed dataset will be made public.

[2]http://hosted.ap.org/dynamic/fronts/HOME?SITE=AP

[3]https://answers.yahoo.com/

[4]Manually extracted 654 proverbs from Proverb websites

[5]https://code.google.com/p/word2vec/

[6]We take the generic Word2Vec vectors without training new vectors for our specific domain. In addition, vectors associated with senses (Kumar Jauhar et al., 2015) might be alternative advantageous in this task.

meaning. Ambiguity occurs when the words of the surface sentence structure can be grouped in more than one way, thus yielding more than one associated deep structures, as shown in the example below.

Did you hear about the guy whose whole left side was cut off? He's all right now.

The multiple possible meanings of words provide readers with different understandings. To capture the ambiguity contained in a sentence, we utilize the lexical resource WordNet (Fellbaum, 1998) and capture the ambiguity as follows:

- Sense Combination: the sense combination in a sentence computed as follows: we first use a POS tagger (Toutanova et al., 2003) to identify Noun, Verb, Adj, Adv. Then we consider the possible meanings of such words $\{w_1, w_2 \cdots w_k\}$ via WordNet and calculate the sense combinations as $log(\prod_{i=1}^{k} n_{w_i})$. n_{w_i} is the total number of senses of word w_i.
- Sense Farmost: the largest Path Similarity[7] of any word senses in a sentence.
- Sense Closest: the smallest Path Similarity of any word senses in a sentence.

4.3 Interpersonal Effect

Besides humor theories and linguistic style modeling, one important theory behind humor is its social/hostility focus, especially regarding its interpersonal effect on receivers. That is, humor is essentially associated with sentiment (Zhang and Liu, 2014) and subjectivity (Wiebe and Mihalcea, 2006). For example, a sentence is likely to be humorous if it contains some words carrying strong sentiment, such as 'idiot' as follows.

Your village called. They want their Idiot back.

Each word is associated with positive or negative sentiments and such measurements reflect the emotion expressed by the writer. To identify the word-associated sentiment, we use the word association resource in the work by (Wilson et al., 2005), which provides annotations and clues to measure the subjectivity and sentiment associated with words. This enables us to design the following features.

- Negative (Positive) Polarity: the number of occurrences of all Negative (Positive) words.
- Weak (Strong) Subjectivity: the number of occurrences of all Weak (Strong) Subjectivity oriented words in a sentence. It is the linguistic expression of people's opinions, evaluations, beliefs or speculations.

4.4 Phonetic Style

Many humorous texts play with sounds, creating incongruous sounds or words. Some studies (Mihalcea and Strapparava, 2005) have shown that the phonetic properties of humorous sentences are at least as important as their content. Many one-liner jokes contain linguistic phenomena such as alliteration, word repetition and rhyme that produce a comic effect even if the jokes are not necessarily meant to be humorous in content.

What is the difference between a nicely dressed man on a tricycle and a poorly dressed man on a bicycle? A tire.

An alliteration chain refers to two or more words beginning with the same phones. A rhyme chain is defined as the relationship that words end with the same syllable. To extract this phonetic feature, we take advantage of the CMU Pronouncing Dictionary [8] and design four features as follows:

- Alliteration: the number of alliteration chains in a sentence, and the maximum length of alliteration chains.
- Rhyme: the number of rhyme chains and the maximum length of rhyme chains.

5 Humor Anchor Extraction

In addition to humor recognition, identifying anchors, or which words prompt humor in a sentence, is also essential in understanding humor language phenomena. In this section, we first define what humor anchors are and then describe how to extract such semantic units that enable humor in a given sentence.

5.1 Humor Anchor Definition

The semantic units or humor anchors enable humor in a given sentence, and are reflected in the form of sentence words. However, not every single word can be a humor anchor. For example, *I am glad that I know sign language; it is pretty handy*. In this one-liner, words such as 'am' and 'is' are not able to enable humor

[7] Path Similarity: `http://www.nltk.org/howto/wordnet.html`

[8] http://www.speech.cs.cmu.edu/cgi-bin/cmudict

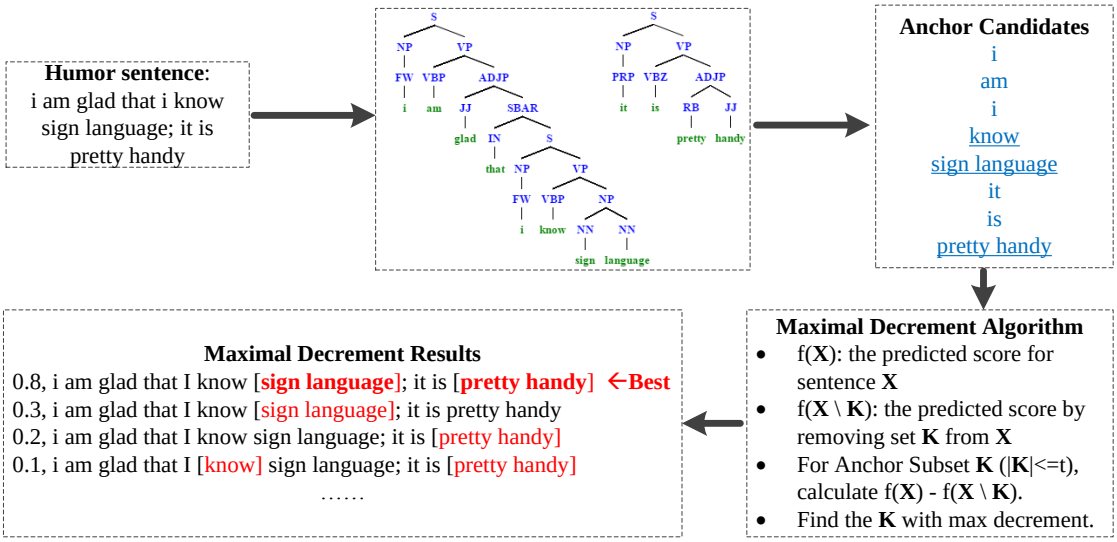


Figure 1: Humor Anchor Extraction Overview. Based on the parsing output of each sentence, we generate its humor anchor candidates. We then apply the Maximal Decrement algorithm to these candidates. The humor anchor subset that gives the maximal decrement is the extracted humor anchors for that sentence.

via themselves. Similarly, 'sign' or 'language' itself are not capable to prompt comic effect. The possible anchors in this example should contain both 'sign language' and 'handy'; it is the combination of these two spans that triggers humor. Therefore, formally defined, a humor anchor is a meaningful, complete, minimal set of word spans in a sentence that potentially enable one of the latent structures of Section 4 to occur. (1) *Meaningful* means humor anchors are meaningful word spans, not meaningless stop words in a sentence; (2) *Completeness* shows that all possible humor anchors should be covered by this anchor set and no individual span in this anchor set is capable enough to enable humor; (3) *Minimal* emphasizes that it is the combination of these anchors together that prompts comic effect; discarding any anchors from this candidate set destroys the humorous effect.

5.2 Anchor Extraction Method

Based on the humor anchor requirements listed above, we scoped humor anchor candidates to words or phrases that belong to the syntactic categories of Noun, Verb, Noun Phrase, Verb Phrase, ADVP or ADJP. Those properties are acquired via a sentence parse tree. To generate anchor candidates, we parsed each sentence and selected words or phrases that satisfy one or more of the latent structure criteria by first extracting the minimal parse subtrees of NP, VP, ADVP and ADJP and then adding remaining Nouns and Verbs into candidate sets.

The above anchor generation process provides us with all possible anchors that might enable humor. It satisfies the *Meaningful* and *Completeness* requirements. To extract a *Minimal* set of anchors, we proposed a simple and effective method of Maximal Decrement. Its basic idea is summarized as follows: Each complete sentence has a predicted humor score, which is computed via a humor recognition classifier trained on all data points. This humor recognizer is not limited to any specific classifiers or features as long as it provides good classification accuracy, which guarantees the generalization ability of our anchor extraction method. We next enumerate a subset of anchors from all potential anchors for this sentence. Then, we recompute the predicted humor score by providing the classifier with features associated with the current sentence, after removing that subset of anchors. Note that our designed humor structural features are all word order free, thereby not distinguishing between complete and incomplete sentences. The subset of humor anchor candidates that provides the maximum decrement of humor predicted scores is then returned as the extracted humor anchor set.

Mathematically, X_i is the word set of sentence i. Let f denote the trained classifier on all data instances. $f(X_i)$ is the predicted humor score

for sentence i before performing any operations. Denote $K_i (K_i \subset X_i)$ as the subset of words that we need to remove from sentence i. The size of K_i should be smaller than a threshold t, $|K_i| \leq t$. $f(X_i/K_i)$ is the recomputed humor score for sentence i after removing K_i. Our Maximal Decrement method tries to maximize the following objective by enumerating all possible K_is. The subset K_i that gives the maximal decrement is returned as our extracted humor anchors for sentence i. The system overview is shown in Figure 1.

$$\arg \min_{|K_i \leq t|} f(X_i) - f(X_i/K_i) \quad (1)$$

6 Experiment

In this section, we validate the performance of different semantic structures we extracted on humor recognition and how the combination of the structures contributes to classification. In addition, both qualitative and quantitative results regarding humor anchor extraction performance are explored.

6.1 Humor Recognition

We formulate humor recognition as a traditional text classification problem, and apply Random Forest to perform 10 fold cross validation on two datasets. Random Forest is an ensemble of decision trees [9] for classification (regression) that constructs a multitude of decision trees at training time and outputs the class that is the mode of the classes output by individual trees. Unlike single decision trees, which are likely to suffer from high variance or high bias, random forests use averaging to find a natural balance between the two extremes.

In addition to the four latent structures behind humor, we also design a set of K Nearest Neighbor (KNN) features that uses the humor classes of the K sentences ($K = 5$) that are the closest to this sentence in terms of meaning distance in the training data. We use several methods to act as baselines for comparison with our classier. **Bag of Words** baseline is used to capture a multiset of words in a sentence that might differentiate humor and non-humor. **Language Model** baseline assigns a humor/nonhumor probability to words in a sentence via probability distributions. **Word2Vec** baseline represents the meaning of sentences via Word2Vec (Mikolov et al., 2013) distributional semantic meaning representation. We implemented an earlier work (Mihalcea and Strapparava, 2005) that exploits stylistic features including alliteration, autonomy and adult slang and ensembles with bag of words representations, denoted as **SaC Ensemble**. It is worth mentioning that our datasets are balanced in terms of positive and negative instances, giving a random classification accuracy of 50%.

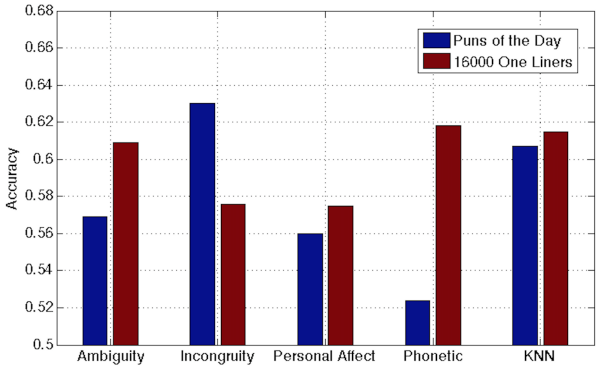


Figure 2: Different Latent Structures' Contribution to Humor Recognition

We first explored how different latent semantic structures affect humor recognition performance and summarize the results in Figure 2. It is evident that *Incongruity* performs the best among all latent semantic structures in the context of Pun of the Day and both *Ambiguity* and *Phonetic* substantially contribute to recognition performance on the 16000 One Liners dataset. The reason behind the differences in performance with *Incongruity* and with *Phonetic* lies in the different nature of the corpus. Most puns are well structured and play with contrasting or incongruous meaning. However, humor sentences in the 16000 One Liners often rely on the reader's awareness of attention-catching sounds (Mihalcea and Strapparava, 2005). This demonstrates that humor characteristics are expressed differently in different contexts and datasets.

We also investigated how the combination of such semantic structures performs compared with our proposed baselines, as shown in Table 2. Here, we denote the combination of four latent structures and KNN features as Human Centric Features (**HCF**). From Table 2, we found that (1) HCF (21 features in total) has a bigger contribution to humor recognition, compared with Bag of Words and Language Model (LM). The

[9] https://www.kaggle.com/wiki/RandomForests

	Pun of the Day				16000 One Liners			
	Accuracy	Precision	Recall	F1	Accuracy	Precision	Recall	F1
HCF	0.705	0.696	0.736	0.715	0.701	0.685	0.746	0.714
Bag of Words	0.632	0.623	0.686	0.650	0.673	0.708	0.662	0.684
Language Model	0.627	0.602	0.762	0.673	0.635	0.645	0.596	0.620
Word2Vec	0.833	0.804	0.880	0.841	0.781	0.767	0.809	0.787
SaC Ensemble	0.763	**0.838**	0.655	0.735	0.662	0.628	0.796	0.701
Word2Vec+HCF	**0.854**	0.834	**0.888**	**0.859**	**0.797**	**0.776**	**0.836**	**0.805**

Table 2: Comparison of Different Methods of Humor Recognition

inadequacy of LM also indicates that we can alleviate the domain differences and capture the real humor. (2) SaC Ensemble is inferior to the combination of Word2Vec and HCF because it does not involve enough latent structures such as Interpersonal Effect and distributional semantics. (3) The combination of Word2Vec and HCF (Word2Vec+HCF) gives the best classification performance because it takes into account both latent structures and semantic word meanings. Such a conclusion is consistent across two datasets. This indicates that our extracted latent semantic structures are effective in capturing humorous meaning.

6.2 Anchor Extraction

Qualitative Evaluation

The above humor recognition classifier provides us with decent accuracy in identifying humor in the text. To better understand which words or semantic units enable humor in sentences, we performed humor anchor extraction as described in Section 5.2. We set the size of the humor anchor set as 3, i.e. $t = 3$. The classifier that is used to predict the humor score is trained on all data instances. Then all predicted humorous instances are collected and input into the humor anchor extraction component. Based on the Maximal Decrement method, a set of humor anchors is extracted for each instance.

Table 3 presents selected extracted humor anchor results, including both successful and unsatisfying extractions. As we can see, extracted humor anchors are quite reasonable in explaining the humor causes or focuses. For example, in the sentence "*I used to be a watchmaker; it is a great job and I made my own hours*", our method selected 'watchmaker', 'made' and 'hours' as humor anchors. It makes sense because each word is necessary and essential to enable humor. Deleting 'watchmaker' will make the combination of 'made' and 'hours' helpless to the comic effect. To sum up, our extracted anchor extraction works fairly well in identifying the focus and meaning of humor language.

Quantitative Evaluation

In addition to the above qualitative exploration, we also conducted quantitative evaluations. For each dataset, we randomly sampled 200 sentences. Then for each sentence, 3 annotators are asked to annotate and label the possible humor anchors. To assess the consistency of the labeling in this context, we introduced an Annotation Agreement Ratio (AAR) measurement as follows:

$$AAR(A, B) = \frac{1}{N_s} \sum_{i=1}^{N_s} \frac{|A_i \cap B_i|}{|A_i \cup B_i|}$$

Here, N_s is the total number of sentences. A_i and B_i are the humor anchor sets of sentence i provided by annotator A and B respectively. The AARs on Pun of the Day and 16000 One Liners datasets are 0.618 and 0.433 respectively, computed by averaging the AAR scores between any two different annotators, which indicate relatively reasonable agreement.

As a further step to validate the effectiveness of our anchor extraction method, we also introduced two baselines. The **Random Extraction** baseline selects humor anchors by sampling words in a sentence randomly. Similarly, **POS Extraction** baseline generates anchors by narrowing down all the words in a sentence to a set of certain POS, e.g. Noun, Verb, Noun Phrase, Verb Phrase, ADVP and ADJP and then sampling words from this set.

To evaluate whether our extracted anchors are consistent with human annotation, we used each annotator's extracted anchor list as the ground truth, and compared with anchor list provided by our method. To identify whether two anchors

Result Category	Representative Sentences
Good	Did you hear about the guy who got **hit** in the head with a can of **soda**? He was lucky it was a **soft drink**.
	I was struggling to figure out how **lightning works** then it **struck** me.
	The one who **invented** the **door knocker** got a **No-bell prize**.
	I used to be a **watchmaker**; it is a great job and I **made** my own **hours**.
Bad	I wanted to lose weight, so I **went** to the **paint store**. I **heard** I could get thinner there.
	I **used** to be a **banker** but I **lost** interest

Table 3: Representative Extracted Humor Anchors. Highlighted parts are the extracted humor anchors in a sentence.

are the same, we introduce two measurements: Exact (EX) Matching and At-Least-One (ALO) Matching. Exact Matching requires the two anchors to be exactly the same. For ALO, two anchors are considered the same if they have at least one word in common. Recall, Precision and F1 Score are act as evaluation metrics. We then average the three annotators' individual scores to get the final extraction performance.

Metrics	Recall	Precision	F1
Pun of the Day Dataset			
MDE EX	0.444	0.446	0.438
POS EX	0.166	0.170	0.165
Random EX	0.121	0.116	0.116
MDE ALO	0.782	0.784	0.756
POS ALO	0.364	0.371	0.360
Random ALO	0.297	0.287	0.285
16000 One Liners Dataset			
MDE EX	0.314	0.281	0.288
POS EX	0.104	0.110	0.104
Random EX	0.087	0.075	0.079
MDE ALO	0.675	0.638	0.616
POS ALO	0.386	0.363	0.356
Random ALO	0.341	0.334	0.319

Table 4: Quantitative Result Comparison of Humor Anchor Extraction

The quantitative evaluation results are summarized in Table 4. Maximal Decrement Extraction is denoted as MDE; POS Extraction is denoted as POS, and Random Extraction is denoted as Random. We report both ALO and EX results for MED, POS and Random. From Table 4, we found that MDE performs quite well under the measurement of human annotation in terms of both ALO and EX settings. This again validates our assumption towards humor anchors and the effectiveness of our anchor extraction method.

6.3 Discussion

The above two subsections described the performance of both humor recognition and humor anchor extraction tasks. In terms of humor recognition, incongruity, ambiguity, personal affect and phonetic style are taken into consideration to assist the identification of humorous language. We focus on discovering generalized structures behind humor, and did not take into account sexual oriented words such as adult slang in modeling humorous language. Based on our results, these four latent structures are effective in capturing humor characteristics and such characteristics are expressed to different extents in different contexts. Note that we can apply any classification methods with our humor latent structures. Once such structures help us acquire high recognition accuracy, we can perform the generalized Maximal Decrement extraction method to identify anchors in humorous text.

Both humor recognition and humor anchor extraction suffer from several common issues. (1) **Phrase Meaning**: For example, a humorous sentence "*How does the earth get clean? It takes a meteor shower*" is predicted as non-humorous, because the recognizer does not fully understand the meaning of 'meteor shower', let alone the comic effect caused by 'earth', 'clean' and 'meteor shower'. For the unsatisfying example in Table 3 "*I used to be a banker but I lost interest*", anchor extraction would work better if it recognizes 'lost interest' correctly as a basic semantic unit. (2) **External Knowledge**: For jokes that involve idioms or social phenomena, or need some external knowledge such as "*Veni, Vidi, Visa: I came, I saw, I did a little shopping*", both humor recognition and anchor extraction fail because a broader and implicit comparison of this sentence and its origin ("*Veni, Vidi, Vici: I came, I*

saw, I conquered... ") is hard to be captured from a sentence. (3) **Humor Categorization**: Moreover, a fine granularity categorization of humor might aid in understanding humorous language, because humor has different types of manifestations, such as irony, sarcasm, creativity, insult and wordplay. Therefore, more sophisticated techniques in modeling phrase meaning, external knowledge, humor types, etc., are needed to better expose and define humor for automatic recognition and extraction.

7 Conclusion

In this work, we focus on understanding humorous language through two subtasks: humor recognition and humor anchor extraction. For this purpose, we first designed four semantic structures behind humor. Based on the designed sets of features associated with each structure, we constructed different computational classifiers to recognize humor. Then we proposed a simple and effective Maximal Decrement method to automatically extract anchors that enable humor in a sentence. Experimental results conducted on two datasets demonstrate the effectiveness of our proposed latent structures. The performances of humor recognition and anchor extraction are superior compared to several baselines. In the future, we would like to step further into the discovery of humor characteristics and apply our findings to the process of humor generation.

Acknowledgement

The authors would like to thank Li Zhou, Anna Kasunic, the anonymous reviewers, our annotators and all colleagues who have contributed their valuable comments and suggestions.

References

Salvatore Attardo. 1994. *Linguistic theories of humor*, volume 1. Walter de Gruyter.

Tristan A Bekinschtein, Matthew H Davis, Jennifer M Rodd, and Adrian M Owen. 2011. Why clowns taste funny: the relationship between humor and semantic ambiguity. *The Journal of Neuroscience*, 31(26):9665–9671.

Kim Binsted and Graeme Ritchie. 1997. Computational rules for generating punning riddles. *Humor: International Journal of Humor Research*.

Chiara Bucaria. 2004. Lexical and syntactic ambiguity as a source of humor: The case of newspaper headlines. *Humor*, 17(3):279–310.

Christiane Fellbaum. 1998. *WordNet*. Wiley Online Library.

Bryan Anthony Hong and Ethel Ong. 2009. Automatically extracting word relationships as templates for pun generation. In *Proceedings of the Workshop on Computational Approaches to Linguistic Creativity*, CALC '09, pages 24–31, Stroudsburg, PA, USA. Association for Computational Linguistics.

Chloe Kiddon and Yuriy Brun. 2011. That's what she said: double entendre identification. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 89–94. Association for Computational Linguistics.

Sujay Kumar Jauhar, Chris Dyer, and Eduard Hovy. 2015. Ontologically grounded multi-sense representation learning for semantic vector space models. In *The 2015 Conference of the North American Chapter of the Association for Computational Linguistics*.

Igor Labutov and Hod Lipson. 2012. Humor as circuits in semantic networks. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pages 150–155. Association for Computational Linguistics.

Herbert M Lefcourt. 2001. *Humor: The psychology of living buoyantly*. Springer Science & Business Media.

Rada Mihalcea and Carlo Strapparava. 2005. Making computers laugh: Investigations in automatic humor recognition. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 531–538. Association for Computational Linguistics.

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

Tristan Miller and Iryna Gurevych. 2015. Automatic disambiguation of english puns. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 719–729, Beijing, China, July. Association for Computational Linguistics.

Gözde Ozbal and Carlo Strapparava. 2012. Computational humour for creative naming. *Computational Humor 2012*, page 15.

John Allen Paulos. 2008. *Mathematics and humor: A study of the logic of humor*. University of Chicago Press.

Amruta Purandare and Diane Litman. 2006. Humor: Prosody analysis and automatic recognition for f*r*i*e*n*d*s*. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, EMNLP '06, pages 208–215, Stroudsburg, PA, USA. Association for Computational Linguistics.

Victor Raskin. 1985. *Semantic mechanisms of humor*, volume 24. Springer.

Yishay Raz. 2012. Automatic humor classification on twitter. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Student Research Workshop*, pages 66–70. Association for Computational Linguistics.

Graeme Ritchie. 2005. Computational mechanisms for pun generation. In *Proceedings of the 10th European Natural Language Generation Workshop*, pages 125–132. Citeseer.

Oliviero Stock and Carlo Strapparava. 2003. Getting serious about the development of computational humor. In *IJCAI*, volume 3, pages 59–64.

Oliviero Stock and Carlo Strapparava. 2005. Hahacronym: A computational humor system. In *Proceedings of the ACL 2005 on Interactive poster and demonstration sessions*, pages 113–116. Association for Computational Linguistics.

J Taylor and L Mazlack. 2004. Computationally recognizing wordplay in jokes. *Proceedings of CogSci 2004*.

Julia M Taylor. 2009. Computational detection of humor: A dream or a nightmare? the ontological semantics approach. In *Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology-Volume 03*, pages 429–432. IEEE Computer Society.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, pages 173–180, Stroudsburg, PA, USA. Association for Computational Linguistics.

Janyce Wiebe and Rada Mihalcea. 2006. Word sense and subjectivity. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 1065–1072. Association for Computational Linguistics.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 347–354, Vancouver, British Columbia, Canada, October. Association for Computational Linguistics.

Renxian Zhang and Naishi Liu. 2014. Recognizing humor on twitter. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 889–898. ACM.

Topic Identification and Discovery on Text and Speech

Chandler May, Francis Ferraro, Alan McCree, Jonathan Wintrode,
Daniel Garcia-Romero, and **Benjamin Van Durme**
Human Language Technology Center of Excellence
Johns Hopkins University
cjmay@jhu.edu, ferraro@cs.jhu.edu, alan.mccree@jhu.edu, jcwintr@cs.jhu.edu,
dgromero@jhu.edu, vandurme@cs.jhu.edu

Abstract

We compare the multinomial i-vector framework from the speech community with LDA, SAGE, and LSA as feature learners for topic ID on multinomial speech and text data. We also compare the learned representations in their ability to discover topics, quantified by distributional similarity to gold-standard topics and by human interpretability. We find that topic ID and topic discovery are competing objectives. We argue that LSA and i-vectors should be more widely considered by the text processing community as pre-processing steps for downstream tasks, and also speculate about speech processing tasks that could benefit from more interpretable representations like SAGE.

1 Introduction

The text processing and speech processing research communities have similar problems and goals, but the technical approaches in these two communities develop largely independently. In this paper we compare dimensionality reduction techniques on multinomial language data from the text and speech communities. We consider a multinomial formulation of the i-vector model (hereafter "mi-vector" model) from the speech community (Soufifar et al., 2011), the sparse additive generative (SAGE) (Eisenstein et al., 2011) and latent Dirichlet allocation (LDA) (Blei et al., 2003b) topic models from the text community, and latent semantic analysis (LSA) (Deerwester et al., 1990). Both the mi-vector model and the SAGE topic model represent a multinomial parameter vector as the softmax of a sum of vectors, one of which is a background vector representing overall word usage in the corpus, and so we might expect mi-vectors and SAGE to produce similar results on real-world data. We evaluate these two recent models and two more conventional models, LDA and LSA (a term describing a class of methods based on the singular value decomposition, or SVD, which is used broadly in both research communities). We assess the similarity of mi-vectors and SAGE and expose the strengths and weaknesses of all four learned representations by evaluating them on the supervised task of topic identification (topic ID), depicted in Figure 1. We also evaluate the representations on the unsupervised, less easily-measurable task of topic discovery. As a proxy for controlled human annotations, we quantify topic discovery performance by distributional similarity to gold-standard topics.

We use the bag-of-words multinomial representation of text data, i.e., each document is represented by a vector of counts over the word vocabulary. For speech data, we use a modern automatic speech recognition (ASR) system to produce frame-wise triphone state cluster posteriors and we take the sum of these posteriors across all frames in a document to obtain a document-level vector of triphone state cluster soft counts. Modern topic ID systems for speech use ASR output instead of a lower-resource representation like these soft counts to improve performance (Hazen et al., 2007). ASR word counts are high-resource and can be viewed as a noisy version of word counts from text. We wish to assess the relative performance of our learned representations, not the quality of the data pre-processing scheme, and we desire to strengthen our results by evaluating performance on two distinct views of a corpus. Hence we break from convention and use triphone state cluster soft counts as speech data.

While previous work has juxtaposed the mi-vector model against LDA (Chen et al., 2014; Morchid et al., 2014), the current study is the first to provide *cross-community* evaluations of mi-vectors and a contemporaneous model from

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 2377–2387,
Lisbon, Portugal, 17-21 September 2015.

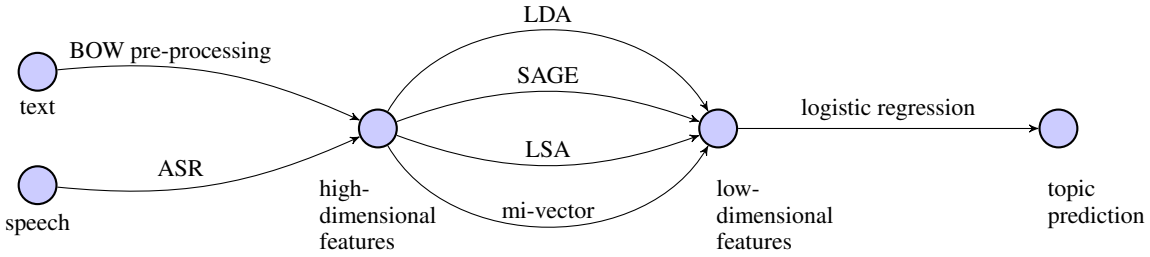


Figure 1: Depiction of the topic ID pipeline. Raw text or speech data is processed into multinomial counts, which are then transformed into a learned representation, and a classifier then predicts the topic of each document based on its representation.

the text community on both text and speech data. This study is also novel in its direct application of the mi-vector model to topic ID and topic discovery, two separate tasks with different motivations and preferring different types of models, and in its use of low-resource triphone state cluster soft counts as speech data for topic ID. The low-resource setting reflects constraints often faced in real-world applications, and we report topic ID performance under limited supervision to better illuminate the practical strengths and weaknesses of the learned representations. Finally, we believe that the centralized comparison herein of several prominent learned representations on two complementary tasks on both text and speech will provide a useful point of reference for future research.

2 Background

Previous work has compared and composed the mi-vector model with older dimensionality reduction techniques, including LDA. Chen et al. (2014) compared a mi-vector language model against LDA and other models on the task of spoken document retrieval, and found the mi-vector model to significantly outperform the other models on words, but not on subwords (syllable pairs), derived from ASR. The syllable pairs are similar in granularity to the triphone state clusters used as multinomial speech data in the current work.

Morchid et al. (2014) improved conversation theme identification by employing LDA and a *Gaussian* i-vector model in a pipeline. They learn LDA models of varying dimensions (numbers of topics) on ASR output and use them to generate a suite of feature vectors. The feature vector for each document-dimension pair is created by marginalizing over topics according to the document's inferred topic proportions. A Gaussian i-vector model is then learned on those feature vectors; the i-vectors are normalized and used to identify document themes via the Bayes decision rule.

Note that we have fundamentally different approaches, goals and methodology from that of Morchid et al. (2014). First, in an effort to provide a scientific comparison of independently created models, we use *multinomial* i-vectors, whereas Morchid et al., focusing on a particular task setting, used traditional *Gaussian* i-vectors. Similarly, while we treat multiple types of topic models as goals in their own rights, directly comparing SAGE and LDA, Morchid et al. use LDA as a pre-processing step to Gaussian i-vectors. Second, we use triphone state cluster soft counts instead of ASR word counts, hence our representation of speech data is significantly lower-resource. Third, we also evaluate performance on text data, and where Morchid et al. limit their vocabulary (from ASR) to 166 task-specific words, we use all 26,606 words present in our training data.

3 Input Representations

Our data is drawn from Part 1 of the Fisher English speech corpus (Cieri et al., 2004c), which contains audio recordings (Cieri et al., 2004a) and manual transcriptions (Cieri et al., 2004b) of telephone conversations. Specifically, we use the topic ID training and evaluation test subsets defined in prior work (Hazen et al., 2007). In each conversation in these subsets of the data, two study participants are prompted to speak on one of a predefined set of forty topics. There are 1374 training conversations and 686 test conversations. We represent each conversation by two documents, one for each side (speaker), resulting in a training set of 2748 documents and a test set of 1372 documents. The deep neural network (DNN) used to infer the triphone state cluster posteriors forming the basis of our speech data was trained on Parts 1 and 2 of the Fisher English speech corpus (Cieri et al., 2004a; Cieri et al., 2005); see the supplement for further

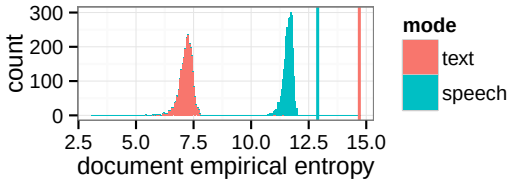


Figure 2: Distributions of the empirical entropy (in bits) of documents under the two multinomial views of our corpus. The vertical lines are the respective upper bounds (entropy of the uniform distributions). The distribution of the entropy of the text documents has median 7.2, over seven bits away from the upper bound of 14.7, thus the text representation is approximately sparse. The speech distribution has median 11.6, within two bits of the upper bound of 12.9, thus the speech representation is nearly uniform.

details about our dataset and ASR system.

To quantify the sparsity of the raw text (word count) and speech (triphone state cluster soft count) representations, we consider the representation density (number of non-zero entries) on our training set. The text representation is sparse, with median density 292 and maximum 500 (out of 26,606 dimensions); the speech representation is dense, with median density 7586 and maximum 7591 (out of 7591 dimensions).

To assess approximate sparsity, we plot histograms of the entropy of the normalized multinomial views of our training set in Figure 2. The median entropy for speech is less than two bits away from the uniform entropy, so the speech data is neither sparse nor approximately sparse.

Finally, we note that topic occurrence in the Fisher English training set is unbalanced, with quartiles (including minimum and maximum) of 6, 18.75, 29.5, 50.25, and 87.

4 Learned Representations

We consider four main dimensionality reduction models: the mi-vector model from the speech community, the SAGE and LDA topic models from the text community, and LSA. The learned representations we consider explain which words appear in a document d via a latent, lower-dimensional representation $\boldsymbol{\theta}^{(d)}$. All representations operate under a bag-of-words assumption. To compare mi-vectors, topic models and LSA, we find it useful to formulate each learned representation as operating on different "areas" or "contexts" a of a document; such a formulation does not negate the fundamental bag-of-words assumption. The four models represent the words that appear in an area a—either the entire document or each token—via multinomial-style parameters $\boldsymbol{\phi}^{(a)}$.[1,2] Each model consists of K *components* (e.g., a K-dimensional affine subspace), and shared parameters $H_{k,v}$ prescribe the amount of weight each component k places on each vocabulary word v. The models construct $\boldsymbol{\phi}^{(a)}$ by combining $\boldsymbol{H}$ and $\boldsymbol{\theta}^{(d)}$; often empirical word statistics $\mathbf{m}$ are also used to stabilize the representations.

4.1 LSA

LSA (Deerwester et al., 1990) factorizes a term-document matrix by truncated SVD, learning the projection of the data onto a linear subspace of fixed rank such that the approximation error of the reconstructed term-document matrix (as measured by the Frobenius norm) is minimized. In the basic version of LSA, SVD is applied to the raw term counts, giving the low-dimensional representation

$$\boldsymbol{\phi}^{(d)} = \boldsymbol{H}\boldsymbol{\theta}^{(d)}, \tag{1}$$

where $\boldsymbol{\phi}^{(d)}$ is the vector of observed multinomial counts in document d, $\boldsymbol{H}$ is the matrix of left singular vectors of the term-document count matrix, and $\boldsymbol{\theta}^{(d)}$ is the inferred representation of $\boldsymbol{\phi}^{(d)}$. In practice, LSA is often applied instead to the term-document matrix weighted by term frequency–inverse document frequency (tf-idf) in order to normalize terms by importance. We can also apply further pre-processing steps, such as term-wise centering by subtracting the column-wise mean $\mathbf{m}$ of the data, in which case LSA finds an affine subspace that approximates the data.

4.2 Mi-vector Model

The original acoustic i-vector model represents continuous, high-dimensional ASR system state (namely, Gaussian mixture model supervectors) in an affine subspace (Dehak et al., 2011). Prior work has found this dense, low-dimensional representation to be effective for a number of tasks, including language recognition (Martínez et al., 2011) and speaker recognition (Dehak et al., 2011; Garcia-Romero and Espy-Wilson, 2011).

Recently the i-vector model was augmented for multinomial observations (Soufifar et al., 2011)

[1] Other efforts have modeled documents with intermediate granularity, e.g., sentence-level (Titov and McDonald, 2008) or entity-level (Newman et al., 2006) granularity.

[2] For brevity, we use the multinomial distribution and its parameter interchangeably throughout.

and applied accordingly to language recognition (Soufifar et al., 2011; McCree and Garcia-Romero, 2015), speaker recognition (Kockmann et al., 2010), and spoken document retrieval (Chen et al., 2014). In this version of the i-vector model the observations are draws from a multinomial and the (unnormalized) natural parameters of that distribution are represented in an affine subspace:

$$\begin{aligned} \boldsymbol{\phi}^{(d)} &= \text{softmax}\left(\mathbf{m} + \boldsymbol{H}\boldsymbol{\theta}^{(d)}\right) \\ \boldsymbol{\theta}^{(d)} &\sim \mathcal{N}(0, \mathbf{I}). \end{aligned} \tag{2}$$

We call this multinomial version of the i-vector model the *mi-vector* model. The latent variable $\boldsymbol{\theta}^{(d)}$ is the *multinomial i-vector*, or *mi-vector*. $\boldsymbol{H}$ is an unconstrained linear transformation. The bias term $\mathbf{m}$ is computed as the log of the l_1-normalized background word count vector. The Gaussian prior on the mi-vector $\boldsymbol{\theta}^{(d)}$ is effectively an l_2 regularizer; mi-vectors are neither non-negative nor sparse in general.

Unlike many Bayesian topic models, word occurrences in the mi-vector model are i.i.d. draws from a document-level multinomial $\boldsymbol{\phi}^{(d)}$; as in LSA, each latent component contributes equally to each word within a given document. Specifically, in the mi-vector model, the natural parameter vector of the multinomial for *all* words in a given document is determined by an additive offset from a background parameter vector.

4.3 Bayesian Discrete Topic Models

Bayesian topic models explain word occurrences via K latent components $\boldsymbol{H}_k$ (topics) each drawn from some prior distribution G. Unlike mi-vectors and LSA, multinomial topic models are admixture models: each token n is drawn from a particular distribution $\boldsymbol{H}_k$. Latent token assignment variables $z_n^{(d)}$, taking integral values between 1 and K, dictate the token's topic choice. A document d controls how often each topic is chosen via the K-dimension multinomial distribution $\boldsymbol{\theta}^{(d)}$. In the parametric settings we consider, Dirichlet priors are often placed on $\boldsymbol{\theta}^{(d)}$, allowing experimentation with the topic representation $\boldsymbol{H}$.[3] A mapping $Q(\boldsymbol{H}_k)$, possibly the identity, ensures $\boldsymbol{\phi}^{(d,n)}$ are probability vectors. A general formulation is then

$$\begin{aligned} \boldsymbol{\phi}^{(d,n)} &= Q\left(\boldsymbol{H}_{z_n^{(d)}}\right) \\ \boldsymbol{H}_k &\sim G\left(\boldsymbol{\eta}\right) \\ z_n^{(d)} &\sim \text{Discrete}\left(\boldsymbol{\theta}^{(d)}\right) \\ \boldsymbol{\theta}^{(d)} &\sim \text{Dirichlet}\left(\boldsymbol{\alpha}\right). \end{aligned} \tag{3}$$

The hyperparameters $\boldsymbol{\alpha}$ and $\boldsymbol{\eta}$ dictate the informativeness of the priors over $\boldsymbol{H}_k$ and $\boldsymbol{\theta}^{(d)}$: often (empirically optimized) symmetric hyperparameters are employed, resulting in a form of Laplace smoothing during topic estimation. In the current work, we follow this strategy, noting that there have been concerted efforts to encode domain or expert knowledge via the hyperparameters (Gormley et al., 2012; Paul and Dredze, 2015).

SAGE Topic Model The Sparse Additive Generative (SAGE) model (Eisenstein et al., 2011) is a generative Bayesian modeling framework in which $\boldsymbol{\phi}^{(d,n)}$ are formed by summing a background vector and one or more sparse vectors generated from appropriate priors. The additive components can reflect the contributions of documents, aspects, topics, or other factors chosen by the modeler. A basic SAGE topic model sets $\boldsymbol{\phi}^{(d,n)} = \text{softmax}\left(\mathbf{m} + \boldsymbol{H}_{z_n^{(d)}}\right)$, and draws $\boldsymbol{H}_k$ from some sparsity-inducing distribution G, e.g., the Laplace distribution. As $\mathbf{m}$ is a shared background frequency vector, $\boldsymbol{H}_k$ is the learned residual frequency vector of topic k.

Replacing the topic assignment in SAGE by its conditional expectation gives

$$\begin{aligned} \tilde{\boldsymbol{\phi}}^{(d,n)} &= \text{softmax}\left(\mathbf{m} + \mathbb{E}_{z_n^{(d)}}\left[\boldsymbol{H}_{z_n^{(d)}} \middle| \boldsymbol{\theta}^{(d)}, \boldsymbol{H}\right]\right) \\ &= \text{softmax}\left(\mathbf{m} + \boldsymbol{H}\boldsymbol{\theta}^{(d)}\right). \end{aligned} \tag{4}$$

This modification of the SAGE topic model is the same as the mi-vector model but with different regularization on the representation vector $\boldsymbol{\theta}^{(d)}$ and l_1 regularization on the basis vectors $\boldsymbol{H}_k$. This "marginal SAGE" model could be useful in future work: the marginalization may mitigate the problem of topic-switching, yielding a more identifiable (but perhaps less interpretable) model and lending to downstream tasks such as topic ID.

LDA Latent Dirichlet Allocation (LDA) (Blei et al., 2003b) is a generative Bayesian topic model similar to SAGE, but in which each topic is drawn

[3] There have been many efforts to provide or induce latent structure among the topics (Blei et al., 2003a; Li and McCallum, 2006; Wallach et al., 2009; Paul and Girju, 2010), but most models ground out to Dirichlet and discrete random variables.

from a Dirichlet prior G rather than a sparsity-inducing distribution. LDA does not explicitly account for the background distribution; to account for this, it is common practice to threshold the vocabulary *a priori* to remove very common and very rare words (though in our experiments, we do not do this). Therefore, $\boldsymbol{\phi}^{(d,n)}$ is exactly $\boldsymbol{H}_{z_n^{(d)}}$, and $\boldsymbol{H}_k \sim \text{Dirichlet}\,(\boldsymbol{\eta})$.

5 Experiments

We compare these four models of learned representations empirically on two distinct tasks, topic ID and topic discovery. The essential implementation details of the models are as follows; further details are provided in the supplement. We learn the **mi-vector** model in a maximum a posteriori framework as in McCree and Garcia-Romero (2015). Our own C++ implementation of **SAGE**, available online,[4] uses approximate mean-field variational inference, as in Eisenstein et al. (2011). We learn the **LDA** model using Gibbs sampling, implemented in MALLET (McCallum, 2002).[5] We perform **LSA** using centered tf-idf–weighted word counts and centered l_2-normalized triphone state cluster soft counts. We implement tf-idf by scaling the raw term count by the log inverse document frequency. We apply l_2 normalization rather than tf-idf weighting to the speech data because it is dense and tf-idf is thus inappropriate. On both text and speech, mean-centering is performed *after* the respective normalization, as this pre-processing recipe performed best of all the variants we tried.[6]

For each of the four models, the low-dimensional real vector $\boldsymbol{\theta}^{(d)}$ represents a given document d in our experiments. We also consider two high-dimensional baseline representations: **raw** (soft) counts on both the text and speech data, and, only on the text data, **tf-idf**–weighted word counts. These tf-idf weights constitute a high-dimensional *learned* representation.

5.1 Topic ID

In our first topic ID experiment we evaluate topic ID error on raw multinomial views of the data. To our knowledge, we are the first to adopt a multinomial view of triphone state clusters and apply it to topic ID. In subsequent experiments we explore the interaction of representation dimension with each model and dataset, and evaluate relative performance when the classifier is only given a fraction of the available data for training. This latter configuration is the most interesting, as it reflects the cost of obtaining supervised data in practice.

Given feature vectors for some representation of the documents in a corpus, topic ID is performed in a one-versus-all framework. We use logistic regression as the per-class binary classifier, implemented using LIBLINEAR (Fan et al., 2008). Results were similar when logistic regression was replaced by support vector machines. All document representations are length-normalized (divided by their l_2 norm) before they are input to the classifier. Performance is measured by topic ID error, the error of multi-class prediction where the class predicted for each document is that of the per-class classifier that gave it the highest weight. Baseline performance on the test set (where the baseline classifier chooses the most prevalent topic in the training set for all test examples) is 96.2% error. Note that this error rate differs from the uniform-at-random classification error rate of 97.5% because of the uneven distribution of topics.

Document Construction Prior work (Hazen et al., 2007; Wintrode and Khudanpur, 2014) treated whole conversations as documents in addition to separating each conversation into its two sides. We perform a small topic ID experiment in this configuration to probe the impact of this design choice. Ten-fold cross-validation (CV) is used to tune the logistic regression regularizers. On the test set, the classifier achieves topic ID error of 12.4% and 15.6% for whole-conversation and individual-side text data, respectively, and 20.1% and 29.5% for whole-conversation and individual-side speech data, respectively. These results correspond roughly to results listed in Table 3 of Hazen et al. (2007), specifically, the topic ID error of 8.2% and 12.4% for whole-conversation and individual-side transcriptions, respectively, and 22.9% and 35.3% for whole-conversation and individual-side triphones derived from ASR lattices, respectively (Hazen et al., 2007). However, we use logistic regression without feature selection instead of Naïve Bayes with feature selection, and we apply our classifier to triphone state cluster soft counts inferred by a DNN instead of triphone

[4] https://github.com/fmof/sagepp

[5] For Gibbs sampling, fractional counts are truncated.

[6] Results for other versions of LSA are provided in the supplement. We did not present the conventional, uncentered tf-idf weighting scheme here because although it performs best in topic ID, it yields extremely variable V-measure.

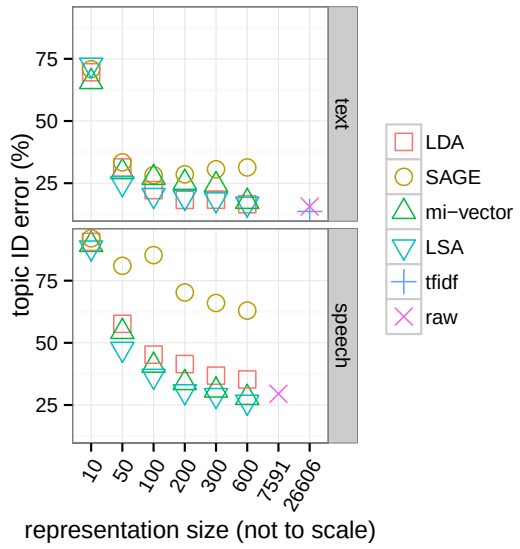


Figure 3: Topic ID error (%) on the test set for raw and tf-idf representations and lower-dimensional learned representations at dimensions $K \in \{10, 50, 100, 200, 300, 600\}$. We see many of the learned representations approach the error rate of the raw representation, but at much lower dimensionality.

counts from ASR lattices. We believe that the discrepancies in performance with respect to prior work are due to these differences in experimental configuration. Our results and those of prior work show that using whole-conversation documents instead of individual-side documents make the topic ID task easier. As a result, we expect that differences in performance between the different learned representations will be more clearly pronounced on individual conversation sides and we restrict the rest of our study to that setting.

Dimensionality Study We perform topic ID on learned representations at dimensions $K \in \{10, 50, 100, 200, 300, 600\}$ on individual conversation sides, using ten-fold cross-validation to tune the logistic regression regularizers. Figure 3 gives topic ID error results on the test set, varying K. (Selected values are listed in Table 1.) In both datasets, as the dimension K increases, topic ID error decreases, approaching (approximately) the raw baseline. On text, tf-idf performs slightly better than the raw representation. LSA is marginally the best-performing lower-dimensional learned representation; LDA and mi-vectors perform well at some representation sizes, depending on the data source, but their performance is less consistent. SAGE performs poorly overall.

view	model	dimension	error
text	LDA	600	16.5
text	SAGE	600	31.3
text	mi-vector	600	17.6
text	LSA	600	16.7
text	tf-idf	26,606	13.6
text	raw	26,606	15.6
speech	LDA	600	35.3
speech	SAGE	600	63.0
speech	mi-vector	600	27.9
speech	LSA	600	26.2
speech	raw	7591	29.5

Table 1: Selected topic ID error (%) values from Figure 3.

Limited Data Study The raw text and speech representations (multinomial observations) are very high-dimensional, and the classifier is likely to overfit to specific components (words or triphone state clusters) in these representations. To measure this effect and attempt to separate the predictive power of logistic regression from the quality of the learned representations in our analysis, we experiment with reducing the number of *labeled* training examples the *classifier* can use; we still learn representations on the full (unlabeled) training set. This experiment represents the limited-supervision setting in which supervised data is costly to obtain but unlabeled data abounds.

We run this experiment twice, using $\ell = 2$ and $\ell = 6$ labeled examples per topic, for a total of 80 and 240 classifier training examples, respectively. Ten-fold cross-validation is used to fit the regularizer; per-class loss coefficients are set according to the class prior in the original training set in order to counteract the artificial balancing of the classes in the limited-supervision dataset. We report cross-validation estimates of the topic ID error on the training set for $K = 10$ (Figure 4), $K = 100$ (Figure 5), and $K = 600$ (Figure 6). For $K = 100$ and $K = 600$, LSA dominates in the limited-supervision setting. Mi-vectors perform as well as or better than other low-dimensional learned representations at $K = 10$, and exhibit mixed performance for larger K. SAGE performs poorly overall.[7] LDA performs significantly bet-

[7] We believe that approximately sparse posterior $\boldsymbol{\theta}^{(d)}$ values result in a kind of topic switching, contributing to the poor performance of SAGE. To examine this we "tested on train" and analyzed the top topics inferred for each document: while the highest-weighted topic tended to be consistent, SAGE infers approximately sparse $\boldsymbol{\theta}^{(d)}$ with large variation in the next four highest-weighted topics (the remaining topics are assigned trace mass). Second, a phenomenon known as conversation drift, explained in Section 3 of the supplement, is so pronounced in Fisher that the first 25% percent of words of each conversation side are nearly as predictive as the entire

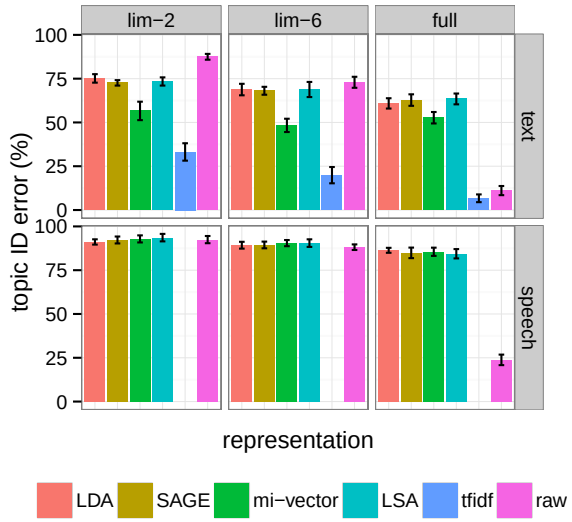


Figure 4: CV topic ID error (%) for raw and tf-idf representations and lower-dimensional learned representations of size $K = 10$. Error bars denote plus and minus one standard deviation according to the CV empirical distribution. We see underparametrized mi-vectors excel at compressing the topic label information for text, particularly in the limited-supervision settings.

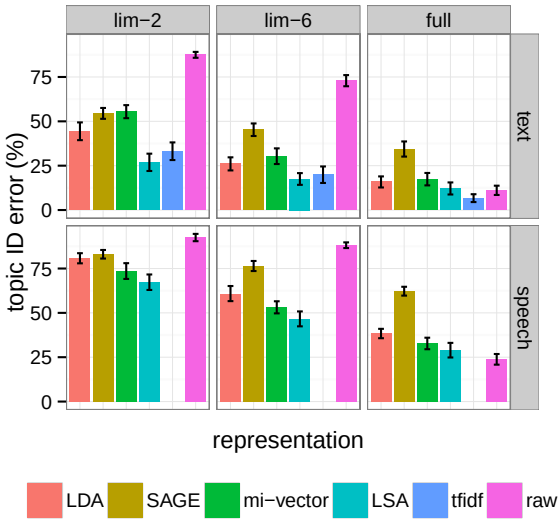


Figure 5: CV topic ID error (%) for raw and tf-idf representations and lower-dimensional learned representations of size $K = 100$. Error bars denote plus and minus one standard deviation according to the CV empirical distribution.

view	model	dimension	V-measure
text	LDA	600	0.517
text	SAGE	600	0.663
text	mi-vector	600	0.507
text	LSA	600	0.525
text	tf-idf	26,606	0.626
text	raw	26,606	0.134
speech	LDA	600	0.358
speech	SAGE	600	0.511
speech	mi-vector	600	0.468
speech	LSA	600	0.190
speech	raw	7591	0.132

Table 2: Selected V-measure values from Figure 7.

ter than SAGE, but not as well as mi-vectors. Finally, tf-idf–weighted word counts perform very well on text, often achieving the best performance of all representations, even under limited supervision (but at the same dimension as the raw data).

5.2 Topic Discovery

To quantitatively assess representations' potential for topic discovery we compute their V-measure against the gold-standard labels. V-measure is an unsupervised measure of similarity between two partitions (Rosenberg and Hirschberg, 2007) and is equivalent to the mutual information normalized by the sum of the entropy (Becker, 2011).

For all representations, we compute V-measure between a partition induced by that representation and the gold-standard topic labels on the test set. A partition is induced on a representation by assigning each document d to the cluster indexed by the coordinate of $\boldsymbol{\theta}^{(d)}$ with highest value (the argmax). Results of this analysis are displayed in Figure 7. (Selected values are listed in Table 2.) On the text data, SAGE dominates the lower-dimensional representations, LSA is next best overall, and LDA and mi-vectors exhibit relatively low performance; the high-dimensional tf-idf weights are surpassed by SAGE for $K > 10$ but beat other representations by a significant margin. On speech, SAGE is best overall, mi-vectors exhibit similar but generally lower performance, LDA performs worse, and LSA is worst.

We also measure the topic discovery potential of the mi-vector and SAGE representations more directly. First, we provide a manual inspection of the learned topics: in Table 3 we show the top-five word lists for five random topics from the 600-dimensional mi-vector and SAGE models (respectively) learned on the text data. In both models, the top five words in a topic are selected according to the five largest positive values in the corresponding vector $\boldsymbol{H}_k$. Qualitatively, the SAGE topics are considerably more interpretable than the mi-vector topics: the SAGE topics represent issues of censorship, foreign relations, coffee fran-

document (Wintrode, 2013). All representations must contend with this drift, but $\boldsymbol{\theta}^{(d)}$ sparsity may make SAGE particularly susceptible. These two issues may make the classification we use much less robust.

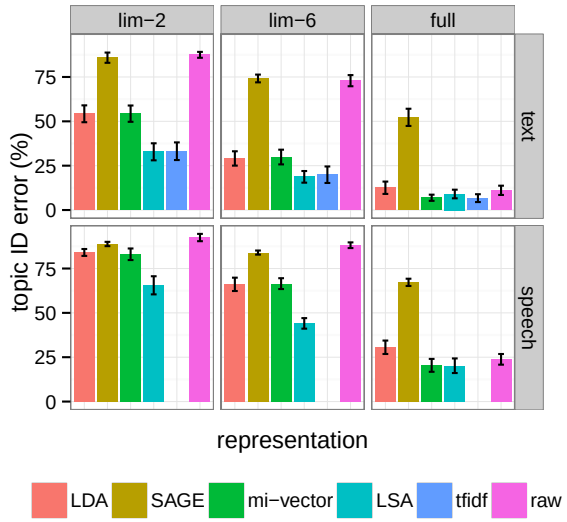


Figure 6: CV topic ID error (%) for raw and tf-idf representations and lower-dimensional learned representations of size $K = 600$. Error bars denote plus and minus one standard deviation according to the CV empirical distribution.

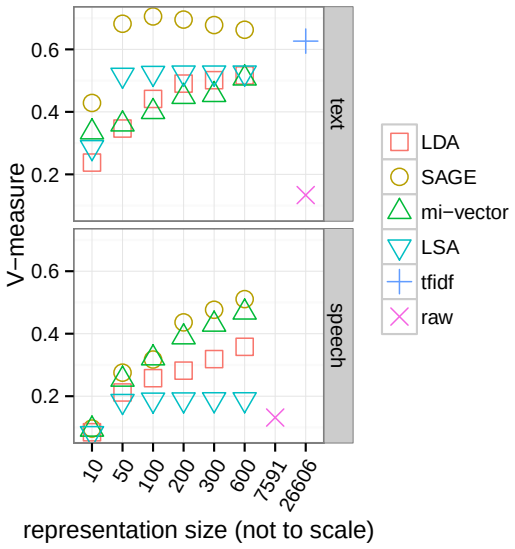


Figure 7: V-measure on the Fisher English text and speech data, respectively, for raw and tf-idf representations and lower-dimensional learned representations at selected dimensions. As in topic ID, we see underparametrized mi-vectors perform well on the text data.

chises, welfare, and professional basketball, while the mi-vector topics are less succinctly characterizable and more polluted by uninformative words.

We complement this qualitative analysis with Mimno et al. (2011)'s intrinsic coherence measure, a standard quantitative method. This scoring function, which correlates well with human quality judgments, averages estimates of the conditional log-likelihoods of each topic's M highest-weighted words across all topics. Using $K = 600$ models on text as before and picking $M = 20$, we compute mi-vector coherence as -453.34 and SAGE coherence (averaged over three runs) as -407.52, indicating that SAGE is more amenable to topic discovery and human interaction.

mi-vector
you've, florida, each, a-, bit
hours, never, couldn't, check, communicate
pregnant, water, lifestyle, awful, called
forgot, ran, social, topics, unique
tough, way, let's, fifties, hand

SAGE
censor, books, censorship, neat, agree
sanctions, siblings, democratic, rely, u._n.
starbucks, franchise, coffee, franchising, studio
welfare, wage, minimum, cents, tips
team, role, professional, blazers, basketball

Table 3: Top five words in five random mi-vector and SAGE topics learned on text data at $K = 600$.

6 Discussion

We have theoretically and empirically compared several content-bearing representations of text and speech from prior work. We have measured the relative performance of these representations on topic ID, an easy-to-evaluate task familiar to both text and speech research communities. We have also assessed the representations in their ability to discover the topics inherent in a corpus, a task that is more prominent in the text community and more difficult to evaluate. On our subset of the Fisher English data, these tasks appear to have competing objectives: the best representations in one task are not necessarily the best in the other. In particular, while SAGE yields the worst performance as a feature learner for topic ID, it is demonstrably superior to other low-dimensional learned representations in topic discovery. We have evaluated performance in topic discovery by distributional similarity to gold-standard topics as a proxy for human-annotated judgments of topic quality, and briefly compared the interpretability of mi-vectors and SAGE; future work could pursue expert or crowd-sourced human evaluations.

In the full-supervision setting of topic ID, the lower-dimensional learned representations converge in performance to the raw representation as the dimension K increases. However, if only a couple of labeled examples per class are available,

reflecting the expense of obtaining labels in practice, then learned representations generally outperform the raw representation, which is more prone to overfitting. It is surprising that tf-idf performs so well in the limited supervision setting; it is learned from the data, but it should be prone to overfitting due to its high dimensionality. It is also surprising that SAGE performance on text degrades significantly at high dimensions; we suspect this is due to topic switching, but further investigation is warranted. Overall, though, for topic ID on word counts or triphone state cluster soft counts, if labeled data is scarce, we benefit from training on unsupervised learned representations.

In the V-measure experiment, the documents were partitioned according to the heaviest coordinate in their representations. This choice of experimental protocol is a nuisance variable in our results; other partition constructions may yield different conclusions. In particular, the heaviest-coordinate partition may favor topic models, whose representations are probability vectors, and disfavor mi-vectors and LSA, whose representations may have positive and negative coordinates encoding general linear combinations.

Within each task, the ranking of the representations (by performance) is generally consistent between the text and speech data; however, mi-vectors often outperform LDA on the speech data, while LDA often outperforms mi-vectors on the text data. This may be evidence that the two communities have already independently identified appropriate dimensionality reduction techniques for their respective data sources. However, our results support that the speech community can benefit from broader use of sparsity-inducing graphical models such as SAGE in tasks like spoken topic discovery and recommendation, in which human-interpretable representations are desired. The text community may similarly benefit from parsimonious models such as LSA or mi-vectors in downstream tasks; underparametrized mi-vectors perform particularly well on text, and future work may benefit from investigating this setting.

Word counts and triphone state cluster soft counts provide only one view of text and speech (respectively), and other input representations may yield different conclusions. The particular LSA approach we used for text, based on tf-idf weighting, is not as appropriate for our speech data, which is dense. Future work could evaluate other implementations of LSA or use a higher-level view of speech, such as triphone state cluster n-grams, that more naturally exhibits sparsity and lends to tf-idf weighting. In particular, weighting by a likelihood ratio test statistic and applying a log transform has generated better performance in several other tasks (Lapesa and Evert, 2014). Future work could also test our conclusions on higher-resource views of speech, such as ASR word counts, or lower-resource views such as mel-frequency cepstral coefficients (MFCCs).

We have provided a brief cross-community evaluation of learned representations on multinomial text and speech data. Some prior work has evaluated related learned representations on text data alone, surveying parameters and tasks at greater breadth (Lapesa and Evert, 2014; Levy et al., 2015). A similarly comprehensive evaluation spanning the text and speech research communities would demand great effort but provide a large and versatile resource. In complement, a detailed, case-by-case analysis of errors made by the models in our study could illuminate future modeling efforts by exposing exactly how and why each model errs or excels in each task.

7 Conclusion

Topic ID and topic discovery are competing objectives in our setting: we found that the best-performing representations per task were the same whether considering text- or speech-based communications. By evaluating learned representations from both the text and speech communities on a common set of data and tasks, we have provided a framework for better understanding the topic ID and topic discovery objectives, among others. More generally, we hope to encourage cross-community collaboration to accelerate convergence toward comprehensive models of language.

Acknowledgments

We would like to thank the three anonymous reviewers for their feedback. A National Science Foundation Graduate Research Fellowship, under Grant No. DGE-1232825, supported the second author. We would like to thank the Johns Hopkins HLTCOE for providing support. Any opinions expressed in this work are those of the authors.

References

Hila Becker. 2011. *Identification and Characterization of Events in Social Media*. Ph.D. thesis, Columbia University.

David M. Blei, Thomas L. Griffiths, Michael I. Jordan, and Joshua B. Tenenbaum. 2003a. Hierarchical topic models and the nested chinese restaurant process. In *Advances in Neural Information Processing Systems 16 (NIPS)*.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003b. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Kuan-Yu Chen, Hung-Shin Lee, Hsin-Min Wang, Berlin Chen, and Hsin-Hsi Chen. 2014. I-vector based language modeling for spoken document retrieval. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 7083–7088.

Christopher Cieri, David Graff, Owen Kimball, Dave Miller, and Kevin Walker. 2004a. Fisher english training speech part 1 speech LDC2004S13. DVD.

Christopher Cieri, David Graff, Owen Kimball, Dave Miller, and Kevin Walker. 2004b. Fisher english training speech part 1 transcripts LDC2004T19. Web Download.

Christopher Cieri, David Miller, and Kevin Walker. 2004c. The fisher corpus: a resource for the next generations of speech-to-text. In *International Conference on Language Resources and Evaluation (LREC)*, pages 69–71.

Christopher Cieri, David Graff, Owen Kimball, Dave Miller, and Kevin Walker. 2005. Fisher english training part 2, speech LDC2005S13. DVD.

Scott C. Deerwester, Susan T Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American Society For Information Science*, 41(6):391–407.

Najim Dehak, Patrick J. Kenny, Réda Dehak, Pierre Dumouchel, and Pierre Ouellet. 2011. Front-end factor analysis for speaker verification. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(4):788–798.

Jacob Eisenstein, Amr Ahmed, and Eric P. Xing. 2011. Sparse additive generative models of text. In *Proceedings of the 28th International Conference on Machine Learning (ICML)*, pages 1041–1048.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874.

Daniel Garcia-Romero and Carol Y. Espy-Wilson. 2011. Analysis of i-vector length normalization in speaker recognition systems. In *Interspeech*, pages 249–252.

Matthew R Gormley, Mark Dredze, Benjamin Van Durme, and Jason Eisner. 2012. Shared components topic models. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL HLT)*, pages 783–792.

Timothy J. Hazen, Fred Richardson, and Anna Margolis. 2007. Topic identification from audio recordings using word and phone recognition lattices. In *IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pages 659–664.

Marcel Kockmann, Lukáš Burget, Ondřej Glembek, Luciana Ferrer, and Jan Černocký. 2010. Prosodic speaker verification using subspace multinomial models with intersession compensation. In *Interspeech*, pages 1061–1064.

Gabriella Lapesa and Stefan Evert. 2014. A large scale evaluation of distributional semantic models: Parameters, interactions and model selection. *Transactions of the Association for Computational Linguistics*, 2:531–545.

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

Wei Li and Andrew McCallum. 2006. Pachinko allocation: Dag-structured mixture models of topic correlations. In *Proceedings of the 23th International Conference on Machine Learning (ICML)*, pages 577–584.

David Martínez, Oldřich Plchot, Lukáš Burget, Ondřej Glembek, and Pavel Matějka. 2011. Language recognition in ivectors space. In *Interspeech*, pages 861–864.

Andrew Kachites McCallum. 2002. Mallet: A machine learning for language toolkit. http://mallet.cs.umass.edu.

Alan McCree and Daniel Garcia-Romero. 2015. DNN senone MAP multinomial i-vectors for phonotactic language recognition. In *Interspeech*. To appear.

David Mimno, Hanna M. Wallach, Edmund Talley, Miriam Leenders, and Andrew McCallum. 2011. Optimizing semantic coherence in topic models. In *Proceedings of the Conference on Empirical Methods on Natural Language Processing (EMNLP)*, pages 262–272.

Mohamed Morchid, Mohamed Bouallegue, Richard Dufour, Georges Linarès, Driss Matrouf, and Renato de Mori. 2014. An i-vector based approach to compact multi-granularity topic spaces representation of textual documents. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 443–454.

David Newman, Chaitanya Chemudugunta, and Padhraic Smyth. 2006. Statistical entity-topic models. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 680–686.

Michael Paul and Mark Dredze. 2015. Sprite: Generalizing topic models with structured priors. *Transactions of the Association for Computational Linguistics*, 3:43–57.

Michael Paul and Roxana Girju. 2010. A two-dimensional topic-aspect model for discovering multi-faceted topics. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence*, pages 545–550.

Andrew Rosenberg and Julia Hirschberg. 2007. V-measure: A conditional entropy-based external cluster evaluation measure. In *Proceedings of the 2007 EMNLP-CoNLL Joint Conference*, pages 410–420.

Mehdi Soufifar, Marcel Kockmann, Lukáš Burget, Oldřich Plchot, Ondřej Glembek, and Torbjørn Svendsen. 2011. iVector approach to phonotactic language recognition. In *Interspeech*, pages 2913–2916.

Ivan Titov and Ryan McDonald. 2008. Modeling online reviews with multi-grain topic models. In *Proceedings of the 17th International World Wide Web Conference (WWW)*, pages 111–120.

Hanna M. Wallach, David Mimno, and Andrew McCallum. 2009. Rethinking lda: Why priors matter. In *Advances in Neural Information Processing Systems 22 (NIPS)*.

Jonathan Wintrode and Sanjeev Khudanpur. 2014. Limited resource term detection for effective topic identification of speech. In *IEEE International Conference on Acoustic, Speech and Signal Processing (ICASSP)*, pages 7118–7122.

Jonathan Wintrode. 2013. Leveraging locality for topic identification of conversational speech. In *Interspeech*, pages 1579–1583.

A Dynamic Programming Algorithm for Computing N-gram Posteriors from Lattices

Doğan Can
Department of Computer Science
University of Southern California
Los Angeles, CA, USA
dogancan@usc.edu

Shrikanth S. Narayanan
Department of Electrical Engineering
University of Southern California
Los Angeles, CA, USA
shri@sipi.usc.edu

Abstract

Efficient computation of n-gram posterior probabilities from lattices has applications in lattice-based minimum Bayes-risk decoding in statistical machine translation and the estimation of expected document frequencies from spoken corpora. In this paper, we present an algorithm for computing the posterior probabilities of all n-grams in a lattice and constructing a minimal deterministic weighted finite-state automaton associating each n-gram with its posterior for efficient storage and retrieval. Our algorithm builds upon the best known algorithm in literature for computing n-gram posteriors from lattices and leverages the following observations to significantly improve the time and space requirements: i) the n-grams for which the posteriors will be computed typically comprises all n-grams in the lattice up to a certain length, ii) posterior is equivalent to expected count for an n-gram that do not repeat on any path, iii) there are efficient algorithms for computing n-gram expected counts from lattices. We present experimental results comparing our algorithm with the best known algorithm in literature as well as a baseline algorithm based on weighted finite-state automata operations.

1 Introduction

Many complex speech and natural language processing (NLP) pipelines such as Automatic Speech Recognition (ASR) and Statistical Machine Translation (SMT) systems store alternative hypotheses produced at various stages of processing as weighted acyclic automata, also known as lattices. Each lattice stores a large number of hypotheses along with the raw system scores assigned to them. While single-best hypothesis is typically what is desired at the end of the processing, it is often beneficial to consider a large number of weighted hypotheses at earlier stages of the pipeline to hedge against errors introduced by various subcomponents. Standard ASR and SMT techniques like discriminative training, rescoring with complex models and Minimum Bayes-Risk (MBR) decoding rely on lattices to represent intermediate system hypotheses that will be further processed to improve models or system output. For instance, lattice based MBR decoding has been shown to give moderate yet consistent gains in performance over conventional MAP decoding in a number of speech and NLP applications including ASR (Goel and Byrne, 2000) and SMT (Tromble et al., 2008; Blackwood et al., 2010; de Gispert et al., 2013).

Most lattice-based techniques employed by speech and NLP systems make use of posterior quantities computed from probabilistic lattices. In this paper, we are interested in two such posterior quantities: i) n-gram expected count, the expected number of occurrences of a particular n-gram in a lattice, and ii) n-gram posterior probability, the total probability of accepting paths that include a particular n-gram. Expected counts have applications in the estimation of language model statistics from probabilistic input such as ASR lattices (Allauzen et al., 2003) and the estimation term frequencies from spoken corpora while posterior probabilities come up in MBR decoding of SMT lattices (Tromble et al., 2008), relevance ranking of spoken utterances and the estimation of document frequencies from spoken corpora (Karakos et al., 2011; Can and Narayanan, 2013).

The expected count $c(x|A)$ of n-gram x given lattice A is defined as

$$c(x|A) = \sum_{y \in \Sigma^*} \#_y(x) p(y|A) \qquad (1)$$

where $\#_y(x)$ is the number of occurrences of n-

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 2388–2397,
Lisbon, Portugal, 17-21 September 2015.

gram x in hypothesis y and $p(y|A)$ is the posterior probability of hypothesis y given lattice A. Similarly, the posterior probability $p(x|A)$ of n-gram x given lattice A is defined as

$$p(x|A) = \sum_{y \in \Sigma^*} 1_y(x) p(y|A) \quad (2)$$

where $1_y(x)$ is an indicator function taking the value 1 when hypothesis y includes n-gram x and 0 otherwise. While it is straightforward to compute these posterior quantities from weighted n-best lists by examining each hypothesis separately and keeping a separate accumulator for each observed n-gram type, it is infeasible to do the same with lattices due to the sheer number of hypotheses stored. There are efficient algorithms in literature (Allauzen et al., 2003; Allauzen et al., 2004) for computing n-gram expected counts from weighted automata that rely on weighted finite state transducer operations to reduce the computation to a sum over n-gram occurrences eliminating the need for an explicit sum over accepting paths. The rather innocent looking difference between Equations 1 and 2, $\#_y(x)$ vs. $1_y(x)$, makes it hard to develop similar algorithms for computing n-gram posteriors from weighted automata since the summation of probabilities has to be carried out over paths rather than n-gram occurrences (Blackwood et al., 2010; de Gispert et al., 2013).

The problem of computing n-gram posteriors from lattices has been addressed by a number of recent works (Tromble et al., 2008; Allauzen et al., 2010; Blackwood et al., 2010; de Gispert et al., 2013) in the context of lattice-based MBR for SMT. In these works, it has been reported that the time required for lattice MBR decoding is dominated by the time required for computing n-gram posteriors. Our interest in computing n-gram posteriors from lattices stems from its potential applications in spoken content retrieval (Chelba et al., 2008; Karakos et al., 2011; Can and Narayanan, 2013). Computation of document frequency statistics from spoken corpora relies on estimating n-gram posteriors from ASR lattices. In this context, a spoken document is simply a collection of ASR lattices. The n-grams of interest can be word, syllable, morph or phoneme sequences. Unlike in the case of lattice-based MBR for SMT where the n-grams of interest are relatively short – typically up to 4-grams –, the n-grams we are interested in are in many instances relatively long sequences of subword units.

In this paper, we present an efficient algorithm for computing the posterior probabilities of all n-grams in a lattice and constructing a minimal deterministic weighted finite-state automaton associating each n-gram with its posterior for efficient storage and retrieval. Our n-gram posterior computation algorithm builds upon the custom forward procedure described in (de Gispert et al., 2013) and introduces a number of refinements to significantly improve the time and space requirements:

- The custom forward procedure described in (de Gispert et al., 2013) computes unigram posteriors from an input lattice. Higher order n-gram posteriors are computed by first transducing the input lattice to an n-gram lattice using an order mapping transducer and then running the custom forward procedure on this higher order lattice. We reformulate the custom forward procedure as a dynamic programming algorithm that computes posteriors for successively longer n-grams and reuses the forward scores computed for the previous order. This reformulation subsumes the transduction of input lattices to n-gram lattices and obviates the need for constructing and applying order mapping transducers.

- Comparing Eq. 1 with Eq. 2, we can observe that posterior probability and expected count are equivalent for an n-gram that do not repeat on any path of the input lattice. The key idea behind our algorithm is to limit the costly posterior computation to only those n-grams that can potentially repeat on some path of the input lattice. We keep track of repeating n-grams of order n and use a simple impossibility argument to significantly reduce the number of n-grams of order $n+1$ for which posterior computation will be performed. The posteriors for the remaining n-grams are replaced with expected counts. This filtering of n-grams introduces a slight bookkeeping overhead but in return dramatically reduces the runtime and memory requirements for long n-grams.

- We store the posteriors for n-grams that can potentially repeat on some path of the input lattice in a weighted prefix tree that we construct on the fly. Once that is done, we com-

Table 1: Common semirings.

SEMIRING	SET	$\oplus$	$\otimes$	$\overline{0}$	$\overline{1}$
Boolean	$\{0, 1\}$	$\vee$	$\wedge$	0	1
Probability	$\mathbb{R}_+ \cup \{+\infty\}$	$+$	$\times$	0	1
Log	$\mathbb{R} \cup \{-\infty, +\infty\}$	$\oplus_{\log}$	$+$	$+\infty$	0
Tropical	$\mathbb{R} \cup \{-\infty, +\infty\}$	$\min$	$+$	$+\infty$	0

$a \oplus_{\log} b = -\log(e^{-a} + e^{-b})$

pute the expected counts for all n-grams in the input lattice and represent them as a minimal deterministic weighted finite-state automaton, known as a factor automaton (Allauzen et al., 2004; Mohri et al., 2007), using the approach described in (Allauzen et al., 2004). Finally we use general weighted automata algorithms to merge the weighted factor automaton representing expected counts with the weighted prefix tree representing posteriors to obtain a weighted factor automaton representing posteriors that can be used for efficient storage and retrieval.

2 Preliminaries

This section introduces the definitions and notation related to weighted finite state automata and transducers (Mohri, 2009).

2.1 Semirings

Definition 1 *A semiring is a 5-tuple* $(\mathbb{K}, \oplus, \otimes, \overline{0}, \overline{1})$ *where* $(\mathbb{K}, \oplus, \overline{0})$ *is a commutative monoid,* $(\mathbb{K}, \otimes, \overline{1})$ *is a monoid,* $\otimes$ *distributes over* $\oplus$ *and* $\overline{0}$ *is an annihilator for* $\otimes$.

Table 1 lists common semirings. In speech and language processing, two semirings are of particular importance. The *log semiring* is isomorphic to the probability semiring via the negative-log morphism and can be used to combine probabilities in the log domain. The *tropical semiring*, provides the algebraic structure necessary for shortest-path algorithms and can be derived from the log semiring using the Viterbi approximation.

2.2 Weighted Finite-State Automata

Definition 2 *A weighted finite-state automaton (WFSA)* A *over a semiring* $(\mathbb{K}, \oplus, \otimes, \overline{0}, \overline{1})$ *is a 7-tuple* $A = (\Sigma, Q, I, F, E, \lambda, \rho)$ *where:* Σ *is the finite input alphabet;* Q *is a finite set of states;* $I, F \subseteq Q$ *are respectively the set of initial and final states;* $E \subseteq Q \times (\Sigma \cup \{\varepsilon\}) \times \mathbb{K} \times Q$ *is a finite set of arcs;* $\lambda : I \to \mathbb{K}$, $\rho : F \to \mathbb{K}$ *are respectively the initial and final weight functions.*

Given an arc $e \in E$, we denote by $i[e]$ its input label, $w[e]$ its weight, $s[e]$ its source or origin state and $t[e]$ its target or destination state. A path $\pi = e_1 \cdots e_k$ is an element of E^* with consecutive arcs satisfying $t[e_{i-1}] = s[e_i]$, $i = 2, \ldots, k$. We extend t and s to paths by setting $t[\pi] = s[e_k]$ and $s[\pi] = t[e_1]$. The labeling and the weight functions can also be extended to paths by defining $i[\pi] = i[e_1] \ldots i[e_k]$ and $w[\pi] = w[e_1] \otimes \ldots \otimes w[e_k]$. We denote by $\Pi(q, q')$ the set of paths from q to q' and by $\Pi(q, x, q')$ the set of paths from q to q' with input string $x \in \Sigma^*$. These definitions can also be extended to subsets $S, S' \subseteq Q$, e.g.

$$\Pi(S, x, S') = \bigcup_{q \in S, q' \in S'} \Pi(q, x, q').$$

An *accepting* path in an automaton A is a path in $\Pi(I, F)$. A string x is accepted by A if there exists an accepting path π labeled with x. A is *deterministic* if it has at most one initial state and at any state no two outgoing transitions share the same input label. The weight associated by an automaton A to a string $x \in \Sigma^*$ is given by

$$[\![A]\!](x) = \bigoplus_{\pi \in \Pi(I, x, F)} \lambda(s[\pi]) \otimes w[\pi] \otimes \rho(t[\pi])$$

and $[\![A]\!](x) \triangleq \overline{0}$ when $\Pi(I, x, F) = \emptyset$.

A weighted automaton A defined over the probability semiring $(\mathbb{R}_+, +, \times, 0, 1)$ is said to be *probabilistic* if for any state $q \in Q$, the sum of the weights of all cycles at q, $\oplus_{\pi \in \Pi(q,q)} w[\pi]$, is well-defined and in $\mathbb{R}_+$ and $\sum_{x \in \Sigma^*} [\![A]\!](x) = 1$.

2.3 N-gram Mapping Transducer

We denote by Φ_n the n-gram mapping transducer (Blackwood et al., 2010; de Gispert et al., 2013)

of order n. This transducer maps label sequences to n-gram sequences of order n. Φ_n is similar in form to the weighted finite-state transducer representation of a backoff n-gram language model (Allauzen et al., 2003). We denote by A_n the n-gram lattice of order n obtained by composing lattice A with Φ_n, projecting the resulting transducer onto its output labels, i.e. n-grams, to obtain an automaton, removing ε-transitions, determinizing and minimizing (Mohri, 2009). A_n is a compact lattice of n-gram sequences of order n consistent with the labels and scores of lattice A. A_n typically has more states than A due to the association of distinct n-gram histories with states.

2.4 Factor Automata

Definition 3 *Given two strings $x, y \in \Sigma^*$, x is a factor (substring) of y if $y = uxv$ for some $u, v \in \Sigma^*$. More generally, x is a factor of a language $L \subseteq \Sigma^*$ if x is a factor of some string $y \in L$. The factor automaton $S(y)$ of a string y is the minimal deterministic finite-state automaton recognizing exactly the set of factors of y. The factor automaton $S(A)$ of an automaton A is the minimal deterministic finite-state automaton recognizing exactly the set of factors of A, that is the set of factors of the strings accepted by A.*

Factor automaton (Mohri et al., 2007) is an efficient and compact data structure for representing a full index of a set of strings, i.e. an automaton. It can be used to determine if a string x is a factor in time linear in its length $O(|x|)$. By associating a weight with each factor, we can generalize the factor automaton structure to weighted automata and use it for efficient storage and retrieval of n-gram posteriors and expected counts.

3 Computation of N-gram Posteriors

In this section we present an efficient algorithm based on the n-gram posterior computation algorithm described in (de Gispert et al., 2013) for computing the posterior probabilities of all n-grams in a lattice and constructing a weighted factor automaton for efficient storage and retrieval of these posteriors. We assume that the input lattice is an ε-free acyclic probabilistic automaton. If that is not the case, we can use general weighted automata ε-removal and weight-pushing algorithms (Mohri, 2009) to preprocess the input automaton.

Algorithm 1 reproduces the original algorithm of (de Gispert et al., 2013) in our notation. Each iteration of the outermost loop starting at line 1 computes posterior probabilities of all unigrams in the n-gram lattice $A_n = (\Sigma_n, Q_n, I_n, F_n, E_n, \lambda_n, \rho_n)$, or equivalently all n-grams of order n in the lattice A. The inner loop starting at line 6 is essentially a custom forward procedure computing not only the standard forward probabilities $\alpha[q]$, the marginal probability of paths that lead to state q,

$$\alpha[q] = \bigoplus_{\pi \in \Pi(I,q)} \lambda(s[\pi]) \otimes w[\pi] \tag{3}$$

$$= \bigoplus_{\substack{e \in E \\ t[e]=q}} \alpha[s[e]] \otimes w[e] \tag{4}$$

but also the label specific forward probabilities $\tilde{\alpha}[q][x]$, the marginal probability of paths that lead to state q and include label x.

$$\tilde{\alpha}[q][x] = \bigoplus_{\substack{\pi \in \Pi(I,q) \\ \exists\, u,v \in \Sigma^*:\ i[\pi]=uxv}} \lambda(s[\pi]) \otimes w[\pi] \tag{5}$$

$$= \bigoplus_{\substack{e \in E \\ t[e]=q \\ i[e]=x}} \alpha[s[e]] \otimes w[e]$$

$$\oplus \bigoplus_{\substack{e \in E \\ t[e]=q \\ i[e]\neq x}} \tilde{\alpha}[s[e]][x] \otimes w[e] \tag{6}$$

Just like in the case of the standard forward algorithm, visiting states in topological order ensures that forward probabilities associated with a state has already been computed when that state is visited. At each state s, the algorithm examines each arc $e = (s, x, w, q)$ and updates the forward probabilities for state q in accordance with the recursions in Equations 4 and 6 by propagating the forward probabilities computed for s (lines 8-12). The conditional on line 11 ensures that the label specific forward probability $\tilde{\alpha}[s][y]$ is propagated to state q only if label y is different from label x, the label on the current arc. In other words, if a label y repeats on some path π leading to state q, then π contributes to $\tilde{\alpha}[q][y]$ only once. This is exactly what is required by the indicator function in Equation 2 when computing unigram posteriors. Whenever a final state is processed, the posterior probability accumulator for each label observed on paths reaching that state is updated by multiplying the label specific forward probability and the final weight associated with that state

Algorithm 1 Compute N-gram Posteriors

```
1  for n ← 1, ..., N do
2      A_n ← Min(Det(RmEps(ProjOut(A ∘ Φ_n))))
3      α[q] ← λ_n(q), ∀ state q ∈ Q_n
4      α̃[q][x] ← 0̄, ∀ state q ∈ Q_n, ∀ label x ∈ Σ_n
5      p(x|A) ← 0̄, ∀ label x ∈ Σ_n
6      for each state s ∈ Q_n do                ▷ In topological order
7          for each arc (s, x, w, q) ∈ E_n do
8              α[q] ← α[q] ⊕ α[s] ⊗ w
9              α̃[q][x] ← α̃[q][x] ⊕ α[s] ⊗ w
10             for each label y ∈ α̃[s] do
11                 if y ≠ x then
12                     α̃[q][y] ← α̃[q][y] ⊕ α̃[s][y] ⊗ w
13         if s ∈ F_n then
14             for each label x ∈ α̃[s] do
15                 p(x|A) ← p(x|A) ⊕ α̃[s][x] ⊗ ρ_n(s)
16 P ← Min(ConstructPrefixTree(p))
```

and adding the resulting value to the accumulator (lines 13-15). It should be noted that this algorithm is a form of marginalization (de Gispert et al., 2013), rather than a counting procedure, due to the conditional on line 11. If that conditional were to be removed, this algorithm would compute n-gram expected counts instead of posterior probabilities.

The key idea behind our algorithm is to restrict the computation of posteriors to only those n-grams that may potentially repeat on some path of the input lattice and exploit the equivalence of expected counts and posterior probabilities for the remaining n-grams. It is possible to extend Algorithm 1 to implement this restriction by keeping track of repeating n-grams of order n and replacing the output labels of appropriate arcs in Φ_{n+1} with ε labels. Alternatively we can reformulate Algorithm 1 as in Algorithm 2. In this formulation we compute n-gram posteriors directly on the input lattice A without constructing the n-gram lattice A_n. We explicitly associate states in the original lattice with distinct n-gram histories which is implicitly done in Algorithm 1 by constructing the n-gram lattice A_n. This explicit association lets us reuse forward probabilities computed at order n while computing the forward probabilities at order $n+1$. Further, we can directly restrict the n-grams for which posterior computation will be performed.

In Algorithm 2, $\acute{\alpha}[n][q][h]$ represents the history specific forward probability of state q, the marginal probability of paths that lead to state q and include length n string h as a suffix.

$$\acute{\alpha}[n][q][h] = \bigoplus_{\substack{\pi \in \Pi(I,q) \\ \exists\, z \in \Sigma^* \colon\, i[\pi] = zh}} \lambda(s[\pi]) \otimes w[\pi] \tag{7}$$

$$= \bigoplus_{\substack{e \in E \\ t[e] = q \\ g \in \acute{\alpha}[n-1][s[e]] \\ gi[e] = h}} \acute{\alpha}[n-1][s[e]][g] \otimes w[e] \tag{8}$$

$\acute{\alpha}[n][q][h]$ is the analogue of $\alpha[q]$ in Algorithm 1. It splits the forward probability of state q (Equation 3), among length n suffixes (or histories) of paths that lead to state q. We can interpret $\acute{\alpha}[n][q][h]$ as the forward probability of state (q, h) in the n-gram lattice A_{n+1}. Here $(q, h) \in Q_{n+1}$ denotes the unique state corresponding to state q in the original lattice A and state h in the mapping transducer Φ_{n+1}. $\hat{\alpha}[q][h][x]$ represents the history and n-gram specific forward probability of state q, the marginal probability of paths that lead to state q, include length $n-1$ string h as a suffix and

Algorithm 2 Compute N-gram Posteriors (Reformulation)

1 $R[0] \leftarrow \{\varepsilon\}$
2 $\acute{\alpha}[0][q][\varepsilon] \leftarrow \alpha[q],\ \forall$ state $q \in Q$
3 **for** $n \leftarrow 1, \ldots, N$ **do**
4 $\quad R[n] \leftarrow \emptyset$
5 $\quad \acute{\alpha}[n][q][x] \leftarrow \overline{0},\ \forall$ state $q \in Q,\ \forall$ ngram $x \in \Sigma^n$
6 $\quad \hat{\alpha}[q][h][x] \leftarrow \overline{0},\ \forall$ state $q \in Q,\ \forall$ history $h \in \Sigma^{n-1},\ \forall$ ngram $x \in \Sigma^n$
7 $\quad p(x|A) \leftarrow \overline{0},\ \forall$ ngram $x \in \Sigma^n$
8 $\quad$ **for each** state $s \in Q$ **do** ▷ In topological order
9 $\quad\quad$ **for each** history $g \in \acute{\alpha}[n-1][s]$ **where** $g \in R[n-1]$ **do**
10 $\quad\quad\quad$ **for each** arc $(s, i, w, q) \in E$ **do**
11 $\quad\quad\quad\quad x \leftarrow gi$ ▷ Concatenate history and label
12 $\quad\quad\quad\quad h \leftarrow x[1:n]$ ▷ Drop first label
13 $\quad\quad\quad\quad$ **if** $h \in R[n-1]$ **then**
14 $\quad\quad\quad\quad\quad \acute{\alpha}[n][q][x] \leftarrow \acute{\alpha}[n][q][x] \oplus \acute{\alpha}[n-1][s][g] \otimes w$
15 $\quad\quad\quad\quad\quad \hat{\alpha}[q][h][x] \leftarrow \hat{\alpha}[q][h][x] \oplus \acute{\alpha}[n-1][s][g] \otimes w$
16 $\quad\quad\quad\quad\quad$ **for each** ngram $y \in \hat{\alpha}[s][g]$ **do**
17 $\quad\quad\quad\quad\quad\quad$ **if** $y \neq x$ **then**
18 $\quad\quad\quad\quad\quad\quad\quad \hat{\alpha}[q][h][y] \leftarrow \hat{\alpha}[q][h][y] \oplus \hat{\alpha}[s][g][y] \otimes w$
19 $\quad\quad\quad\quad\quad\quad$ **else**
20 $\quad\quad\quad\quad\quad\quad\quad R[n] \leftarrow R[n] \cup \{y\}$
21 $\quad\quad$ **if** $s \in F$ **then**
22 $\quad\quad\quad$ **for each** history $g \in \hat{\alpha}[s]$ **do**
23 $\quad\quad\quad\quad$ **for each** ngram $x \in \hat{\alpha}[s][g]$ **do**
24 $\quad\quad\quad\quad\quad p(x|A) \leftarrow p(x|A) \oplus \hat{\alpha}[s][g][x] \otimes \rho(s)$
25 $P' \leftarrow \text{ConstructPrefixTree}(p)$
26 $C \leftarrow \text{ComputeExpectedCounts}(A, N)$
27 $P \leftarrow \text{Min}(\text{Det}(\text{RmEps}((C - \text{RmWeight}(P')) \oplus P')))$

include n-gram x as a substring.

$$\hat{\alpha}[q][h][x] = \bigoplus_{\substack{\pi \in \Pi(I,q) \\ \exists\, z \in \Sigma^*:\ i[\pi] = zh \\ \exists\, u,v \in \Sigma^*:\ i[\pi] = uxv}} \lambda(s[\pi]) \otimes w[\pi] \tag{9}$$

$$= \bigoplus_{\substack{e \in E \\ t[e] = q \\ g \in \acute{\alpha}[|h|][s[e]] \\ gi[e] = x}} \acute{\alpha}[|h|][s[e]][g] \otimes w[e] \;\oplus \bigoplus_{\substack{e \in E \\ t[e] = q \\ g \in \hat{\alpha}[s[e]] \\ gi[e] \neq x}} \hat{\alpha}[s[e]][g][x] \otimes w[e] \tag{10}$$

$\hat{\alpha}[q][h][x]$ is the analogue of $\tilde{\alpha}[q][x]$ in Algorithm 1. $R[n]$ represents the set of n-grams of order n that repeat on some path of A. We start by defining $R[0] \triangleq \{\varepsilon\}$, i.e. the only repeating n-gram of order 0 is the empty string ε, and computing $\acute{\alpha}[0][q][\varepsilon] \equiv \alpha[q]$ using the standard forward algorithm. Each iteration of the outermost loop starting at line 3 computes posterior probabilities of all n-grams of order n directly on the lattice A. At iteration n, we visit the states in topological order and examine each length $n-1$ history g associated with s, the state we are in. For each history g, we go over the set of arcs leaving state s, construct the current n-gram x by concatenating g with the current arc label i (line 11), construct the length $n-1$ history h of the target state q (line 12), and update the forward probabilities for the target state history pair (q, h) in accordance with the recursions in Equations 8 and 10 by propagating the forward probabilities computed for the state history pair (s, g) (lines 14-18). Whenever a final state is processed, the posterior probability accumulator for

each n-gram of order n observed on paths reaching that state is updated by multiplying the n-gram specific forward probability and the final weight associated with that state and adding the resulting value to the accumulator (lines 21-24).

We track repeating n-grams of order n to restrict the costly posterior computation operation to only those n-grams of order $n+1$ that can potentially repeat on some path of the input lattice. The conditional on line 17 checks if any of the n-grams observed on paths reaching state history pair (s, g) is the same as the current n-gram x, and if so adds it to the set of repeating n-grams. At each iteration n, we check if the current length $n-1$ history g of the state we are in is in $R[n-1]$, the set of repeating n-grams of order $n-1$ (line 9). If it is not, then no n-gram $x = gi$ can repeat on some path of A since that would require g to repeat as well. If g is in $R[n-1]$, then for each arc $e = (s, i, w, q)$ we check if the length $n-1$ history $h = g[1:n-1]i$ of the next state q is in $R[n-1]$ (line 13). If it is not, then the n-gram $x = g[0]h$ can not repeat either.

We keep the posteriors $p(x|A)$ for n-grams that can potentially repeat on some path of the input lattice in a deterministic WFSA P' that we construct on the fly. P' is a prefix tree where each path π corresponds to an n-gram posterior, i.e. $i[\pi] = x \implies w[\pi] = \rho(t[\pi]) = p(x|A)$. Once the computation of posteriors for possibly repeating n-grams is finished, we use the algorithm described in (Allauzen et al., 2004) to construct a weighted factor automaton C mapping all n-grams observed in A to their expected counts, i.e. $\forall \pi$ in C, $i[\pi] = x \implies w[\pi] = c(x|A)$. We use P' and C to construct another weighted factor automaton P mapping all n-grams observed in A to their posterior probabilities, i.e. $\forall \pi$ in P, $i[\pi] = x \implies w[\pi] = p(x|A)$. First we remove the n-grams accepted by P' from C using the difference operation (Mohri, 2009),

$$C' = C - \mathrm{RmWeight}(P')$$

then take the union of the remaining automaton C' and P', and finally optimize the result by removing ε-transitions, determinizing and minimizing

$$P = \mathrm{Min}(\mathrm{Det}(\mathrm{RmEps}(C' \oplus P'))).$$

4 Experiments and Discussion

In this section we provide experiments comparing the performance of Algorithm 2 with Algorithm 1 as well as a baseline algorithm based on the approach of (Tromble et al., 2008). All algorithms were implemented in C++ using the OpenFst Library (Allauzen et al., 2007). Algorithm 1 implementation is a thin wrapper around the reference implementation. All experiments were conducted on the 88K ASR lattices (total size: #states + #arcs = 33M, disk size: 481MB) generated from the training subset of the IARPA Babel Turkish language pack, which includes 80 hours of conversational telephone speech. Lattices were generated with a speaker dependent DNN ASR system that was trained on the same data set using IBM's Attila toolkit (Soltau et al., 2010). All lattices were pruned to a logarithmic beam width of 5.

Figure 1 gives a scatter plot of the posterior probability computation time vs. the number of lattice n-grams (up to 5-grams) where each point

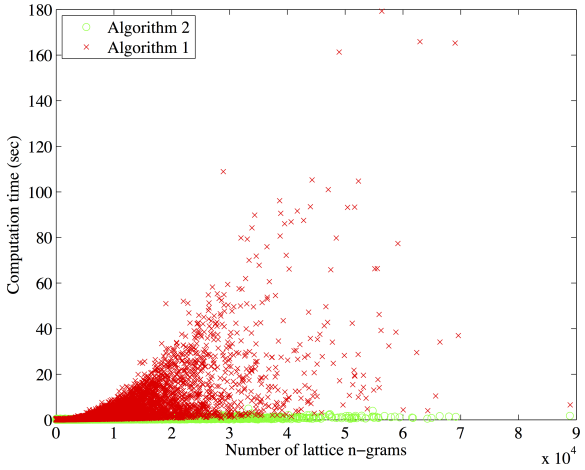


Figure 1: Runtime comparison

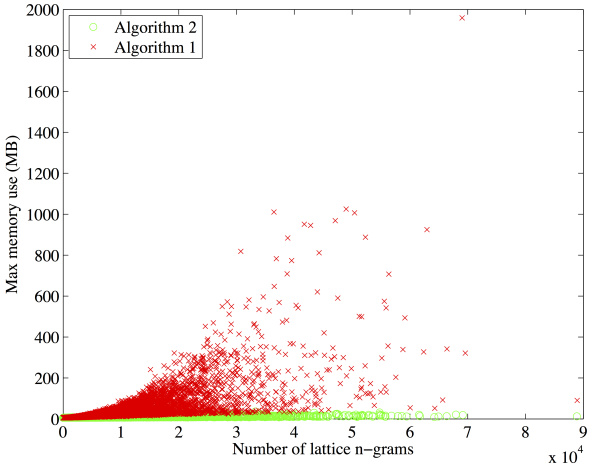


Figure 2: Memory use comparison

Table 2: Runtime Comparison

Max n-gram length	1	2	3	4	5	6	10	all
$\log_{10}(\#n\text{-grams})$	3.0	3.8	4.2	4.5	4.8	5.1	6.3	11.2
Baseline (sec)	5	15	32	69	147	311	5413	-
Algorithm 1 (sec)	0.5	0.6	0.9	1.6	3.9	16	997	-
Algorithm 2 (sec)	0.7	0.8	0.9	1.1	1.2	1.3	1.7	1.0
Expected Count (sec)	0.3	0.4	0.5	0.6	0.7	0.8	1.0	0.5

represents one of the 88K lattices in our data set. Similarly, Figure 2 gives a scatter plot of the maximum memory used by the program (maximum resident set size) during the computation of posteriors vs. the number of lattice n-grams (up to 5-grams). Algorithm 2 requires significantly less resources, particularly in the case of larger lattices with a large number of unique n-grams.

To better understand the runtime characteristics of Algorithms 1 and 2, we conducted a small experiment where we randomly selected 100 lattices (total size: #states + #arcs = 81K, disk size: 1.2MB) from our data set and analyzed the relation between the runtime and the maximum n-gram length N. Table 2 gives a runtime comparison between the baseline posterior computation algorithm described in (Tromble et al., 2008), Algorithm 1, Algorithm 2 and the expected count computation algorithm of (Allauzen et al., 2004). The baseline method computes posteriors separately for each n-gram by intersecting the lattice with an automaton accepting only the paths including that n-gram and computing the total weight of the resulting automaton in log semiring. Runtime complexities of the baseline method and Algorithm 1 are exponential in N due to the explicit enumeration of n-grams and we can clearly see this trend in the 3rd and 4th rows of Table 2. Algorithm 2 (5th row) takes advantage of the WFSA based expected count computation algorithm (6th row) to do most of the work for long n-grams, hence does not suffer from the same exponential growth. Notice the drops in the runtimes of Algorithm 2 and the WFSA based expected count computation algorithm when all n-grams are included into the computation regardless of their length. These drops are due to the expected count computation algorithm that processes all n-grams simultaneously using WFSA operations. Limiting the maximum n-gram length requires pruning long n-grams, which in general can increase the sizes of intermediate WFSAs used in computation and result in longer runtimes as well as larger outputs.

When there is no limit on the maximum n-gram length, the output of Algorithm 2 is a weighted factor automaton mapping each factor to its posterior. Table 3 compares the construction and storage requirements for posterior factor automata with similar factor automata structures. We use the approach described in (Allauzen et al., 2004) for constructing both the unweighted and the expected count factor automata. We construct the unweighted factor automata by first removing the weights on the input lattices and then applying the determinization operation on the tropical semiring so that path weights are not added together. The storage requirements of the posterior factor automata produced by Algorithm 2 is similar to those of the expected count factor automata. Unweighted factor automata, on the other hand, are significantly more compact than their weighted counterparts even though they accept the same set of strings. This difference in size is due to accommodating path weights which in general can significantly impact the effectiveness of automata determinization and minimization.

5 Related Work

Efficient computation of n-gram expected counts from weighted automata was first addressed in (Allauzen et al., 2003) in the context of estimating n-gram language model statistics from ASR lattices. Expected counts for all n-grams of interest observed in the input automaton are computed by composing the input with a simple counting transducer, projecting on the output side, and removing ε-transitions. The weight associated by the resulting WFSA to each n-gram it accepts is simply the expected count of that n-gram in the input automaton. Construction of such an automaton for all substrings (factors) of the input automaton was later explored in (Allauzen et al., 2004) in the con-

Table 3: Factor Automata Comparison

FA Type	Unweighted	Expected Count	Posterior
#states + #arcs (M)	16	20	21
On disk size (MB)	219	545	546
Runtime (min)	5.5	11	22

text of building an index for spoken utterance retrieval (SUR) (Saraclar and Sproat, 2004). This is the approach used for constructing the weighted factor automaton C in Algorithm 2. While expected count works well in practice for ranking spoken utterances containing a query term, posterior probability is in theory a better metric for this task. The weighted factor automaton P produced by Algorithm 2 can be used to construct an SUR index weighted with posterior probabilities.

The problem of computing n-gram posteriors from lattices was first addressed in (Tromble et al., 2008) in the context of lattice-based MBR for SMT. This is the baseline approach used in our experiments and it consists of building a separate FSA for each n-gram of interest and intersecting this automaton with the input lattice to discard those paths that do not include that n-gram and summing up the weights of remaining paths. The fundamental shortcoming of this approach is that it requires separate intersection and shortest distance computations for each n-gram. This shortcoming was first tackled in (Allauzen et al., 2010) by introducing a counting transducer for simultaneous computation of posteriors for all n-grams of order n in a lattice. This transducer works well for unigrams since there is a relatively small number of unique unigrams in a lattice. However, it is less efficient for n-grams of higher orders. This inefficiency was later addressed in (Blackwood et al., 2010) by employing n-gram mapping transducers to transduce the input lattices to n-gram lattices of order n and computing unigram posteriors on the higher order lattices. Algorithm 1 was described in (de Gispert et al., 2013) as a fast alternative to counting transducers. It is a lattice specialization of a more general algorithm for computing n-gram posteriors from a hypergraph in a single inside pass (DeNero et al., 2010). While this algorithm works really well for relatively short n-grams, its time and space requirements scale exponentially with the maximum n-gram length. Algorithm 2 builds upon this algorithm by exploiting the equivalence of expected counts and posteriors for non-repeating n-grams and eliminating the costly posterior computation operation for most n-grams in the input lattice.

6 Conclusion

We have described an efficient algorithm for computing n-gram posteriors from an input lattice and constructing an efficient and compact data structure for storing and retrieving them. The runtime and memory requirements of the proposed algorithm grow linearly with the length of the n-grams as opposed to the exponential growth observed with the original algorithm we are building upon. This is achieved by limiting the posterior computation to only those n-grams that may repeat on some path of the input lattice and using the relatively cheaper expected count computation algorithm for the rest. This filtering of n-grams introduces a slight bookkeeping overhead over the baseline algorithm but in return dramatically reduces the runtime and memory requirements for long n-grams.

Acknowledgments

The authors would like to thank Cyril Allauzen and Graeme W. Blackwood for helpful discussions. This work uses IARPA-babel105b-v0.4 Turkish full language pack from the IARPA Babel Program language collection and is supported by the Intelligence Advanced Research Projects Activity (IARPA) via Department of Defense U.S. Army Research Laboratory (DoD/ARL) contract number W911NF-12-C-0012. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DoD/ARL, or the U.S. Government.

References

Cyril Allauzen, Mehryar Mohri, and Brian Roark. 2003. Generalized algorithms for constructing statistical language models. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, ACL '03, pages 40–47, Stroudsburg, PA, USA. Association for Computational Linguistics.

Cyril Allauzen, Mehryar Mohri, and Murat Saraclar. 2004. General indexation of weighted automata: Application to spoken utterance retrieval. In *HLT-NAACL Workshop on Interdisciplinary Approaches to Speech Indexing and Retrieval*, pages 33–40, Boston, MA, USA.

Cyril Allauzen, Michael Riley, Johan Schalkwyk, Wojciech Skut, and Mehryar Mohri. 2007. OpenFst: A general and efficient weighted finite-state transducer library. In *Proceedings of the Ninth International Conference on Implementation and Application of Automata, (CIAA 2007)*, volume 4783 of *Lecture Notes in Computer Science*, pages 11–23. Springer. `http://www.openfst.org`.

Cyril Allauzen, Shankar Kumar, Wolfgang Macherey, Mehryar Mohri, and Michael Riley. 2010. Expected sequence similarity maximization. In *HLT-NAACL*, pages 957–965. The Association for Computational Linguistics.

Graeme Blackwood, Adrià de Gispert, and William Byrne. 2010. Efficient path counting transducers for minimum bayes-risk decoding of statistical machine translation lattices. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 27–32, Uppsala, Sweden, July. Association for Computational Linguistics.

Dogan Can and Shrikanth Narayanan. 2013. On the computation of document frequency statistics from spoken corpora using factor automata. In *INTERSPEECH 2013, 14th Annual Conference of the International Speech Communication Association*, pages 6–10, Lyon, France.

Ciprian Chelba, Timothy J. Hazen, and Murat Saraclar. 2008. Retrieval and browsing of spoken content. *Signal Processing Magazine, IEEE*, 25(3):39–49, May.

Adrià de Gispert, Graeme Blackwood, Gonzalo Iglesias, and William Byrne. 2013. N-gram posterior probability confidence measures for statistical machine translation: an empirical study. *Machine Translation*, 27(2):85–114.

John DeNero, Shankar Kumar, Ciprian Chelba, and Franz Och. 2010. Model combination for machine translation. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 975–983, Stroudsburg, PA, USA. Association for Computational Linguistics.

Vaibhava Goel and William J Byrne. 2000. Minimum bayes-risk automatic speech recognition. *Computer Speech & Language*, 14(2):115–135.

Damianos Karakos, Mark Dredze, Ken Ward Church, Aren Jansen, and Sanjeev Khudanpur. 2011. Estimating document frequencies in a speech corpus. In David Nahamoo and Michael Picheny, editors, *ASRU*, pages 407–412. IEEE.

Mehryar Mohri, Pedro Moreno, and Eugene Weinstein. 2007. Factor automata of automata and applications. In *Implementation and Application of Automata*, pages 168–179. Springer.

Mehryar Mohri. 2009. Weighted automata algorithms. In Manfred Droste, Werner Kuich, and Heiko Vogler, editors, *Handbook of Weighted Automata*, Monographs in Theoretical Computer Science. An EATCS Series, pages 213–254. Springer Berlin Heidelberg.

Murat Saraclar and Richard Sproat. 2004. Lattice-based search for spoken utterance retrieval. In *Proc. HLT-NAACL*, pages 129–136, Boston, MA, USA.

Hagen Soltau, George Saon, and Brian Kingsbury. 2010. The ibm attila speech recognition toolkit. In *Spoken Language Technology Workshop (SLT), 2010 IEEE*, pages 97–102, Dec.

Roy W Tromble, Shankar Kumar, Franz Och, and Wolfgang Macherey. 2008. Lattice minimum bayes-risk decoding for statistical machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 620–629. Association for Computational Linguistics.

Bilingual Structured Language Models for Statistical Machine Translation

Ekaterina Garmash and **Christof Monz**
Informatics Institute, University of Amsterdam
Science Park 904, 1098 XH Amsterdam, The Netherlands
{e.garmash,c.monz}@uva.nl

Abstract

This paper describes a novel target-side syntactic language model for phrase-based statistical machine translation, bilingual structured language model. Our approach represents a new way to adapt structured language models (Chelba and Jelinek, 2000) to statistical machine translation, and a first attempt to adapt them to phrase-based statistical machine translation. We propose a number of variations of the bilingual structured language model and evaluate them in a series of rescoring experiments. Rescoring of 1000-best translation lists produces statistically significant improvements of up to 0.7 BLEU over a strong baseline for Chinese-English, but does not yield improvements for Arabic-English.

1 Introduction

Many model components of competitive statistical machine translation (SMT) systems are based on rather simplistic definitions with little linguistic grounding, which includes the definitions of phrase pairs, lexicalized reordering, and n-gram language models. However, earlier work has also shown that statistical MT can benefit from additional linguistically motivated models. Most prominent among the linguistically motivated approaches are syntax-based MT systems which take into account the syntactic structure of sentences through CKY decoding and categorial labels (Zollmann and Venugopal, 2006; Shen et al., 2008). On the other hand, the commonly used phrase-based SMT approaches can also reap some of the benefits of using syntactic information by integrating linguistic components addressing specific phenomena, such as Cherry (2008), Carpuat et al. (2010), Crego and Yvon (2010), Ge (2010), Xiang et al. (2011), Lerner and Petrov (2013), Garmash and Monz (2014).

This paper is a contribution to the existing body of work on how syntactically motivated models help translation performance. We work with the phrase-based SMT (PBSMT) (Koehn et al., 2003) framework as the baseline system. Our choice is motivated by the fact that PBSMT is a conceptually simple and therefore flexible framework. It is typically quite straightforward to integrate an additional model into the system. Also, PBSMT is the most widely used framework in the SMT research community, which ensures comparability of our results to other people's work on the topic.

There is a variety of ways syntax can be used in a PBSMT model. Typically a syntactic representation of a source sentence is used to define constraints on the order in which the decoder translates it. For example, Cherry (2008) defines soft constraints based on the notion of syntactic cohesion (Section 2). Ge (2010) captures reordering patterns by defining soft constraints based on the currently translated word's POS tag and the words structurally related to it. On the other hand, target syntax is more challenging to use in PBSMT, since a target-side syntactic model does not have access to the whole target sentence at decoding. Post and Gildea (2008) is one of the few target-side syntactic approaches applicable to PBSMT, but it has been shown not to improve translation. Their approach uses a target side parser as a language model: one of the reasons why it fails is that a parser assumes its input to be grammatical and chooses the most likely parse for it. What we are interested in during translation is how gram-

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 2398–2408,
Lisbon, Portugal, 17-21 September 2015.

matical the target sentence actually is.

In addition to reordering constraints, source syntax can be used for target-side language modeling. A target side string can be encoded with source-syntactic building blocks and then scored as to how well-formed it is. Crego and Yvon (2010), Niehues et al. (2011), Garmash and Monz (2014) model target sequences as strings of tokens built from the target POS tag and the POS tags of the source words related to it through alignment and the source parse. In this paper, we define a target-side syntactic language model that takes structural constraints from the source sentence, but uses the words from the target side (as 'building blocks'). We do it by adapting an existing monolingual model of Chelba and Jelinek (2000), *structured language models*, to the bilingual setting. Our contributions can be summarized as follows:

- we propose a novel method to adapt monolingual structured language models (Chelba and Jelinek, 2000) (Section 3) to a PBSMT system (Section 4), which does not require an external on-the-fly parser, but only uses the given source-side syntactic analysis to infer structural relations between target words;
- building on the existing literature, we propose a set of deterministic rules that incrementally build up a parse of a target translation hypothesis based on the source parse (Section 4);
- we evaluate our models in a series of rescoring experiments and achieve statistically significant improvements of up to 0.7 BLEU for Chinese-English (Section 5).

Before describing the models, we motivate our method with a common assumption about cross-lingual correspondence (Section 2).

2 Direct correspondence assumption and syntactic cohesion in SMT

Before we apply the syntactic model introduced in Section 3 to the bilingual setting (Section 4), we first explain two widely used assumptions about syntactic correspondence across languages.

We take a *dependency tree* to be a syntactic representation of a sentence and reason about other syntactic assumptions and models in its terms. In this work, we choose a dependency structure over a constituency structure because the former is more primitive.[1] A dependency parse D is a dependency tree analysis of a sentence W, and we will think of it as a relation between words of W, such that $D(w, v)$ if w is a parent (head) of v (v being a child/modifier). D can be generalized to D^* which is an relation between words that are connected by a continuous path in a dependency tree (i.e. $D^*(w, v)$ if $D(w, v)$ or if $\exists u$ s.t. $D(w, u) \wedge D^*(u, v)$). We assume *unlabeled* dependency trees. Finally, we make a projectivity assumption, which is supported by empirical data in many languages (Kuhlmann and Nivre, 2006; Havelka, 2007), and makes a model computationally less expensive. A dependency parse D of a sentence $W = w_1, \ldots, w_n$ is *projective*, if for every word pair $w_i, w_j \in W$ s.t. $D(w_i, w_j)$ it holds that every $w_k \in W$ s.t. $i < k < j$ or $j < k < i$ is a descendant of w_i, i.e., $D^*(w_i, w_k)$; see Figure 1.

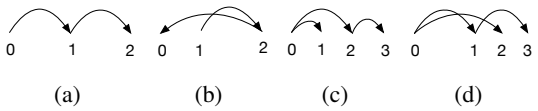


Figure 1: Examples of projective and non-projective parses. (a-b): projective (a) and non-projective (b) parses of the same dependency tree. (b) is non-projective because node 1 is not a descendant of either 0 or 2 (it is the parent of 2). (c-d): projective (c) and non-projective (d) parses of the same dependency tree. Node 2 in (d) is placed between its sibling (node 1) and the child of its sibling (node 3), neither of which is its ancestor.

Most NLP models that address the interaction of two or more languages are based (explicitly or implicitly) on the *direct correspondence assumption* (DCA) (Hwa et al., 2002). It states that close translation equivalents in different languages have the same dependency structure. This is grounded linguistically, as translation equivalence implies semantic equivalence and therefore thematic relations are preserved (Hwa et al., 2002). Thus dependency relations are preserved, as they are defined based on thematic relations between words. On the other hand, there is plenty empirical evidence supporting the violation of DCA under certain conditions (Hwa et al., 2002). For instance, even semantically very close sentences in different languages may have a different number of

[1] A dependency parse (a dependency tree analysis of a sentence) is more primitive because every constituency parse can be formalized as a projective dependency parse with labeled relations, but not vice versa (Osborne, 2008).

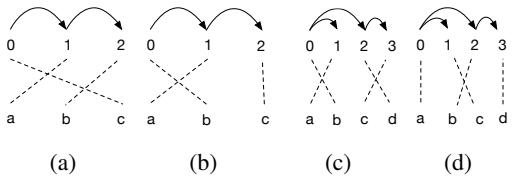


Figure 2: Examples of cohesive and uncohesive translations. (a-b): cohesive (a) and uncohesive (b) translations of the same dependency parse. (b) is uncohesive because words a and c translate the source subtree $\{(1, 2)\}$, but the target word b does not translate this subtree. (c-d): cohesive (c) and uncohesive (d) translations. (d) is uncohesive because a and c translate the source subtree $\{(0, 1)\}$, but b does not translate it.

words. Syntactic divergence increases if the two languages are typologically different.

Even though DCA only holds up to a certain level of precision, it is widely used in NLP. There are models of cross-lingual transfer that define syntactic structure of one language by conditioning it on the structure of semantically equivalent sentences in another language (Naseem et al., 2012). DCA has also been used in SMT. In particular, syntax-based SMT is built implicitly around this assumption (Wu, 1997; Yamada and Knight, 2001). In Quirk and Menezes (2006) DCA is explicitly implemented by defining a translation model in terms of treelet pairs where target-side treelets are produced by projecting source dependencies via word alignments.

Closely related to DCA is the notion of *syntactic cohesion* of translation (Fox, 2002; Cherry, 2008). This is a constraint that does not allow for non-projective reordering: Given a source parse D_S, a translation W is cohesive if all translated target words w_i, w_j do not have any word w_k between them such that there is a source subtree sub in D_S such that some parts of it are translated by w_i and w_j but not by w_k (Figure 2). Cherry (2008) and Bach et al. (2009) define a set of soft constraints based on the syntactic cohesion assumption which are applicable to PBSMT decoding. They only require phrase applications, and not necessarily individual target words, to conform to the cohesion principle. For example, if we imagine a situation where a subtree as in Figure 2(b) is translated as a whole with one phrase application (and not word by word), then it does not violate the cohesion principle, although it is internally uncohesive. Both our approach and Cherry (2008) implement the idea of conforming the target translation to the source syntactic structure, but in different ways. Approaches like Cherry (2008) define principles that constrain the decoder in order to produce better translations. Our goal is to have a model that allows for a more direct way of evaluation of how well-formed the target translation is. In Section 5 we compare translation performance of the two approaches.

3 Structured language models

As discussed in Sections 1 and 2, we would like to test how much a PBSMT can benefit from an additional syntax-based LM. In this section, we describe a syntactic language model, structured LM (SLM) (Chelba and Jelinek, 2000), that we extend to a bilingual setting and apply to SMT in Section 4. SLMs have been applied in SMT before (Yamada and Knight, 2001; Yu et al., 2014), but as we show in Section 4, we provide a much simpler method to integrate it into the system. While a SLM is not the only syntactically defined LM, it is one of the few that models sentence generation sequentially. And due to the way the decoding procedure of PBSMT is defined, it is natural and straightforward to use models whose score can be computed sequentially. Other syntactic language models define sentence generation hierarchically (Shen et al., 2008; Sennrich, 2015), which complicates their integration into a PBSMT system.

The linguistic intuition behind SLMs is that the structural children of a word do not essentially change its distributional properties but just provide additional specification. In Figure 3(a) the word *president* has two modifiers: *the* and *former* and it follows *yesterday* (an adjunct) and precedes *met* (a predicate). This ordering is correct in English. If instead its modifier was *a* or an entire relative clause, it would not make it incorrect.

To capture this observation, (Chelba and Jelinek, 2000) propose a language model where each word w_i of a sentence W is predicted by an ordered subset of the words preceding w_i. This conditioning subset is selected based on the syntactic properties of the preceding sequence W_{i-1}: the strong predictors are kept and the weak ones are left out. The strong predictors are the set of *exposed heads*. Given a subsequence W_{i-1} and its associated parse D_{i-1}, exposed heads are the roots of all the disconnected subtrees in D_{i-1}. Note that

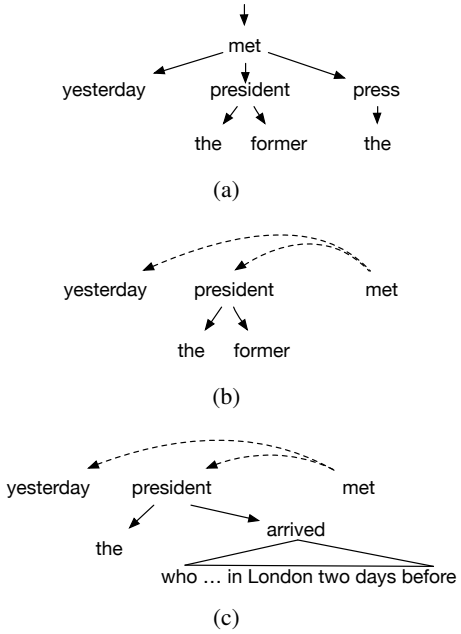


Figure 3: A fully parsed sentence (a) and its partial parse (b) during sequential generation. The partial parse in (b) has two disconnected subtrees with roots *yesterday* and *president*. These roots are the exposed heads for *met*. (c) is an alternative sentence with a similar structure: *president* is still a root of a subtree, and thus and an exposed head.

a parse D_{i-1} is not necessarily fully connected and thus a word can have multiple conditioning words.

For an example, consider again Figure 3(a). In a left-to-right scenario, when *met* is generated, a regular n-gram LM conditions it on *yesterday the former president*, while a SLM conditions it on *yesterday president*, since these two words are the exposed heads with respect to *met* (Figure 3(b)). The words *the* and *former* are modifiers of *president* and they get filtered out. Thus we obtain a less specific conditioning history, which may lead to the resulting model being less sparse. Another potential benefit is that SLMs can capture long-distance reordering: If *president* had as its modifier a relative clause (Figure 3(c)) then a simple n-gram LM would be conditioned on *days before* (assuming $n = 3$), while an SLM would condition *met* on *yesterday president*.

Summarizing the ideas of words being conditioned on a structurally defined subset of the preceding sentence, Chelba and Jelinek (2000) formalize the generation process of W as follows:[2] Each new word w_i is conditioned on a sequence of exposed heads $\mathit{Expos}(W, D)$. Then a tag t_i is predicted, and the parse D_{i-i} of W_{i-1} is extended to D_i incorporating w_i and t_i (where W_{i-1} is the prefix of W preceding w_i):

$$p(W, D) = \prod_{i=1}^{|W|} p(w_i | \mathit{Expos}(W_{i-1}, D_{i-1})) \cdot p(t_i | w_i, \mathit{Expos}(W_{i-1}, D_{i-1})) \cdot p(D_i | w_i, t_i, \mathit{Expos}(W_{i-1}, D_{i-1})). \quad (1)$$

They use a shift-reduce parser with reduce-left, reduce-right, and shift operations.

4 Bilingual structured language models

In this section, we combine the direct correspondence assumption (Section 2) and SLMs (Section 3), and define bilingual structured language models (BiSLMs) for PBSMT. Structured LMs have been successfully applied in SMT before. Yamada and Knight (2001) use SLMs in a string-to-tree SMT system where a derivation of a target-side parse tree is part of the decoding algorithm, and target syntactic representations are obtained 'for free'. Yu et al. (2014) use an on-the-fly shift-reduce parser to build an incremental target parse.

The approaches sketched above rely on resources that a standard PBSMT system does not have access to by default. Phrase-based decoders do not provide us with a parse of the target sentence, and inferring the parse of a target string with an external parser is computationally expensive and potentially unreliable (see Section 1). Our main insight is that in a bilingual setting one does not need an additional probabilistic target parsing model. We assume that the source parse is given (precomputed) and that the DCA (Section 2) holds, and project the parse deterministically onto the target side via word alignments[3]. We obtain the following equation:

$$p(T|S, D_S) = \prod_{i=1}^{|T|} p(t_i | \mathit{Expos}(T_{i-1}, \mathit{ProjP}(D_S, S, T_{i-1}))), \quad (2)$$

where T is a target sentence, T_{i-1} is the sequence in T preceding the i-th target word t_i, S is a

[2]The original model by (Chelba and Jelinek, 2000) is defined in terms of a lexicalized constituency grammar, but as we discussed in Section 2, constituency parses can be transformed into dependency parses.

[3]Phrase-internal word alignments are stored in the phrase table and are available at decoding time, see Section 4.4.

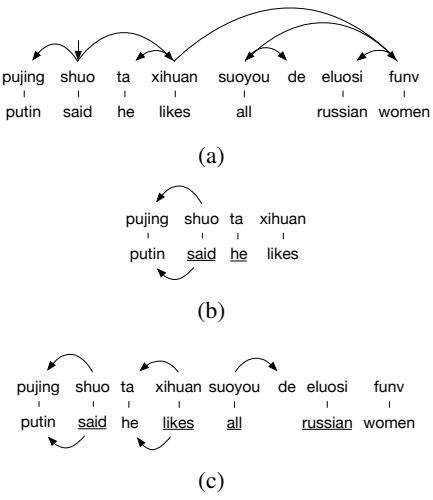


Figure 4: Chinese-English sentence pair (a) and sets of exposed heads (underlined) at different generation (b and c) steps of a bilingual SLM.

source sentence, D_S is a source dependency parse, and *ProjP* is a function that returns a partial target parse D_{Ti-1} by projecting D_S onto T_{i-1}. In words, at each time step i we predict the next word t_i conditioned on the exposed heads of the partial parse of T_{i-1} projected from the source side. We limit *Expos* to returning the four preceding exposed heads.[4] Because the function *ProjP* is deterministic and because we do not have to predict tags for words, Equation 2 is simpler than Equation 1.

We first illustrate Equation 2 with an example in Figure 4. Since word alignment is monotonic in Figure 4(a), it is straightforward to project the source dependencies onto the target side. We aim to imitate a monolingual parser in the way we build up our projected parse: Reduce operations should be invoked whenever both of the subtrees involved in the operation are complete, i.e., are not expected to have any more modifiers (Section 4.2). For example, when the target word *likes* is produced its exposed heads are *said* and *he* (Figure 4(b)), since *Putin* is a modifier of *said*. Likewise, the exposed heads for *women* are *said likes all Russian* (Figure 4(c)).

In what follows we discuss how to define *ProjP*. Compared to projection approaches like (Quirk and Menezes, 2006), we would like our model to project a source parse incrementally, allowing it to be used in a PBSMT decoder. We think of *ProjP* as a function that computes the output in two stages: first, it infers from the source parse the dependency relations between target words (Section 4.1), second, it decides how to parse the target sequence, i.e. in which order to assign these dependencies (Section 4.2). Additionally, in Section 4.3 we propose to use additional labelings of target words, and in Section 4.4 we describe some important implementation details.

4.1 Dependency graph projection

Adoption of DCA (Section 2) allows to build up a target dependency tree from a source tree by projecting the latter through word alignments. The definition of DCA can be rephrased as requiring a one-to-one correspondence *map* between words of a sentence pair, allowing one to unambiguously map dependencies: Given a source parse, if t_1 is the head of t_2, then *map*(t_1) is the head of *map*(t_2). The correspondence relation that we have in PBSMT is the word alignment *align*: in the most general case, it is a many-to-many correspondence, and the straightforward projection described above can lead to incorrect dependency structures. To overcome these problems, we describe a simple ordered set of projection rules, based on the ones specified by (Quirk and Menezes, 2006) (and we point out if otherwise).

The general idea behind this set of rules is to extract a one-to-one function ***align***$_{1-1}$ from source words to target words from *align* and use it to project source dependencies as described in the paragraph above (**R1** below). We then use additional rules (**R2-R4** below) for the target words that are not in *align*$_{1-1}$. Given a source sentence S with a parse D_S, a target sentence T and word alignment *align*, ***align***$_{1-1}$ is extracted as follows: For all $t_i \in T$ with multiple aligned source words $\{s_{i_1}, s_{i_2}, ...\}$ only *align*$_{1-1}(s_{i_1}) = t_i$ (only leftmost source word is kept, the links from the rest of the source words are removed[5]). For all $s_i \in S$ with aligned target words $\{t_{i_1}, t_{i_2}, ...\}$ keep the link only for the leftmost aligned target word: *align*$_{1-1}(s_i) = t_{i_1}$. For example, in Figure 5(b) the link between f_0 and e_1 is not in *align*$_{1-1}$, and in Figure 5(c) the link between f_1 and e_0 is removed (and the arc from f_2 to f_1 is not projected).

[4] As written above, we choose the dependency structures over the lexicalized constituency ones because the latter can be mapped to the former. It is thus more likely that a projected dependency tree is still be a well-formed parse, than a projected constituency tree. We decided to work with structural models that are more flexible, but one may also define BiSLM in terms of the more constraining constituency trees and see if the such model has better generalization power.

[5] This is an ad-hoc solution, other heuristics could be used.

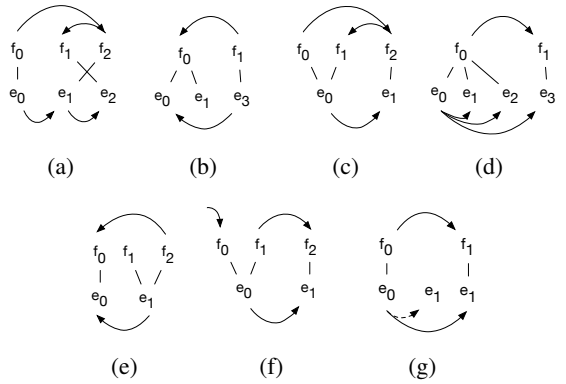


Figure 5: Examples for dependency projection rules. (a): no alignment links get removed (R1). (b): $f_0 - e_1$ link is removed from $align_{1-1}$ (R1). (c): $f_1 - e_0$ link gets removed (R1). (d): e_1 and e_2 get adjoined to e_0 (R2). (e): R3a. (f): R3b. (g) demonstrates two versions of R4: the dashed arrow gets 'realized' only if we adjoin unaligned words to the preceding head.

The following rules should be applied in order (as *else-if* conditions). Given a source sentence S with a parse D_S, a target sentence T and word alignment *align* between them, $t_i \in T$ is a head of $t_j \in T$ (i.e. $D_T(t_i, t_j)$):

(R1) if there are $s_k, s_l \in S$ s.t. $D_S(s_k, s_l)$ and $align_{1-1}(s_k) = t_i$ and $align_{1-1}(s_l) = t_j$; see Figures 5(a)-5(c);

(R2) if $\exists s \in S$ s.t. $align_{1-1}(s) = t_i$ and $(s, t_j) \in align$. This rule deals with one-to-many alignments; see Figure 5(d);

(R3a) if $\exists s_k$ s.t. $align_{1-1}(s_k) = t_i$ and $\exists s_l$ s.t. $(s_l, t_j) \in align$ and and $D_S(s_l, s_k)$, and t_i linearly precedes t_j. In words: if two target words are in $align_{1-1}$ but do not get connected via **R1**, find a source word aligned to the second target word that may get them connected; see Figure 5(e);

(R3b) same as R3a, but in case t_j precedes t_i (i.e., find an additional source word aligned to the first target word; see Figure 5(f)).[6]

(R4) In case $\neg\exists s\ (s, t_j) \in align$ (t_j is unaligned), we consider two strategies: We simplify the rule of Quirk and Menezes (2006) (dealing with the same situation) by adjoining it to the immediately preceding head. We also consider a strategy whereby the word remains unconnected to any word in the sentence; see Figure 5(g).

[6] R3a and R3b differ from the rules proposed in Quirk and Menezes (2006) dealing with the same situation, since we had to adapt it to the left-to-right parsing scenario.

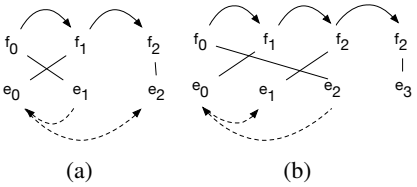


Figure 6: (a): The dashed lines are the dependency arcs that would project through word alignment, resulting in a non-projective projective (impossible under strong source-completeness). (b): The dashed lines are the parse produced under weak source-completeness. Under strong completeness none of the words will get connected.

4.2 BiSLM parsing procedure

Given an inference procedure for dependency relations between target words (Section 4.1), one can specify in which order the corresponding dependency arcs are assigned to the target sentence. We define an incremental parsing procedure in terms of three operations: *shift*, *left-reduce*, and *right-reduce*. The operations are applied as soon as the sufficient conditions hold: We specify the conditions using the following structural properties. A target subtree is *source-complete* if all the descendants of $align^{-1}_{1-1}(root(sub))$ (source correspondent of the root of the current subtree) (Section 4.1) have been translated and reduced. A target subtree is *complete* if it is source-complete and all the target words that are its children through non-projected arcs (through R2 or R4 in Section 4.1) have been translated and reduced. The bilingual parsing operations and the sufficient conditions for them are defined as follows:

Shift: after the word is produced it is shifted onto the stack as an elementary subtree.

Left-reduce: if a disconnected subtree sub_i and a disconnected subtree sub_{i-1} immediately preceding it are both complete and $D_T(root(sub_i), root(sub_{i-1}))$, adjoin sub_{i-1} to sub_i so that $root(sub_{i-1})$ is a modifier of $root(sub_i)$.

Right-reduce: analogous to *left-reduce*, but $D_T(root(sub_{i-1}), root(sub_i))$.

In the case of non-cohesive translation the resulting target dependencies are non-projective. Our definition of left- and right-reduce only produces projective parses. For a non-cohesive translation, certain subtrees will never be source-complete and will never be reduced; see Figure 6(a). Note that this is not a disadvantage

of our model. Cherry (2008) simply assumes that non-cohesive reordering should be penalized, and our model is able to learn this pattern. We also consider an alternative to incorporating non-cohesive alignments by relaxing the definition of completeness for subtrees: A projected subtree sub is *weakly source-complete* if all descendants of all source word(s) which are aligned to the root of sub have been translated and, *only* if the definition of reduce applies, reduced; see Figure 6(b).

4.3 Syntactic labeling of tokens

One of the problems with SLMs in general is that at time steps i and j the sets of exposed heads for t_i and t_j can differ in size, which may imply different predictive power. To this end, we add an additional detail to our model: Each time a reduction occurs, we label the root of the subtree to which another subtree has been adjoined, thus making the conditioning history more specific. We use the following labelings:

Reduction labeling: if a subtree is adjoint to *sub* from the left, then label *root(sub)* with **LR**. If it is adjoint from the right, then label it with **RR**.

Reduction POS-labeling: same as in simple reduction labeling, but add the POS tag of the root of the reduced subtree to the label.

4.4 Implementation and training

To use BiSLM during decoding, one needs access to phrase-internal alignments and target POS tags. We store phrase-internal alignments and target-side POS annotations of each phrase in the phrase table, based on the most frequent internal alignment during training and the most likely target-side POS labeling $\hat{t}$ given the phrase pair: $\hat{t} = \arg\max_{\bar{t}} p(\bar{t}|\bar{e}, \bar{f})$. We train BiSLMs on the parallel training data (Section 5.1) and use the Stanford dependency parser (Chang et al., 2009) for Chinese and and the Stanford constituency parser (Green and Manning, 2010) for Arabic[7]. POS-tagging of the training data is produced with the Stanford POS-tagger (Toutanova et al., 2003). We learn a 5-gram model using SRILM (Stolcke et al., 2011) with modified Kneser-Ney smoothing.

5 Experiments

To evaluate the effectiveness of BiSLMs for PB-SMT, we performed rescoring experiments for Arabic-English and Chinese-English. We compare the resulting 1-best translation lists with an output of the baseline system and the baseline augmented with soft cohesion constraints from Bach et al. (2009).

System	MT06	MT08	MT06+MT08
baseline	32.60	25.94	29.56
cohesion	32.52	25.98	29.54

Table 1: Chinese-English baseline and comparison model (Cherry, 2008; Bach et al., 2009) results.

System	MT08	MT09	MT08+MT09
baseline	45.84	48.61	47.18
cohesion constr.	45.61	48.49	47.02

Table 2: Arabic-English baseline and comparison model (Cherry, 2008; Bach et al., 2009) results.

5.1 Experimental setup

This section provides information about our baseline system. Word-alignment is produced with GIZA++ (Och and Ney, 2003). We use an in-house implementation of a PBSMT system similar to Moses (Koehn et al., 2007). Our baseline has all standard PBSMT features including language model, lexical weighting, and lexicalized reordering. The distortion limit is set to 5. A 5-gram LM is trained on the English Gigaword corpus (1.6B tokens) using SRILM with modified Kneser-Ney smoothing and linear interpolation. Information about the training data for the Arabic-English and Chinese-English systems is in Table 3.[8] Feature weights are tuned using pairwise ranking optimization (Hopkins and May, 2011) on the MT04 benchmark (for both language pairs). For testing, we use MT08 and MT09 for Arabic, and MT06 and MT08 for Chinese. We use case-insensitive BLEU (Papineni et al., 2002) as evaluation metric. Approximate randomization (Noreen, 1989; Riezler and Maxwell, 2005) is used to detect statistically significant differences.

5.2 Baseline and comparison systems

As a comparison model, we implemented six features from Cherry (2008) and Bach et al. (2009)[9] and added them to the log-linear interpolation used

[7] We extract dependency parses from its output based on Collins (1999)

[8] The standard LDC corpora were used for training.

[9] Exhaustive and non-exhaustive interruption check, exhaustive and non-exhaustive interruption count, verb- and noun-dominated subtree interruption count.

Training set	N. of lines	N. of tokens
Source side of Ar-En set	4,376,320	148M
Target side of Ar-En set	4,376,320	146M
Source side of Ch-En set	2,104,652	20M
Target side of Ch-En set	2,104,652	28M

Table 3: Training data for Arabic-English and Chinese-English experiments.

by the baseline system. Since these features are binary or count-based, we cannot use them directly in rescoring. For that reason we integrated the features into the decoder and tuned the corresponding weights. The results for Chinese-English and Arabic-English translation experiments are presented in Table 1 and 2, respectively. We see that adding the cohesion constraints does not improve performance. This finding is different from, for example, Feng et al. (2010), where they get improvement for Chinese-English: however, we note that their training set is smaller than ours, and their baseline is weaker as it does not contain lexicalized distortion models.

5.3 Rescoring experiments

Rescoring with BiSLMs is performed as follows: For the test runs of the baseline system we compute the $n = 1000$ best translation hypotheses for each source sentence and extract their derivations (sequence of phrase pair applications). Each phrase pair in our implementation is associated with a unique phrase-internal alignment and target POS-sequence. We fully reconstruct word-alignment for each pair of a source sentence and its translation hypothesis. We project a precomputed source parse onto the target side and compute representations of the target sentence to be computed by a BiSLM. For each hypothesis, we take its BiSLM score and its score assigned by the baseline system and compute the final score as a weighted sum of the original baseline score and a length-normalized BiSLM score[10], where the weight λ is empirically set to 0.3:

$$\lambda \cdot \frac{score_{\text{BiSLM}}}{length_{\text{Hypothesis}}} + (1 - \lambda) \cdot score_{\text{Baseline}} \quad (3)$$

5.3.1 Chinese-English

Our main focus here is Chinese-English, since it has more instances of longer-distance reordering, at which syntax-based models are typically good.

[10]Normalization is needed to ensure comparability of scores for translation hypotheses of different lengths, since longer translation hypotheses will have lower scores.

labeling	**complete**	**unalign -adjoin**	BLEU	diff.
plain	strong	+	30.09▲	+0.53
		-	30.20▲	+0.64
	weak	+	30.11▲	+0.55
		-	30.22▲	+0.66
reduce	strong	+	29.94△	+0.40
		-	30.19▲	+0.63
	weak	+	30.09▲	+0.53
		-	30.24▲	+0.68
reduce-POS	strong	+	30.09▲	+0.53
		-	30.25▲	+0.69
	weak	+	30.05▲	+0.49
		-	30.25▲	+0.69

Table 4: Rescoring experiments for Chinese MT06+08 1000-best translation sets. Unrescored BLEU is 29.56. The column **labeling** contains information about the kind of labeling used on the target side of a BiSLM: just target words, target words with a reduction label, or target words with a reduction label and a POS of the root of the reduced subtree (Section 4.3). The column **complete** indicates whether we use a strong or weak definition of a complete subtree (Section 4.2). The column **unalign-adjoin** indicates whether we adjoin an unaligned target word to the preceding subtree (Section 4.1). Statistically significant improvements over the baseline are marked ▲ at the $p < .01$ level and △ at the $p < .05$ level. ▼ marks significant decrease at the $p < .01$ level.

SLMs by design are good at capturing longer-distance dependencies. We try out several variations of BiSLM. First, we test whether to use a strong or weak definition of a complete subtree (Section 4.2). Second, we investigate whether to adjoin unaligned target words to a preceding head (Section 4.1; **unalign-adjoin+/-**). Third, we compare several target-side labeling methods (Section 4.3): plain (just target words), reduce (**LR** or **RR**) or reduce-POS (**LR**_POS or **RR**_POS, where POS is the tag of the root of the reduced subtree). The rescoring results are presented in Table 4.

The results show statistically significant improvement over the baseline of up to 0.7 BLEU (for all of the employed BiSLM variants except one). The rescoring experiments also demonstrate the tendency of the **unalign-adjoin-** feature value to produce higher scores than **unalign-adjoin+**. But the other two distinguishing features do not have an effect on BLEU scores. As future work, we are interested in examining if these features produce the same distribution of scores when a BiSLM is fully integrated into the decoder.

labeling	complete	unalign -adjoin	BLEU	diff.
plain	strong	+	47.20	-0.02
		-	47.00▼	-0.18
	weak	+	47.22	+0.04
		-	46.98▼	-0.20
reduce	strong	+	47.15	-0.03
		-	46.99▼	-0.19
	weak	+	47.09	-0.09
		-	46.98▼	-0.20
reduce-POS	strong	+	47.15	-0.03
		-	46.98▼	-0.20
	weak	+	47.17	+0.01
		-	47.00▼	-0.18

Table 5: Rescoring experiments for Arabic MT08+09 n-best translation sets. Unrescored BLEU for is 47.18. For notation see Table 4.

5.3.2 Arabic-English

We also rescore the n-best lists for the output of the Arabic-English baseline system and results are shown in Table 5. Arabic and English are typologically very different, but the range of reordering is much smaller than for Chinese-English. We expect reordering-related models to have lesser effect on Arabic as compared to Chinese (Carpuat et al., 2010). Experimental results on Arabic-English could indicate what kind of translation aspect benefits from BiSLMs. We see that for Arabic-English, just as for the cohesion constraint, BiSLM have little effect on BLEU scores, or even decrease them. This is a weak indication that BiSLMs are better at capturing reordering aspects. As for the varying features defining different BiSLM versions, we again see little effect of the labeling type or subtree completeness definition. On the other hand, we see the opposite pattern for the **unalign-adjoin** feature, where **unalign-adjoin+** is preferred.

To gain further insight into the different effect of BiSLM on the two language pairs, we evaluated our experimental output against a reordering-sensitive metric LRscore (Birch et al., 2010). We use the version of LRscore which is an average of the inverse Kendall's Tau distance and the Hamming distance. In order to compute alignments for test sets which are needed to compute the score we concatenated the parallel text with an additional 250K lines of parallel text from the training data to ensure better generalization of the alignment algorithm (GIZA++). The LRscores of the baseline are compared to the best performing BiSLM system with respect to BLEU, for each of the language pair. The results are provided in Tables 6 and 7.

system	LRscore MT06+08
baseline	0.4736
BiSLM	0.4907

Table 6: LRscores (average inverse Kendall's Tau distance and Hamming distance) for Chinese-English baseline and BiSLM with reduce-labeling, weak completeness, unalign-adjoin-.

system	LRscore MT08+09
baseline	0.6671
BiSLM	0.6719

Table 7: LRscores for Arabic-English baseline and BiSLM with plain-labeling, weak completeness, unalign-adjoin+.

As expected, the scores for Chinese-English are much lower than for Arabic-English, which is consistent with the observation reordering is more difficult for Chinese-English. BiSLM yields larger improvements for Chinese-English suggesting that the proposed model helps addressing difficult reordering problems. While there are also small improvements for Arabic-English the they may be too small to be detectable by BLEU.

6 Conclusions

In this paper we proposed a novel way to adapt structured language models to phrase-based SMT. Our method requires minimal changes to the PB-SMT pipeline. We tried a number of variations of our model and evaluated them in rescoring experiments, resulting in statistically significant improvement for Chinese-English. The model is based on the idea of syntactic transfer (DCA; Section 2) and the positive result indicates its ability to capture syntactic patterns across languages. For Arabic-English, we did not observe any improvements, suggesting that our models indeed mainly improve reordering aspects. Improvements in rescoring are a positive indication that our model may be a strong feature during decoding. As future work, we will fully integrate our model into a PBSMT decoder and evaluate it on other language pairs with different reordering distributions.

Acknowledgments

We thank the reviewers for their useful comments. This research was funded in part by the Netherlands Organization for Scientific Research (NWO) under project numbers 639.022.213 and 612.001.218.

References

Nguyen Bach, Stephan Vogel, and Colin Cherry. 2009. Cohesive constraints in a beam search phrase-based decoder. In *Proceedings of the 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 1–4. Association for Computational Linguistics.

Alexandra Birch, Miles Osborne, and Phil Blunsom. 2010. Metrics for mt evaluation: evaluating reordering. *Machine Translation*, 24(1):15–26.

Marine Carpuat, Yuval Marton, and Nizar Habash. 2010. Improving Arabic-to-English statistical machine translation by reordering post-verbal subjects for alignment. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 178–183. Association for Computational Linguistics.

Pi-Chuan Chang, Huihsin Tseng, Dan Jurafsky, and Christopher D. Manning. 2009. Discriminative reordering with chinese grammatical relations features. In *Proceedings of the Third Workshop on Syntax and Structure in Statistical Translation*, pages 51–59. Association for Computational Linguistics.

Ciprian Chelba and Frederick Jelinek. 2000. Structured language modeling. *Computer Speech and Language*, 14(4):283–332.

Colin Cherry. 2008. Cohesive phrase-based decoding for statistical machine translation. In *Proceedings of Association for Computational Linguistics*, pages 72–80.

Michael Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

Josep M. Crego and François Yvon. 2010. Improving reordering with linguistically informed bilingual n-grams. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 197–205. Association for Computational Linguistics.

Minwei Feng, Arne Mauser, and Hermann Ney. 2010. A source-side decoding sequence model for statistical machine translation. In *Conference of the Association for Machine Translation in the Americas, Denver, Colorado, USA*.

Heidi J. Fox. 2002. Phrasal cohesion and statistical machine translation. In *Proceedings the Conference on Empirical Methods in Natural Language Processing*, pages 304–311. Association for Computational Linguistics.

Ekaterina Garmash and Christof Monz. 2014. Dependency-based bilingual language models for reordering in statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1689–1700, Doha, Qatar, October. Association for Computational Linguistics.

Niyu Ge. 2010. A direct syntax-driven reordering model for phrase-based machine translation. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 849–857. Association for Computational Linguistics.

Spence Green and Christopher D. Manning. 2010. Better arabic parsing: Baselines, evaluations, and analysis. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 394–402. Association for Computational Linguistics.

Jiri Havelka. 2007. Beyond projectivity: Multilingual evaluation of constraints and measures on non-projective structures. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 608–615. Association for Computational Linguistics.

Mark Hopkins and Jonathan May. 2011. Tuning as ranking. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1352–1362. Association for Computational Linguistics.

Rebecca Hwa, Philip Resnik, Amy Weinberg, and Okan Kolak. 2002. Evaluating translational correspondence using annotation projection. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 392–399. Association for Computational Linguistics.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 48–54. Association for Computational Linguistics.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics on Interactive Poster and Demonstration Sessions*, pages 177–180. Association for Computational Linguistics.

Marco Kuhlmann and Joakim Nivre. 2006. Mildly non-projective dependency structures. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, pages 507–514, Sydney, Australia, July. Association for Computational Linguistics.

Uri Lerner and Slav Petrov. 2013. Source-side classifier preordering for machine translation. In *Proceedings of the Empirical Methods in Natural Language Processing*.

Tahira Naseem, Regina Barzilay, and Amir Globerson. 2012. Selective sharing for multilingual dependency

parsing. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 629–637. Association for Computational Linguistics.

Jan Niehues, Teresa Herrmann, Stephan Vogel, and Alex Waibel. 2011. Wider context by using bilingual language models in machine translation. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 198–206. Association for Computational Linguistics.

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

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

Timothy Osborne. 2008. Major constituents and two dependency grammar constraints on sharing in coordination. *Linguistics*, 46(6):1109–1165.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318. Association for Computational Linguistics.

Matt Post and Daniel Gildea. 2008. Parsers as language models for statistical machine translation. In *Proceedings of the Eighth Conference of the Association for Machine Translation in the Americas*, pages 172–181. Citeseer.

Chris Quirk and Arul Menezes. 2006. Dependency treelet translation: The convergence of statistical and example-based machine translation? *Machine Translation*, 20:43–65, March.

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

Rico Sennrich. 2015. Modelling and optimizing on syntactic n-grams for statistical machine translation. *Transactions of the Association for Computational Linguistics*, 3:169–182.

Libin Shen, Jinxi Xu, and Ralph M. Weischedel. 2008. A new string-to-dependency machine translation algorithm with a target dependency language model. In *Proceedings of the Association for Computational Linguistics*, pages 577–585.

Andreas Stolcke, Jing Zheng, Wen Wang, and Victor Abrash. 2011. Srilm at sixteen: Update and outlook. In *Proceedings of IEEE Automatic Speech Recognition and Understanding Workshop*, page 5.

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

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

Bing Xiang, Niyu Ge, and Abraham Ittycheriah. 2011. Improving reordering for statistical machine translation with smoothed priors and syntactic features. In *Proceedings of the Fifth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 61–69. Association for Computational Linguistics.

Kenji Yamada and Kevin Knight. 2001. A syntax-based statistical translation model. In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, pages 523–530. Association for Computational Linguistics.

Heng Yu, Haitao Mi, Liang Huang, and Qun Liu. 2014. A structured language model for incremental tree-to-string translation. *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics*, pages 1133–1143.

Andreas Zollmann and Ashish Venugopal. 2006. Syntax augmented machine translation via chart parsing. In *Proceedings of the Workshop on Statistical Machine Translation*, pages 138–141. Association for Computational Linguistics.

Compact, Efficient and Unlimited Capacity: Language Modeling with Compressed Suffix Trees

Ehsan Shareghi,[♭] **Matthias Petri,**[♮] **Gholamreza Haffari**[♭] **and Trevor Cohn**[♮]
[♭] Faculty of Information Technology, Monash University
[♮] Computing and Information Systems, The University of Melbourne
first.last@{monash.edu,unimelb.edu.au}

Abstract

Efficient methods for storing and querying language models are critical for scaling to large corpora and high Markov orders. In this paper we propose methods for modeling extremely large corpora without imposing a Markov condition. At its core, our approach uses a succinct index – a *compressed suffix tree* – which provides near optimal compression while supporting efficient search. We present algorithms for on-the-fly computation of probabilities under a Kneser-Ney language model. Our technique is *exact* and although slower than leading LM toolkits, it shows promising scaling properties, which we demonstrate through ∞-order modeling over the full Wikipedia collection.

1 Introduction

Language models (LMs) are critical components in many modern NLP systems, including machine translation (Koehn, 2010) and automatic speech recognition (Rabiner and Juang, 1993). The most widely used LMs are $m$gram models (Chen and Goodman, 1996), based on explicit storage of $m$grams and their counts, which have proved highly accurate when trained on large datasets. To be useful, LMs need to be not only accurate but also fast and compact.

Depending on the order and the training corpus size, a typical $m$gram LM may contain as many as several hundred billions of $m$grams (Brants et al., 2007), raising challenges of efficient storage and retrieval. As always, there is a trade-off between accuracy, space, and time, with recent papers considering small but approximate *lossy* LMs (Chazelle et al., 2004; Talbot and Osborne, 2007; Guthrie and Hepple, 2010), or *loss-less* LMs backed by tries (Stolcke et al., 2011), or related compressed structures (Germann et al., 2009; Heafield, 2011; Pauls and Klein, 2011; Sorensen and Allauzen, 2011; Watanabe et al., 2009). However, none of these approaches scale well to very high-order m or very large corpora, due to their high memory and time requirements. An important exception is Kennington et al. (2012), who also propose a language model based on a suffix tree which scales well with m but poorly with the corpus size (requiring memory of about $20\times$ the training corpus).

In contrast, we[1] make use of recent advances in *compressed suffix trees* (CSTs) (Sadakane, 2007) to build compact indices with much more modest memory requirements ($\approx$ the size of the corpus). We present methods for extracting frequency and unique context count statistics for $m$gram queries from CSTs, and two algorithms for computing Kneser-Ney LM probabilities on the fly using these statistics. The first method uses two CSTs (over the corpus and the reversed corpus), which allow for efficient computation of the number of unique contexts to the left and right of an $m$gram, but is inefficient in several ways, most notably when computing the number of unique contexts to both sides. Our second method addresses this problem using a single CST backed by a wavelet tree based FM-index (Ferragina et al., 2007), which results in better time complexity and considerably faster runtime performance.

Our experiments show that our method is practical for large-scale language modelling, although querying is substantially slower than a SRILM benchmark. However our technique scales much more gracefully with Markov order m, allowing unbounded 'non-Markov' application, and enables training on large corpora as we demonstrate on the complete Wikipedia dump. Overall this paper illustrates the vast potential succinct indexes have

[1] For the implementation see: https://github.com/eehsan/lm-sdsl.

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 2409–2418,
Lisbon, Portugal, 17-21 September 2015.

for language modelling and other 'big data' problems in language processing.

2 Background

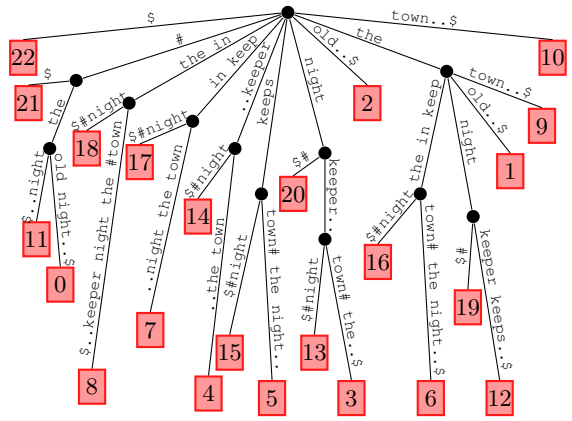
Suffix Arrays and Suffix Trees Let $\mathcal{T}$ be a string of size n drawn from an alphabet Σ of size σ. Let $\mathcal{T}[i..n-1]$ be a *suffix* of $\mathcal{T}$. The *suffix tree* (Weiner, 1973) of $\mathcal{T}$ is the compact labeled tree of $n+1$ leaves where the root to leaf paths correspond to all suffixes of $\mathcal{T}\$$, where \$ is a terminating symbol not in Σ. The *path-label* of each node v corresponds to the concatenation of edge labels from the root node to v. The *node depth* of v corresponds to the number of ancestors in the tree, whereas the *string depth* corresponds to the length of the path-label. Searching for a pattern α of size m in $\mathcal{T}$ translates to finding the *locus* node v closest to the root such that α is a prefix of the path-label of v in $O(m)$ time. We refer to this approach as *forward search*. Figure 1a shows a suffix tree over a sample text. A suffix tree requires $O(n)$ space and can be constructed in $O(n)$ time (Ukkonen, 1995). The children of each node in the suffix tree are lexicographically ordered by their edge labels. The i-th smallest suffix in $\mathcal{T}$ corresponds to the path-label of the i-th leaf. The starting position of the suffix can be associated its corresponding leaf in the tree as shown in Figure 1a. All occurrences of α in $\mathcal{T}$ can be retrieved by visiting all leaves in the subtree of the locus of α. For example, pattern "the night" occurs at positions 12 and 19 in the sample text. We further refer the number of children of a node v as its *degree* and the number of leaves in the subtree rooted at v as the *size* of v.

The *suffix array* (Manber and Myers, 1993) of $\mathcal{T}$ is an array $SA[0 \dots n-1]$ such that $SA[i]$ corresponds to the starting position of the i-th smallest suffix in $\mathcal{T}$ or the i-th leaf in the suffix tree of $\mathcal{T}$. The suffix array requires $n \log n$ bits of space and can also be constructed in $O(n)$ time (Kärkkäinen et al., 2006). Using only the suffix array and the text, pattern search can be performed using binary search in $O(m \log n)$ time. For example, the pattern "the night" is found by performing binary search using SA and $\mathcal{T}$ to determine $SA[18, 19]$, the interval in SA corresponding the the suffixes in $\mathcal{T}$ prefixed by the pattern. In practice, suffix arrays use $4-8n$ bytes of space whereas the most efficient suffix tree implementations require at least $20n$ bytes of space (Kurtz, 1999) which are both much larger than $\mathcal{T}$ and prohibit the use of these structures for all but small data sets.

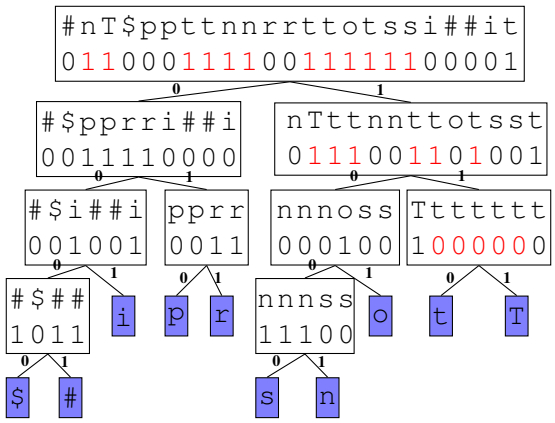
Compressed Suffix Structures Reducing the space usage of suffix based index structure has recently become an active area of research. The space usage of a suffix array can be reduced significantly by utilizing the compressibility of text combined with succinct data structures. A *succinct* data structure provides the same functionality as an equivalent uncompressed data structure, but requires only space equivalent to the information-theoretic lower bound of the underlying data. For simplicity, we focus on the *FM-Index* which emulates the functionality of a suffix array over $\mathcal{T}$ using $nH_k(\mathcal{T}) + o(n \log \sigma)$ bits of space where H_k refers to the k-th order entropy of the text (Ferragina et al., 2007). In practice, the FM-Index of $\mathcal{T}$ uses roughly space equivalent to the compressed representation of $\mathcal{T}$ using a standard compressor such as `bzip2`. For a more comprehensive overview on succinct text indexes, see the excellent survey of Ferragina et al. (2008).

The FM-Index relies on the duality between the suffix array and the BWT (Burrows and Wheeler, 1994), a permutation of the text such that $\mathcal{T}^{bwt}[i] = \mathcal{T}[SA[i]-1]$ (see Figure 1). Searching for a pattern using the FM-Index is performed in reverse order by performing $\text{RANK}(\mathcal{T}^{bwt}, i, c)$ operations $O(m)$ times. Here, $\text{RANK}(\mathcal{T}^{bwt}, i, c)$ counts the number of times symbol c occurs in $\mathcal{T}^{bwt}[0 \dots i-1]$. This process is usually referred to as *backward search*. Let $SA[l_i, r_i]$ be the interval corresponding to the suffixes in $\mathcal{T}$ matching $\alpha[i \dots m-1]$. By definition of the BWT, $\mathcal{T}^{bwt}[l_i, r_i]$ corresponds to the symbols in $\mathcal{T}$ preceding $\alpha[i \dots m-1]$ in $\mathcal{T}$. Due to the lexicographical ordering of all suffixes in SA, the interval $SA[l_{i-1}, r_{i-1}]$ corresponding to all occurrences of $\alpha[i-1 \dots m-1]$ can be determined by computing the rank of all occurrences of $c = \alpha[i-1]$ in $\mathcal{T}^{bwt}[l_i, r_i]$. Thus, we compute $\text{RANK}(\mathcal{T}^{bwt}, l_i, c)$, the number of times symbol c occurs before l_i and $\text{RANK}(\mathcal{T}^{bwt}, r_i+1, c)$, the number of occurrences of c in $\mathcal{T}^{bwt}[0, r_i]$. To determine $SA[l_{i-1}, r_{i-1}]$, we additionally store the starting positions C_s of all suffixes for each symbol s in Σ at a negligible cost of $\sigma \log n$ bits. Thus, the new interval is computed as $l_{i-1} = C_c + \text{RANK}(\mathcal{T}^{bwt}, l_i, c)$ and $r_{i-1} = C_c + \text{RANK}(\mathcal{T}^{bwt}, r_i+1, c)$.

The time and space complexity of the FM-index thus depends on the cost of storing and pre-



(a) Word-based Suffix Tree.



(b) Wavelet tree and RANK($\mathcal{T}^{bwt}$, 17, 't') = 5.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
$\mathcal{T}[SA[i]]$	$	#	#	#	i	i	p	p	r	r	s	s	n	n	n	o	t	t	t	t	t	t	T
SA	22	21	11	0	18	8	17	7	14	4	15	5	20	13	3	2	16	6	19	12	1	9	10
$\mathcal{T}^{bwt}$	#	n	T	$	p	p	t	t	n	n	r	r	t	t	o	t	s	s	i	#	#	i	t

Figure 1: Data structures for the sample text $\mathcal{T}$="#the old night keeper keeps the keep in the town# the night keeper keeps the keep in the night#$" with alphabet Σ={the, old, night, keeper, keeps, keep, in, town, #} and code words *$=0000*, *#=0001*, *i=in=001*, *p=keep=010*, *r=keeper=011*, *s=keeps=1000*, *o=old=101*, *t=the=110*, *n=night=1001* and *T=town=111*.

processing $\mathcal{T}^{bwt}$ to answer RANK efficiently. A *wavelet tree* can be used to answer RANK over $\mathcal{T}^{bwt}$ in $O(\log \sigma)$ time. The wavelet tree reduces RANK over an alphabet Σ into multiple RANK operations over a binary alphabet which can be answered in $O(1)$ time and $o(n)$ bits extra space by periodically storing absolute and relative RANK counts (Munro, 1996). The alphabet is reduced by recursively splitting symbols based on their code words into subgroups to form a binary tree as shown in Figure 1b for $\mathcal{T}^{bwt}$. To answer RANK($\mathcal{T}^{bwt}, i, c$), the tree is traversed based on the code word of c, performing binary RANK at each level. For example, RANK($\mathcal{T}^{bwt}$, 17, 't') translates to performing RANK($WT_{root}, 17, 1$) = 12 on the top level of the wavelet tree, as `t=the=110`. We recurse to the right subtree of the root node and compute RANK($WT_1, 12, 1$) as there were 12 ones in the root node and the next bit in the codeword of 'the' is also one. This process continues until the correct leaf node is reached to answer RANK($\mathcal{T}^{bwt}$, 17, 't') = 5 in $O(\log \sigma)$ time. The space usage of a regular wavelet tree is $n \log \sigma + o(n \log \sigma)$ bits which roughly matches the size of the text.[2] If locations of matches are required, additional space is needed to access $SA[i]$ or the inverse suffix array $SA^{-1}[SA[i]] = i$. In the simplest scheme, both values are periodically sampled using a given sample rate *SAS* (e.g. 32) such that $SA[i] \mod SAS = 0$. Then, for any $SA[i]$ or $SA^{-1}[i]$, at most $O(SAS)$ RANK operations on $\mathcal{T}^{bwt}$ are required to access the value. Different sample rates, bitvector implementations and wavelet tree types result in a wide variety of time-space tradeoffs which can be explored in practice (Gog et al., 2014).

In the same way the FM-index emulates the functionality of the suffix array in little space, *compressed suffix trees* (CST) provide the functionality of suffix trees while requiring significantly less space than their uncompressed counterparts (Sadakane, 2007). A CST uses a compressed suffix array (CSA) such as the FM-Index but stores additional information to represent the shape of the suffix tree as well as information about path-labels. Again a variety of different storage schemes exist, however for simplicity we focus on the CST of Ohlebusch et al. (2010) which we use in our experiments. Here, the shape of the tree is stored using a balanced-parenthesis (BP) sequence which for a tree of p nodes requires $\approx 2p$

[2] However, if code-words for each symbol are chosen based on their Huffman-codes the size of the wavelet tree reduces to $nH_0(\mathcal{T})(1 + o(1))$ bits which can be further be reduced to to $nH_k(\mathcal{T}) + o(n \log \sigma)$ bits by using entropy compressed bitvectors.

bits. Using little extra space and advanced bit-operations, the BP-sequence can be used to perform operations such as string-depth(v), parent(v) or accessing the i-th leaf can be answered in constant time. To support more advanced operations such as accessing path-labels, the underlying CSA or a compressed version of the LCP array are required which can be more expensive.[3] In practice, a CST requires roughly $4 - 6n$ bits in addition to the cost of storing the CSA. For a more extensive overview of CSTs see Russo et al. (2011).

Kneser Ney Language Modelling Recall our problem of efficient $m$gram language modeling backed by a corpus encoded in a succinct index. Although our method is generally applicable to many LM variants, we focus on the Kneser-Ney LM (Kneser and Ney, 1995), specifically the interpolated variant described in Chen and Goodman (1996), which has been shown to outperform other $n$gram LMs and has become the de-facto standard.

Interpolated Kneser-Ney describes the conditional probability of a word w_i conditioned on the context of $m-1$ preceding words, w_{i-m+1}^{i-1}, as

$$P(w_i|w_{i-m+1}^{i-1}) = \frac{\max\left[c(w_{i-m+1}^{i}) - D_m, 0\right]}{c(w_{i-m+1}^{i-1})} + \frac{D_m N^{1+}(w_{i-m-1}^{i-1}\ \bullet)}{c(w_{i-m+1}^{i-1})} \bar{P}(w_i|w_{i-m+2}^{i-1}), \quad (1)$$

where lower-order smoothed probabilities are defined recursively (for $1 < k < m$) as

$$\bar{P}(w_i|w_{i-k+1}^{i-1}) = \frac{\max\left[N^{1+}(\bullet\ w_{i-k+1}^{i}) - D_k, 0\right]}{N^{1+}(\bullet\ w_{i-k+1}^{i-1}\ \bullet)} + \frac{D_k N^{1+}(w_{i-k+1}^{i-1}\ \bullet)}{N^{1+}(\bullet\ w_{i-k+1}^{i-1}\ \bullet)} \bar{P}(w_i|w_{i-k+2}^{i-1})\,. \quad (2)$$

In the above formula, D_k is the $k$gram-specific discount parameter, and the *occurrence count* $N^{1+}(\alpha\ \bullet) = |\{w : c(\alpha w) > 0\}|$ is the number of observed word types following the pattern α; the occurrence counts $N^{1+}(\bullet\ \alpha)$ and $N^{1+}(\bullet\ \alpha\ \bullet)$ are defined accordingly. The recursion stops at unigram level where the unigram probabilities are defined as $\bar{P}(w_i) = N^{1+}(\bullet\ w_i)/N^{1+}(\bullet\ \bullet)$.[4]

[3] See *Supplementary Materials* Table 1 for an overview of the complexities of the functionality of the CST that is used in our experiments.

[4] Modified Kneser-Ney, proposed by Chen and Goodman (1996), typically outperforms interpolated Kneser-Ney through its use of context-specific discount parameters. The implementation of this with our data structures is straightforward in principle, but brings a few added complexities in terms of dynamic computing other types of occurrence counts, which we leave for future work.

3 Using CSTs for KN Computation

The key requirements for computing probability under a Kneser-Ney language model are two types of counts: raw frequencies of $m$grams and occurrence counts, quantifying how many different contexts the $m$gram has occurred in. Figure 2 (right) illustrates the requisite counts for calculating the probability of an example 4-gram. In electing to store the corpus directly in a suffix tree, we need to provide mechanisms for computing these counts based on queries into the suffix tree.

The raw frequency counts are the simplest to compute. First we identify the locus node v in the suffix tree for the query $m$gram; the frequency corresponds to the node's *size*, an $O(1)$ operation which returns the number of leaves below v. To illustrate, consider searching for c(*the night*) in Figure 1a, which matches a node with two leaves (labelled 19 and 12), and thus $c = 2$.

More problematic are the occurrence counts, which come in several flavours: right contexts, $N^{1+}(\alpha\ \bullet)$, left contexts, $N^{1+}(\bullet\ \alpha)$, and contexts to both sides of the pattern, $N^{1+}(\bullet\ \alpha\ \bullet)$. The first of these can be handled easily, as

$$N^{1+}(\alpha\ \bullet) = \begin{cases} \text{degree}(v), & \text{if } \alpha = \text{label}(v) \\ 1, & \text{otherwise} \end{cases}$$

where v is the node matching α, and label(v) denotes the *path-label* of v.[5] For example, *keep in* has two child nodes in Figure 1a, and thus there are two unique contexts in which it can occur, N^{1+}(*keep in* $\bullet$) $= 2$, while *the keep* partially matches an edge in the forward suffix tree in Figure 1a as it can only be followed by *in*, N^{1+}(*the keep* $\bullet$) $= 1$. A similar line of reasoning applies to computing $N^{1+}(\bullet\ \alpha)$. Assuming we also have a second suffix tree representing the *reversed corpus*, we first identify the reversed pattern (e.g., *in keep*$_R$) and then use above method to compute the occurrence count (denoted hereafter N1P(t, v, α)[6], where t is the CST.).

[5] See the *Supplementary Materials* for the explicit algorithm, but note there are some corner cases involving sentinels # and \$, which must be excluded when computing occurrence counts. Such tests have been omitted from the presentation for clarity.

[6] In the presented algorithms, we overload the pattern argument in function calls for readability, and use $\bullet$ to denote the query context.

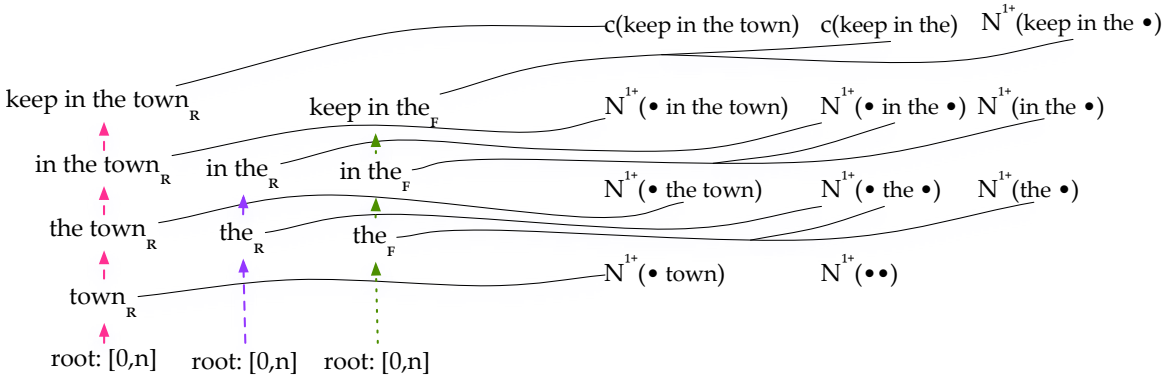


Figure 2: Counts required for computing $P(town|keep\ in\ the)$ (right) and the suffix tree nodes required for computing each value (left). The two left-most columns correspond to $v_{\mathrm{R}}^{\mathrm{all}}$ and v_{R} and are updated using *forward-search* in the reverse CST, while the righter-most column correspond to v_{F} and is updated using *backward-search* in the forward CST. See Algorithm 2 for details.

The final component of the Kneser-Ney LM computation is $N^{1+}(\bullet\ \alpha\ \bullet)$, the number of unique contexts considering symbols on both sides of the pattern. Unfortunately this does not map to a simple suffix tree operation, but instead requires enumeration, $N^{1+}(\bullet\ \alpha\ \bullet) = \sum_{s \in F(\alpha)} N^{1+}(\bullet\ \alpha s)$, where $F(\alpha)$ is the set of symbols that can follow α. Algorithm 1 shows how this is computed, with lines 7 and 8 enumerating $s \in F(\alpha)$ using the *edge* labels of the children of v. For each symbol, line 9 searches for an extended pattern incorporating the new symbol s in the reverse CSA (part of the reverse CST), by refining the existing match v_{R} using a single backward search operation after which we can compute $N^{1+}(\bullet\ \alpha s)$.[7] Line 5 deals with the special case where the pattern does not match a complete edge, in which case there is only only unique right context and therefore $N^{1+}(\bullet\ \alpha\ \bullet) = N^{1+}(\bullet\ \alpha)$.

N1P and N1PFRONTBACK can compute the requisite occurence counts for $m$gram language modelling, however at considerable cost in terms of space and time. The need for twin reverse and forward CSTs incurs a significant storage overhead, as well as the search time to match the pattern in both CSTs. We show in Section 5 how we can avoid the need for the reversed suffix tree, giving rise to lower memory requirements and faster runtime. Beyond the need for twin suffix trees, the highest time complexity calls are *string-depth*, *edge* and *backward-search*. Calling *string-depth* is constant time for internal nodes, but $O(SAS \log \sigma)$ for leaf nodes; fortunately we can avoid this call for leaves, which by definition extend to the end of the corpus and consequently extend further than our pattern.[8] The costly calls to *edge* and *backward-search* however cannot be avoided. This leads to an overall time complexity of $O(1)$ for N1P and $O(F(\alpha) \times SAS \times \log \sigma)$ for N1PFRONTBACK, where $F(\alpha)$ is the number of following symbols and *SAS* is the suffix array value sample rate described in Section 2.

Algorithm 1 Two-sided occ., $N^{1+}(\bullet\ \alpha\ \bullet)$

Precondition: v_{F} in forward CST t_{F} matches α
Precondition: v_{R} in reverse CST t_{R} matches α

```
1:  function N1PFRONTBACK(t_F, v_F, t_R, v_R, α)
2:      o ← 0
3:      d ← string-depth(v_F)
4:      if d > |α| then
5:          o ← N1P(t_R, v_R, • α)
6:      else
7:          for u_F ← children(v_F) do
8:              s ← edge(u_F, d + 1)
9:              u_R ← back-search(v_R, s)
10:             o ← o + N1P(t_R, u_R, • αs)
11:     return o
```

4 Dual CST Algorithm

The methods above for computing the frequency and occurrence counts provide the ingredients necessary for computing $m$gram language model probabilities. This leaves the algorithmic problem

[7] Backward search in the reverse tree corresponds to searching for the reversed pattern appended with one symbol.

[8] We assume search patterns do not extend beyond a single sentence, and thus will always be shorter than the edge labels.

Algorithm 2 KN probability $P\big(w_k|w_{k-(m-1)}^{k-1}\big)$

```
1: function ProbKneserNey(t_F, t_R, w, m)
2:     v_F ← root(t_F)                    ▷ match for suffix of w_{k-(m-1)}^{k-1}
3:     v_R ← root(t_R)                    ▷ match for suffix of w_{k-(m-1)}^{k-1}
4:     v_R^all ← root(t_R)                ▷ match for suffix of w_{k-(m-1)}^{k}
5:     p ← 1
6:     for i ← 1 to m do
7:         v_R^all ← forw-search(v_R^all, w_{k-i+1})
8:         if i > 1 then
9:             v_F ← back-search(v_F, w_{k-i+1})
10:            if i < m then
11:                v_R ← forw-search(v_R, w_{k-i+1})
12:        D_i ← lookup discount for igram
13:        if i = m then
14:            c ← size(v_R^all)
15:            d ← size(v_F)
16:        else
17:            c ← N1P(t_R, v_R^all, • w_{k-i+1}^{k})
18:            d ← N1PFrontBack(t_F, v_F, t_R, v_R, • w_{k-i+1}^{k-1} •)
19:        if i > 1 then
20:            if v_F is valid then
21:                q ← N1P(t_F, v_F, w_{k-i+1}^{k-1} •)
22:                p ← 1/d (max(c − D_i, 0) + D_i q p)
23:        else if i = 1 then
24:            p ← c/N^{1+}(• •)
25:    return p
```

Algorithm 3 Precompute KN discounts

```
1: function PrecomputeDiscounts(t_R, m)
2:     c_{k,f} ← 0   ∀k ∈ [1, m], f ∈ [1, 2]
3:     N^1_{k,g} ← 0   ∀k ∈ [1, m], g ∈ [1, 2]
4:     N^{1+}(• •) ← 0
5:     for v_R ← descendents(root(t_R)) do        ▷ depth-first
6:         d_P ← string-depth(parent(v_R))
7:         d ← string-depth(v_R)
8:         for k ← d_P + 1 to min(d, d_P + m) do
9:             s ← edge(v_R, k)
10:            if s is the end of sentence sentinel then
11:                skip all children of v_R
12:            else
13:                if k = 2 then
14:                    N^{1+}(• •) ← N^{1+}(• •) + 1
15:                f ← size(v_R)
16:                if 1 ≤ f ≤ 2 then
17:                    c_{k,f} ← c_{k,f} + 1
18:                if k < d then
19:                    g ← 1
20:                else
21:                    g ← degree(v_R)
22:                if 1 ≤ g ≤ 2 then
23:                    N^1_{k,g} ← N^1_{k,g} + 1
24:    return c, N^1, N^{1+}(• •)
```

of efficiently ordering the search operations in forward and reverse CST structures.

This paper considers an interpolated LM formulation, in which probabilities from higher order contexts are interpolated with lower order estimates. This iterative process is apparent in Figure 2 (right) which shows the quantities required for probability scoring for an example $m$gram. Equivalently, the iteration can be considered in reverse, starting from unigram estimates and successively growing to large $m$grams, in each stage adding a single new symbol to left of the pattern. This suits incremental search in a CST in which search bounds are iteratively refined, which has a substantially lower time complexity compared to searching over the full index in each step.

Algorithm 2 presents an outline of the approach. This uses a forward CST, t_{F}, and a reverse CST, t_{R}, with three CST nodes (lines 2–4) tracking the match progress for the full $i$gram ($v_{\text{R}}^{\text{all}}$) and the $(i-1)$gram context ($v_{\text{F}}, v_{\text{R}}$), $i = 1 \ldots m$. The need to maintain three concurrent searches arises from the calls to *size*, $N^{1+}(\bullet\ \alpha)$, $N^{1+}(\alpha\ \bullet)$ and $N^{1+}(\bullet\ \alpha\ \bullet)$ (lines 14, 15; 17; 21; and 18, respectively). These calls impose conditions on the direction of the suffix tree, e.g., such that the edge labels and node degree can be used to compute the number of left or right contexts in which a pattern appears. The matching process is illustrated in Figure 2 where the three search nodes are shown on the left, considered bottom to top, and their corresponding count operations are shown to the right. The $N^{1+}(\bullet\ \alpha)$ calls require a match in the reverse CST (left-most column, $v_{\text{R}}^{\text{all}}$), while the $N^{1+}(\alpha\ \bullet)$ require a match in the forward CST (right-most column, v_{F}, matching the $(i-1)$gram context). The $N^{1+}(\bullet\ \alpha\ \bullet)$ computation reuses the forward match while also requiring a match for the $(i-1)$gram context in the reversed CST, as tracked by the middle column (v_{R}). Because of the mix of forward and reverse CSTs, coupled with search patterns that are revealed right-to-left, incremental search in each of the CSTs needs to be handled differently (lines 7–11). In the forward CST, we perform *backward search* to extend the search pattern to the left, which can be computed very efficiently from the BWT in the CSA.[9] Conversely in the reverse CST, we must use *forward search* as we are effectively extending the reversed pattern to the right; this operation is considerably more costly.

The discounts D on line 12 of Algorithm 2 and $N^{1+}(\bullet\ \bullet)$ (a special case of line 18) are precomputed directly from the CSTs thus avoiding several costly computations at runtime. The precomputa-

[9] See *Supplementary Materials* Table 1 for the time complexities of this and other CSA and CST methods.

tion algorithm is provided in Algorithm 3 which operates by traversing the nodes of the reverse CST and at each stage computing the number of $m$grams that occur 1–2 times (used for computing D_m in eq. 1), or with $N^{1+}(\bullet\ \alpha) \in [1-2]$ (used for computing D_k in eq. 2), for various lengths of $m$grams. These quantities are used to compute the discount parameters, which are then stored for later use in inference.[10] Note that the PRECOMPUTEDISCOUNTS algorithm can be slow, although it is significantly faster if we remove the *edge* calls and simply include in our counts all $m$grams finishing a sentence or spanning more than one sentence. This has a negligible (often beneficial) effect on perplexity.

5 Improved Single CST Approach

The above dual CST algorithm provides an elegant means of computing LM probabilities of arbitrary order and with a limited space complexity ($O(n)$, or roughly n in practice). However the time complexity is problematic, stemming from the expensive method for computing N1PFRONTBACK and repeated searches over the CST, particularly *forward-search*. Now we outline a method for speeding up the algorithm by doing away with the reverse CST. Instead the critical counts, $N^{1+}(\bullet\ \alpha)$ and $N^{1+}(\bullet\ \alpha\ \bullet)$ are computed directly from a single forward CST. This confers the benefit of using only backward search and avoiding redundant searches for the same pattern (cf. lines 9 and 11 in Algorithm 2).

The full algorithm for computing LM probabilities is given in Algorithm 4, however for space reasons we will not describe this in detail. Instead we will focus on the method's most critical component, the algorithm for computing $N^{1+}(\bullet\ \alpha\ \bullet)$ from the forward CST, presented in Algorithm 5. The key difference from Algorithm 1 is the loop from lines 6–9, which uses the *interval-symbols* (Schnattinger et al., 2010) method. This method assumes a *wavelet tree* representation of the *SA* component of the CST, an efficient encoding of the BWT as describes in section 2. The *interval-symbols* method uses RANK operations to efficiently identify for a given pattern the set of preceding symbols $P(\alpha)$ and the ranges $SA[l_s, r_s]$ corresponding to the patterns $s\alpha$ for all $s \in P(\alpha)$

Algorithm 4 KN probability $P(w_k|w_{k-(m-1)}^{k-1})$ using a single CST

```
1: function ProbKneserNey1(t_F, w, m)
2:     v_F ← root(t_F)                          ▷ match for context w_{k-i}^{k-1}
3:     v_F^all ← root(t_F)                      ▷ match for w_{k-i}^{k}
4:     p ← 1
5:     for i ← 1 to m do
6:         v_F^all ← back-search([lb(v_F^all), rb(v_F^all)], w_{k-i+1})
7:         if i > 1 then
8:             v_F ← back-search([lb(v_F), rb(v_F)], w_{k-i+1})
9:         D_i ← discount parameter for igram
10:        if i = m then
11:            c ← size(v_F^all)
12:            d ← size(v_F)
13:        else
14:            c ← N1pBack1(t_F, v_F^all, • w_{k-i+1}^{k-1})
15:            d ← N1pFrontBack1(t_F, v_F, • w_{k-i+1}^{k-1} •)
16:        if i > 1 then
17:            if v_F is valid then
18:                q ← N1p(t_F, v_F, w_{k-i+1}^{k-1} •)
19:                p ← 1/d (max(c − D_i, 0) + D_i q p)
20:        else
21:            p ← c/N^{1+}(• •)
22:    return p
```

Algorithm 5 $N^{1+}(\bullet\ \alpha\ \bullet)$, using forward CST

Precondition: v_F in forward CST t_F matches α

```
1: function N1pFrontBack1(t_F, v_F, α)
2:     o ← 0
3:     if string-depth(v_F) > |α| then
4:         o ← N1pBack1(t_F, v_F, • α)
5:     else
6:         for ⟨l, r, s⟩ ← int-syms(t_F, [lb(v_F), rb(v_F)]) do
7:             l' ← C_s + l
8:             r' ← C_s + r
9:             o ← o + N1p(t_F, node(l', r'), sα •)
10:    return o
```

by visiting all leaves of the wavelet tree of symbols occurring in $\mathcal{T}^{bwt}[l, r]$ (corresponding to α) in $O(|P(\alpha)| \log \sigma)$ time (lines 6-8). These ranges $SA[l', r']$ can be used to find the corresponding suffix tree node for each $s\alpha$ in $O(1)$ time. To illustrate, consider the pattern α = "night" in Figure 1a. From $\mathcal{T}^{bwt}$ we can see that this is preceeded by s ="old" (1^{st} occurrence in $\mathcal{T}^{bwt}$) and s ="the" (3^{rd} and 4^{th}); from which we can compute the suffix tree nodes, namely $[15, 15]$ and $[16 + (3-1), 16 + (4-1)] = [18, 19]$ for "old" and "the" respectively.[11]

N1PBACK1 is computed in a similar way, using the *interval-symbols* method to compute the number of unique preceeding symbols (see *Supplementary Materials*, Algorithm 7). Overall the time complexity of inference for both N1PBACK1

[10] Discounts are computed up to a limit on $m$gram size, here set to 10. The highest order values are used for computing the discount of $m$grams above the limit at runtime.

[11] Using the offsets into the *SA* for each symbol, $C_{\text{old}} = 15$ and $C_{\text{the}} = 16$, while -1 adjusts for counting from 1.

Language	Size(MiB)	Tokens(M)	Word Types	Sentences(K)
BG	36.11	8.53	114930	329
CS	53.48	12.25	174592	535
DE	171.80	44.07	399354	1785
EN	179.15	49.32	124233	1815
FI	145.32	32.85	721389	1737
FR	197.68	53.82	147058	1792
HU	52.53	12.02	318882	527
IT	186.67	48.08	178259	1703
PT	187.20	49.03	183633	1737
Wikipedia	8637	9057	196	87835

Table 1: Dataset statistics, showing total uncompressed size; and tokens, types and sentence counts for the training partition. For Wikipedia the Word Types, and Tokens are computed based on characters.

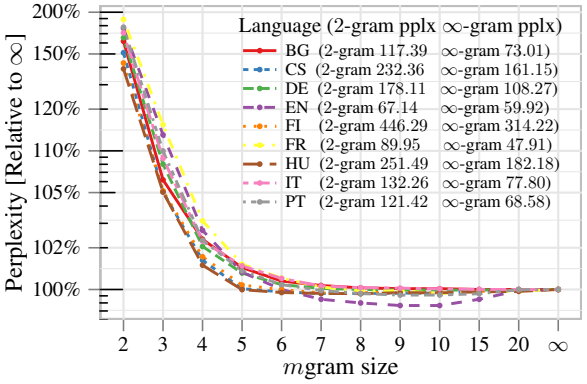


Figure 3: Perplexity results on several Europarl languages for different $m$gram sizes, $m = 2 \ldots 10, 15, 20, \infty$.

and N1PFRONTBACK1 is $O(P(\alpha) \log \sigma)$ where $P(\alpha)$ is the number of preceeding symbols of α, a considerable improvement over N1PFRONTBACK using the forward and reverse CSTs. Overall this leads to considerably faster computation of $m$gram probabilities compared to the two CST approach, and although still slower than highly optimised LM toolkits like SRILM, it is fast enough to support large scale experiments, and has considerably better scaling performance with the Markov order m (even allowing unlimited order), as we will now demonstrate.

6 Experiments

We used Europarl dataset and the data was numberized after tokenizing, splitting, and excluding XML markup. The first $10k$ sentences were used as the test data, and the last 80% as the training data, giving rise to training corpora of between 8M and 50M tokens and uncompressed size of up to 200 MiB (see Table 1 for detailed corpus statistics). We also processed the full 52 GiB uncompressed "20150205" English Wikipedia articles dump to create a character level language model consisting of $72M$ sentences. We excluded $10k$ random sentences from the collection as test data. We use the SDSL library (Gog et al., 2014) to implement all our structures and compare our indexes to SRILM (Stolcke, 2002). We refer to our dual-CST approach as D-CST, and the single-CST as S-CST.

We evaluated the perplexity across different languages and using $m$grams of varying order from $m = 2$ to ∞ (unbounded), as shown on Figure 3. Our results matched the perplexity results from SRILM (for smaller values of m in which SRILM training was feasible, $m \leq 10$). Note that perplexity drops dramatically from $m = 2 \ldots 5$ however the gains thereafter are modest for most languages. Despite this, several large $m$gram matches were found ranging in size up to a 34-gram match. We speculate that the perplexity plateau is due to the simplistic Kneser-Ney discounting formula which is not designed for higher order $m$gram LMs and appear to discount large $m$grams too aggressively. We leave further exploration of richer discounting techniques such as Modified Kneser-Ney (Chen and Goodman, 1996) or the Sequence Memoizer (Wood et al., 2011) to our future work.

Figure 4 compares space and time of our indexes with SRILM on the German part of Europarl. The construction cost of our indexes in terms of both space and time is comparable to that of a 3/4-gram SRILM index. The space usage of D-CST index is comparable to a compact 3-gram SRILM index. Our S-CST index uses only 177 MiB RAM at query time, which is comparable to the size of the collection (172 MiB). However, query processing is significantly slower for both our structures. For 2-grams, D-CST is 3 times slower than a 2-gram SRILM index as the expensive $N^{1+}(\bullet\, \alpha\, \bullet)$ is not computed. However, for large $m$grams, our indexes are much slower than SRILM. For $m > 2$, the D-CST index is roughly six times slower than S-CST. Our fastest index, is 10 times slower than the slowest SRILM 10-gram index. However, our run-time is independent of m. Thus, as m increases, our index will become more competitive to SRILM while using a constant amount of space.

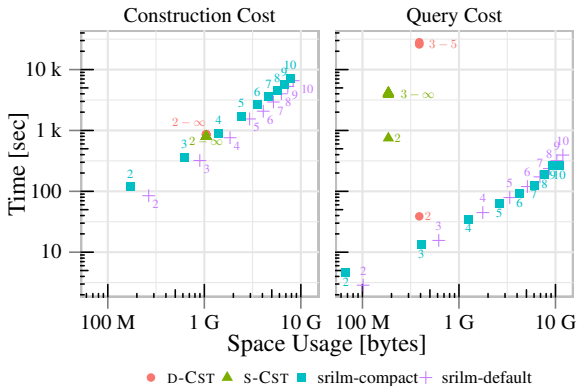


Figure 4: Time versus space tradeoffs measured on Europarl German (de) dataset, showing memory and time requirements.

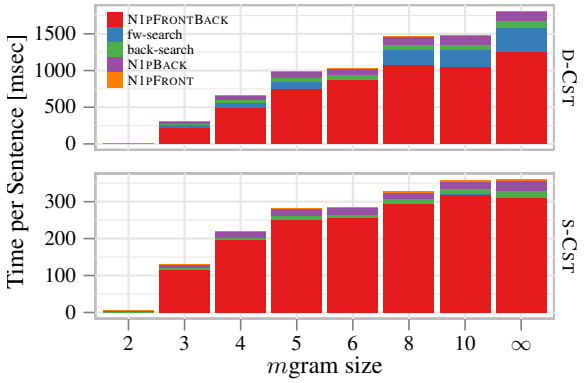


Figure 5: Runtime breakdown of a single pattern averaged over all patterns for both methods over the Wikipedia collection.

Next we analyze the performance of our index on the large Wikipedia dataset. The S-CST, character level index for the data set requires 22 GiB RAM at query time whereas the D-CST requires 43 GiB. Figure 5 shows the run-time performance of both indexes for different $m$grams, broken down by the different components of the computation. As discussed above, 2-gram performance is much faster. For both indexes, most time is spent computing N1PFRONTBACK (i.e., $N^{1+}(\bullet\ \alpha\ \bullet)$) for all $m > 2$. However, the wavelet tree traversal used in S-CST roughly reduces the running time by a factor of three. The complexity of N1PFRONTBACK depends on the number of contexts, which is likely small for larger $m$grams, but can be large for small $m$grams, which suggest partial precomputation could significantly increase the query performance of our indexes. Exploring the myraid of different CST and CSA configurations available could also lead to significant improvements in runtime and space usage also remains future work.

7 Conclusions

This paper has demonstrated the massive potential that succinct indexes have for language modelling, by developing efficient algorithms for on-the-fly computing of $m$gram counts and language model probabilities. Although we only considered a Kneser-Ney LM, our approach is portable to the many other LM smoothing method formulated around similar count statistics. Our complexity analysis and experimental results show favourable scaling properties with corpus size and Markov order, albeit running between 1-2 orders of magnitude slower than a leading count-based LM. Our ongoing work seeks to close this gap: preliminary experiments suggest that with careful tuning of the succinct index parameters and caching expensive computations, query time can be competitive with state-of-the-art toolkits, while using less memory and allowing the use of unlimited context.

Acknowledgments

Ehsan Shareghi and Gholamreza Haffari are grateful to National ICT Australia (NICTA) for generous funding, as part of collaborative machine learning research projects. Matthias Petri is the recipient of an Australian Research Councils Discovery Project scheme (project DP140103256). Trevor Cohn is the recipient of an Australian Research Council Future Fellowship (project number FT130101105).

References

Thorsten Brants, Ashok C Popat, Peng Xu, Franz J Och, and Jeffrey Dean. 2007. Large language models in machine translation. In *Proc. EMNLP-CoNLL*.

M. Burrows and D. Wheeler. 1994. A block sorting lossless data compression algorithm. Technical Report 124, DEC.

Bernard Chazelle, Joe Kilian, Ronitt Rubinfeld, and Ayellet Tal. 2004. The bloomier filter: An efficient data structure for static support lookup tables. In *Proc. SODA*, pages 30–39.

Stanley F Chen and Joshua Goodman. 1996. An empirical study of smoothing techniques for language modeling. In *Proc. ACL*, pages 310–318.

P. Ferragina, G. Manzini, V. Mäkinen, and G. Navarro. 2007. Compressed representations of sequences and full-text indexes. *ACM Trans. on Algorithms*, 3(2):article 20.

Paolo Ferragina, Rodrigo González, Gonzalo Navarro, and Rossano Venturini. 2008. Compressed text indexes: From theory to practice. *ACM J. of Exp. Algorithmics*, 13.

Ulrich Germann, Eric Joanis, and Samuel Larkin. 2009. Tightly packed tries: How to fit large models into memory, and make them load fast, too. In *Proc. of the Workshop on Software Engineering, Testing, and Quality Assurance for Natural Language Processing*, pages 31–39.

Simon Gog, Timo Beller, Alistair Moffat, and Matthias Petri. 2014. From theory to practice: Plug and play with succinct data structures. In *Proc. SEA*, pages 326–337.

David Guthrie and Mark Hepple. 2010. Storing the web in memory: Space efficient language models with constant time retrieval. In *Proc. EMNLP*, pages 262–272.

Kenneth Heafield. 2011. KenLM: Faster and smaller language model queries. In *Proc. WMT*.

Juha Kärkkäinen, Peter Sanders, and Stefan Burkhardt. 2006. Linear work suffix array construction. *J. ACM*, 53(6):918–936.

Casey Redd Kennington, Martin Kay, and Annemarie Friedrich. 2012. Suffix trees as language models. In *Proc. LREC*, pages 446–453.

Reinhard Kneser and Hermann Ney. 1995. Improved backing-off for m-gram language modeling. In *Proc. ICASSP*, volume 1, pages 181–184.

Philipp Koehn. 2010. *Statistical Machine Translation*. Cambridge University Press, New York, NY, USA.

Stefan Kurtz. 1999. Reducing the space requirement of suffix trees. *Softw., Pract. Exper.*, 29(13):1149–1171.

Udi Manber and Eugene W. Myers. 1993. Suffix arrays: A new method for on-line string searches. *SIAM J. Comput.*, 22(5):935–948.

Ian Munro. 1996. Tables. In *Proc. FSTTCS*, pages 37–42.

Enno Ohlebusch, Johannes Fischer, and Simon Gog. 2010. CST++. In *Proc. SPIRE*, pages 322–333.

Adam Pauls and Dan Klein. 2011. Faster and smaller n-gram language models. In *Proc. ACL-HLT*.

Lawrence Rabiner and Biing-Hwang Juang. 1993. *Fundamentals of speech recognition*. Prentice-Hall.

L. Russo, G. Navarro, and A. Oliveira. 2011. Fully-compressed suffix trees. *ACM Trans. Algorithms*, 7(4):article 53.

Kunihiko Sadakane. 2007. Compressed suffix trees with full functionality. *Theory Comput. Syst.*, 41(4):589–607.

Thomas Schnattinger, Enno Ohlebusch, and Simon Gog. 2010. Bidirectional search in a string with wavelet trees. In *Proc. CPM*, pages 40–50.

Jeffrey Sorensen and Cyril Allauzen. 2011. Unary data structures for language models. In *Proc. INTERSPEECH*, pages 1425–1428.

Andreas Stolcke, Jing Zheng, Wen Wang, and Victor Abrash. 2011. Srilm at sixteen: Update and outlook. In *Proc. ASRU*, page 5.

Andreas Stolcke. 2002. SRILM–an extensible language modeling toolkit. In *Proc. INTERSPEECH*.

David Talbot and Miles Osborne. 2007. Randomised language modelling for statistical machine translation. In *Proc. ACL*.

Esko Ukkonen. 1995. On-line construction of suffix trees. *Algorithmica*, 14(3):249–260.

Taro Watanabe, Hajime Tsukada, and Hideki Isozaki. 2009. A succinct n-gram language model. In *Proc. ACL Short Papers*, pages 341–344.

Peter Weiner. 1973. Linear pattern matching algorithms. In *Proc. SWAT*, pages 1–11.

Frank Wood, Jan Gasthaus, Cédric Archambeau, Lancelot James, and Yee Whye Teh. 2011. The sequence memoizer. *CACM*, 54(2):91–98.

ERSOM: A Structural Ontology Matching Approach Using Automatically Learned Entity Representation

Chuncheng Xiang, Tingsong Jiang, Baobao Chang, Zhifang Sui
Key Laboratory of Computational Linguistics, Ministry of Education
School of Electronics Engineering and Computer Science, Peking University
Collaborative Innovation Center for Language Ability, Xuzhou 221009 China
{ccxiang,tingsong,chbb,szf}@pku.edu.cn

Abstract

As a key representation model of knowledge, ontology has been widely used in a lot of NLP related tasks, such as semantic parsing, information extraction and text mining etc. In this paper, we study the task of ontology matching, which concentrates on finding semantically related entities between different ontologies that describe the same domain, to solve the semantic heterogeneity problem. Previous works exploit different kinds of descriptions of an entity in ontology directly and separately to find the correspondences without considering the higher level correlations between the descriptions. Besides, the structural information of ontology haven't been utilized adequately for ontology matching. We propose in this paper an ontology matching approach, named ERSOM, which mainly includes an unsupervised representation learning method based on the deep neural networks to learn the general representation of the entities and an iterative similarity propagation method that takes advantage of more abundant structure information of the ontology to discover more mappings. The experimental results on the datasets from Ontology Alignment Evaluation Initiative (OAEI[1]) show that ERSOM achieves a competitive performance compared to the state-of-the-art ontology matching systems.

[1]The OAEI is an international initiative organizing annual campaigns for evaluating ontology matching systems. All of the ontologies provided by OAEI are described in OWL-DL language, and like most of the other participates our ERSOM also manages the OWL ontology in its current version. OAEI: `http://oaei.ontologymatching.org/`

1 Introductions

In the recent years, it becomes evident that one of the most important directions of improvement in natural language processing (NLP) tasks, like word sense disambiguation, coreference resolution, relation extraction, and other tasks related to knowledge extraction, is by exploiting semantics resources (Bryl et al., 2010). Nowadays, the Semantic Web made available a large amount of logically encoded information (e.g. ontologies, RDF(S)-data, linked data, etc.), which constitutes a valuable source of semantics. However, extending the state-of-the-art natural language applications to use these resources is not a trivial task mainly due to the heterogeneity and the ambiguity of the schemes adopted by the different resources of the Semantic Web. How to utilize these resources in NLP tasks comprehensively rather than choose just one of them has attracted much attention in recent years.

An effective solution to the ontology heterogeneity problem is ontology matching (Euzenat et al., 2007; Shvaiko and Euzenat, 2013), whose main task is to establish semantic correspondences between entities (i.e., classes, properties or instances) from different ontologies.

Ontology matching is usually done by measuring the similarity between two entities from two different ontologies. To effectively calculate the similarities, almost all types of descriptions of an entity should be used. In previous works, given the different nature of different kinds of descriptions, similarities are normally measured separately with different methods and then aggregated with some kind of combination strategy to compute the final similarity score. For example, Mao et al. (2010) defined three single similarities (i.e., *Name similarity*, *Profile similarity* and *Structural similarity*) based on the descriptions of an entity, then they employed a harmony-based method to aggregate

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 2419–2429,
Lisbon, Portugal, 17-21 September 2015.

the single similarities to get a final similarity for extracting the final mappings. However, treating different kinds of descriptions of an entity separately suffers from two limitations. First, it limits the capacity of modeling the interactions between different descriptions. For example, entity's label is always a specific substitution of its ID; entity's comment is a semantic definition for its ID; a class can be characterized with its related properties, and a property is usually restricted by its domain and range. These potential correlations of the descriptions are very important to measure the similarity between entities since they can be treated as some potential features describing an entity. Second, it is difficult to estimate how many and which kind of single similarities are needed for an aggregation method to get a satisfactory result.

On the other hand, in order to find more mappings, many structural ontology matching methods are proposed. To the best of our knowledge, previous structural methods are either local methods (Le et al., 2004; Sunna and Cruz, 2007) or global (i.e. iterative) methods but only use part of the structure information of the ontology (Li et al., 2009; Ngo and Bellahsene, 2012). For example, the ontology matching system YAM++ (Ngo and Bellahsene, 2012) utilizes a global structural method but it only uses the structure information of classes and properties to create the propagation graph to find mappings between classes and properties. A large amount of instances and their relations to classes and properties in the ontology haven't been exploited in this system.

To overcome the existing limitations, we propose in this paper a representation learning method to capture the interactions among entity's descriptions; then we present our global structural method which exploits more abundant structure information of the ontology. We summarize our contributions as follows.

- We propose to use the deep neural network model to learn the high-level abstract representations of classes and properties from their descriptions to acquire the potential correlations for the computing of the similarities between classes and properties. Moreover, there is no need to select and aggregate different single similarities in the similarity computation.
- We propose a global similarity propagation method that utilizes more abundant structure information including all kinds of entities and their relations in the ontology, to find more mappings.

To evaluate the effectiveness of our approach, we conduct experiments on the public datasets from OAEI campaign (We select the OAEI data sets mainly because evaluation metrics have been well defined on these data sets and comparision can be easily made). The experimental results show that our matching approach can achieve a competitive matching performance compared to the state-of-the-art systems.

2 Problem Statement

Ontology is a formal, explicit specification of a shared conceptualization in terms of classes, properties and relations (Euzenat et al., 2004). The process of ontology matching is to find mappings (or correspondences) between entities (classes, properties or individuals) from two ontologies. A mapping is defined as a four-tuple as written in Eq.(1), where e^1 and e^2 represent the entity in ontology O^1 and O^2 respectively, r is a kind of matching relation (e.g., equivalent, subsume) and $k \rightarrow [0, 1]$ is the degree of confidence of matching relation between e^1 and e^2 (Mao et al., 2010).

$$m =< e^1, e^2, r, k > \quad (1)$$

Similar with most of the OAEI systems (Li et al., 2009; Ngo and Bellahsene, 2012; Cheatham and Hitzler, 2013b), we focus on discovering only equivalent mappings between classes and properties with cardinality 1:1. That is, one class (property) in ontology O^1 can be matched to at most one class (property) in ontology O^2 and vise versa.

3 ERSOM: Entity Representation and Structure based Ontology Matching

In this paper, we propose a structural ontology matching approach using automatically learned entity representation, which we call ERSOM. Fig.1 shows the architecture of our approach. The details of its major modules are given in the following sections.

3.1 Learning the representation of entity

In this section, we present how to learn the high-level abstract representations for ontology entities. The motivations are: 1) we regard different kinds of the descriptions of an entity as a whole to avoid

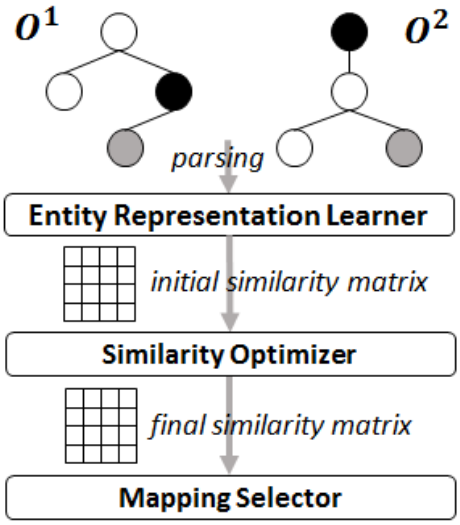


Figure 1: The architecture of ERSOM. Given the two to-be-matched ontologies, we first extract the descriptive information for each entity, then learn the entity's abstract representation based on its descriptions, and finally utilize the representations to compute the similarities between entities to initialize the similarity propagation method to find final mappings.

separatively calculating the similarities and aggregating them later with a combination method; 2) the learned representation can not only express the meaning of the original descriptions of an entity but also captures the interactions among different descriptions.

3.1.1 Creating term vector for entity

We first generate a combination of the entity's descriptions (CDs for short) and then create a term vector for each entity. In particular, the CDs of a class = the class's ID + label + comments + its properties' descriptions + its instances' descriptions. The CDs of a property = the property's ID + label + its domain + its range (or its textual value when the property is a datatype property). And the CDs of an instance = the instance's ID + label + its properties' values. A binary term vector is created for each entity with the pre-processing that consists of tokenizing, removing stop words, stemming and deleting superfluous words. In the binary term vectors, element 1 and 0 refer to the existence and inexistence of a specific word, respectively.

3.1.2 Learning entity representations

In the ontology matching area, training data usually refers to a pair of ontologies with correct mappings created by domain experts between their entities. The acquisition of such dataset is time-consuming and laborious. We state in this section how to learn the abstract representations for entities from their binary term vectors with an unsupervised way. The deep neural network (DNN) (Hinton et al., 2006; Bengio et al., 2007) is a multilayer learning model. It is mainly used for learning the high-level abstract representations of original input data. Given the generalization and the abstraction introduced in the representation learning procedure, DNN allows us to better model the interactions among different kinds of input features, and measure the similarity at a more general level. Inspired by the work in (Hinton, 2007; Bengio et al., 2012; He et al., 2013; Cui et al., 2014), we use auto-encoder (Bourlard and Kamp, 1988; Hinton and Zemel, 1994) to learn the representations for classes and properties. The auto-encoder is one of the neural network variants that can automatically discover interesting abstractions to represent an unlabeled dataset.

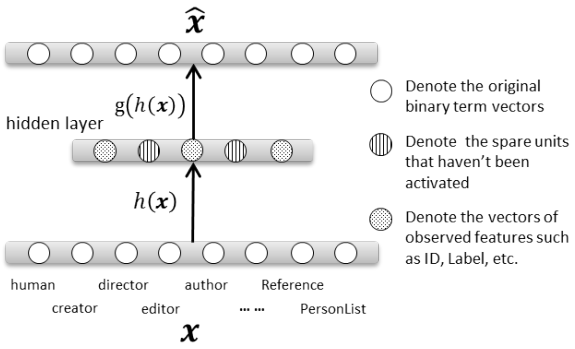


Figure 2: Unsupervised representation learning.

As shown in Fig.2, the input to the auto-encoder is denoted as x, which indicates a binary term vector of a class or a property. Auto-encoder tries to learn an approximation to the identity function $h(x)$, so as to output $\hat{x}$ that is similar to x. More specifically, the auto-encoder consists of an encoding process $h(x) = f(Wx + b_1)$ and a decoding process $g(h(x)) = f(W^T h(x) + b_2)$, where f is a activation function like the sigmoid function $f(x) = 1/(1 + exp(-x))$, W is the weight matrix and b is the bias term. The goal of the auto-encoder is to minimize the reconstruction error $L(x, g(h(x))) = |x - g(h(x))|^2$, thus retaining

maximum information. Through the combination and transformation, auto-encoder learns the abstract representation $h(x)$ of the input binary term vector. The representation is a real vector with values between 0 and 1.

In consideration of the large number of units in the hidden layer (as marked in Fig.2), a sparsity constraint is imposed on the hidden units to hold the capacity to discover interesting structure in the data. We use sparse auto-encoder (Coates et al., 2011) to learn the correlations between descriptions from their binary term vectors. The sparse auto-encoder attempts to minimize the following loss function:

$$J(W,b) = \sum_{i=1}^{m} \left\| x^{(i)} - \widehat{x}^{(i)} \right\|^2 + \lambda \left(\|W_1\|_F + \|W_2\|_F \right) + \beta \sum_{h \in H} KL\left(\rho \| \widehat{\rho}_h\right) \tag{2}$$

where $x^{(i)}$ is the binary term vector of the ith entity (a class or a property), $\widehat{x}^{(i)}$ is the reconstruction of $x^{(i)}$, λ is a regularization parameter, original $\| \cdot \|_F$ is the Frobenius norm, β controls the weight of the sparsity penalty term, H is the set of hidden units, and ρ is the sparsity parameter. Formally, $KL\left(\rho \| \widehat{\rho}_h\right) = \rho \log \frac{\rho}{\widehat{y}_h} + (1-\rho) \log \frac{1-\rho}{1-\widehat{y}_h}$ is the Kullback-Leibler (KL) divergence between a Bernoulli random variable with mean ρ and a Bernoulli random variable with mean $\widehat{y}_h$.

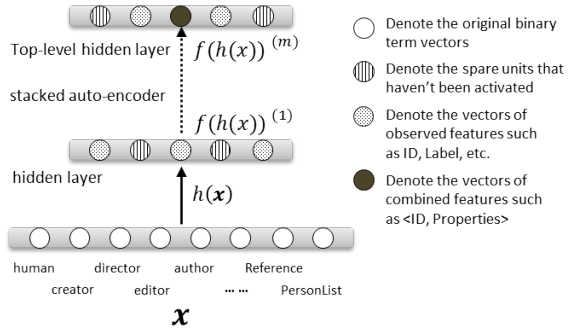


Figure 3: Learning higher level representations.

3.1.3 Learning higher level representations

The auto-encoder which only has one hidden layer may not be enough to learn the complex interactions between input features. Inspired by the work of Vincent et al. (2010) and He et al. (2013), we build multi-layer model to learn more abstract entity representations. To achieve this, we repeatedly stack new sparse auto-encoder on top of the previously learned $h(x)$ (i.e., the higher level representations are formed by combination of lower level representations). This model is called Stacked Auto-Encoder (SAE) by Bengio et al. (2007). In this way, when we input the binary term vector to the network, we can get its abstract representations in different levels. In other words, with the layer-by-layer learning, we obtain different levels of representations. The top-level representation, which models the final interactions of the original descriptions, can be used to measure the similarity between classes and properties.

The prototype of Stacked Auto-Encoder (SAE) is shown in Fig.3, where $f(h(x))^{(m)}$ denotes the final representation learned by the top-level hidden layer and superscript m means the SAE consists of m sparse auto-encoders.

3.2 Optimizing with the ontology structure

The above method can only consider the local descriptions (such as ID, label and comments etc.) of entities in ontology. According to the study in (Melnik et al., 2002), we present our structural method or called Similarity Propagation (SP) method, which exploits more abundant structure information of the ontology to discover more mappings globally. The intuition behind the propagation idea is that the similarity between two entities is not only about their neighbor entities, but it is about all other entities (neighbor entities' neighbor entities) in the ontologies. This idea has also been used in the ontology matching systems RiMOM(Li et al., 2009) and YAM++(Ngo and Bellahsene, 2012) in order to find mappings between classes and properties. But the nodes in their propagation graph are just limited to class pairs and property pairs, and the propagation edges are transformed from relations between two classes, two properties or a class and a property. The difference of our SP method is that we consider the instances and its relations with classes and properties when creating the propagation graph even if we also only find mappings between classes and properties. This is because (1) the similar degree of two classes will be increased if they have some of similar instances; (2) the similar degree of two properties will be increased if the instances that own these properties are similar. The propagation graph in our SP method will be much more complete compared with the previous ones.

Algorithm 1 presents the procedures of our SP method. In the first two steps of it, we represent each to-be-matched ontology to a Directed Labeled Graph (DLG). Each edge in the DLG has format (s, p, o), where s and o are the source and target nodes (each node represents a class, a property or an instance), and the edge's label p comes from one of the seven ontology relations including *HasSubClass*, *HasSubProperty*, *HasObjectProperty*, *HasDataProperty*, *HasInstance*, *HasDomain*, *HasRange*. Then we create a Pairwise Connectivity Graph (PCG) from two DLGs by merging edges having the same labels.

Algorithm 1: Our SP Algorithm

Input: The to-be-matched ontologies, O_R and O_T; The initial similarity matrix, M_0; The edges' weight matrix, W;

Output: The updated similarity matrix, M_1;

$DLG_1 \leftarrow Transform(O_R)$;
$DLG_2 \leftarrow Transform(O_T)$;
$PCG \leftarrow Merge(DLG_1, DLG_2)$;
$IPG \leftarrow Initiate(PCG, M_0, W)$;
$M_1 \leftarrow Propagation(IPG, Normalized)$;

In the fourth step of Algorithm 1, for a PCG, we assign weight values to edges as the inverse of the number of out-linking relationships of its source node (Melnik et al., 2002). For the nodes that consist of two classes or two properties, we assign them values calculated with the cosine similarity between their representations learned in section 2.1.3. For the node consisting of two instances, the similarity value assigned to it is measured with the ScaledLevenstein[2] between the IDs of instances. In this way, we construct an Induced Propagation Graph (IPG) on which the propagation algorithm will run iteratively. Let $\sigma(x, y)$ denotes the similarity score between entities x and y for node (x, y) in the IPG. At the $(i+1)$th iteration, the similarity score is updated as follows:

$$\sigma^{i+1} = \frac{1}{z}\left(\sigma^0 + \sigma^i + \varphi(\sigma^0 + \sigma^i)\right), \quad (3)$$

$$\varphi\left(\sigma^0 + \sigma^i\right) = \sum_{j=1}^{m}\left(\sigma^0 + \sigma_j^i\right) w_j \quad (4)$$

[2] `http://sourceforge.net/projects/secondstring/`. ScaledLevenstein is a good method for computing string similarity (Cheatham and Hitzler, 2013a), of course, it can be replaced by other methods.

where z is the normalization factor defined as $z = \max_{x' \in IPG}\left(\sigma^{i+1}\right)$. σ^0 and σ^i are similarities at the initial time and ith iterations, respectively. $\varphi()$ is the function to compute the similarities propagated from the adjacent node σ_j^i connected to node (x, y) in the $(i+1)$th iteration. And w_j is the weight of edge between the node (x, y) and its jth neighboring node.

During each iteration in the final step of Algorithm 1, only the similarity value between two entities in the node will be updated. At the end of each iteration, all similarity values are normalized by a *Normalized* function to all in range $[0, 1]$. The iteration stops after a predefined number of steps.

3.3 Mapping selection

Similar with the work in (Wang and Xu, 2007; Huang et al., 2007; Ngo and Bellahsene, 2012), we use the Stable Marriage (SM) algorithm (Melnik et al., 2002) to choose the 1:1 mappings from the M rows and N columns similarity matrix, where M and N is the number of classes and properties in ontologies O_1 and O_2, respectively. In addition, before we run the SM algorithm we set the value of cell $[i, j]$ of the similarity matrix to zero if i and j correspond to different types of entities. Thus, we remove lots of redundant data and only find the mappings between classes or properties.

4 Experiments

4.1 Data sets and evaluation criteria

The annual OAEI campaign is an authoritative contest in the area of ontology matching, we choose the data from OAEI in our experiments, because the evaluation metrics have been well defined and the comparision can be easily made. We observe strong structure similarities lies between OAEI ontologies and ontologies used in NLP tasks, such as WordNet and HowNet for WSD (Li et al., 1995; Agirre et al., 2009), and Freebase, YAGO, and knowledge graph for IE, text mining and QA (Yao and Van Durme, 2014; Yao et al., 2014), both describe entities and their relations with class, properties and instances.

Development dataset: the Standard Benchmark 2012 dataset that OAEI provides for developers to test their system before participating in the competition is used as the development dataset in our experiments. This dataset contains one reference ontology and 109 target ontologies. We use

this dataset to test various values for the parameters in our ERSOM and apply the best ones to the experiments on the testing datasets.

Testing dataset: (1) the Benchmark-Biblio 2012 dataset which contains one reference ontology and 94 target ontologies; (2) the Benchmark-Biblioc 2013 dataset which has five sub-datasets and there are one reference ontology and 93 target ontologies in each sub-dataset. We use these two datasets to evaluate the performance of our ERSOM approach.

In the matching scenario, each target ontology should be mapped to the reference ontology. We followed the standard evaluation criteria from the OAEI, calculating the precision, recall and f-measure over each test. The version computed here is the harmonic mean of precision and recall.

4.2 Experimental design and results

4.2.1 Evaluation for representation learning

We first use Jena[3] parsing the ontologies and extract descriptions for entities according to the description in section 2.1.1, then we create a vocabulary based on the dataset and denote each class and property as a binary term vector. We apply the L-BFGS algorithm (Ngiam et al., 2011) to train the stacked auto-encoder described in section 2.1.3. The size of the input layer is equals to the length of the vocabulary created from the dataset. We fix the parameters $\lambda = 1e-4$, $\beta = 3$ and $\rho = 0.25$ in Eq.2, and set the size of the first and second hidden layer of the stacked auto-encoder to 200 and 100, respectively, by experience. The number of iterations of the L-BFGS algorithm is set to 500. We use the learned representations to measure the similarities between classes and properties and apply the strategy presented in section 2.3 to extract final mappings. The matching results of our Unsupervised Representation Learning (URL) method on the development dataset and testing datasets are shown in Table 1 and Table 2, respectively.

Method	Prec.	Rec.	F-m.
TV	0.841	0.715	0.748
URL(1)	0.819	0.822	0.820
URL(2)	0.843	0.846	0.844

Table 1: Representation learning on dev. dataset.

In Table 1, TV denotes the matcher in which the similarities are calculated between binary term vectors of classes and properties by using cosine measure. URL(i), where $i \in \{1, 2\}$, represents that we use the representations learned by the ith hidden layer of the stacked auto-encoder to measure the similarities between classes and properties to find mappings. Table 1 shows that on the development dataset, the F-measure of TV is 0.748 and it is improved 9.6% and 12.8% when we use the representations learned by the single-layer and double-layer auto-encoder to find the mappings, respectively. It illustrates that we have learned some useful information from the term vectors, which can be explained as the interactions between descriptions of entities. From the last two rows in Table 1, we can find that the F-measure improved by 2.9% when we use the representations learned by the second hidden layer (i.e., URL(2)) to measure the similarities.

	Benchmark-Biblio 2012			Benchmark-Biblioc 2013		
	Prec.	***Rec.***	***F-m.***	***Prec.***	***Rec.***	***F-m.***
TV	0.870	0.719	0.761	0.865	0.715	0.757
URL(1)	0.805	0.808	0.806	0.780	0.783	0.786
URL(2)	0.814	0.817	0.815	0.787	0.790	0.793

Table 2: Representation learning on test datasets.

From Table 2 we can see that the F-measures are increased on both of the testing datasets when we use the learned representations to measure similarities between classes and properties compared with using term vectors, but the amount of improvements are less than that on the development dataset. This is because we estimate the parameters of the representation learning model on the development dataset and then apply them on the test tasks directly. The precision is reduced when we use URL method, this may be due to the learned representations of entities are too general. In addition, in the parameter adjustment process, we try to make the F value maximization, but not to care about mapping precision. This is because we usually compare the performance of the systems based on their matching F values.

4.2.2 Comparison with aggregation methods

Aggregating different similarities is pervasive in ontology matching systems that contain multiple single matchers, for example, Falcon-AO(Qu et al., 2006), RiMOM(Li et al., 2009), YAM++(Ngo and Bellahsene, 2012), etc. Since our representation learning method also combines all descriptions of an entity together in an unsu-

[3] https://jena.apache.org/

pervised way, we compare it with previous unsupervised aggregation strategies, that is, Max, Average, Sigmoid, Weighted(Cruz et al., 2010) and Harmony(Mao et al., 2008, 2010). As the work in (Mao et al., 2010; Ngo and Bellahsene, 2012), we first define three context profiles including individual profile, semantic profile and external profile for each class and property (this equivalent to divide the collection of descriptions of a class or a property into three different parts). Then we apply a vector space model with TFIDF weighting scheme and cosine similarity measure to compute similarity scores between profiles. And finally, we aggregate these three single similarities using different aggregation methods.

	Dev.	**Tes.1**	**Tes.2**
Individual Profile	0.668	0.612	0.611
Semantic Profile	0.434	0.472	0.477
External Profile	0.222	0.224	0.224
MAX	0.739	0.705	0.712
Average	0.792	0.786	0.786
Sigmoid	0.763	0.753	0.757
Weighted	0.755	0.716	0.728
Harmony	0.794	0.789	0.785
URL	**0.844**	**0.815**	**0.793**

Table 3: Comparison with aggregation methods.

Table 3 shows the F-measure of the single matchers and aggregation methods on the development dataset (Dev. for short) and two testing datasets (i.e., Tes.1 and Tes.2, which refer to the Benchmark-Biblio 2012 dataset and the Benchmark-Biblioc 2013 dataset, respectively). First, the performance of single matcher is poor, the highest F-measures are 0.668, 0.612 and 0.611 on the datasets Dev., Tes.1 and Tes.2, respectively. And when we use external profile to calculate the similarities, the F-measures are reduced to 22%. Second, the performance is dramatically boosted by aggregation methods and they all achieve F-measures higher than 0.7, so the aggregation methods are very effective in improving the performance of mapping approaches that rely on measuring multiple similarities. And finally, our Unsupervised Representation Learning (URL) method holds the highest F-measure both on the development dataset and on the testing datasets.

4.2.3 Evaluation for our structural method

In this experiment, we compare our Similarity Propagation (SP) method to other structure based methods, that is, ADJACENTS and ASCOPATH in (Le et al., 2004); DSI and SSC in (Sunna and Cruz, 2007); Li's SP (Li et al., 2009) and Ngo's SP (Ngo and Bellahsene, 2012). We first use entity's ID to compute the similarity between classes and properties to provide an unified initial similarity matrix as input (or initialization) for our SP and other structural methods. Then, a new similarity matrix will be created and updated by considering the initial similarities and different structure information. And finally, we extract the mappings from the newly created similarity matrix with the strategy described in section 2.3.

	Dev.	**Tes.1**	**Tes.2**
Initial Matcher	0.616	0.524	0.523
ADJACENTS	0.622	0.569	0.570
ASCOPATH	0.604	0.540	0.552
DSI	0.641	0.576	0.575
SSC	0.642	0.569	0.568
Li's SP	0.747	0.769	0.772
Ngo's SP	0.751	0.768	0.764
Our SP	**0.810**	**0.834**	**0.839**
Our SP with URL	**0.903**	**0.865**	**0.866**

Table 4: Comparison with structural methods.

In ADJACENTS method, the parameter W_k, where $k \in \{1, 2, 3\}$, is set to 1/3. The parameter MCP in the methods DSI and SSC is set to 0.75 as reported in their work. The iterative times to SP algorithm are fixed to 50. Table 4 reports the matching F-measures of these structure based methods on the development dataset (Dev. for short) and testing datasets (Tes.1, Tes.2 for short).

From table 4 we can see that the local-structure based methods (i.e., ADJACENTS, ASCOPATH, DSI and SSC) provide low matching quality. It means that these methods did not discover enough additional correct mappings or even find some incorrect mappings. For example, the F-measure even reduced on the development dataset when use ASCOPATH method. This is because if two entities don't have any common entity in their ancestors, their similarity is equal to 0. Whereas, Li's SP and Ngo's SP are global-structure based methods, and they seem to work well. The F-measure has even improved by 21.9% when using the Ngo's SP compared with the initial matcher. This

is because in the SP method, the similarity score of a pair of entities depends on not only their current status but also on the status of the other pairs. That explains why SP outperforms all other local based structural methods. In our SP, all instances and their relations to other entities in ontology are exploited to help find the mappings between classes and properties, therefore the matching quality is distinctly improved.

The last two rows of Table 4 shows that when we use the learned representations to create the initial similarity matrix to initialize our SP method, the matching quality is significantly improved. For example, the F-measure is improved from 0.810 to 0.903 on the development dataset. This illustrates that the initialization step is very important to the SP method.

4.2.4 Comparison with other ontology matching systems

We compare our ontology matching approach, called ERSOM, with other multi-strategy matching systems on the testing datasets. Fig.4 lists the results of top five matching systems according to their F-measures on the Benchmark-Biblio 2012 dataset and Benchmark-Biblioc 2013 dataset.

As shown in Fig.4, ERSOM outperforms most of the participates except the systems YAM++ and CroMatcher whose F-measures are 0.89 and 0.88 in 2013, respectively. CroMatcher achieves the same level of recall as YAM++ but with consistently lower precision (Grau et al., 2014). Unlike MapSSS, our approach does not use any external resources such as Google queries in its current version. In YAM++ approach, the gold standard datasets that taken from Benchmark dataset published in OAEI 2009 are used to generate training data to train a decision tree classifier. And in the classifying phase, each pair of elements from two to-be-matched ontologies is predicted as matched or not according to its attributes. However, ERSOM is an unsupervised approach, but it does not exclude using external resources and training data to help learning the representations of entities and provide the initial similarity matrix for the SP method to further improve the performance.

5 Related work

There are many studies on Ontology Matching (Euzenat et al., 2007; Shvaiko and Euzenat, 2013). Currently, almost all ontology matching systems exploit various kinds of information provided in

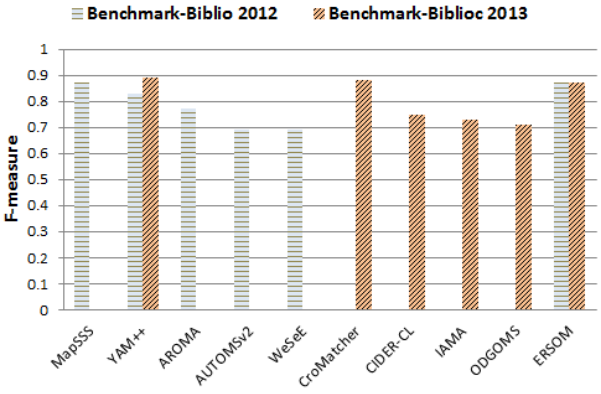


Figure 4: Comparison with other OAEI systems.

ontologies to get better performance. To aggregate the different similarity matrixes, various approaches have been proposed.

Pirró and Talia (2010) is a generic schema matching system. It exploits *Max*, *Min*, *Average* and *Weighted strategies* for the combination. The weighted method assigns a relative weight to each similarity matrix, and calculates a weighted sum of similarity for all similarity matrixes. The *Average method* is a special case of *Weighted*, which considers each similarity matrix equally important in the combination. *Max* and *Min* are two extreme cases that return the highest and lowest similarities in all similarity matrixes respectively. Ji et al. (2011) use the Ordered Weighted Average (OWA) to combine different matchers. It is a kind of ontology-independent combination method which can assign weights to the entity level, i.e., it use a specific ordered position rather than a weight associated with a specific similarity matrix to aggregate multiple matchers. Jean-Mary et al. (2009) combines different matchers by using a weighted sum strategy that adjusts weights empirically, or based on some static rules. This approach cannot automatically combine different matchers in various matching tasks.

There are several works which exploit the supervised machine learning techniques for ontology matching. Eckert et al. (2009), string-based, linguistic and structural measures (in total 23 features) were used as input to train a SVM classifier to align ontologies. CSR (Classification-based learning of Subsumption Relations) is a generic method for automatic ontology matching between concepts based on supervised machine learning (Spiliopoulos et al., 2010). It specifically focusses on discovering subsumption correspon-

dences. SMB (Schema Matcher Boosting) is an approach to combining matchers into ensembles (Gal, 2011). It is based on a machine learning technique called boosting, that is able to select (presumably the most appropriate) matchers that participate in an ensemble.

The difference of our work is that the textual descriptions are not been directly used to measure the similarities between entities. We learn a representation for each ontology entity in an unsupervised way to capture the interactions among the descriptions, which avoid the problem of selecting and aggregating different individual similarities.

6 Conclusions

The successful ontology matching is very important to link heterogeneous ontologies for NLP. In this paper, we have proposed an ontology matching approach, ERSOM, which describes the classes and properties in ontology with abstract representations learned from their descriptions and improves the overall matching quality using an iterative Similarity Propagation (SP) method based on more abundant structure information. Experimental results on the datasets from OAEI demonstrate that our approach performs better than most of the participants and achieves a competitive performance. In our future work, we will consider to use the ontology matching approach to the matching between different NLP-oriented ontologies such as wordnet, Freebase, YAGO, etc.

Acknowledgments

This research is supported by National Key Basic Research Program of China (No.2014CB340504) and National Natural Science Foundation of China (No.61375074,61273318). The corresponding authors of this paper are Baobao Chang and Zhifang Sui.

References

Agirre, E., De Lacalle, O. L., Soroa, A., and Fakultatea, I. (2009). Knowledge-based wsd and specific domains: Performing better than generic supervised wsd. In *IJCAI*, pages 1501–1506. Citeseer.

Bengio, Y., Courville, A. C., and Vincent, P. (2012). Unsupervised feature learning and deep learning: A review and new perspectives. *CoRR abs/1206.5538*.

Bengio, Y., Lamblin, P., Popovici, D., and Larochelle, H. (2007). Greedy layer-wise training of deep networks. *Advances in neural information processing systems*, 19:153.

Bourlard, H. and Kamp, Y. (1988). Auto-association by multilayer perceptrons and singular value decomposition. *Biological cybernetics*, 59(4-5):291–294.

Bryl, V., Giuliano, C., Serafini, L., and Tymoshenko, K. (2010). Supporting natural language processing with background knowledge: Coreference resolution case. In *The Semantic Web–ISWC 2010*, pages 80–95. Springer.

Cheatham, M. and Hitzler, P. (2013a). String similarity metrics for ontology alignment. In *The Semantic Web–ISWC 2013*, pages 294–309. Springer.

Cheatham, M. and Hitzler, P. (2013b). Stringsauto and mapsss results for oaei 2013. *Ontology Matching*, page 146.

Coates, A., Ng, A. Y., and Lee, H. (2011). An analysis of single-layer networks in unsupervised feature learning. In *International Conference on Artificial Intelligence and Statistics*, pages 215–223.

Cruz, I. F., Stroe, C., Caci, M., Caimi, F., Palmonari, M., Antonelli, F. P., and Keles, U. C. (2010). Using agreementmaker to align ontologies for oaei 2010. In *ISWC International Workshop on Ontology Matching (OM). CEUR Workshop Proceedings*, volume 689, pages 118–125.

Cui, L., Zhang, D., Liu, S., Chen, Q., Li, M., Zhou, M., and Yang, M. (2014). Learning topic representation for smt with neural networks. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 133–143.

Eckert, K., Meilicke, C., and Stuckenschmidt, H. (2009). Improving ontology matching using meta-level learning. In *The Semantic Web: Research and Applications*, pages 158–172. Springer.

Euzenat, J., Euzenat, J., Shvaiko, P., et al. (2007). *Ontology matching*. Springer.

Euzenat, J., Le Bach, T., Barrasa, J., Bouquet, P., De Bo, J., Dieng-Kuntz, R., Ehrig, M., Hauswirth, M., Jarrar, M., Lara, R., et al.

(2004). State of the art on ontology alignment. *Knowledge Web Deliverable D*, 2:2–3.

Gal, A. (2011). Uncertain schema matching. *Synthesis Lectures on Data Management*, 3(1):1–97.

Grau, B. C., Dragisic, Z., Eckert, K., Euzenat, J., Ferrara, A., Granada, R., Ivanova, V., Jiménez-Ruiz, E., Kempf, A., Lambrix, P., et al. (2014). Results of the ontology alignment evaluation initiative 2013. In *International Workshop on Ontology Matching, collocated with the 12th International Semantic Web Conference-ISWC 2013*, pages pp–61.

He, Z., Liu, S., Li, M., Zhou, M., Zhang, L., and Wang, H. (2013). Learning entity representation for entity disambiguation. In *ACL (2)*, pages 30–34.

Hinton, G. E. (2007). Learning multiple layers of representation. *Trends in cognitive sciences*, 11(10):428–434.

Hinton, G. E. and Zemel, R. S. (1994). Autoencoders, minimum description length, and helmholtz free energy. *Advances in neural information processing systems*, pages 3–3.

Huang, J., Dang, J., Vidal, J. M., and Huhns, M. N. (2007). Ontology matching using an artificial neural network to learn weights. In *Proc. IJCAI Workshop on Semantic Web for Collaborative Knowledge Acquisition (SWeCKa-07), India*.

Jean-Mary, Y. R., Shironoshita, E. P., and Kabuka, M. R. (2009). Ontology matching with semantic verification. *Web Semantics: Science, Services and Agents on the World Wide Web*, 7(3):235–251.

Ji, Q., Haase, P., and Qi, G. (2011). Combination of similarity measures in ontology matching using the owa operator. In *Recent Developments in the Ordered Weighted Averaging Operators: Theory and Practice*, pages 281–295. Springer.

Le, B. T., Dieng-Kuntz, R., and Gandon, F. (2004). On ontology matching problems. *ICEIS (4)*, pages 236–243.

Li, J., Tang, J., Li, Y., and Luo, Q. (2009). Rimom: A dynamic multistrategy ontology alignment framework. *Knowledge and Data Engineering, IEEE Transactions on*, 21(8):1218–1232.

Li, X., Szpakowicz, S., and Matwin, S. (1995). A wordnet-based algorithm for word sense disambiguation. In *IJCAI*, volume 95, pages 1368–1374. Citeseer.

Mao, M., Peng, Y., and Spring, M. (2008). A harmony based adaptive ontology mapping approach. In *SWWS*, pages 336–342.

Mao, M., Peng, Y., and Spring, M. (2010). An adaptive ontology mapping approach with neural network based constraint satisfaction. *Web Semantics: Science, Services and Agents on the World Wide Web*, 8(1):14–25.

Melnik, S., Garcia-Molina, H., and Rahm, E. (2002). Similarity flooding: A versatile graph matching algorithm and its application to schema matching. In *Data Engineering, 2002. Proceedings. 18th International Conference on*, pages 117–128. IEEE.

Ngiam, J., Coates, A., Lahiri, A., Prochnow, B., Le, Q. V., and Ng, A. Y. (2011). On optimization methods for deep learning. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 265–272.

Ngo, D. and Bellahsene, Z. (2012). Yam++: a multi-strategy based approach for ontology matching task. In *Knowledge Engineering and Knowledge Management*, pages 421–425. Springer.

Pirró, G. and Talia, D. (2010). Ufome: An ontology mapping system with strategy prediction capabilities. *Data & Knowledge Engineering*, 69(5):444–471.

Qu, Y., Hu, W., and Cheng, G. (2006). Constructing virtual documents for ontology matching. In *Proceedings of the 15th international conference on World Wide Web*, pages 23–31. ACM.

Shvaiko, P. and Euzenat, J. (2013). Ontology matching: state of the art and future challenges. *Knowledge and Data Engineering, IEEE Transactions on*, 25(1):158–176.

Spiliopoulos, V., Vouros, G. A., and Karkaletsis, V. (2010). On the discovery of subsumption relations for the alignment of ontologies. *Web Semantics: Science, Services and Agents on the World Wide Web*, 8(1):69–88.

Sunna, W. and Cruz, I. F. (2007). Structure-based methods to enhance geospatial ontology alignment. In *GeoSpatial Semantics*, pages 82–97. Springer.

Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., and Manzagol, P.-A. (2010). Stacked denoising

autoencoders: Learning useful representations in a deep network with a local denoising criterion. *The Journal of Machine Learning Research*, 11:3371–3408.

Wang, P. and Xu, B. (2007). Lily: the results for the ontology alignment contest oaei 2007. In *Proceedings of the Second International Workshop on Ontology Matching*, pages 179–187. Citeseer.

Yao, X., Berant, J., and Van Durme, B. (2014). Freebase qa: Information extraction or semantic parsing? *ACL 2014*, page 82.

Yao, X. and Van Durme, B. (2014). Information extraction over structured data: Question answering with freebase. In *Proceedings of ACL*.

A Single Word is not Enough: Ranking Multiword Expressions Using Distributional Semantics

Martin Riedl and **Chris Biemann**
Language Technology
Computer Science Department, Technische Universität Darmstadt
Hochschulstrasse 10, D-64289 Darmstadt, Germany
{riedl, biem}@cs.tu-darmstadt.de

Abstract

We present a new unsupervised mechanism, which ranks word n-grams according to their multiwordness. It heavily relies on a new *uniqueness* measure that computes, based on a distributional thesaurus, how often an n-gram could be replaced in context by a single-worded term. In addition with a downweighting mechanism for incomplete terms this forms a new measure called DRUID. Results show large improvements on two small test sets over competitive baselines. We demonstrate the scalability of the method to large corpora, and the independence of the measure of shallow syntactic filtering.

1 Introduction

While it seems intuitive to treat certain sequences of tokens as single terms, there is still considerable controversy about the definition of what exactly such a multiword expression (MWE) constitutes. Sag et al. (2001) pinpoint the need of treating MWEs correctly and classify a range of syntactic formations that could form MWEs and define MWEs as being non-compositional with respect to the meaning of their parts. While the exact requirements on MWEs is bound to specific tasks (such as parsing, keyword extraction, etc.), we operationalize the notion of non-compositionality by using distributional semantics and introduce a new measure that works well for a range of task-based MWE definitions.

Most previous MWE ranking approaches use the following mechanisms to determine multiwordness: part-of-speech (POS) tags, word/multiword frequency and significance of co-occurrence of the parts. In this paper we do not want to introduce "yet another ranking function" but rather an additional mechanism, which performs ranking based on distributional semantics.

Distributional semantics has already been used for MWE identification, but mainly to discriminate between compositional and non-compositional MWEs (Schone and Jurafsky, 2001; Salehi et al., 2014; Hermann and Blunsom, 2014). Here we introduce a new concept to describe the multiwordness of a term by its *uniqueness*. Using the *uniqueness* score we measure how likely a term in context can be replaced by a single word. This measure is motivated by the semiotic consideration that due to parsimony concepts are often expressed as single words. Furthermore, we implement a context-aware punishment, called *incompleteness*, which degrades the score of terms that seem incomplete regarding their contexts. Both concepts are combined into a single score we call DRUID, which is calculated based on a distributional thesaurus. In the following, we show the impact of that new method for French and English and also examine the effect of corpus size on MWE extraction. Additionally, we report on results without using any linguistic preprocessing except tokenization.

2 Related Work

The generation of MWE dictionaries has drawn much attention in the field of Natural Language Processing (NLP). Early computational approaches (e.g. Justeson and Katz (1995)) use POS sequences as MWE extractors. Other approaches, relying on word frequency, statistically verify the hypothesis whether the parts of the MWE occur more often together than would be expected by chance (Manning and Schütze, 1999; Evert, 2005; Ramisch, 2012). One of the first measures that consider context information (co-occurrences) are the C-value and the NC-value introduced by Frantzi et al. (1998). These methods first extract candidates using POS information

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 2430–2440,
Lisbon, Portugal, 17-21 September 2015.

and then compute scores based on the frequency of the MWE and the frequency of nested MWE candidates. The method described by Wermter and Hahn (2005) computes a score by multiplying the frequency of a candidate when placing wildcards for each word. A newer method is introduced in Lossio-Ventura et al. (2014), which reranks scores based on an extension of the C-value, which uses a POS-based probability and an inverse document frequency. Using different measures and learning a classifier that predicts the multiwordness was first proposed by Pecina (2010), who, however, restricts his experiments to two-word MWEs for the Czech language only. Korkontzelos (2010) comparatively evaluates several MWE ranking measures. The best MWE extractor reported in his work is the scorer by (Nakagawa and Mori, 2002; Nakagawa and Mori, 2003), who use the un-nested frequency (called marginal frequency) of each candidate and multiply these by the geometric mean of the distinct neighbor of each word within the candidate.

Distributional semantics is mostly used to detect compositionality of MWEs (Salehi et al., 2014; Katz and Giesbrecht, 2006). Most approaches therefore compare the context vector of a MWE with the combined vectors based on the constituent words of the MWE. The similarity between the vectors is then used as degree for compositionality. In machine translation, words are sometimes considered as multiwords if they can be translated as single term (cf. (Bouamor et al., 2012; Anastasiou, 2010)). Whereas this follows the same intuition as our *uniqueness* measure, we do not require any bilingual corpora.

Regarding the evaluation, mostly precision at k ($P@k$) and recall at k ($R@k$) are applied (e.g. (Evert, 2005; Frantzi et al., 1998; Lossio-Ventura et al., 2014)). Another general approach is using the average precision (AP), which is also used in Information Retrieval (IR) (Thater et al., 2009) and has also been applied by Ramisch et al. (2012).

3 Baselines and Previous Approaches

We will evaluate our method by comparing our MWE ranking to multiword lists that have been annotated in corpora. Here, we introduce an upper bound and two baseline methods and give a brief description of the competitive methods used in this paper. Most of these methods require a list of candidate terms T, usually extracted with POS sequences (see Section 5).

3.1 Upper Bound

We use a perfect ranking as upper bound, where we rank all positive candidates before all negative ones.

3.2 Lower Baseline and Frequency Baseline

The ratio between true candidates and all candidates serves as lower baseline, which is also called baseline precision (Evert, 2008). The second baseline is the frequency baseline, which ranks candidate terms $t \in T$ according to their frequency $freq(t)$.

3.3 C-value/NC-value

The commonly used C-value (see Eq. 1) was developed by Frantzi et al. (1998). The first factor, logarithm of the term length in words, favors longer MWEs. The second factor is the frequency of the term reduced by the average frequency of all candidate terms T, which nest the term t, i.e. t is a substring of the terms we denote as T_t.

$$\mathrm{c}(t) = \log_2(|t|)(\mathrm{freq}(t) - \frac{1}{|T_t|}\sum_{b \in T_t}\mathrm{freq}(b)) \quad (1)$$

An extension of the C-value was proposed by Frantzi et al. (1998) as well and is named NC-value. It takes advantage of context words C_t by assigning weights to them. As context words only *nouns*, *adjectives* and *verbs* are considered[1]. Context words are weighted with Equation 2, where k denotes the number of times the context word $c \in C_t$ occurs with any of the candidate terms. This number is normalized by the number of candidate terms.

$$w(c) = \frac{k}{|T|} \quad (2)$$

The NC-value is a weighted sum of the C-value and the product of the term t occurring with each context c which form the term t_c:

$$\mathrm{nc}(t) = 0.8 * \mathrm{c}(t) + 0.2 \sum_{c \in C_t} freq(t_c)w(c). \quad (3)$$

3.4 t-test

The t-test (see e.g. (Manning and Schütze, 1999, p.163)) is a statistical test for the significance of

[1] the context window size is not specified in Frantzi et al. (1998)

co-occurrence of two words. It relies on the probabilities of the term and its single words. The probability of a word $p(w)$ is defined as the frequency of the term divided by the total number of terms of the same length. The *t-test* statistic is computed using Equation 4 with $freq(.)$ being the total frequency of unigrams.

$$t(w_1 \dots w_n) \approx \frac{p(w_1 \dots w_n) - \prod_{i=1}^{n} p(w_i)}{\sqrt{p(w_1 \dots w_n)/freq(.)}} \quad (4)$$

We then use this score to rank the candidate terms.

3.5 FGM Score

Another method inspired by the C/NC-value is proposed in (Nakagawa and Mori, 2002; Nakagawa and Mori, 2003). The method was developed on a Japanese dataset and outperformed a modified C-value[2] measure. The method is composed of two scoring mechanisms for the candidate term t as shown in Equation 5.

$$FGM(t) = GM(t) \times MF(t) \quad (5)$$

The first term in the equation is a geometric mean $GM(.)$ of the number of distinct direct left $l(.)$ and right $r(.)$ neighboring words for each single word t_i within t.

$$GM(t) = \sqrt[2|t|]{\sum_{t_i \in t} (|l(t_i)| + 1)(|r(t_i)| + 1)} \quad (6)$$

The neighboring words are extracted directly from the corpus; the method does neither rely on candidate lists nor POS tags. To the contrary, the marginal frequency $MF(t)$ relies on the candidate list and the underlying corpus. This frequency counts how often the candidate term occurs within the corpus and is not a subset of a candidate. In Korkontzelos (2010) it was shown that while scoring according to Equation 5 leads to comparatively good results, it is consistently outperformed by MF only.

4 Semantic Uniqueness and Incompleteness

We present two new mechanisms relying on a Distributional Thesaurus (DT), which we use to rank terms regarding their multiwordness: A score for the *uniqueness* of a term and a punishing score that conveys the *incompleteness*.

[2]They adjust the logarithmic length in order to be able to use the C-value to detect single worded terms.

4.1 Similarity Computation

The DT is computed based on Biemann and Riedl (2013). First we extract n-grams from text and consider the left and the right neighbor of each n-gram as context feature. Then, we calculate the Lexicographer's Mutual Information (LMI) significance score (Bordag, 2008) between n-grams and features and remove all context features, which co-occur with more than 1000 terms, as these features tend to be to general. In the next step we keep for each n-gram only the 1000 context features, with the highest LMI score. The similarity score is then computed based on the overlap of features between two terms. Due to pruning this overlap-based significance measure is proportional to the Jaccard similarity measure, albeit we do not consider any normalization. After computing the feature overlap between two terms, we keep for each n-gram the 200 most similar n-grams. An example for the most similar n-grams to the terms *red blood cell* and *red blood* including their feature overlap are shown in Table 1.

4.2 Uniqness Computation

The first mechanism of our MWE ranking method is based on the following hypothesis: n-grams, which are MWE, could be substituted by single words, thus they have many single words amongst their most similar terms. This is motivated by semiotic considerations: Because of parsimony, concepts are usually expressed in single words. When a semantically non-compositional word combination is added to the vocabulary, it expresses a concept that is necessarily similar to other concepts. Hence, if a candidate multiword is similar to many single word terms, this indicates multiwordness.

To compute the *uniqueness* score (uq) of an n-gram t, we first extract the n-grams it is similar to using the DT as described in Section 4.1. The function $similarities(t)$ returns the 200 most similar n-grams to the given n-gram t. We then compute the ratio between unigrams and all similar n-grams considered using the formula:

$$uq(t) = \frac{\sum_{s:|s|=1}^{similarities(t)} 1}{|similarities(t)|}. \quad (7)$$

We illustrate the computation of our measure based on the MWE *red blood cell* and the non-MWE *red blood*. When considering only the ten most similar entries for both n-grams as illustrated

in Figure 1, we observe an uniqueness score of $7/10 = 0.7$ for both n-grams. If considering the

red blood cell		*red blood*	
Sim. term	Sc.	Sim. term	Sc.
erythrocyte	133	**red**	148
red cell	129	white blood	111
RBC	95	**Sertoli**	93
platelet	70	**Leydig**	92
red-cell	37	**NK**	86
reticulocyte	34	**mast**	85
white blood	33	**granulosa**	81
leukocyte	29	**endothelial**	81
granulocyte	28	hematopoietic stem	79
the erythrocyte	28	peripheral blood monon	78

Table 1: We show the ten most similar entries for the term *red blood cell* (left) and *red blood* (right). Here, seven out of ten terms are single words.

top 200 similar n-grams, which are also used in our experiments we will obtain 135 unigrams for the candidate *red blood cell* and 100 unigrams for the n-gram *red blood*. We will use these counts for showing the workings of the method in the remainder.

4.3 Incompleteness Computation

Similar to the C/NC-value method, we also assign a context weighting function that punishes incomplete terms, which we call *incompleteness (ic)*. For this function we extract the 1000 most significant context features using the function $context(t)$, which yields tuples of left and right contexts. These context features are the same that are used for the similarity computation in Section 4.1 and have been ranked according to the LMI measure. For the example term *red blood*, some of the contexts are ⟨*extravasated, cells*⟩, ⟨*uninfected, cells*⟩, ⟨*nucleated, corpuscles*⟩. In the next step we split each tuple to its left and right word including its relative position (left/right) to the candidate term. Using the first context feature results in: ⟨*extravasated, left*⟩, ⟨*cells, right*⟩. Then, we sum up the occurrences of for each single context, as shown in Table 2 for the two terms.

We subsequently select the maximal count and normalize it by the counts of features $|context(t)|$ considered, which is 1000. This results into the incompleteness measure $ic(t)$. For our example terms we achieve the values $ic(red\ blood) = 557/1000$ and $ic(red\ blood\ cell) = 48/1000$. Whereas the uniqueness scores for the most similar entries were equal, we now have a measure that indicates the incompleteness of an n-gram, with higher scores indicating more incomplete terms.

Context term	Position	Count
red blood cell		
transfusions	right	48
(	right	42
transfusion	right	33
red blood		
cells	right	557
cell	right	344
corpuscles	right	13

Table 2: Top three most frequent context words for the term *red blood cell* and *red blood* in the Medline corpus.

4.4 Combining Both Measures

As shown in the previous two sections, a high uniqueness score indicates the multiwordness and a high incompleteness score should decrease the overall score. In experiments, we found the best combination if we subtract[3] the incompleteness score from the uniqueness score. This mechanism is inspired by the C-value and motivated as terms that are often preceded/followed by the same word do not cover the full multiword expression and need to be downranked. This leads to Equation 8, which we call **D**ist**R**ibutional **U**niqueness and **I**ncompleteness **D**egree:

$$\texttt{DRUID}(t) = \mathrm{uq}(t) - \mathrm{ic}(t). \quad (8)$$

Applying the DRUID score to our example terms (considering the 200 most similar terms) we will achieve the scores $\texttt{DRUID}(red\ blood\ cell) = 135/200 - 48/1000 = 0.627$ and $\texttt{DRUID}(red\ blood) = 100/200 - 557/1000 = -0.057$. As a higher DRUID score indicates the multiwordness of an n-gram, we can summarize that the n-gram *red blood cell* is a better MWE than the n-gram *red blood*.

5 Experimental Setting

We examine two experimental settings: First, we compute all measures on a small corpus that has been annotated for MWEs, which serves as the gold standard. In the second setting we compute the measures on a larger in-domain corpus. The evaluation is again performed for the same candidate terms as given by the gold standard. Results for the top k ranked entries are reported using the precision at k ($P@k = \frac{1}{k}\sum_{i=1}^{k} x_i$ with x_i equals 1 if the i-th ranked candidate is annotated as MWE and 0 otherwise). For an overall performance we

[3] multiplicative combinations consistently performed worse

use the average precision (AP) as defined in Thater et al. (2009): $AP = \frac{1}{|T_{mwe}|} \sum_{k=1}^{|T|} x_k P@k$, with T_{mwe} beeing the set of positive MWE. When facing tied scores we mix false and true candidates randomly cf. Cabanac et al. (2010).

5.1 Corpora

For the experiments we consider two annotated (small) corpora and two unannotated (large) corpora.

5.1.1 GENIA corpus and SPMRL 2013: French Treebank

In the first experiments we use two small annotated corpora that serve the gold standard MWEs. We use the medical GENIA corpus (Kim et al., 2003)[4] which consists of 1999 abstracts from Medline[5] and encompasses 0.4 million words. This corpus has annotations regarding important and biomedical terms. Also single terms are annotated in this data set, which we ignore.

The second small corpus is based on the French Treebank (Abeillé and Barrier, 2004), which was extended for the SPMRL task (Seddah et al., 2013). This version of the corpus also contains compounds annotated as MWEs. In our experiments we use the training data, which covers 0.4 million words.

Whereas the GENIA MWEs target term matching and medical information retrieval, the SPMRL MWEs mainly focus on improving parsing through compound recognition.

5.1.2 Medline Corpus and Est Républican Corpus (ERC)

In a second experiment the scalability to larger corpora is tested. For this, we make use of the entire Medline[5] abstracts, which consist of about 1.1 billion words. The Est Républican Corpus (ERC) (Seddah et al., 2012)[6] is our large French corpus. It consists of local French news from the eastern part of France and comprises of 150 million words.

5.2 Candidate Selection

In the first two experiments, we use POS filters to select candidates. We concentrate on filters that extract noun MWEs and avoid further preprocessing like lemmatization. We use the filter introduced by Justeson and Katz (1995)[7] for the English medical datasets. Considering only terms that appear more than ten times leads to 1,340 candidates for the GENIA dataset and 29,790 candidates for the Medline dataset. According to Table 3 we observe that most candidates are bigrams. Whereas for both corpora still around 20% of trigrams are contained, the number of 4-grams is only marginally represented. For the French datasets we apply the filter proposed by Daille et al. (1994)[8], which is suited to match nominal MWEs. Applying the same filtering as for the medical corpora leads to 330 candidate terms for the SPMRL and 7,365 candidate terms for the ERC. Here the ratio between bi- and trigrams is more balanced but again the number of 4-grams constitutes the smallest class.

Corpus	Candidates	2-gram	3-gram	4-gram
GENIA	1,340	1,056	243	41
Medline	29,790	22,236	6,400	1,154
SPMRL	330	197	116	17
ERC	7,365	3,639	2,889	837

Table 3: Number of candidates after filtering for the expected POS-tag and their distribution over n-grams with $n \in \{1, 2, 3, 4\}$.

In comparison to the Medline dataset, the ratio of multiwords extracted by the POS filter on the French corpus is much lower. The reason for that property is that in the French data, many adverbial, prepositional MWEs are annotated, which are not covered by the POS filter.

The third experiment shows the performance of the method in absence of language-specific preprocessing. Thus, we only filter the candidates by frequency and do not make use of POS filtering. As most previous methods rely on POS-filtered data we cannot make use of them in this setting.

For the evaluation, we compute the scores of the competitive methods in two settings: First, we compute the scores based on the full candidate list without any frequency filter and prune low-frequent candidates only for the evaluation (post-prune). In the second setting we filter candidates

[4] freely available at `http://www.nactem.ac.uk/genia/genia-corpus/pos-annotation`.

[5] `http://www.nlm.nih.gov/bsd/licensee/access/medline_pubmed.html`

[6] `http://www.cnrtl.fr/corpus/estrepublicain`

[7] A regular expression for matching POS tag sequences is summarized by Korkontzelos (2010): `(([JN]+[JN]?[NP]?[JN]?)N)`. Each letter is a truncated POS tag of length one where J is an adjective N a noun and P a preposition.

[8] Following the same convention as for English the regular expression can be expressed as `N[J]?|NN|NPDN`

according to their frequency before the computation of scores (pre-prune). This leads to differences for context-aware measures, since in the pre-pruned case, a lower number of less noisier contexts is used.

6 Results

6.1 Small Corpora Results

First we show the results based on the GENIA corpus (see Table 4). Almost all competitive methods

Method	P@100	P@500	AP
upper baseline	1.000	1.000	1.0000
lower baseline	0.713	0.713	0.7134
frequency	0.790	0.750	0.7468
t-test	0.790	0.750	0.7573
C-value (pre-pruned)	0.880	0.846	0.8447
NC-value (pre-pruned)	0.880	0.840	0.8405
GM	0.590	0.662	0.6740
MF (pre-pruned)	0.920	0.872	0.8761
FGM (pre-pruned)	0.910	0.840	0.8545
MF (post-pruned)	0.900	0.876	0.8866
FGM (post-pruned)	0.900	0.900	0.8948
DRUID	0.930	0.852	0.8663
log(freq)(DRUID)	**0.970**	0.860	0.8661
MF(post-pruned)DRUID	0.950	0.926	**0.9241**
FGM(post-pruned)DRUID	**0.960**	**0.940**	**0.9262**

Table 4: Results for the GENIA dataset.

beat the lower baseline. The C/NC-value perform best when the pruning is done after a frequency filter. In line with the findings of Korkontzelos (2010) and in contrast to Frantzi et al. (1998) the AP of the C-value is slightly higher than for the NC-value. All the FGM based methods except the GM measure alone outperform the C-value. The results in Table 4 indicate that the best competitive system is the post-pruned FGM system as it has much higher average precision scores and misses only 50 MWEs in the first 500 entries. A slightly different picture is presented in Figure 1 where the $P@k$ scores against the number of candidates are plotted. Here DRUID performs well for the top-k list for small k, i.e. finds many good MWEs with high confidence thus combines well with MF, which extends to larger k, but places too much importance of frequency when used alone. Common errors are frequent chunks such as "in patience", see Table 9 in Section 7. Whereas for the post-pruned case FGM scores higher than MF, the inverse is observed when pre-pruning. Using our DRUID methods can surmount the FGM method only for the first 300 ranked terms (see Figure 1 and Table 4). Multiplying the logarithmic frequency to the DRUID, the results improve

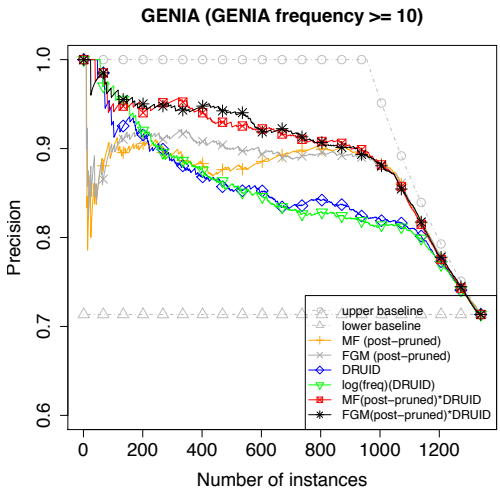


Figure 1: Results for the GENIA corpus.

slightly and the best P@100 with 0.97 is achieved. All FGM results are outperformed when combining the post-pruned FGM scores with our measure. According to Figure 1 this combination achieves high precision for the first ranked candidates and still exploits the good performance of the post-pruned FGM based method for the middle-ranked candidates.

Different results are achieved for the SPMRL dataset as can be seen in Table 5. Whereas the pre-pruned C-value again receives better results than frequency, it scores below the lower baseline. Also the post-pruned FGM and MF method

Scoring	P@100	P@200	AP
upper baseline	1.000	0.860	1.0000
lower baseline	0.521	0.521	0.5212
frequency	0.500	0.480	0.4876
t-test	0.500	0.485	0.4934
C-value (pre-pruned)	0.490	0.540	0.5107
MF (post-pruned)	0.510	0.495	0.5017
FGM (post-pruned)	0.460	0.480	0.4703
DRUID	**0.790**	**0.690**	**0.7794**
log(freq)DRUID	0.770	0.675	0.7631
MF(post-pruned)DRUID	0.700	0.630	0.6850
FGM(post-pruned)DRUID	0.600	0.570	0.5948

Table 5: Results for the French SPMRL dataset

do not exceed the lower baseline. Data analysis revealed that for the French dataset only ten out of the 330 candidate terms are nested within any of the candidates. This is much lower than the 637 terms nested in the 1340 candidate terms for the GENIA dataset. As both the FGM-based methods and the C/NC-value heavily rely on nested candidates, they cannot profit from the candidates of this dataset and achieve similar scores as ordering candidates according to frequency. Comparing the

baselines to our scoring method, this time we obtain the best result for DRUID without additional factors. However, multiplying DRUID with MF or log(frequency) still outperforms the other methods and the baselines.

6.2 Large Corpora Results

Most MWE evaluations have been performed on rather small corpora. Here we want to inspect the performance of the measures for large corpora, so as to realistically simulate a situation where the MWEs should be found automatically for an entire domain.

Using the Medline corpus, all methods except the GM score outperform the lower baseline and the frequency baseline (see Table 6). Regarding the AP the best results are obtained when combining our DRUID method with the MF, whereas for $P@100$ and $P@500$ the log-frequency weighted DRUID scores best. Using solely the DRUID method or the combined variation with the log-frequency lead to the best ranking for the first 1000 ranked candidates and is then outperformed by the MF based DRUID variations. In this experiment the C-value achieves the best performance from the competitive methods for the $P@100$ and $P@500$, followed by the t-test. But the highest AP is reached with the post-pruned MF method, which also outperforms the sole DRUID slightly. Contrary to the GENIA results, the MF scores are consistently higher than the FGM scores.

Scoring	$P@100$	$P@500$	AP
upper baseline	1.000	1.000	1.0000
lower baseline	0.416	0.416	0.4161
frequency	0.720	0.534	0.4331
C-value (pre-pruned)	0.750	0.564	0.4519
t-test	0.720	0.542	0.4483
GM	0.210	0.272	0.3502
MF (pre-pruned)	0.550	0.542	0.4578
FGM (pre-pruned)	0.580	0.478	0.4200
MF (post-pruned)	0.530	0.500	0.4676
FGM (post-pruned)	0.490	0.446	0.4336
DRUID	0.770	0.686	0.4608
log(freq)*DRUID	**0.860**	**0.720**	0.4693
GM*DRUID	0.770	0.634	0.4497
MF(pre-pruned)*DRUID	0.730	0.634	**0.4824**
MF(post-pruned)*DRUID	0.730	0.626	**0.4889**

Table 6: Results computed on the Medline corpus.

When using the French ERC we figured out that no nested terms are found within the candidates. Thus, the post- and pre-pruned settings are equivalent and thus MF equals frequency. The best results are again obtained with our method with and without the logarithmic frequency weighting (see Table 7). Again the AP of the C-value and most of the FGM-based methods are inferior to the frequency scoring. Only the t-test and the MF are slightly higher than the frequency[9]. But in contrast to the results based on the smaller SPMRL dataset, the MF, FGM and C-value can outperform the lower baseline. In comparison to the smaller corpora, the performance for the larger corpora is much lower. Especially low-frequent terms in the small corpora that have high frequencies in the larger corpora have not been annotated as MWEs.

Method	$P@100$	$P@500$	AP
upper baseline	1.000	1.000	1.0000
lower baseline	0.220	0.220	0.2201
frequency	0.370	0.354	0.3105
C-value	0.420	0.366	0.3059
t-test	0.390	0.360	0.3134
GM	0.010	0.052	0.1694
MF	0.370	0.356	0.3148
FGM	0.280	0.260	0.2405
DRUID	0.700	0.568	0.3962
log(freq)DRUID	**0.760**	**0.582**	**0.4075**
MF*DRUID	0.570	0.516	0.3776
FGM*DRUID	0.510	0.418	0.3234

Table 7: Results computed based on the ERC.

6.3 Results without POS Filtering

In the last experiment, we apply our method to candidates without any POS filtering and report results using a frequency threshold of ten. As the competitive methods from the previous section rely on POS tags, we use the t-test for comparison. Analysis revealed that the top-scored candidates according to the t-test begin with stop words.

		Medical		French	
	Method	$P@100$	AP	$P@100$	AP
small corpora	upper baseline	1.000	1.0000	1.000	1.0000
	lower baseline	0.107	0.1071	0.083	0.0832
	frequency	0.150	0.1135	0.060	0.0906
	t-test	0.160	0.1261	0.080	0.1097
	t-test + sw	0.530	0.3643	0.180	0.1481
	DRUID	**0.700**	**0.4048**	**0.670**	**0.2986**
	log(freq)DRUID	0.690	0.3644	0.460	0.2527
large corpora	upper baseline	1.000	1.0000	1.000	1.0000
	lower baseline	0.036	0.0361	0.019	0.0191
	frequency	0.010	0.0361	0.060	0.0366
	t-test	0.020	0.0412	0.080	0.0440
	t-test + sw	0.000	0.0989	0.200	0.0485
	DRUID	0.610	0.1378	**0.660**	**0.1009**
	log(freq)DRUID	**0.760**	**0.1649**	0.600	0.0988

Table 8: Results without linguistic pre-processing.

[9]This is achieved by chance for the MF, as it is equal to the frequency. The different scores are due to the randomly sorted tied scores used during our evaluation and reflect the variance of the randomness.

As an additional heuristic for the t-test, we shift MWEs, which start or end with one of the ten most frequent words, to the last ranks. For the smaller dataset the best results are achieved with the sole DRUID (see Table 8) and the frequency weighting does not seem to be beneficial, as highly frequent n-grams ending with stopwords are ranked higher in absence of POS filtering. This, however, is not observed for larger corpora. Here the best results for Medline are achieved with the frequency weighted DRUID. Whereas for French, the sole DRUID method performs best, the difference between the DRUID and the log-frequency-weighted DRUID is rather small. The low APs throughout can be explained by the large number of considered candidates. The second best scores are achieved with stop word based t-test (t-test + sw). C-value performs en par with frequency.

6.4 Components of DRUID

Here, we show different parameters for DRUID, relying on the English GENIA dataset without POS filtering of MWE candidates and by considering only terms with a frequency of 10 or more. Inspecting the two different components of the DRUID measure (see upper graph in Figure 2), we observe that the uniqueness measure contributes most to the DRUID score. The main effect of the incompleteness component is the downranking of a rather small number of terms with high uniqueness scores, which improves the overall ranking. We can also see that for the top ranked terms the negative incompleteness score does not improve over the frequency baseline but outperforms the frequency in the middle ranked candidates. Used in DRUID we observe a slight improvement for the complete ranking. We achieve a P@500 of 0.474 for the uniqueness scoring and 0.498 for the DRUID score.

When filtering similar entries, used for the uq scoring, by their similarity score (see bottom graph in Figure 2), we observe that the amount of similar n-grams considered seems to be more important then the quality of the similar entries: With the increasing filtering also the quality of extracted candidate MWEs diminishes.

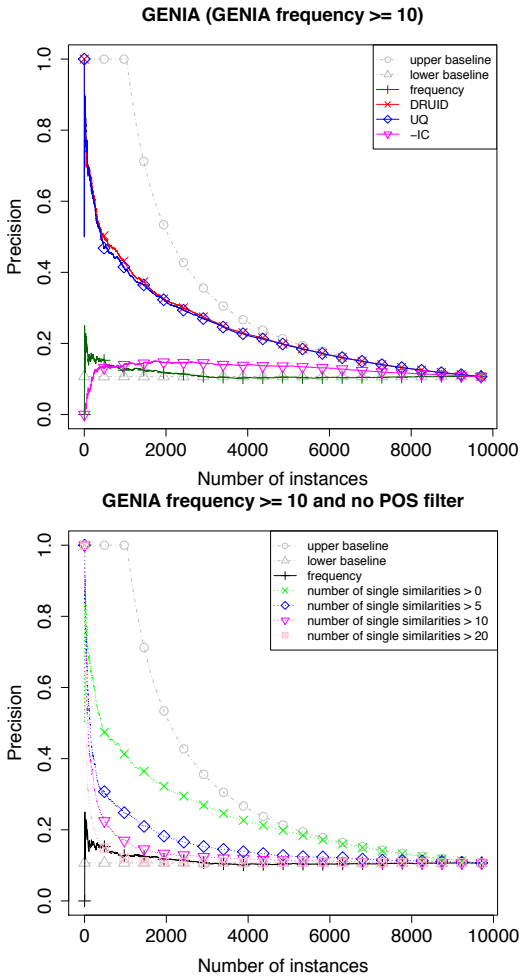


Figure 2: Results for the components of the DRUID measure (top) and for different filtering thresholds of the similar entries considered for the uniqueness scoring (bottom).

7 Discussion and Data Analysis

The experiments confirm that our DRUID measure, either weighted with the MF or alone, works best across two languages and across different corpus sizes. It also achieves the best results in absence of POS filtering for candidate term extraction. The optimal weighting of DRUID depends on the nestedness of the MWEs: Using DRUID with the MF should be used when there are more than 20% of nested candidates and using the log-frequency or no frequency weighting when there are almost no nested candidate terms.

We show the best-ranked candidates obtained with our method and with the best competitive method in terms of P@100 for the two smaller corpora. Using the GENIA dataset, our log-frequency based DRUID (see left column in Table 9) ranks only true MWE within the 15 top-scored candidates.

The right-hand side shows results extracted with the pre-pruned MF method that yields three non-MWE terms. Whereas that could be a POS error,

log(freq)DRUID		MF (pre-pruned)	
NF-kappa B	1	T cells	1
transcription factors	1	NF-kappa B	1
transcription factor	1	transcription factors	1
I kappa B alpha	1	activated T cells	1
activated T cells	1	T lymphocytes	1
nuclear factor	1	human monocytes	1
human monocytes	1	I kappa B alpha	1
gene expression	1	nuclear factor	1
T lymphocytes	1	gene expression	1
NF-kappa B activation	1	NF-kappa B activation	1
binding sites	1	in patients	0
MHC class II	1	important role	0
tyrosine phosphorylation	1	binding sites	1
transcriptional activation	1	in B cells	0
nuclear extracts	1	transcriptional activation	1

Table 9: Top ranked candidates from the GENIA dataset using our method (left) and the competitive method (right). Each term is marked, whether the term is an MWE (1) or not (0).

the MF and also the C-value are not capable to remove terms starting with stop words. The DRUID score alleviates this problem with the uniqueness factor. For the French dataset our method ranks only one false candidate whereas the MF (post-pruned) ranks eight non-annotated candidates in the top 15 list (see Table 10).

DRUID		MF (post-pruned)	
hausse des prix	1	milliards de francs	0
mise en oeuvre	1	millions de francs	0
prise de participation	1	Etats - Unis	1
chiffre d' affaires	1	chiffre d' affaires	1
formation professionnelle	1	taux d' intérêt	1
population active	1	milliards de dollars	0
taux d' intérêt	1	millions de dollars	0
politique monétaire	1	Air France	1
Etats - Unis	1	% du capital	0
Réserve fédérale	1	milliard de francse	0
comit d' tablissement	1	directeur général	1
projet de loi	1	M. Jean	0
système européen	0	an dernier	1
conseil des ministres	1	années	1
Europe centrale	1	% par rapport	0

Table 10: Top ranked candidates from the SPMRL dataset for the best DRUID method (left) and the best competitive method (right). Each term is marked, if it is an MWE (1) or not (0).

Whereas the unweighted DRUID method scores better than its competitors on the large corpora, the best results are achieved when using DRUID with frequency-based weights on the smaller corpora. For a direct comparison we evaluated the small and large corpora using an equal candidate set. We observed that all methods computed on the large corpora achieve slightly inferior results than when computing them using the small corpora. Data analysis revealed that we would consider many of the high ranked "false" candidates as MWE.

Therefore we extracted the top ten ranked terms, which are not annotated as MWE from the methods with the best $P@100$ performance, resulting to the log(freq) DRUID and the pre-pruned C-value methods.

First, we observed that the first 'false' candidate for our method appears at rank 26 and at rank 1 for the C-value. Additionally, only ten out of the top 74 candidates are not annotated as MWEs for our method and 48 for the competitor. When searching the terms within the MeSH dictionary[10], we find seven terms ranked from our method and two for the competitive method.

8 Conclusion

Uniqueness is a new mechanism in MWE modeling. Whereas frequency and co-occurrence have been captured in many previous approaches (see Manning and Schütze (1999), Ramisch et al. (2012) and Korkontzelos (2010) for a survey), we boost multiword candidates t by their grade of distributional similarity with single word terms. We implement such contextual substitutability with a model where the term t can consist of multiword tokens and similarity is measured based on the right and neighboring word between all (single and multiword) terms. Since it is the default to express concepts with single words, a high uniqueness score is given to multiwords that belong to a category just as single words would. E.g. for an English open-domain corpus *hot dog* is most similar to the terms: *food*, *burger*, *hamburger*, *sausage* and *roadside*. Candidates with a low number of single word similarities also serve the same function, but more frequently we observe single n-grams with function words or modifying adjectives concatenated with content words, i.e. *small dog* is most similar to "*various cat*", "*large amount of*", "*large dog*", "*certain dog*", "*dog*". To be able to kick in, the measure requires a certain minimum frequency for candidates in order to find enough contextual overlap with other terms. Additionally, we also demonstrate effective performance on larger corpora and show its applicability when used in a complete unsupervised evaluation setting.

[10] http://www.nlm.nih.gov/mesh/

Acknowledgment

This work has been supported by the German Federal Ministry of Education and Research (BMBF) within the context of the Software Campus project LiCoRes under grant No. 01IS12054 and the Deutsche Forschungsgemeinschaft (DFG) within the SeMSch project.

References

Anne Abeillé and Nicolas Barrier. 2004. Enriching a French Treebank. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC '04)*, pages 2233–2236, Lisbon, Portugal.

Dimitra Anastasiou. 2010. *Idiom Treatment Experiments in Machine Translation*. Ph.D. thesis, Universität des Saarlandes, Saarbrücken, Germany.

Chris Biemann and Martin Riedl. 2013. Text: Now in 2D! A Framework for Lexical Expansion with Contextual Similarity. *Journal of Language Modelling*, 1(1):55–95.

Stefan Bordag. 2008. A Comparison of Co-occurrence and Similarity Measures As Simulations of Context. In *Proceedings of the 9th International Conference on Computational Linguistics and Intelligent Text Processing (CICLing '14)*, pages 52–63, Haifa, Israel.

Dhouha Bouamor, Nasredine Semmar, and Pierre Zweigenbaum. 2012. Identifying bilingual Multi-Word Expressions for Statistical Machine Translation. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC '12)*, pages 674–679, Istanbul, Turkey.

Guillaume Cabanac, Gilles Hubert, Mohand Boughanem, and Claude Chrisment. 2010. Tie-breaking Bias: Effect of an Uncontrolled Parameter on Information Retrieval Evaluation. In *Conference on Multilingual and Multimodal Information Access Evaluation (CLEF)*, pages 112–123, Padua, Italy.

Béatrice Daille, Éric Gaussier, and Jean-Marc Langé. 1994. Towards automatic extraction of monolingual and bilingual terminology. In *Proceedings of the 15th Conference on Computational Linguistics - Volume 1*, COLING '94, pages 515–521, Kyoto, Japan.

Stefan Evert. 2005. *The Statistics of Word Cooccurrences: Word Pairs and Collocations.* Ph.D. thesis, Institut für maschinelle Sprachverarbeitung, University of Stuttgart.

Stefan Evert. 2008. A lexicographic evaluation of German adjective-noun collocations. In *Proceedings of the LREC 2008 Workshop Towards a Shared Task for Multiword Expressions (MWE '08)*, pages 3–6, Marrakech, Morocco.

Katerina T. Frantzi, Sophia Ananiadou, and Jun-ichi Tsujii. 1998. The C-value/NC-value Method of Automatic Recognition for Multi-Word Terms. In *Proceedings of the Second European Conference on Research and Advanced Technology for Digital Libraries (ECDL '98)*, pages 585–604, Heraklion, Greece.

Karl Moritz Hermann and Phil Blunsom. 2014. Multilingual Models for Compositional Distributed Semantics. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL '14)*, pages 58–68, Baltimore, MA, USA.

John S. Justeson and Slava M. Katz. 1995. Technical terminology: some linguistic properties and an algorithm for identification in text. *Natural Language Engineering*, 1:9–27, 3.

Graham Katz and Eugenie Giesbrecht. 2006. Automatic Identification of Non-compositional Multi-word Expressions Using Latent Semantic Analysis. In *Proceedings of the Workshop on Multiword Expressions: Identifying and Exploiting Underlying Properties (MWE '06)*, pages 12–19, Sydney, Australia.

Jin-Dong Kim, Tomoko Ohta, Yuka Tateisi, and Jun ichi Tsujii. 2003. GENIA corpus - a semantically annotated corpus for bio-textmining. *Bioinformatics*, 19(suppl 1):i180–182.

Ioannis Korkontzelos. 2010. *Unsupervised Learning of Multiword Expressions*. Ph.D. thesis, University of York, UK.

Juan Antonio Lossio-Ventura, Clement Jonquet, Mathieu Roche, and Maguelonne Teisseire. 2014. Yet Another Ranking Function for Automatic Multi-word Term Extraction. In *Proceedings of the 9th International Conference on Natural Language Processing (PolTAL '14)*, pages 52–64, Warsaw, Poland.

Chris Manning and Hinrich Schütze. 1999. *Foundations of statistical natural language processing*. MIT Press, Cambridge, MA.

Hiroshi Nakagawa and Tatsunori Mori. 2002. A simple but powerful automatic term extraction method. In *COLING-02 on COMPUTERM 2002: Second International Workshop on Computational Terminology - Volume 14*, COMPUTERM '02, pages 1–7, Taipei, Taiwan.

Hiroshi Nakagawa and Tatsunori Mori. 2003. Automatic term recognition based on statistics of compound nouns and their components. *Terminology*, 9(2):201–219.

Pavel Pecina. 2010. Lexical association measures and collocation extraction. *Language Resources and Evaluation*, 44:137–158.

Carlos Ramisch, Vitor De Araujo, and Aline Villavicencio. 2012. A broad evaluation of techniques for automatic acquisition of multiword expressions. In *Proceedings of the Student Research Workshop of the 50th Meeting of the Association for Computational Linguistics (ACL Student Workshop '12)*, pages 1–6, Jeju Island, Korea.

Carlos Ramisch. 2012. *A generic and open framework for multiword expressions treatment: from acquisition to applications*. Ph.D. thesis, Universidade Federal Do Rio Grande do Sul.

Ivan A. Sag, Timothy Baldwin, Francis Bond, Ann Copestake, and Dan Flickinger. 2001. Multiword expressions: A pain in the neck for nlp. In *Proceedings of the 3rd International Conference on Intelligent Text Processing and Computational Linguistics (CICLing-2002)*, pages 1–15, Mexico City, Mexico.

Bahar Salehi, Paul Cook, and Timothy Baldwin. 2014. Using distributional similarity of multi-way translations to predict multiword expression compositionality. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics (EACL '14)*, pages 472–481, Gothenburg, Sweden.

Patrick Schone and Daniel Jurafsky. 2001. Is Knowledge-Free Induction of Multiword Unit Dictionary Headwords a Solved Problem? In *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing (EMNLP '01)*, pages 100–108, Pittsburgh, PA, USA.

Djamé Seddah, Marie Candito, Benoit Crabbé, and Enrique Henestroza Anguiano. 2012. Ubiquitous Usage of a Broad Coverage French Corpus: Processing the Est Republicain corpus. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC '12)*, pages 3249–3254, Istanbul, Turkey.

Djamé Seddah, Reut Tsarfaty, Sandra Kübler, Marie Candito, Jinho D. Choi, Richárd Farkas, Jennifer Foster, Iakes Goenaga, Koldo Gojenola Galletebeitia, Yoav Goldberg, Spence Green, Nizar Habash, Marco Kuhlmann, Wolfgang Maier, Joakim Nivre, Adam Przepiórkowski, Ryan Roth, Wolfgang Seeker, Yannick Versley, Veronika Vincze, Marcin Woliński, Alina Wróblewska, and Eric Villemonte de la Clergerie. 2013. Overview of the SPMRL 2013 shared task: A cross-framework evaluation of parsing morphologically rich languages. In *Proceedings of the Fourth Workshop on Statistical Parsing of Morphologically-Rich Languages*, pages 146–182, Seattle, WA, USA.

Stefan Thater, Georgiana Dinu, and Manfred Pinkal. 2009. Ranking Paraphrases in Context. In *Proceedings of the 2009 Workshop on Applied Textual Inference (TextInfer '09) in conjunction with the ACL '09*, pages 44–47, Suntec, Singapore.

Joachim Wermter and Udo Hahn. 2005. Effective grading of termhood in biomedical literature. In *Annual AMIA Symposium Proceedings*, pages 809–813, Washington D.C., USA.

Syntactic Dependencies and Distributed Word Representations for Chinese Analogy Detection and Mining

Likun Qiu[1,2], **Yue Zhang**[2], **Yanan Lu**[3]
[1]School of Chinese Language and Literature, Ludong University, China
[2]Singapore University of Technology and Design, Singapore
[3]Computer School, Wuhan University, China
qiulikun@pku.edu.cn, yue_zhang@sutd.edu.sg, luyanan@whu.edu.cn

Abstract

Distributed word representations capture relational similarities by means of vector arithmetics, giving high accuracies on analogy detection. We empirically investigate the use of syntactic dependencies on improving Chinese analogy detection based on distributed word representations, showing that a dependency-based embeddings does not perform better than an ngram-based embeddings, but dependency structures can be used to improve analogy detection by filtering candidates. In addition, we show that distributed representations of dependency structure can be used for measuring relational similarities, thereby help analogy mining.

1 Introduction

Relational similarity measures the correspondence between word-word relations (Medin et al., 1990). It is relevant to many tasks in NLP (Turney, 2006), such as word sense disambiguation, information extraction, question answering, information retrieval, semantic role identification and metaphor detection. Typical tasks on relational similarity include *analogy detection*, which measures the degree of relational similarities, and *analogy mining*, which extracts analogous word pairs from unstructured text.

Recently, distributed word representations (i.e. *embeddings*) (Mikolov et al., 2013a; Mikolov et al., 2013b; Levy and Goldberg, 2014b) have been used for unsupervised analogy detection. Mikolov et al. use attributional similarities between words in a relation to compute relational similarities, and show that the method outperforms the best system in the SemEval 2012 shared task on analogy detection. Levy and Goldberg (2014b) further improve Mikolov's relational similarity measure method using novel arithmetic combinations of attributional similarities. For simplicity, we call the method of Mikolov et al. *embedding-based analogy detection*, without stressing the difference between *distributed* and *distributional* (i.e. counting-based) word representations.

Most work on embedding-based analogy detection uses relational similarities as a measure of the quality of embeddings. However, relatively little has been done in the opposite direction, exploring how to leverage embeddings for improving relational similarity algorithms. We empirically study the use of word embeddings for Chinese analogy detection and mining, leveraging syntactic dependencies, which has been shown to be closely associated with semantic relations (Levin, 1993; Chiu et al., 2007). Compared with many other languages, this association is particularly strong for Chinese, which is fully configurational and lacks morphology. To our knowledge, relatively little work has been reported on Chinese relational similarities, compared to other tasks in Chinese NLP, including syntactic parsing, information extraction and machine translation.

We work on three specific problems. First, we study the effect of dependency-based word embeddings for analogy detection. There are two variations of Mikolov et al's *skip-gram* embedding model, one training the distributed word representation of a word using its context words in local ngram window (Mikolov et al., 2013a), and the other training the distributed representation of a word using words in a syntactic dependency context (Levy and Goldberg, 2014b; Bansal et al., 2014). The latter has attracted much recent atten-

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 2441–2450,
Lisbon, Portugal, 17-21 September 2015.

tion due to its potential in capturing more syntactic regularities. It has been shown to outperform the former in a variety of NLP tasks, and can potentially also improve relation similarity. Our experiments on both English and Chinese show that the dependency-context embeddings consistently under-perform ngram-context embeddings. We give some theoretical justifications to the findings.

Second, we propose to use syntactic dependencies as a context for improving embedding-based analogy detection, pruning the search space and filtering noise using syntactic dependencies. While highly useful for measuring relational similarities, attributional similarities between words are not the only source of information for analogy detection. Traditional methods, such as Turney and Littman (2005), Turney (2006), Chiu et al. (2007) and Ó Séaghdha and Copestake (2009), also leverage context between word pairs in a corpus for better accuracies, which the current embedding-based methods ignore. Results show that our proposed method achieves significant improvements for this task.

Third, we show that a novel distributed representation of syntactic dependencies between word pairs can be used to mine analogous dependencies from a large Chinese corpus. Inspired by the fact that distributed word representations can be used to measure word similarities, we use our distributed dependency representations to measure relation similarities. We propose a bootstrapping algorithm for analogy mining using dependency embeddings, and experiments on a large Chinese corpus show that the method can achieve a precision of 95.2% at a recall of 56.8%.

Our automatically-parsed corpus, trained embeddings and evaluation datasets are released publicly at `http://people.sutd.edu.sg/~yue_zhang/publication.html`. To our knowledge, we are the first to present results on Chinese analogy detection and to release large-scale Chinese word embeddings.

2 Background

2.1 Relational Similarity Tasks

There are three main tasks for relational similarity. This first is *relation classification*, which has been used in Task 2 of SemEval 2012 (Jurgens et al., 2012). In this task, all four words in two word pairs are given, and one needs to judge whether they belong to a same relation type. In order to address this task, various supervised methods have been used (Bollegala et al., 2008; Herdağdelen and Baroni, 2009; Turney, 2013).

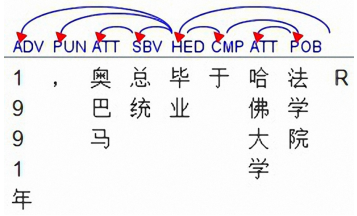


Figure 1: Dependency tree of the sentence "1991年 (in 1991) ， (,) 奥巴马 (Obama) 总统 (President) 毕业 (graduate) 于 (from) 哈佛 (Harvard) 法学院 (Law School)".

The second task is *analogy detection* (Mikolov et al., 2013b), which takes three words in two word pairs, and searches for a most suitable word from the vocabulary to recover the hidden word. This task has been addressed using word embeddings (Mikolov et al., 2013b; Levy and Goldberg, 2014b).

The third task is *analogy mining* (Chiu et al., 2007), which takes one word pair belonging to a certain semantic relation as a seed, and searches for all the word pairs that share the same relation with the seed. Compared with relation classification and analogy detection, analogy mining can be practically more useful because it requires less given information, and provides a large quantity of analogous word pairs automatically.

2.2 Skip-gram Word Embeddings

As a by-product of neural language models (Bengio et al., 2003; Mnih and Hinton, 2007), word embeddings are distributed vector representations of words, trained using local contexts. They capture linguistic regularities in languages (Mikolov et al., 2013b) and have been used in various tasks (Collobert and Weston, 2008; Turian et al., 2010; Socher et al., 2011).

In this paper, we apply the Skip-gram method of Mikolov et al. (2013a) for training embeddings, which works by maximizing the probability of a word given a context of multiple words. Mikolov et al. (2013b) use an ngram window as the context, and observe that the resulting embeddings are highly useful for unsupervised analogy detection.

2.3 Embedding-based Analogy Detection

Formally, the task of analogy detection is to find a word $b*$ given a pair of words *a:b* and a word $a*$ such that *a*:b** is analogous to *a:b*. Mikolov et al. (2013b) show that the task can be solved by finding a word that maximizes:

$$score = sim(b*, b - a + a*) \quad (1)$$

where sim is a similarity measure, typically the $cosine$ function. Levy and Goldberg (2014b) show that the Equation 1 is equivalent to:

$$score = cos(b*, b) - cos(b*, a) + cos(b*, a*) \quad (2)$$

As a result, the goal of analogy detection is to find a word $b*$ which is similar to b and $a*$ but different from a. Levy and Goldberg (2014b) further propose to substitute the addictive functions in Equation 2 with multiplicative functions:

$$score = cos(b*, b)cos(b*, a*)/(cos(b*, a) + \varepsilon) \quad (3)$$

Here $\varepsilon = 0.001$ is used to prevent division by zero. Their experiments show that the use of Equation 3 can improve the state-of-the-art. Following Levy and Goldberg (2014b), we refer to Equation 1 and 2 as 3COSADD and Equation 3 as 3COSMUL, respectively.

2.4 Chinese Relational Similarity

There are various types of relational similarities. Syntactically, inflections can be treated as a type of word-word relation (Mikolov et al., 2013b). For example, the comparative pairs "good:better" and "rough:rougher" are analogous, and the past tense inflections "see:saw" and "return:returned" are analogous. However, such inflectional relations do not apply to Chinese, which is fully configurational and lacks morphology. Consequently, our main focus is semantic similarities, which include antonymy (e.g. (热 (hot):冷 (cold)) VS (快 (fast):慢 (slow))), meronymy (e.g. (车 (car):轮子 (wheel)) VS (熊 (bear):掌 (paw))), gender (e.g. (男人 (man):女人 (woman)) VS (国王 (king):女王 (queen))) and function relations (e.g. (衣服 (clothing):穿 (wear)) VS (帽子 (hat):戴 (wear))), etc.

Chiu et al. (2007) show that English semantic relations are also reflected by syntactic dependencies. Their finding coincides with Levin (1993), who study English verbs. We find that this observation is even more prevalent for Chinese. In our automatically-parsed Chinese corpus of 3.4 billion words (Section 5.1), 86.4% word pairs from the analogy test dataset (Section 5.2) have corresponding dependencies, each of which appearing at least ten times.

The frequent correlation between semantic relations and syntactic dependencies can be due to the lack of morphology and function words in Chinese. In fact, Chinese syntactic ambiguities often need to be resolved by leveraging semantic information (Xiong et al., 2005; Zhang et al., 2014). Although not all occurrences of semantically-related word pairs must also form a syntactic dependency in a corpus, we show that syntactic dependencies can effectively improve analogy detection.

3 Dependency-context Word Embeddings for Analogy Detection

A first use of syntactic dependencies for embedding-based analogy detection is to use them directly for embeddings. Recently, a dependency context has been used for the skip-gram method, for capturing more syntactic regularities. Taking the sentence in Figure 1 for example, a *bi-gram* context for the word "毕业 (graduate)" can be "奥巴马 (Obama), 总统 (President), 于 (from), 哈佛 (Harvard)", while a *dependency context* of the same word can be "1991年/ADV, 总统/SBV, 于/CMP, 法学院/POB_于"[1], where "ADV, SBV, CMP, POB" indicate adverbial modifier, subject, complement and prepositional object, respectively.

It has been shown that a dependency context leads to embeddings that better help parsing (Bansal et al., 2014) and measuring word similarity (Levy and Goldberg, 2014a), compared with ngram contexts. However, little previous work has systematically compared dependency contexts with ngram contexts in analogy detection. We empirically study this problem (c.f Section 6.3), finding that dependency context leads to significantly worse analogy detection results for both Chinese and English using state-of-the-art embedding-based methods (Levy and Goldberg, 2014b). We give analysis in Section 6.4.

[1]The last token is a grand-child of "毕业 (graduate)", via the preposition "于 (at)" (Levy and Goldberg, 2014a).

4 Search Space Pruning Using Syntactic Dependencies

We study an alternative way of making use of syntactic dependencies, by using them to prune the vocabulary-sized search space of analogy detection. Given two word pairs *a:b* and *a*:b**, where *b** is hidden and *a* is the head word, we search for dependencies, taking *a** as the head word. The dependent words in the search candidates need to share the POS tag of *b*. If there are several types of dependencies between *a* and *b*, only the one with highest frequency is used. We rank all resulting dependencies using the 3COSMUL objective, and take the word *b** in the highest-scored dependencies as the answer.

For example, given the word pair (萨拉热窝 *(Sarajevo):*波黑 *(Bosnia and Herzegovina)*), whose most frequency dependency is <萨拉热窝 (Sarajevo), 波黑 (Bosnia and Herzegovina), ATT>, and the unknown pair (伦敦 *(London):b**), we acquire a list of dependencies, including <伦敦 (London), 美国 (USA), ATT>, <伦敦 (London), 巴黎 (Paris), COO>, <伦敦 (London), 加拿大 (Canada), ATT> and <伦敦 (London), 英国 (England), ATT>. Some of these dependencies, such as <伦敦 (London), 巴黎 (Paris), COO>, are parsed as the coordinate relation (COO), and thus pruned because the target syntactic relation is *ATT*. From the resulting list, the 3COSMUL objective successfully ranks the triple <伦敦 (London), 英国 (England), ATT> as the top candidate. In contrast, Levy and Goldberg's method takes "南非 (South Africa)" as the answer, which does not form an attributive-head phrase with "伦敦 (London)".

5 Analogy Mining Using Dependency Embeddings

Formally, *analogy mining* is the task of mining analogous dependencies $<x_1, y_1, r>, <x_2, y_2, r>$ ...$<x_n, y_n, r>$ that share the same relation *r* with a given dependency $<a, b, r>$. We mine analogous dependencies by considering relational similarity and attributional similarity simultaneously using the skip-gram model for embeddings.

5.1 Dependency Embedding

Inspired by the fact that *word* similarities can be measured by using distributed *word* representations, we hypothesize that *relation* similarities can be measured by distributed *relation* representations. Based on the observation in Section 2.4, semantically analogous word pairs typically have syntactic dependencies. We use the skip-gram algorithm to train *distributed representations of syntactic dependencies*, and use them for mining analogous word pairs.

With respect to the skip-gram model, *words* are the most common target for embeddings (Levy and Goldberg, 2014b; Levy and Goldberg, 2014a; Mikolov et al., 2013a), although continuous vector representations can be trained for other structures. For example, Mikolov et al. (2013a) take idiomatic *phrases* as embedding targets. *Dependencies*, which consist of a modifier word, a head word and a syntactic relation between them, can also be represented by continuous embeddings using the same algorithm.

To induce dependency embeddings, we take the union of the dependency context of both the dependent and the head of a dependency as the context. For instance, in the example sentence, the context of the dependency <总统 *(Presiden-*

```
Input  : dependency embedding DT, word
         embedding DW, seed dependency s,
         threshold α and β.
Output: set of ranked dependencies WP.
Function Mine (DT,DW,s,WP,α,β):
begin
    DTSet =∅;
    MScore =0;
    SimDT =GetSimDT (DT,s);
    for each Triple ∈ SimDT do
        MWS =GetMWord (s);
        HWS =GetHWord (s);
        MWD =GetMWord (Triple);
        HWD =GetHWord (Triple);
        ScoreX =Sim (MWS,MWD,DW);
        ScoreY =Sim (HWS,HWD,DW);
        ScoreXY =ScoreX × ScoreY;
        MScore =Max (ScoreXY,MScore);
        TopK (ScoreXY,Triple,DTSet,α)
    end
    MScore =MScore × β ;
    for each Triple, ScoreXY ∈ DTSet do
        if ScoreXY > MScore and Triple ∉
        WP then
            AddToSet (Triple,WP);
            s =Triple;
            Mine (DT,DW,s,WP,α,β);
        end
    end
end
WP =∅;
Mine (DT,DW,s,WP,α,β);
```

Algorithm 1: Bootstrapping for analogy mining.

t), 毕业 (graduate), SBV> consists of four tokens: "1991年/ADV", "奥巴马/ATT", "于/CMP" and "法学院/POB_于". The same skip-gram algorithm is used to train embeddings for dependency structures.

5.2 Analogy Mining by Bootstrapping

A bootstrapping algorithm is used to mine analogous word pairs based on dependency-context word embeddings and dependency embeddings. Algorithm 1 shows pseudocode of the recursive bootstrapping algorithm.

The recursive function `Mine` (Algorithm 1) contains three steps with six parameters, including the dependency embeddings *DT*, word embeddings *DW*, a seed dependency *s*, and two thresholds α and β. Step 1 (lines 3 to 5) is an initialization process, where the dependency embedding is used to return up to 100 most similar dependencies for the given seed *s*. These dependencies are stored in *SimDT*, and the candidate analogous dependency set *DTSet* is initialized to an empty set.

In Step 2 (lines 6 to 16), an analogous score *ScoreXY* is computed for each dependency *Triple* in *SimDT* by multiplying the similarity scores between the two dependents and the two heads in *Triple* and *s*, respectively. *Triple* is stored into the set *DTSet* if *ScoreXY* is ranked top α. The top 1 score in *DTSet* is referred to as *MScore*. In Step 3 (lines 17 to 24), if the score of a dependency *Triple* in *DTSet* is larger than $\beta \times$*MScore*, it is used as a new seed for mining more analogous dependencies, by calling the function `Mine` recursively.

We take the seed dependency <弹 (play), 钢琴 (piano), VOB> as an example to illustrate the work-flow of the `Mine` function. In Step 1, a set of similar dependencies (e.g., <弹 (play), 吉他 (guitar), VOB>, <弹 (play), 琴 (lyra), VOB>), is calculated using the dependency embeddings *DT* and stored in *SimDT*. Each dependency in *SimDT* is scored in Step 2, and the top α scores are put into the set *DTSet*. Finally, a dependency is used as seed to mine new analogous dependencies if its score is larger than a threshold ($\beta \times$*MScore*). For instance, the dependency <弹 (play), 琴 (lyra), VOB> is used to mine the new dependency <弹 (play), 古筝 (zheng), VOB>, which is then used to mine other dependencies such as <吹 (blow), 葫芦丝 (cucurbit flute), VOB> and <吹 (blow), 萨克斯 (sax), VOB>.

6 Experiments

6.1 Word Embeddings

We train three sets of word embeddings: NG5 (n-gram context with 5 words to the left of the target word and 5 words to the right), NG2 (2 words to the left and right) and DEP (dependency context), and one set of dependency embeddings DT (dependency context), using the Skip-Gram model. WORD2VEC[2] is used to train NG5 and NG2, and WORD2VECF[3] is used to train DEP and DT. The negative-sampling parameter is set to 15 in all the training processes.

All embeddings are trained on a free Chinese news archive[4] that contains about 170 millions sentences and 3.4 billions words. We segment and parse these sentences using the MVT implementation of ZPar 0.7[5] (Zhang and Clark, 2011), which is trained on a large-scale annotated corpus and achieves state-of-the-art analyzing accuracy on contemporary Chinese (Qiu et al., 2014)[6]. Targets and contexts for word and dependency embeddings were filtered with a minimum frequency of 100 and 10, respectively, and all the four types of embeddings are trained with 200 dimensions.

6.2 Datasets and Evaluation Metrics

Three datasets are used for evaluating Chinese embeddings. First, we construct a set of semantic analogy questions. This set contains five types of semantic analogy questions, including capital-country (136 word pairs, and 18354 analogy questions), provincial capital-province (28, 756), city-province (637, 386262), family member (male-female) (18, 306) and currency-country (62, 3782). We collect the five types of word pairs and then produce analogy questions automatically by concatenating two word pairs. The resulting analogy dataset contains 400K analogy questions. We refer to this dataset as the Chinese Analogy Question Set (CAQS).

[2] `http://code.google.com/p/word2vec/`

[3] `https://bitbucket.org/yoavgo/word2vecf`

[4] This dataset contains news articles in 2014 from various news websites, and can be downloaded from `http://pan.baidu.com/s/1o6wRjp4`

[5] `http://people.sutd.edu.sg/\%7Eyue_zhang/doc/doc/multiview.html`

[6] The system achieves 96.1% , 92.6% and 83.28% F1-score for words segmentation, joint POS-tagging and dependency parsing, respectively, on 1493 manually annotated sentences.

Data	Metrics	NG5	NG2	DEP
Cilin	P@1	43.3%	**45.9%**	43.6%
	P@5	31.1%	**33.3%**	32.6%
	P@10	25.5%	**27.5%**	**27.5%**
	P@20	20.5%	22.2%	**22.7%**
	P@50	15.0%	16.2%	**17.0%**
	P@100	11.5%	12.2%	**12.8%**
CWS	Kendall's τ	38.6%	**44.1%**	42.4%
	Spearman's ρ	54.5%	**62.2%**	60.7%

Table 1: Results on *Cilin* and CWS.

Because embeddings are central for analogy detection, yet there is little large-scale evaluation results on Chinese embeddings in the literature, we perform embedding evaluation on two datasets. The first one is the Chinese WordSim (CWS), translated from the English WordSim-353 Set and re-scored by native Chinese speakers (Jin and Wu, 2012). This dataset consists of 297 word pairs.

The second one is the Chinese thesaurus *Tongyicicilin* (Cilin) (Che et al., 2010), which groups 74,000 Chinese words into five-layer hierarchies and has been used for evaluating the accuracy of word similarity by traditional sparse vector space models (Qiu et al., 2011; Jin et al., 2012). The third level of *Cilin*, which contains 1428 classes, is used to evaluate whether two words are semantically similar.

For comparison between Chinese and English, we also use an English analogy question dataset, the Google dataset[7] (Mikolov et al., 2013a), to evaluate the English word embeddings of Levy and Goldberg (2014a)[8] on analogy detection.

On both the CAQS and the Google datasets, the 3COSMUL method (Levy and Goldberg, 2014b) is used to to answer analogy questions based on given embeddings. The results on the CWS dataset are evaluated using the two standard metrics for the task, namely Spearman's ρ and Kendall's τ rank correlation coefficients. The results on *Cilin* are evaluated using Precision@*K*: the percentage of words from the top-*K* candidates that belong to the *Cilin* category of the target word. If one of the top-*K* candidates belongs to the same third-level category in *Cilin* as the target word, the candidate word is taken as correct.

6.3 Dependency-based and Word-based Word Similarity and Analogy Detection

Word Similarity

[7] http://code.google.com/p/word2vec/source/browse/trunk/questions-words.txt

[8] http://levyomer.wordpress.com/2014/04/25/dependency-based-word-embeddings/

	Relation	NG5	NG2	DEP
MUL	capital-country	68.8%	52.7%	9.9%
	capital-province	84.0%	87.7%	50.0%
	city-province	80.9%	80.3%	22.6%
	family	39.7%	45.1%	41.5%
	currency	10.4%	9.9%	2.5%
	All	80.0%	78.8%	22.0%
IMP	capital-country	87.9%	88.0%	87.6%
	capital-province	84.9%	86.8%	84.9%
	city-province	91.8%	92.0%	90.5%
	family	45.3%	48.0%	47.1%
	currency	7.9%	7.0%	25.9%
	All	90.9%	91.1%	89.8%

Table 2: Results on CAQS. MUL and IMP indicate 3COSMUL and our improved method, respectively.

Relation	NG5	NG2	DEP
capital-country	94.6%	84.5%	38.5%
capital-world	71.5%	64.7%	14.2%
city-in-state	53.2%	42.5%	13.1%
family	82.0%	81.2%	81.0%
currency	10.5%	10.7%	6.0%
All	63.7%	60.7%	38.8%

Table 3: English results on the Google set.

Table 1 shows the results of the three Chinese embedding on *Cilin* and CWS, where NG2 performs much better than NG5 on both datasets. This demonstrates that one does not need to use large window sizes in training word-based embeddings for capturing word similarities. The result is similar to the finding of Shi et al. (2010), which indicates that a window size of 2 is better than a window size of 4 for capturing word similarity by using distributional word representations.

DEP performs slightly worse than NG2 on CWS and *Cilin* in P@1 and P@5. However, it achieves better results on *Cilin* in P@10 to P@100 when more candidate similar words are evaluated. In contrast, NG5 and NG2 mix more semantically related words. This finding is consistent with that of Levy and Goldberg (2014a).

Analogy Detection

Table 2 shows the results of the three Chinese embeddings on CAQS. Unlike on *Cilin* and CWS, NG5 outperforms DEP, and is also slightly better than NG2. Similar tendency is shown in Table 3 for the three English embeddings evaluated on the Google dataset. These results show that dependency embeddings are relatively weak for answering analogy questions. On the other hand, the performance also varies across different relation types.

Target	NG5	NG2	DEP
穿 (wear)	短裤 (shorts), 紧身 (slim-fit), 身穿 (wear), 外套 (coat), 裙子 (skirt)	身穿 (wear), 身着 (wear), 短裤 (shorts), 戴 (wear), 紧身 (slim-fit)	身穿 (wear), 身着 (wear), 戴 (wear), 改穿 (change cloths), 外穿 (wear outside)
关羽 (Guan Yu; P)	赵云(Zhao Yun; P), 刘备 (Liu Bei; P), 诸葛亮 (Zhuge Liang; P), 张飞 (Zhang Fei; P), 曹操 (Cao Cao; P)	赵云 (Zhao Yun; P), 刘备 (Liu Bei; P), 张飞 (Zhang Fei; P), 曹操 (Cao Cao; P), 夏侯渊 (Xiahou Yuan; P)	赵云 (Zhao Yun; P), 韩信 (Han Xin; P), 曹操 (Cao Cao; P), 刘备 (Liu Bei; P), 阿修罗 (Asura; P)
郑州 (Zhengzhou; C)	石家庄 (Shijiazhuang; C), 洛阳 (Luoyang; C), 西安 (Xian; C), 许昌 (Xuchang; C), 太原 (Taiyuan)	石家庄 (Shijiazhuang; C), 太原 (Taiyuan; C), 济南 (Ji-nan; C), 合肥 (Hefei; C), 西安 (Xi-an; C)	合肥 (Hefei; C), 济南 (Jinan; C), 武汉 (Wuhan; C), 石家庄 (Shijiazhuang; C), 南宁 (Nanning; C)

Table 4: Comparison between NG2, NG5 and DEP Embeddings. (P: personal name, C: city name)

6.4 Analysis

To analyze the difference between the three Chinese embeddings methods qualitatively, we manually inspect the words "穿 (wear)", "关羽 (Guan Yu, a person name in the novel '三国演义 *(Romance of the three kingdoms)*')", and "郑州 (Zhengzhou, a city)". Their most similar words are shown in Table 4.

Word Similarity

For the word "穿 (wear)", both NG5 and NG2 yield *similar words* such as "身穿 (wear)", "身着 (wear)", "戴 (wear)" and *related words* such as "短裤 (shorts)", "紧身 (slim-fit)", "外套 (coat)", "裙子 (skirt)", although NG5 gives more related words. In contrast, DEP gives only words that are *similar* both syntactically and semantically. This observation holds for other verbs and nouns, and can be explained by the context extraction methods. For instance, the word "穿 (wear)" usually takes one of the words "短裤 (shorts)", "外套 (coat)", "裙子 (skirt)" as its object, and thus shares similar contexts with them in NG5 and NG2. The context extraction method in DEP, on the other hand, yields different context across syntactic roles, such as verbs (e.g. "穿 (wear)") and their objects (e.g. "短裤 (shorts)" and "外套 (coat)").

Observations on the person name "关羽 (Guan Yu)" and location "郑州 (Zhengzhou)" are similar. For "关羽 (Guan Yu)", NG5 and NG2 can yield more person names in the same novel, while DEP yields person names from other novels (i.e. "韩信 (Hanxin)" and "阿修罗 (Asura)"). For "郑州 (Zhengzhou)", the provincial capital of "河南 (Henan)", NG5 and NG2 give more cities in the same province "河南 (Henan)", while DEP yields capitals of other provinces.

Analogy Detection

As mentioned in Section 2.3, both 3COSADD and 3COSMUL seek a word $b*$ that is similar to b and $a*$ but dissimilar to a. Ideally, the two word pairs b:$b*$ and a:$a*$ should be semantically *similar* while the two word pairs a:b and $a*$:$b*$ should be semantically *related*. Therefore, 3COSADD and 3COSMUL require the embeddings to give higher cosine scores for both semantically similar and related words.

Our analysis above shows that word-context embeddings tend to mix semantically *related* and *similar* words, but dependency-context embeddings only capture semantic *similarity*. This partly explains the reason that dependency-context word embeddings are weak for analogy detection.

It has also been shown in Section 6.3 that the performances of analogy detection vary across different types of relations, which indicates that there are more sophisticated underlying factors. One intuitive explanation is that different semantic relations correspond to different syntactic dependency structures. For example, the male-female family member relation is expected to stand less frequently in a syntactic dependency relation, compared with geographic relations such as city-country, which stand frequently in attributional syntactic relations (e.g. "London, England"). As a result, where the coupling between syntactic and semantic relations is weak, our analysis in Section 6.3 and other work based on syntactic relations can find limitations.

6.5 Syntactic Dependencies for Improved Analogy Detection

The results on CAQS using the method in Section 4 are shown in the IMP rows of Table 2. The method achieves significant improvements (from 80.0% to 90.9% using NG5) compared with Levy and Goldberg's method. In addition, DEP also performs significantly better than with MUL, with an increase from 22.0% to 89.8%. The main reason for this improvement is that the filtering process using syntactic dependencies successfully prunes noisy words.

Seed	Count	Prec
吃 (eat), 苹果 (apple), VOB	572	84.70%
弹 (play), 钢琴 (piano), VOB	142	40.49%
穿 (wear), 衣服 (clothing), VOB	452	67.37%
写 (write), 小说 (novel), VOB	441	53.40%
中国 (China), 北京 (Beijing), ATT	2224	95.23%
湖北 (Hubei), 武汉 (Wuhan), ATT	3201	96.34%

Table 5: Main results of Analogy Mining.

Error analysis shows that the main errors by the improved method are quite different from those by the baseline. For instance, the main errors of Levy and Goldberg's method for the city-province relation are caused by giving another province as the answer, while the improved method gives the name of the country as answer. This is because irrelevant provinces do *not* co-occur frequently with the city in syntactic dependencies, and hence can be filtered by our method. On the other hand, both the country name and province name co-occur frequently with the city name in syntactic dependencies, and our method cannot make a choice between them.

6.6 Dependency Structure Embeddings for Analogy Mining

Shown in Table 5, we use six seeds to mine analogous dependencies. The first seed is used for development and the others for test. The first three seeds, the fourth seed and the last two seeds belong to the *Use:Thing, Produce:Thing and Sub-Location:Location* relations, respectively. α and β are set to 20, and 0.6, respectively. Each set of mined dependencies together with the seed dependency and relation type is shown to two human evaluators, who are required to give a Yes/No answer to each dependency in the set. We take the average scores of the two evaluators (the average inter-annotator agreement is 0.95) as the final precision scores.

As shown in the table, the precisions using different seeds are quite different, ranging from 40% to 96%. One possible reason is that different relations have different numbers of analogous dependencies, ranging from dozens to thousands, and thus the fixed thresholds tuned on a development seed does not apply as effectively to all test cases. For instance, "弹 (play)" and its analogous actions, "吹 (blow)" and "拉 (play)", are all human actions on musical instruments, while the actions "吃 (eat)" and "写 (write)" can apply to many patients. For the seed <弹 (play), 钢琴 (piano), VOB>, irrelevant results such as <拿 (use), 剪子 (scissors), VOB> and <拿 (use), 电筒 (flashlight), VOB>, have the verb "拿 (use)", which is also a human action, yet cannot be considered as usage of the patients "剪子 (scissors)" and "电筒 (flashlight)". Because of the stricter selectional preference of "弹 (play)", its precision of analogy mining is lower.

We tentatively measure the recall of the algorithm by taking the first three types of word pairs in CAQS as the gold set, which contains 801 word pairs. All the three types of word pairs belong to the relation *Sub-Location:Location*. The recall is computed as the percentage of the gold word pairs covered by the mined dependencies. When using the two seeds <武汉 (Wuhan), 湖北 (Hubei), ATT> and <北京 (Beijing), 中国 (China), ATT> for analogy mining, the recalls are 50.2% and 11.3%, respectively. Their union recall is 56.8%. When the precision of each seed is similar, we can achieve better recall without precision loss by using more seeds.

7 Related Work

Turney (2006) introduces a latent relational analysis (LRA) model to measure relational similarity, and apply a novel co-occurrence-based method for analogy filtering. The model can be used for both analogy detection and relation classification, yet cannot scale up well to large datasets due to the complexity of Singular Value Decomposition. Recently, distributed word representations using the skip-gram model (Mikolov et al., 2013a) has been shown to give competitive results on analogy detection. Levy and Goldberg (2014a) extends the skip-gram method with dependency-context embeddings. We study the effect of Levy and Goldberg's embeddings on analogy detection, and further extend their embeddings to dependency-context *dependency structure* embeddings for analogy mining.

Chiu et al. (2007) presents a similarity graph tranversal (SGT) method to mine analogous relations from raw English text automatically, using syntactic dependencies to find candidate relations. The method is unsupervised, and can scale up well to large data sets. However, Chiu et al. (2007) mainly focuses on relations between subjects and objects because of its word-pair extraction method. Ó Séaghdha and Copestake (2009) is a supervised method, which combines lexical similarity and relational similarity to classify se-

mantic relations. These methods are based on distributional word representation models and fit for classifying noun-noun word pairs. In contrast, our methods are based on distributed word representation models, and can mine noun-noun word pairs as well as verb-noun word pairs. In addition, our analogy mining method is unsupervised, while the methods of both Turney (2006) and Ó Séaghdha and Copestake (2009) are supervised.

8 Conclusion

We studied several Chinese relational similarity tasks to train embeddings under the context of distributed word representations using the skip-gram model and syntactic dependencies. For Chinese analogy detection, we compared word-context and dependency-context embeddings, finding that the former results in much better accuracies. Observing that common relations in Chinese are frequently represented by syntactic dependencies, we improved Chinese analogy detection using a dependency context. Further, we empirically studied Chinese analogy mining by proposing a bootstrapping algorithm using a novel distributed representation of syntactic dependencies.

Acknowledgments

We thank the anonymous reviewers for their constructive comments, and gratefully acknowledge the support of the Singapore Ministry of Education (MOE) AcRF Tier 2 grant T2MOE201301, the National Natural Science Foundation of China (No. 61572245, 61170144, 61103089), Major National Social Science Fund of China (No. 12&ZD227), Scientific Research Foundation of Shandong Province Outstanding Young Scientist Award (No. BS2013DX020) and Humanities and Social Science Projects of Ludong University (No. WY2013003).

References

Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2014. Tailoring continuous word representations for dependency parsing. In *Proceedings of ACL*, Baltimore, Maryland, June.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155.

Danushka Bollegala, Yutaka Matsuo, and Mitsuru Ishizuka. 2008. Www sits the sat: Measuring relational similarity on the web. In *ECAI*, pages 333–337.

Wanxiang Che, Zhenghua Li, and Ting Liu. 2010. Ltp: A chinese language technology platform. In *Proceedings of COLING*, pages 13–16.

Andy Chiu, Pascal Poupart, and Chrysanne DiMarco. 2007. Generating lexical analogies using dependency relations. In *Proceedings of EMNLP-CoNLL 2007*, pages 561–570, Prague, Czech Republic, June.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of ICML*, pages 160–167. ACM.

Amaç Herdağdelen and Marco Baroni. 2009. Bagpack: A general framework to represent semantic relations. In *Proceedings of the Workshop on Geometrical Models of Natural Language Semantics*, pages 33–40.

Peng Jin and Yunfang Wu. 2012. Semeval-2012 task 4: evaluating chinese word similarity. In *Proceedings of the Sixth International Workshop on Semantic Evaluation*, pages 374–377.

Peng Jin, John Carroll, Yunfang Wu, and Diana McCarthy. 2012. Distributional similarity for chinese: Exploiting characters and radicals. *Mathematical Problems in Engineering*, 2012.

David A Jurgens, Peter D Turney, Saif M Mohammad, and Keith J Holyoak. 2012. Semeval-2012 task 2: Measuring degrees of relational similarity. In *Proceedings of the Sixth International Workshop on Semantic Evaluation*, pages 356–364. Association for Computational Linguistics.

Beth Levin. 1993. English verb classes and alternations: a preliminary investigation.

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

Omer Levy and Yoav Goldberg. 2014b. Linguistic regularities in sparse and explicit word representations. In *Proceedings of CONLL*, pages 171–180, Ann Arbor, Michigan, June.

Douglas L Medin, Robert L Goldstone, and Dedre Gentner. 1990. Similarity involving attributes and relations: Judgments of similarity and difference are not inverses. *Psychological Science*, 1(1):64–69.

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

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013b. Linguistic regularities in continuous space word representations. In *HLT-NAACL*, pages 746–751. Citeseer.

Andriy Mnih and Geoffrey Hinton. 2007. Three new graphical models for statistical language modelling. In *Proceedings of ICML*, pages 641–648. ACM.

Diarmuid Ó Séaghdha and Ann Copestake. 2009. Using lexical and relational similarity to classify semantic relations. In *Proceedings of EACL 2009*, pages 621–629, Athens, Greece, March.

Likun Qiu, Yunfang Wu, and Yanqiu Shao. 2011. Combining contextual and structural information for supersense tagging of chinese unknown words. In *Computational Linguistics and Intelligent Text Processing*, pages 15–28. Springer.

Likun Qiu, Yue Zhang, Peng Jin, and Houfeng Wang. 2014. Multi-view chinese treebanking. In *Proceedings of COLING*, pages 257–268, Dublin, Ireland, August.

Shuming Shi, Huibin Zhang, Xiaojie Yuan, and Ji-Rong Wen. 2010. Corpus-based semantic class mining: Distributional vs. pattern-based approaches. In *Proceedings of COLING*, pages 993–1001, Beijing, China, August.

Richard Socher, Cliff C Lin, Chris Manning, and Andrew Y Ng. 2011. Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of ICML*, pages 129–136.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of ACL*, pages 384–394.

Peter D Turney and Michael L Littman. 2005. Corpus-based learning of analogies and semantic relations. *Machine Learning*, 60(1-3):251–278.

Peter D Turney. 2006. Similarity of semantic relations. *Computational Linguistics*, 32(3):379–416.

Peter D Turney. 2013. Distributional semantics beyond words: Supervised learning of analogy and paraphrase. *arXiv preprint arXiv:1310.5042*.

Deyi Xiong, Shuanglong Li, Qun Liu, Shouxun Lin, and Yueliang Qian. 2005. Parsing the penn chinese treebank with semantic knowledge. In *Natural Language Processing–IJCNLP 2005*, pages 70–81. Springer.

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

Meishan Zhang, Yue Zhang, Wanxiang Che, and Ting Liu. 2014. A semantics oriented grammar for chinese treebanking. In *Computational Linguistics and Intelligent Text Processing*, pages 366–378. Springer.

Navigating the Semantic Horizon using Relative Neighborhood Graphs

Amaru Cuba Gyllensten and Magnus Sahlgren
Gavagai
Bondegatan 21
116 33 Stockholm
Sweden
{amaru|mange}@gavagai.se

Abstract

This paper introduces a novel way to navigate neighborhoods in distributional semantic models. The approach is based on *relative neighborhood graphs*, which uncover the topological structure of local neighborhoods in semantic space. This has the potential to overcome both the problem with selecting a proper k in k-NN search, and the problem that a ranked list of neighbors may conflate several different senses. We provide both qualitative and quantitative results that support the viability of the proposed method.

1 Introduction

Nearest neighbor search is a fundamental operation in data mining, in which we are interested in finding the closest points to some given reference point. Formally, if we have a reference point r and a set of other points P in a metric space M with some distance function d, the nearest neighbor search task is to find the point $p \in P$ that minimizes $d(p, r)$. In k-Nearest Neighbor search (k-NN), we want to find the k closest points to some given reference point. Nearest neighbor search is a well-studied task, and in particular the complexity of the task (a linear search has a running time of $\mathcal{O}(Ni)$ where N is the cardinality of P and i the complexity of the distance function d) has generated a lot of research; suggestions for reducing the complexity of linear nearest neighbor searches include using various types of space partitioning techniques like k-d trees (Bentley, 1975), or various techniques for doing *approximate* nearest neighbor search (Arya et al., 1998), of which one of the most well-known is locality-sensitive hashing (Indyk and Motwani, 1998).

The problem we are concerned with in this paper is not the complexity of nearest neighbor search, but the question of *how to identify the internal structure of neighborhoods defined by the nearest neighbors*. The problem with a normal k-NN is that the result — a sorted list of the k nearest neighbors — does not say anything about the internal structure of the neighborhood. It is quite possible for two neighborhoods with widely different internal structures to produce identical k-NN results. In the context of *Distributional Semantic Models* (DSMs), which collect and represent co-occurrence statistics in high-dimensional vector spaces, such structural differences may carry significant semantic information, e.g. about the different senses of terms. We argue that the inability of standard k-NN to account for structural properties has been misinterpreted as a shortcoming of the distributional representation (Erk and Padó, 2010).

We will demonstrate in this paper that this is *not* a shortcoming of the distributional representation, but of the *mode of querying* the DSM. We argue that information about the different usages (i.e. senses) of a term is encoded in the structural properties of the nearest neighborhoods, and we propose the use of *relative neighborhood graphs* for identifying these structural properties. Relative neighborhood graphs may also be used for finding a relevant k for a given reference point, which we refer to as the *horizon* with respect to the reference point.

2 Distributional Semantics and Nearest Neighbor Search

Collecting and comparing co-occurrence statistics for terms in language has become a standard approach for computational semantics, and is now commonly referred to as *distributional semantics*. There are many different types of models that can be used for this purpose, but their common objective is to represent terms as vectors that record (some function of) their distri-

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 2451–2460,
Lisbon, Portugal, 17-21 September 2015.

butional properties. The standard approach for generating such vectors is to collect distributional statistics in a *co-occurrence matrix* that records co-occurrence counts between terms and contexts. The co-occurrence matrix is then subject to various types of transformations, ranging from the application of simple frequency filters or association measures to matrix factorization or regression models. The resulting representations are referred to as *distributional vectors* (or *word embeddings*), which are used to compute similarity between terms.

Given a similarity — or distance — measure on such distributional vectors, we can perform a nearest neighbor search. This is a particularly important operation in distributional semantics, since it answers the question "which other terms are similar to this one?" and this is a central question in semantics; lexica and thesauri are built with the main purpose of answering this question. Consequently, nearest neighbor search in a DSM could be seen as a compilation step in a distributional lexicon.

The result of a nearest neighbor search in a DSM is often presented as a list of (the top k) neighbors, sorted by descending similarity with the target term. Table 1 illustrates typical sorted nearest neighbor lists produced with three different DSMs: a standard model based on Pointwise Mutual Information (PMI)[1] that has been reduced to 2,000 dimensions by applying a Gaussian random projection; GloVe, which uses regression to find distributional vectors such that their dot product approximates their log probability of co-occurring (Pennington et al., 2014); and the Skipgram model, which uses stochastic gradient descent and hierarchical softmax combined with negative sampling and subsampling to find distributional vectors that maximize the probability of observed co-occurrence events (Mikolov et al., 2013). We refer to the respective papers for details regarding the various models. The similarity measure used is the cosine similarity: $s(a,b) = \frac{a \cdot b}{\|a\|\|b\|}$.

Table 1 lists the 10 nearest neighbors to *suit* in these three different DSMs using the entire Wikipedia as data. As can be expected, there are both similarities and dissimilarities between these neighborhoods; "suits" and "lawsuit" occur among the 10 nearest neighbors to "suit" in all three models, whereas other terms are specific for one particular model. What is common between the three models is that they all feature neighbors that represent two different usages of "suit": the *law*-sense ("lawsuit") and the *clothes*-sense ("dress", "wearing", "doublebreasted").[2] However, these distinction are not discernible by merely looking at the list of nearest neighbors; the only information it provides is the ranking of the nearest neighbors in descending order of similarity.

Table 1: Sorted list of the nearest neighbors to "suit" in three different distributional models.

PMI	**GloVe**	**Skipgram**
suits	suits	suits
dress	lawsuit	lawsuit
jacket	filed	countersuit
wearing	case	classaction
hat	wearing	doublebreasted
trousers	laiming	skintight
costume	lawsuits	necktie
shirt	alleging	wetsuit
pants	alleges	crossbone
lawsuit	classaction	lawsuits

It has been argued that DSMs that represent terms by a single vector cannot adequately handle polysemy, since they conflate several different usage patterns in one and the same vector (Véronis, 2004; Erk and Padó, 2010). Examples like the one above is often cited as evidence. We argue that this critique is unfounded and misinformed, and that it is *the mode of querying* the DSM that can be susceptible to problems with polysemy. As the above example demonstrates, querying DSMs by k-NN conflates different usages of terms. The reason for this seems quite obvious: simply ranking the nearest neighbors by similarity (or distance) ignores any local structures of the neighborhood. If "suit" has as neighbors both "dress" and "lawsuit", which represent two distinct types of usages of "suit", there will be a *structural* distinction in the neighborhood of "suit" between these different neighbors, since they will be mutually unrelated (i.e. there is a similarity between "suit" and

[1] For observations a and b, PMI$(a,b) = \log \frac{p(a,b)}{p(a)p(b)}$. The probabilities are often replaced in DSMs by co-occurrence counts of a and b and their respective frequency counts.

[2] The Skipgram model also features a *manga*-related sense of "suit" in the neighbor "crossbone," which refers to the mange series "Mobile Suit Crossbone Gundam."

"dress" and between "suit" and "lawsuit", but *not* between "dress" and "lawsuit").

k-NN also gives rise to another problem related to polysemy in DSMs. The problem is that the most frequent senses will populate the top of the nearest neighbor list, while the less frequent senses will not appear until further down the list, and if we set a too restrictive k, we will only see neighbors relating to the most frequent sense. As an example, consider the two different senses of "suit" above. The distributional vector for "suit" can be thought of as a sum $v_{suit} = f_{suit|law}v_{suit|law} + f_{suit|clothes}v_{suit|clothes}$, where $v_{suit|law}$ is an idealized notion of the *true* distributional vector of "suit" in the *law*-sense, and $f_{suit|law}$ is the relative frequency of this sense.[3] From there one can easily argue that a similarity such as $s(v_{suit}, v_{clothes})$ is actually a weighted composite of the similarities $s(v_{suit|law}, v_{clothes})$ and $s(v_{suit|clothes}, v_{clothes})$.[4] If "suit" occurs predominantly in the *law*-sense in our corpus, the k-NN neighborhood of "suit" will be dominated by words pertaining to its *law*-sense, while the less frequent senses might not be present at all. A misguided k may thus obscure any other, less frequent, senses of a term.

3 Word-Sense Induction

Selecting a relevant k for a given term and grouping the neighbors according to which senses they represent is an example of *Word-Sense Induction* (WSI). DSMs are well suited for this task, and there have been a number of different approaches suggested in the literature. One of the earliest approaches is *distributional clustering* (Pereira et al., 1993), which is based on a probabilistic decomposition model that uses maximum likelihood estimation to fit the model to observed data. Another example is *Clustering By Committee* (CBC) (Pantel and Lin, 2002), which first uses average-link clustering to recursively cluster the nearest neighbors of a term into committees, which are then used to define clusters by iteratively adding committees whose similarity to the term exceeds a certain threshold, and that is not too similar to any other added committee. For each added committee, its features are also removed from the distributional representation of the term. This last step ensures that the clusters do not become too similar, and that clusters representing less frequent senses can be discovered.

The idea of iteratively removing features from the distributional vector when a sense cluster as been formed is also present in Dorow and Widdows (2003), who use a graph-based clustering method. Another graph-based approach is the *HyperLex* algorithm (Véronis, 2004), which constructs a graph connecting all pairs of terms that co-occur in the context of an ambiguous term. The resulting graph contains highly connected components, which represent the different senses of the term. Agirre et al. (2006) compare HyperLex to *PageRank* (Brin and Page, 1998) and demonstrates that the two methods perform similarly.

There have also been several attempts to use various types of matrix factorization for WSI. The idea is that the factorization uncovers a set of global senses in the form of the latent factors, and that the sense distribution for a given term can be described as a distribution over these latent factors. Examples of factorization methods that have been used include different versions of *Latent Dirichlet Allocation* ((Brody and Lapata, 2009; Séaghdha and Korhonen, 2011; Yao and Van Durme, 2011; Lau et al., 2012) and *non-negative matrix factorization* (Dinu and Lapata, 2010; Van de Cruys and Apidianaki, 2011).

Tomuro et al. (2007) argue that clustering approaches like distributional clustering or CBC may produce clusters that are themselves polysemous, which may not be a desirable property of a WSI algorithm, and suggests using *feature domain similarity* to solve this problem. The idea is to incorporate similarities between the *features* of items rather than the similarity between the items themselves in a modified version of CBC that enables the algorithm to utilize feature similarities, which inhibit the formation of polysemous clusters.

Koptjevskaja Tamm and Sahlgren (2014) also leverage on the idea of using feature similarity as the basis of sense clustering. The approach, called *syntagmatically labeled partitioning*, relies on a DSM that encodes sequential as well as substitutable relations. The method essentially sorts the k nearest (substitutable) neighbors according to which sequential connections they share. The resulting partitioning of the nearest distributional neighbors does not only constitute a WSI, but it

[3] Weighting schemes muddles this notion quite a bit, but we think the general intuition still holds.

[4] In the case of cosine similarity this follows nicely from the distributive property of dot products: $v = av_1 + bv_2$, $s(v, w) = \frac{v \cdot w}{\|v\|\|w\|} = \frac{a(v_1 \cdot w) + b(v_2 \cdot w)}{\|v\|\|w\|}$

also provides *labels* for the induced senses in the form of the sequential connections the neighbors share.

4 Neighborhood Graphs

Many of the previous WSI approaches operate at a global level, utilizing global structural properties of the semantic spaces, e.g. by matrix factorization techniques. We believe this is as ill-advised as setting a global k or radius for the nearest neighbor search, since it is the *local* structures that are important when analyzing nearest neighbors. Other WSI approaches use various forms of clustering techniques. However, previous studies of the intrinsic dimensionality of distributional semantic spaces using fractal dimensions indicate that neighborhoods in semantic space have a *filamentary* rather than clustered structure (Karlgren et al., 2008).

We therefore propose the use of *topological* models that take the *local* structure of neighborhoods in semantic space into account. The approach discovers different word senses from the local structure of neighborhoods, given nothing but similarities between points. As such it is easy to test on widely different vector models, as long as there exists a well behaved similarity function. The proposed approach not only answers the question which other terms are similar to a given term, but also *how* are they similar.

Relative neighborhoods, first proposed in (Toussaint, 1980), are examples of *empty region graphs* (Cardinal et al., 2009), where points are neighbors if some region between them is empty. For *Relative Neighborhood Graphs* (RNG) this region between two points a and c is defined as the intersection of the two spheres with centers in a and c with radius $d(a, c)$. In other words, a point b lies between points a and c if it is closer to both a and c than a and c are to each other. If no such point b exists, a and c are neighbors. Illustrations of this can be seen in Figure 1.

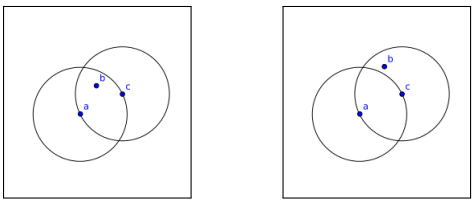


Figure 1: Example of when point b is between point a and c (left), and when it is not (right).

Such neighborhoods have been argued to better preserve local topology (Bremer et al., 2014), and be more robust to deformations of the data than k-NN neighborhoods (Correa and Lindstrom, 2012) as they in some sense contain information about direction whereas k-NN neighborhoods only contain information about distance. Going back to the "suit" example, we can see that if "suit" in the *law*-sense is more similar to the composite "suit" than to its *clothes*-sense, and vice versa, then the composite v_{suit} lies between $v_{suit|law}$ and $v_{suit|clothes}$. This in turn means that out of those two points, both are relative neighbors to "suit", and neither of them lies between the other and "suit".

Formally, the set of points between two points $a, c \in V$ can be characterized and computed in the following way:

$$\text{btw}(V, a, c) = \{b | b \in V, b \text{ is between } a \text{ and } c\}$$

$$\text{rng-nbh}(V, a) = \{c | c \in V, \text{btw}(V, a, c) = \emptyset\}$$

$$E_{\text{rng}}(V) = \{(a, b) | a \in V, b \in \text{rng-nbh}(V, a)\}$$

where E_{rng} is the undirected edge set of the RNG. The function $\text{btw}(V, a, c)$ can be straightforwardly translated to an algorithm taking $\mathcal{O}(|V|)$ time, making the $\text{rng-nbh}(V, a)$ function take $\mathcal{O}(|V|^2)$ time, which in turn makes the computation of the complete graph take $\mathcal{O}(|V|^3)$ time.[5] Clearly unfeasible, but we have not found any alternatives that perform better in the high-dimensional case.[6]

Correa and Lindstrom (2012) note that the intersection of the RNG and the k-NN graph is a more feasible alternative:

$$\text{k-rng-nbh}(V, a) = \text{rng-nbh}(V', a)$$

$$\text{where } V' = k \text{ nearest neighbors of } a.$$

Given a precompiled k-NN lookup, the above takes $\mathcal{O}(k^2)$ time, so using a heap-based $\mathcal{O}(|V| \lg k)$ k-NN algorithm results in an algorithm taking $\mathcal{O}(k^2 + |V| \lg k)$ time.

The same idea can be used to build a tree structure rooted in a reference word a in the following way:

$$\text{rnbh-tree}(V, a) = \{(c, \arg\min_{b \in B_c} d(b, c)) | c \in V\}$$

$$\text{where } B_c = \{a\} \cup \text{btw}(V, a, c)$$

[5] Assuming a constant time distance function.

[6] It should be noted that there are more efficient algorithms for lower-dimensional situations.

which can easily be restricted to the k-nearest neighbors of a in much the same way as above, with the same monotonic behavior.

Computing this for a point a produces a tree where the direct children of a are its relative neighbors, and the parent of a point c further down the tree is the point between a and c that is closest to c. This structure, while similar to a *minimum spanning tree*, differs in some crucial regards: the rnbh-tree(V, a) is rooted in a word a. The difference between rnbh-tree(V, a) and rnbh-tree(V, b) is often quite significant. Furthermore, the restricted k-rnbh-tree is monotonic in k. That property does not hold for a minimum spanning tree of a local neighborhood.

5 Examples of RNGs

To get an intuition of what these neighborhoods look like we present a few examples. The words have been chosen either because they are common examples in similar work — e.g. "heart" and "suit" from Pantel and Lin (2002) — or because they represent different parts-of-speech ("above" is a preposition, "bad" is an adjective, and "service" is a noun) and disparate kinds of ambiguity ("orange" can be both a fruit and a color).

Figure 2 (next page) illustrates what an RNG looks like for the term "heart" and its 100 nearest neighbors in the PMI model. Note that the root "heart" (at the mid-left in the graph) only has two relative neighbors: "cardiac" and "soul," arguably representing a *body*-sense and a *soul*-sense of the term. One advantage of using this type of structure for the neighborhood is that it enables us to examine various depths of the tree. Depth one includes only the direct neighbors ("cardiac" and "soul"), while depth two includes all neighbors two steps away in the graph: "disease," "coronary," "pulmonary," "cardiovascular," "ventricular," and "failure," which are all children to "cardiac." This tree structure can be used to identify neighbors that are themselves polysemous (c.f. the critique mentioned in Section 3 of clustering-based approaches to word-sense induction that they may produce polysemous clusters). One example is the neighbor "disease" at depth two, which has six children that refer to different aspects of disease.

We argue that the RNG can be quite useful for WSI, since the branching structure indicates different usages, and the depth factor enables us to calibrate the granularity of the induced word senses. If we only consider direct neighbors (i.e. depth one), and set $k = V$ (i.e. we do an exhaustive nearest neighbor search), we will extract all terms that have a direct connection to the reference term. We refer to this neighborhood as the *semantic horizon*. At the most coarse level of analysis, this is the neighborhood that represents the main induced senses of a term. Tables 2 and 3 provide examples of 1,000-RNG neighborhoods of depth one.

Table 2: RNG for $k = 1,000$ of the words "suit," "orange," and "heart" in three different semantic models. The numbers in parenthesis indicate the k-NN ranks of the neighbors.

PMI	**GloVe**	**Skipgram**
suit		
suits (1) dress (2) lawsuit (10) dinosaur (53) costly (60) option (76) counterparts (99) predator (107) trump (109) ⋮	suits (1) lawsuit (2) mobile (33) gundam (34) trump (55) zoot (133) rebid (423) serenaders (458) hev (987)	suits (1) lawsuit (2)
orange		
yellow (1) lemon (16)	yellow (1) ktype (12) lemon (14) citrus (17) jersey (21) cherry (24) county (26) peel (42) jumpsuits (57) ⋮	redorange (1)
heart		
cardiac (1) soul (22) hearts (183) ashtray(641) rags(771)	my (1) blood (2) throbs (3) suffering (4) brain (6) cardiac (8) hearts (11) throb (17) lungs (22) ⋮	congestive (1) hearts (2)

These examples demonstrate some interesting similarities and differences between the three models. First of all, there are some direct neighbors that are present in all three models: "suit" has "suits" and "lawsuit" as direct neighbors in all three models, "heart" has "hearts," "service"

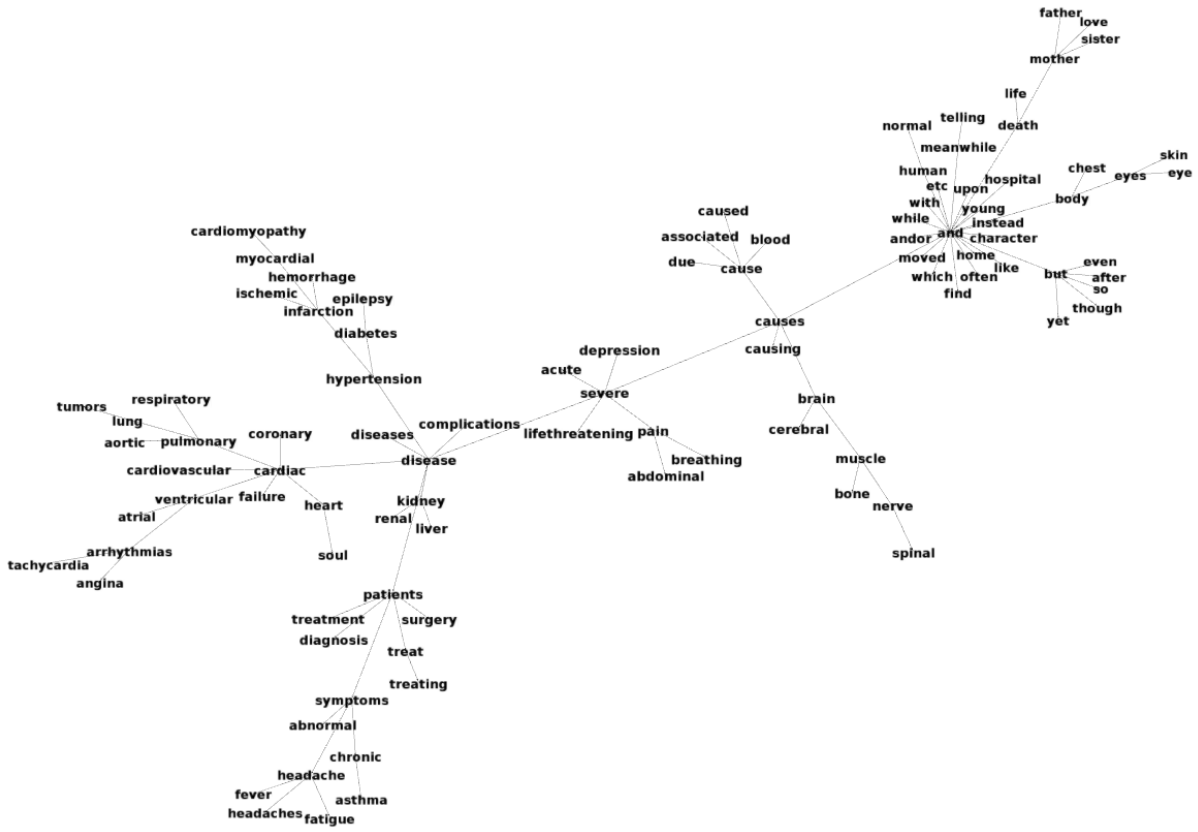


Figure 2: RNG for “heart” in the PMI model, restricted to the 100 nearest neighbors.

has “services,” and “above” has “below”. Plural forms are of course reasonable neighbors of their singular counterparts in a semantic model, but their usefulness for WSI can perhaps be questioned. Taking “suits” to indicate the *clothes*-sense of “suit,” all three models produce both a *clothes*-sense and a *law*-sense. For “orange,” the Skipgram model only represents the *color*-sense, while the PMI and GloVe models also feature a *fruit*-sense. For “heart,” all three models have a *disease*-sense (represented by the neighbors “cardiac” in the PMI and GloVe models, and the neighbor “congestive” in the Skipgram model), and an *organ*-sense (represented by the plural form “hearts”). “Service” is a comparably vague term that has a number of different senses in the PMI and GloVe models, but only one in the Skipgram model. “Bad” produces both a *negativity*-sense and a *German spa town*-sense in all three models, but only the GloVe and Skipgram models have a separate *antonym*-sense (“good” is not a direct neighbor in the PMI model). “Above” has both the antonym and direct neighbors relating to measurements in all three models.

It is interesting to note that GloVe produces a significant amount of sequential relations; “mobile suit gundam”, “cheap suit serenaders”, “orange peel”, and “orange jumpsuit” are just some of many examples of sequential relations found in the relative neighborhood of terms in the GloVe model.

The PMI and GloVe models produce the structurally most similar RNGs in these examples, with on average a handful of direct neighbors, of which some can be very distant. The Skipgram model on the other hand produces very few direct neighbors. This led us to look further into the structural properties of neighborhoods in the Skipgram model. An interesting observation — and possible complication — is that the neighborhoods in the Skipgram model are highly asymmetric: the first neighbor of “information” is “informations”, whereas “information” is the 1,829th neighbor of “informations.” While such asymmetry occurs in all models, it seems much more prevalent in the Skipgram model. Figure 3 confirms this suspicion: each point corresponds to a random word pair (a, b) with x corresponding to where b is in the ordered list of a’s neighbor, and y to where a is in the ordered list of b’s neighbors. The figure

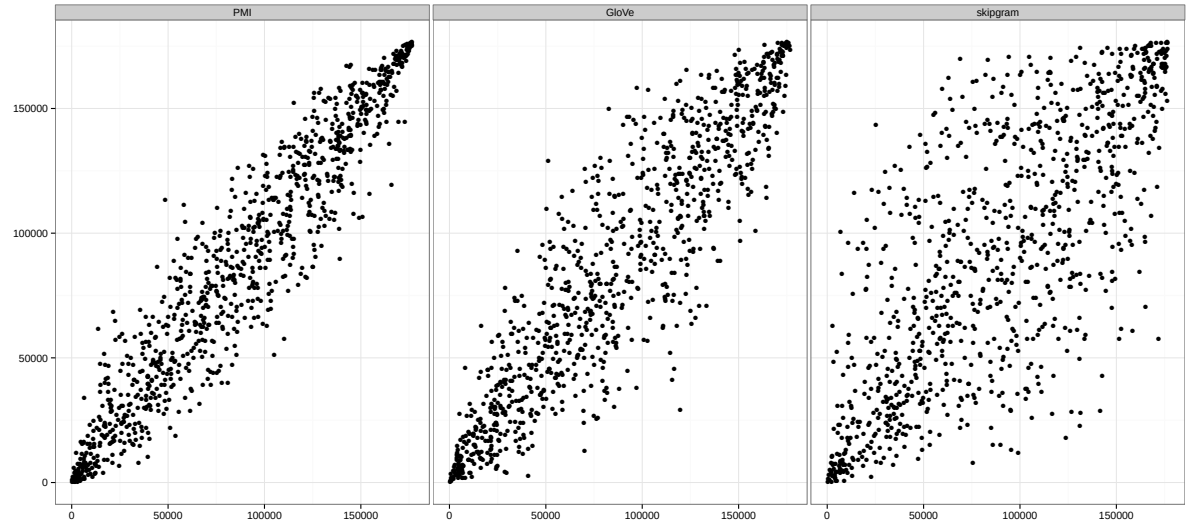


Figure 3: Neighborhood reciprocity in the different models; PMI to the left, GloVe in the middle, and Skipgram to the right.

Table 3: k-RNG for $k = 1,000$ of the words "service," "bad," and "above" in three different semantic models. The numbers in parenthesis indicate the k-NN ranks of the neighbors.

PMI	**GloVe**	**Skipgram**
service		
services (1) network (2) operates (8) launched (18) served (22) intercity(34)	services (1) operated (3) serving (6) military (17) duty (20) passenger (21) dialaride (644) aftersales (759) limitedstop (802)	services (1)
bad		
terrible (1) that (2) luck (39) unfortunate (70) stalling (276) donnersbergkreis (860) rancid (980)	good (1) kissingen (2) ugly (45) nasty (48) dirty (106) omen (328) conkers (360) karma (952)	nauheim (1) good (2) dreadful (5)
above		
below (1) around (2) feet (5) measuring (29) beneath (36) columns (62) atop (102)	below (1) level (2) height (3) just (4) stands (10) lower (11) beneath (12) rise (21) sea (30) ⋮	below (1) 500ft (2)

shows that the local densities vary much more in the Skipgram model than in the others. This is not in itself undesirable, but wild differences in neighborhood reciprocity complicates the choice of k in the k-RNG algorithm, as observed by the particularly sparse neighborhoods of the Skipgram model above.

6 WSI Evaluation

The standard way to evaluate WSI algorithms is to use one the SemEval WSI test collections (Agirre and Soroa, 2007; Manandhar et al., 2010; Navigli and Vannella, 2013; Jurgens and Klapaftis, 2013), which are all designed similarly: systems are expected to first perform WSI and then to assign texts to the induced senses (i.e. in effect doing a word-sense *disambiguation* step). We consider this type of evaluation to be a less useful for our purposes, since the required disambiguation step is a highly non-trivial task in itself. The RNG method proposed in this paper is a pure WSI algorithm, and as such does not offer a solution to the disambiguation problem. We therefore opted to focus solely on the hypothesis that relative neighborhoods cover senses that k-NNs do not. In essence, we investigate whether k-RNG *retrieval* does a better job at covering different senses than k-NN *retrieval*. This was done using *pseudowords*.

Pseudowords are artificially ambiguous words, created by regarding different words as identical. We can, for example, say that the pseu-

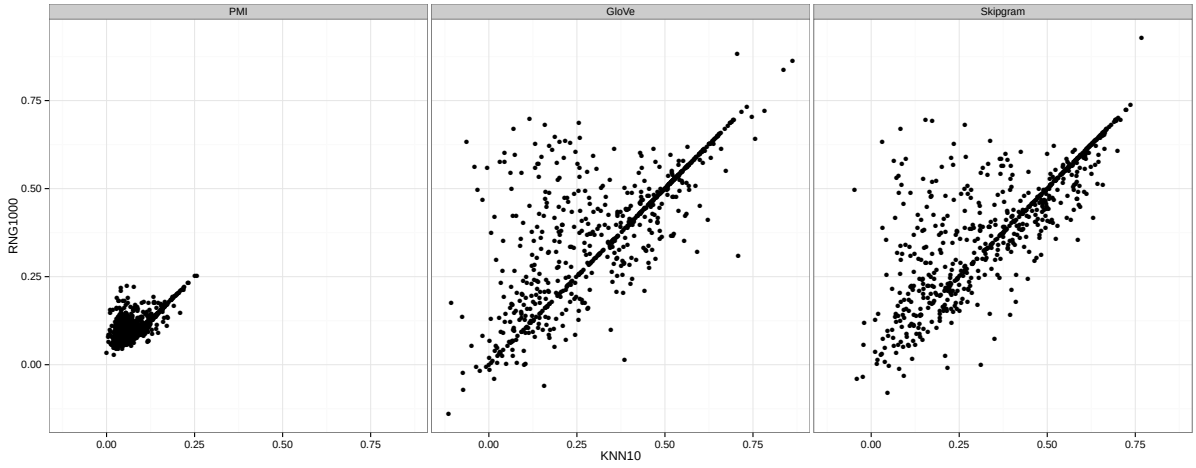


Figure 4: Comparison of minmax pseudosense score for k-RNGs and k-NNs for $k = 1,000$ and $k = 10$ respectively; PMI to the left, GloVe in the middle, and Skipgram to the right.

doword <*deadeye*> is a composite of the two words *marksman* and *loudspeaker*. A corpus with the artificially ambiguous word <*deadeye*> in it can then be created by replacing all occurrences of the words *marksman* and *loudspeaker* with <*deadeye*>.

Using the pseudowords provided by Pilehvar and Navigli (2013) a corpus with 689 non-overlapping pseudowords was created, based on the BNC corpus.[7] Two models were then trained, one on the altered corpus, and one on the unaltered one. To check whether the neighborhood of a pseudoword contains information about its underlying senses we compared each underlying sense to the words in the neighborhood, taking the minimum of all senses' maximum similarity as a score, as demonstrated in Table 4. The similarities were calculated using the model trained on the unaltered corpus, as the one based on the altered corpus will not contain the underlying senses of pseudowords.

Working through the example in Table 4, the neighborhood of the pseudoword <*deadeye* > consists of the three words *shooter*, *stereo*, and *sport*. The pseudoword in itself is made up of the two underlying senses *marksman* and *loudspeaker*. The similarities between the words in the neighborhood of the model trained on the unaltered data and the words of the underlying senses are as presented in Table 4. The closest word to *marksman* is *shooter*, with a similarity score of 0.7. The closest word to *loudspeaker* is *stereo*, with a score of 0.3. So the scoring would, in total, be 0.3. It should be noted that the upper bound for this score is oftentimes significantly lower than 1: The neighborhood could not possibly contain the words *marksman* or *loudspeaker*, as those words are not present in the corpus. This means that the scores are bounded by the similarity of the least similar closest neighbor to the underlying senses.

Table 4: Example scoring of a neighborhood of the word <*deadeye*>.

<*deadeye*>	*shooter*	*stereo*	*sport*	**max**
marksman	0.7	0.04	0.4	0.7
loudspeaker	0.01	0.3	0.05	0.3
min: 0.3				

This score was chosen because of its simplicity and intuitive interpretation: a low score implies that at least one word sense was not represented in the neighborhood whereas a high score means that all senses are represented in the neighborhood. One can then plot these scores for both relative neighborhoods and k-NN neighborhoods for each pseudoword as is done in Figure 4. Each

[7] www.natcorp.ox.ac.uk

point (x, y) represents a pseudoword, with x and y being the score of the k-NN neighborhood and the k-RNG neighborhood respectively.

Figure 5 shows an aggregate of Figure 4, plotting the distribution of $y - x$, i.e. the difference between the scores achieved by the k-RNG and the k-NN. As seen in Figure 4, a lot of points lie on the line $y = x$, meaning both methods achieved the same score. However, when this is not the case, there is a clear bias for the k-RNG to outperform the k-NN, as demonstrated in Figure 5. Here, using the BNC instead of Wikipedia as training data, the GloVe and Skipgram models yielded sparse relative neighborhoods — both with an average of about 8 neighbors — but the PMI model produced quite dense neighborhoods averaging 63 neighbors. Since the scoring function does not penalize neighborhood size there is good reason to be skeptical of its viability, and specifically the performance of the PMI-model based on these figures.

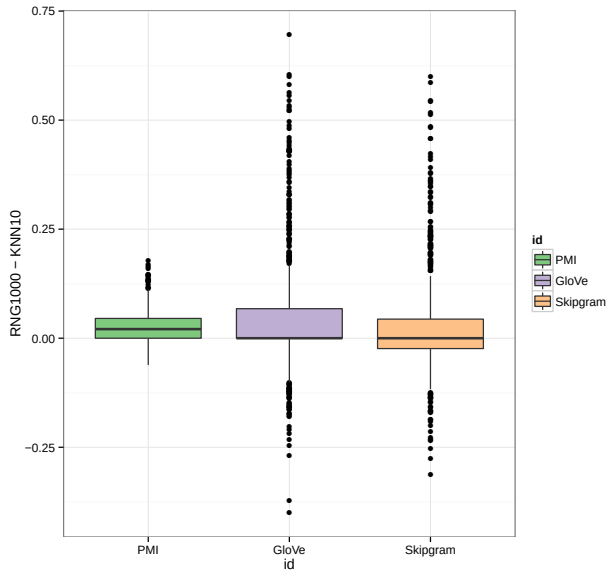


Figure 5: Distribution of difference between scores for k-RNGs and k-NNs. Positive scores means that the k-RNG scored higher than the k-NN

7 Conclusions

This paper has discussed the question how to query semantic models, which is a question that has been long neglected in research on computational semantics. Nearest neighbor search (or k-NN) is often treated as the only available option, which leads to misunderstandings regarding how semantic models represent and handle vagueness and polysemy. We have argued that the structure — or topology — of the local neighborhoods in semantic models carry useful semantic information regarding the different usages — or senses — of a term, and that such topological properties therefore can be used to analyze polysemy and do WSI.

We have introduced relative neighborhood graphs (RNG) as an alternative to standard k-NN, and we have exemplified k-RNG in three different well-known semantic models. The examples demonstrate that k-RNG manages to retrieve disparate and relevant neighbors in all three models, yet the kind of neighbors returned and the nature of the neighborhoods differ. Quantitatively, The k-RNG method consistently outperformed k-NN on underlying sense retrieval.

We have also illustrated how k-RNG can be used as a tool to gain insight into the topological properties of different models. The GloVe model, for example, makes no difference between sequential and substitutable relations, leading to neighborhoods that contain n-grams instead of senses. This can clearly be seen in for example Table 2. Skipgram uses more sophisticated tokenization, which alleviates this issue.

Another interesting result of the paper is that the RNG uncovers otherwise unseen differences between the models, which manifest not as scoring differences but as properties of the word representations themselves. One example is the differences in neighborhood reciprocity observed between the different models.

References

Eneko Agirre and Aitor Soroa. 2007. Semeval-2007 task 02: Evaluating word sense induction and discrimination systems. In *Proceedings of SemEval*, pages 7–12.

Eneko Agirre, David Martínez, Oier López de Lacalle, and Aitor Soroa. 2006. Two graph-based algorithms for state-of-the-art WSD. In *Proceedings of EMNLP*, pages 585–593.

Sunil Arya, David M. Mount, Nathan S. Netanyahu, Ruth Silverman, and Angela Y. Wu. 1998. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *Journal of the ACM*, 45(6):891–923.

Jon Louis Bentley. 1975. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517.

Peer-Timo Bremer, Ingrid Hotz, Valerio Pascucci, and Ronald Peikert. 2014. *Topological Methods in Data Analysis and Visualization III*. Springer.

Sergey Brin and Larry Page. 1998. The anatomy of a large-scale hypertextual web search engine. In *Proceedings of WWW*, pages 107–117.

Samuel Brody and Mirella Lapata. 2009. Bayesian word sense induction. In *Proceedings of EACL*, pages 103–111.

Jean Cardinal, Sébastien Collette, and Stefan Langerman. 2009. Empty region graphs. *Computational geometry*, 42(3):183–195.

Carlos D Correa and Peter Lindstrom. 2012. Locally-scaled spectral clustering using empty region graphs. In *Proceedings of KDD*, pages 1330–1338.

Georgiana Dinu and Mirella Lapata. 2010. Measuring distributional similarity in context. In *Proceedings of EMNLP*, pages 1162–1172.

Beate Dorow and Dominic Widdows. 2003. Discovering corpus-specific word senses. In *Proceedings of EACL*, pages 79–82.

Katrin Erk and Sebastian Padó. 2010. Exemplar-based models for word meaning in context. In *Proceedings of ACL*, pages 92–97.

Piotr Indyk and Rajeev Motwani. 1998. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of STOC*, pages 604–613.

David Jurgens and Ioannis Klapaftis. 2013. Semeval-2013 task 13: Word sense induction for graded and non-graded senses. In *Proceedings of SemEval*, pages 290–299.

Jussi Karlgren, Anders Holst, and Magnus Sahlgren. 2008. Filaments of meaning in word space. In *Proceedings of ECIR*, pages 531–538.

Maria Koptjevskaja Tamm and Magnus Sahlgren. 2014. Temperature in word space. In Benedikt Szmrecsanyi and Bernhard Wälchli, editors, *Aggregating dialectology, typology, and register analysis*, pages 231–267. De Gruyter.

Jey Han Lau, Paul Cook, Diana McCarthy, David Newman, and Timothy Baldwin. 2012. Word sense induction for novel sense detection. In *Proceedings of EACL*, pages 591–601.

Suresh Manandhar, Ioannis P. Klapaftis, Dmitriy Dligach, and Sameer S. Pradhan. 2010. Semeval-2010 task 14: Word sense induction & disambiguation. In *Proceedings of SemEval*, pages 63–68.

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

Roberto Navigli and Daniele Vannella. 2013. Semeval-2013 task 11: Word sense induction and disambiguation within an end-user application. In *Proceedings of SemEval*, pages 193–201.

Patrick Pantel and Dekang Lin. 2002. Discovering word senses from text. In *Proceedings of KDD*, pages 613–619.

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

Fernando Pereira, Naftali Tishby, and Lillian Lee. 1993. Distributional clustering of english words. In *Proceedings of ACL*, pages 183–190.

Mohammad Taher Pilehvar and Roberto Navigli. 2013. Paving the way to a large-scale pseudosense-annotated dataset. In *Proceedings of NAACL-HLT*, pages 1100–1109.

Diarmuid Ó Séaghdha and Anna Korhonen. 2011. Probabilistic models of similarity in syntactic context. In *Proceedings of EMNLP*, pages 1047–1057.

Noriko Tomuro, Steven L. Lytinen, Kyoko Kanzaki, and Hitoshi Isahara. 2007. Clustering using feature domain similarity to discover word senses for adjectives. In *Proceedings of ICSC*, pages 370–377.

Godfried T. Toussaint. 1980. The relative neighbourhood graph of a finite planar set. *Pattern Recognition*, 12(4):261 – 268.

Tim Van de Cruys and Marianna Apidianaki. 2011. Latent semantic word sense induction and disambiguation. In *Proceedings of HLT*, pages 1476–1485.

Jean Véronis. 2004. HyperLex: lexical cartography for information retrieval. *Computer Speech & Language*, 18(3):223–252.

Xuchen Yao and Benjamin Van Durme. 2011. Nonparametric bayesian word sense induction. In *Proceedings of TextGraphs*, pages 10–14.

Multi- and Cross-Modal Semantics Beyond Vision: Grounding in Auditory Perception

Douwe Kiela
Computer Laboratory
University of Cambridge
douwe.kiela@cl.cam.ac.uk

Stephen Clark
Computer Laboratory
University of Cambridge
stephen.clark@cl.cam.ac.uk

Abstract

Multi-modal semantics has relied on feature norms or raw image data for perceptual input. In this paper we examine grounding semantic representations in raw auditory data, using standard evaluations for multi-modal semantics, including measuring conceptual similarity and relatedness. We also evaluate cross-modal mappings, through a zero-shot learning task mapping between linguistic and auditory modalities. In addition, we evaluate multi-modal representations on an unsupervised musical instrument clustering task. To our knowledge, this is the first work to combine linguistic and auditory information into multi-modal representations.

1 Introduction

Although distributional models (Turney and Pantel, 2010; Clark, 2015) have proved useful for a variety of NLP tasks, the fact that the meaning of a word is represented as a distribution over other words implies that they suffer from the *grounding problem* (Harnad, 1990); i.e. they do not account for the fact that human semantic knowledge is grounded in the perceptual system (Louwerse, 2008). Motivated by human concept acquisition, multi-modal semantics enhances linguistic representations with extra-linguistic perceptual input. These models outperform language-only models on a range of tasks, including modelling semantic similarity and relatedness, and predicting compositionality (Silberer and Lapata, 2012; Roller and Schulte im Walde, 2013; Bruni et al., 2014). Although feature norms have also been used, raw image data has become the de-facto perceptual modality in multi-modal models.

However, if the objective is to ground semantic representations in perceptual information, why stop at image data? The meaning of *violin* is surely not only grounded in its visual properties, such as shape, color and texture, but also in its sound, pitch and timbre. To understand how perceptual input leads to conceptual representation, we should use as many perceptual modalities as possible. A recent preliminary study by Lopopolo and van Miltenburg (2015) found that it is possible to derive uni-modal semantic representations from sound data. Here, we explore taking multi-modal semantics beyond its current reliance on image data and experiment with grounding semantic representations in the auditory perceptual modality.

Multi-modal models that rely on raw image data have typically used "bag of visual words" (BoVW) representations (Sivic and Zisserman, 2003). We follow a similar approach for the auditory modality and construct bag of audio words (BoAW) representations. Following previous work in multi-modal semantics, we evaluate these models on measuring conceptual similarity and relatedness, and inducing cross-modal mappings between modalities to perform zero-shot learning. In addition, we evaluate on an unsupervised musical instrument clustering task. Our findings indicate that multi-modal representations enriched with auditory information perform well on relatedness and similarity tasks, particularly on words that have auditory assocations. To our knowledge, this is the first work to combine linguistic and auditory representations in multi-modal semantics.

2 Related Work

Information processing in the brain can be roughly described to occur on three levels: perceptual input, conceptual representation and symbolic reasoning (Gazzaniga, 1995). While research in AI has made great progress in understanding the first and last of these, understanding the middle level is still more of an open problem: how is it that per-

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 2461–2470,
Lisbon, Portugal, 17-21 September 2015.

ceptual input leads to conceptual representations that can be processed and reasoned with?

A key observation is that concepts are, through perception, *grounded* in physical reality and sensorimotor experience (Harnad, 1990; Louwerse, 2008), and there has been a surge of recent work on perceptually grounded semantic models that try to account for this fact. These models learn semantic representations from both textual and perceptual input, using either feature norms (Silberer and Lapata, 2012; Roller and Schulte im Walde, 2013; Hill and Korhonen, 2014) or raw image data (Feng and Lapata, 2010; Leong and Mihalcea, 2011; Bruni et al., 2014) as the source of perceptual information. A popular approach in the latter case is to collect images associated with a concept, and then lay out each image as a set of keypoints on a dense grid, where each keypoint is represented by a robust local feature descriptor such as SIFT (Lowe, 2004). These local descriptors are subsequently clustered into a set of "visual words" using a standard clustering algorithm such as k-means and then quantized into vector representations by comparing the descriptors with the centroids. An alternative to this bag of visual words (BoVW) approach is transferring features from convolutional neural networks (Kiela and Bottou, 2014).

Various ways of aggregating images into visual representations have been proposed, such as taking the mean or the elementwise maximum. Ideally, one would jointly learn multi-modal representations from parallel multi-modal data, such as text containing images (Silberer and Lapata, 2014) or images described with speech (Synnaeve et al., 2014), but such data is hard to obtain, has limited coverage and can be noisy. Hence, image representations are often learned independently. Aggregated visual representations are subsequently combined with a traditional linguistic space to form a multi-modal model. This mixing can be done in a variety of ways, ranging from simple concatenation to more sophisticated fusion methods (Bruni et al., 2014).

Cross-modal semantics, instead of being concerned with improving semantic representations through grounding, focuses on the problem of reference. Using, for instance, mappings between visual and textual space, the objective is to learn which words refer to which objects (Lazaridou et al., 2014). This problem is very much related to the object recognition task in computer vision, but instead of using just visual data and labels, these cross-modal models also utilize textual information (Socher et al., 2014; Frome et al., 2013). This allows for *zero-shot learning*, where the model can predict how an object relates to other concepts just from seeing an image of the object, but without ever having previously encountered an image of that particular object (Lazaridou et al., 2014). Multi-modal and cross-modal approaches have outperformed state-of-the-art text-based methods on a variety of tasks (Bruni et al., 2014; Silberer and Lapata, 2014).

MEN	score	SimLex-999	score
automobile-car	1.00	taxi-cab	0.92
rain-storm	0.98	plane-jet	0.81
cat-feline	0.96	horse-mare	0.83
jazz-musician	0.88	sheep-lamb	0.84
bird-eagle	0.88	bird-hawk	0.79
highway-traffic	0.88	band-orchestra	0.71
guitar-piano	0.86	music-melody	0.70

Table 1: Examples of pairs in the datasets where auditory is relevant, with the similarity score.

3 Evaluations

Following previous work in multi-modal semantics, we evaluate on two standard similarity and relatedness datasets: SimLex-999 (Hill et al., 2014) and the MEN test collection (Bruni et al., 2014). These datasets consist of concept pairs together with a human-annotated similarity or relatedness score, where the former dataset focuses on genuine similarity (e.g., *teacher-instructor*) and the latter focuses more on relatedness (e.g., *river-water*). In addition, following previous work in cross-modal semantics, we evaluate on the zero-shot learning task of inducing a cross-modal mapping to the correct label in the auditory modality from the linguistic one and vice-versa.

3.1 Multi-modal Semantics

Evidence suggests that the inclusion of visual representations only improves performance for certain concepts, and that in some cases the introduction of visual information is detrimental to performance on similarity and relatedness tasks (Kiela et al., 2014). The same is likely to be true for other perceptual modalities: in the case of comparisons such as *guitar-piano*, the auditory modal-

Dataset	MEN	AMEN	SLex	ASLex
Linguistic	3000	258	999	296
Auditory	2590	233	534	216

Table 2: Number of concept pairs for which representations are available in each modality.

ity is certainly meaningful, whereas in the case of *democracy-anarchism* it is probably less so. Therefore, we had two graduate students annotate the datasets according to whether auditory perception is relevant to the pairwise comparison. The annotation criterion was as follows: if both concepts in a pairwise comparison have a distinctive associated sound, the modality is deemed relevant. Inter-annotator agreement was high: $\kappa = 0.93$ for MEN and $\kappa = 0.92$ for SimLex-999. Some examples of relevant pairs can be found in Table 1. Hence, we now have four evaluation datasets: the MEN test collection **MEN** and its auditory-relevant subset **AMEN**; and the SimLex-999 dataset **SLex** and its auditory-relevant subset **ASLex**. Due to the nature of the auditory data sources, it is not possible to build auditory representations for all concepts in the test sets. Hence, unless stated otherwise, we report results for the covered subsets (using the same subsets when comparing across modalities, to ensure a fair comparison). Table 2 shows how much of the test sets are covered for each modality.[1]

3.2 Cross-modal Semantics

In addition to evaluating our models on the MEN and SimLex tasks, we evaluate on the cross-modal task of zero-shot learning. In the case of vision, Lazaridou et al. (2014) studied the possibility of predicting from "we found a cute, hairy wampimuk sleeping behind the tree" that a "wampimuk" will probably look like a small furry animal, even though a wampimuk has never been seen before. We evaluate zero-shot learning, using partial least squares regression (PLSR) to obtain cross-modal mappings from the linguistic to auditory space and vice versa.[2] Thus, given a linguistic representation for e.g. *guitar*, the task is to map it to the appropriate place in auditory space without ever having heard a guitar; or map it to the appropriate place in linguistic space without ever having read about a guitar (having only heard it).

4 Approach

One reason for using raw image data in multi-modal models is that there is a wide variety of resources that contain tagged images, such as ImageNet (Deng et al., 2009) and the ESP Game dataset (Von Ahn and Dabbish, 2004). However, such resources do not exist for audio files, and so we follow a similar approach to Fergus et al. (2005) and Bergsma and Goebel (2011), who use Google Images to obtain images. We use the online search engine Freesound[3] to obtain audio files. Freesound is a collaborative database released under Creative Commons licenses, in the form of snippets, samples and recordings, that is aimed at sound artists. The Freesound API allows users to easily search for audio files that have been tagged using certain keywords.

For each of the concepts in the evaluation datasets, we used the Freesound API to obtain samples encoded in the standard open source OGG format[4]. Because the database contains variable numbers of files, with varying duration per individual file, we restrict the search to a maximum of 50 files and a maximum of 1 minute duration per file. The Freesound API allows for various degrees of keyword matching: we opted for the strictest keyword matching, in that the audio file needs to have been purposely tagged with the given word (the alternative includes searching the text description for matching keywords). For example, if we are searching for audio files of cars, we retrieve up to 50 files with a maximum duration of 1 minute per file that have been tagged with the label "car".

4.1 Linguistic Representations

For the linguistic representations we use the continuous vector representations from the log-linear skip-gram model of Mikolov et al. (2013). Specifically, we trained 300-dimensional vector representations trained on a dump of the English Wikipedia plus newswire (8 billion words in total).[5] These types of representations have been found to yield the highest performance on a variety of semantic similarity tasks (Baroni et al., 2014).

[1] To facilitate further work in multi-modal semantics beyond vision, our code and data have been made publicly available at http://www.cl.cam.ac.uk/~dk427/audio.html.

[2] To avoid introducing another parameter, we set the number of latent variables in the cross-modal PLSR map to a third of the number of dimensions of the perceptual representation.

[3] http://www.freesound.org.

[4] http://www.vorbis.com.

[5] We used the demo-train-big-model-v1.sh script from http://word2vec.googlecode.com to obtain this corpus.

4.2 Auditory Representations

A common approach to obtaining acoustic features of audio files is the Mel-scale Frequency Cepstral Coefficient (MFCC) (O'Shaughnessy, 1987). MFCC features are abundant in a wide variety of applications in audio signal processing, ranging from audio information retrieval, to speech and speaker recognition, and music analysis (Eronen, 2003). Such features are derived from the mel-frequency cepstrum representation of an audio fragment (Stevens et al., 1937). In MFCC, frequency bands are spaced along the mel scale, which has the advantage that it approximates human auditory perception more closely than e.g. linearly-spaced frequency bands. Hence, MFCC takes human perceptual sensitivity to audio frequencies into consideration, which makes it suitable for e.g. compression and recognition tasks, but also for our current objective of modelling auditory perception. We obtain MFCC descriptors for frames of audio files using librosa, a popular library for audio and music analysis written in Python.[6] After having obtained the descriptors, we cluster them using mini-batch k-means (Sculley, 2010) and quantize the descriptors into a "bag of audio words" (BoAW) (Foote, 1997) representation by comparing the MFCC descriptors to the cluster centroids. This gives us BoAW representations for each of the audio files. Auditory representations are obtained by taking the mean of the BoAW representations of the relevant audio files, and finally weighting them using positive point-wise mutual information (PPMI), a standard weighting scheme for improving vector representation quality (Bullinaria and Levy, 2007). We set $k = 300$, which equals the number of dimensions for the linguistic representations.

4.3 Multi-modal Fusion Strategies

Since multi-modal semantics relies on two or more modalities, there are several ways of combining or *fusing* linguistic and perceptual cues (Bruni et al., 2014). When computing similarity scores, for instance, we can either jointly learn the representations; learn them independently, combine (e.g. concatenate) them and compute similarity scores; or learn them independently, compute similarity scores independently and combine the scores. We call these possibilities *early*, *middle* and *late* fusion, respectively, and evaluate multi-modal models in each category.

[6] http://bmcfee.github.io/librosa.

4.3.1 Early Fusion

A good example of early fusion is the recently introduced multi-modal skip-gram model (Lazaridou et al., 2015). This model behaves like a normal skip-gram, but instead of only having a training objective for the linguistic representation, it includes an additional training objective for the visual context, which consists of an aggregated representation of images associated with the given target word. The skip-gram training objective for a sequence of training words $w_1, w_2, ..., w_T$ and a context size c is:

$$\frac{1}{T}\sum_{t=1}^{T} J_\theta(w_t)$$

where J_θ is the log-likelihood $\sum_{-c\leq j\leq c} \log p(w_{t+j}|w_t)$ and $p(w_{t+j}|w_t)$ is obtained via the softmax:

$$p(w_{t+j}|w_t) = \frac{\exp^{u_{w_{t+j}}^\top v_{w_t}}}{\sum_{w'=1}^{W} \exp^{u_{w'}^\top v_{w_t}}}$$

where u_w and v_w are the context and target vector representations for the word w respectively, and W is the vocabulary size. The objective for the multi-modal skip-gram has an additional visual objective J_{vis} (in this case a margin criterion):

$$\frac{1}{T}\sum_{t=1}^{T} J_\theta(w_t) + J_{vis}(w_t)$$

Here, we take a similar but more straightforward approach by making the auditory context a part of the initial training objective, which is possible because linguistic and auditory representations have the same dimensionality. That is, we modified word2vec to predict additional auditory contexts that have been set to the corresponding BoAW representation. We jointly learn linguistic and audio representations by taking the aggregated mean μ_w^a of the auditory vectors for a given word w, and adding this mean vector to the context:

$$\frac{1}{T}\sum_{t=1}^{T} J_\theta(w_t) + \log p(\mu_{w_t}^a|w_t)$$

The intuition is that the induced vector for the target word now has to predict an auditory vector as part of its context, as well as the linguistic ones. As an alternative, we also investigate re-

placing the mean $\mu^a_{w_t}$ with an auditory vector obtained by uniformly sampling from the set of auditory representations for the target word. We refer to these two alternatives as MMSG-MEAN and MMSG-SAMPLED, respectively. For this model, auditory BoAW representations are built for the ten thousand most frequent words in our corpus, based on 10 audio files retrieved from FreeSound for each word (or fewer when 10 are not available).

4.3.2 Middle and Late Fusion

Whereas early fusion requires a joint training objective that takes into account both modalities, middle fusion allows for individual training objectives and independent training data. Similarity between two multi-modal representations is calculated as follows:

$$sim(u, v) = g(f(u^l, u^a), f(v^l, v^a))$$

where g is some similarity function, u^l and v^l are linguistic representations, and u^a and v^a are the auditory representations. A typical formulation in multi-modal semantics for $f(x, y)$ is $\alpha\, x \| (1-\alpha) y$, where $\|$ is concatenation (see e.g. Bruni et al. (2014) and Kiela and Bottou (2014)).

Late fusion can be seen as the converse of middle fusion, in that the similarity function is computed first before the similarity scores are combined:

$$sim(u, v) = h(g(u^l, v^l), g(u^a, v^a))$$

where g is some similarity function and h is a way of combining similarities, in our case a weighted average: $h(x, y) = \frac{1}{2}(\alpha\, x + (1-\alpha) y)$; and we use $g = \frac{x \cdot y}{|x||y|}$ (cosine similarity). Since cosine similarity is the normalized dot-product, and the uni-modal representations are themselves normalized, middle and late fusion are equivalent if $\alpha = 0.5$, which we call MM. However, when $\alpha \neq 0.5$, we distinguish between the two models, calling them MM-MIDDLE and MM-LATE respectively.

5 Results

5.1 Conceptual Similarity and Relatedness

We evaluate performance by calculating the Spearman ρ_s correlation between the ranking of the concept pairs produced by the automatic similarity metric (cosine between the derived vectors) and that produced by the gold-standard similarity scores. To ensure a fair comparison, we evaluate on the common subsets for which there are representations in both modalities (see Table 2).

Modality	MEN	AMEN	SLex	ASLex
Linguistic	0.687	0.603	0.320	0.314
Auditory	0.325	0.510	0.161	0.201
MMSG-MEAN	0.612	0.537	0.274	0.266
MMSG-SAMPLED	**0.690**	0.602	**0.321**	0.314
MM	0.680	**0.662**	0.314	**0.345**

Table 3: Spearman ρ_s correlation comparison of uni-modal and multi-modal representations. The MMSG models perform early fusion, MM represents middle and late fusion with $\alpha = 0.5$ (see Section 4.3.2).

The results are reported in Table 3. We find that, while performance decreases for linguistic representations on the auditory-relevant subsets of the two datasets, performance increases for the uni-modal auditory representations on those subsets. This indicates that our auditory representations are better at judging auditory-relevant comparisons than they are at non-auditory ones, as we might expect.

For all datasets, the accuracy scores for multi-modal models are at least as high as those for the purely linguistic representations. In the case of the full datasets this difference is only marginal, which is to be expected given how few of the words in the datasets are auditory-relevant. However, the results indicate that adding auditory input even for words that are not directly auditory-relevant is not detrimental to overall performance.

In the case of the auditory-relevant subsets, we see a large increase in performance when using multi-modal representations. It is also interesting that this performance increase is found in the simple MM model, compared to the more complicated MMSG models, which seems to indicate that the latter models are still too reliant on linguistic information, which harms their performance when performing auditory-specific comparisons. The model which performs consistently well across the four datasets is MM, the middle-late fusion model with $\alpha = 0.5$.

5.2 Cross-modal Zero-shot Learning

We learn a cross-modal mapping between the linguistic and auditory spaces using partial least squares regression, taking out each concept, training on the others, and then projecting from one

Mapping	P@1	P@5	P@20	P@50
Chance	0.00	0.93	4.01	8.49
Auditory ⇒ Ling.	0.77	6.48	17.54	31.33
Ling. ⇒ Auditory	0.73	6.71	22.16	37.32

Table 4: Cross-modal zero-shot learning accuracy.

space into the other. Zero-shot performance is evaluated using the average percentage correct at N (P@N), which measures how many of the test instances were ranked within the top N highest ranked nearest neighbors. Results are shown in Table 4, with the chance baseline obtained by randomly ranking a concept's nearest neighbors. Insofar as it is possible to make a direct comparison with linguistic-visual zero-shot learning (which uses entirely different data), it appears that the current task may be more difficult: Lazaridou et al. (2014) report a P@1 of 2.4 and P@20 of 33.0 for their linguistic-visual model.

5.3 Qualitative Analysis

We also performed a small qualitative analysis of the BoAW representations for the words in MEN and SLex. As Table 5 shows, the nearest neighbors are remarkably semantically coherent. For example, the model groups together sounds produced by machines, or by water. It even finds that dinner, meal, lunch and breakfast are closely related. In contrast, nearest neighbors for the linguistic model tend to be of a more abstract nature: where we find *mouth* and *throat* as auditory neighbors for *language*, the linguistic model gives us concepts like *word* and *dictionary*; while auditory *gossip* sounds like *maids* and is something you might do in the *corridor*, it is linguistically associated with more abstract concepts like *news* and *newspaper*.

6 Parameter Tuning

There are many parameters that were left fixed in the main results that could have been adjusted to improve performance, particularly in the middle and late fusion models. It is useful to investigate some of the parameters that are likely to have an impact on performance: what the effect of the α mixing parameter is, whether a different k would have yielded better auditory representations, and whether the number and duration of the audio files from FreeSound has any effect.

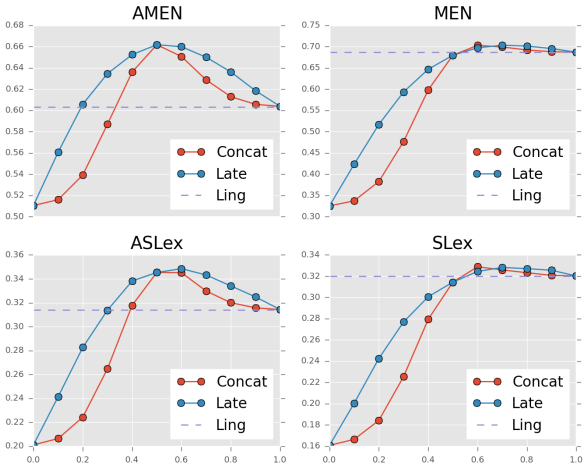


Figure 1: Performance of middle and late multimodal fusion models compared to linguistic representations on the four datasets when varying the α mixing parameter on the x-axis.

6.1 Mixing with α

The mixing parameter α plays an important role in the middle and late fusion models. We kept it fixed at 0.5 for the MM model above, but here we experiment with varying the parameter, yielding results for two different models, MM-MIDDLE and MM-LATE. The results are shown in Figure 1, where moving to the right on the x-axis uses more linguistic input and moving to the left uses more auditory input. The late model consistently outperforms the middle fusion model, which is probably because it is less susceptible to any noise in the auditory representation. Optimal performance seems to be around $\alpha = 0.6$ for both fusion strategies on all four datasets, indicating that it is better to include a little more linguistic than auditory input. It appears that any $0.5 \leq \alpha < 1$ (i.e., where we have more linguistic input but still some auditory signal), outperforms the purely linguistic representation, substantially in the case of the auditory-relevant subsets.

6.2 Number of Auditory Dimensions

We experimented with different values for the number of audio words k (i.e. the number of clusters in the k-means clustering that determines the number of "audio words"). As Figure 2 shows, the quality of the uni-modal auditory representations is highly robust to the number of dimensions. In fact, any choice of k in the range shown provides similar results across the datasets.

Auditory				Linguistic			
navy	language	gossip	dinner	navy	language	gossip	dinner
army	mouth	maid	meal	army	word	news	lunch
aviation	man	guest	lunch	military	words	newspaper	wedding
plane	father	elevator	writer	vessel	literature	cute	meal
jet	adult	danger	breakfast	sunk	dictionary	sexy	breakfast
cannon	throat	corridor	couch	ship	tongue	mirror	cocktail
monster	motor	water	dawn	monster	motor	water	dawn
orchestra	engine	stream	summer	zombie	vehicle	droplets	dusk
demon	rain	bath	child	demon	automobile	salt	sunrise
guitar	beach	river	victor	dragon	car	cold	moon
beast	boat	bathroom	morning	beast	motorcycle	sunlight	night
pilot	car	rain	garden	creatures	truck	milk	misty

Table 5: Example nearest neighbors for auditory (BoAW) representations and linguistic representations.

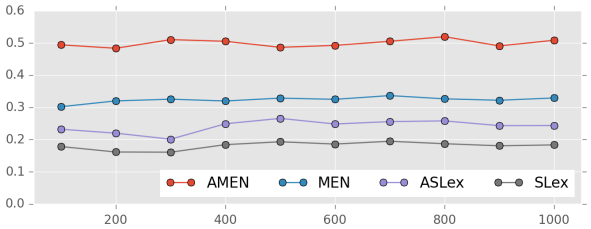


Figure 2: Performance of uni-modal auditory representations on the four datasets when varying the k parameter.

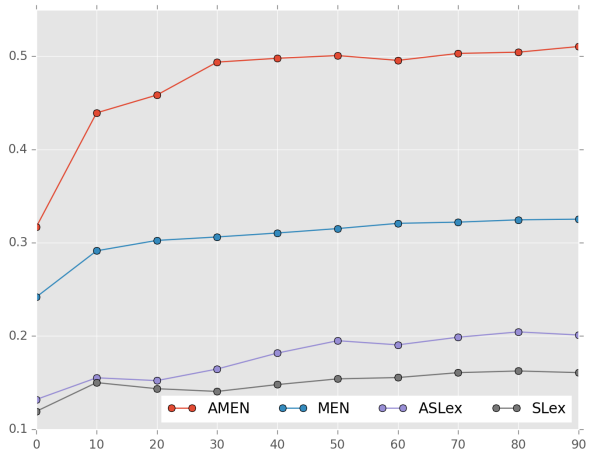


Figure 3: Performance of uni-modal auditory representations on the four datasets when varying the number of audio files per target word.

6.3 Number and Duration of Audio Files

We experimented with the number of audio files by querying FreeSound for up to 100 audio files per search word, while keeping $k = 300$. The results are shown in Figure 3. It appears that "the more the better", although performance does not increase significantly after around 40 audio files.

In order to examine the effect of audio file duration, we experimented with specifying the duration of audio files when querying the database, either taking very short (up to 5 seconds), medium length (up to 1 minute) or files of any duration. The results can be found in Figure 4, showing that performance generally increases as the files get longer (except on AMEN where a duration of 1 minute provides optimal performance).

7 Case Study: Musical Instruments

To strengthen the finding that multi-modal representations perform well on the auditory-relevant subsets of the datasets, we evaluate on an altogether different task, namely that of musical instrument classification. We used Wikipedia to collect a total of 52 instruments and divided them into 5 classes: brass, percussion, piano-based, string and woodwind instruments. For each of the instruments, we collected as many audio files from FreeSound as possible, and used the MM-MIDDLE model with parameter settings that yielded good results in the previous experiments ($k = 300$ and $\alpha = 0.6$). We then performed k-means clustering with five cluster centroids and compared results between auditory, linguistic and multi-modal, evaluating the clustering quality using the standard V-measure clustering evaluation metric (Rosenberg and Hirschberg, 2007).

This is an interesting problem because instrument classes are determined somewhat by conven-

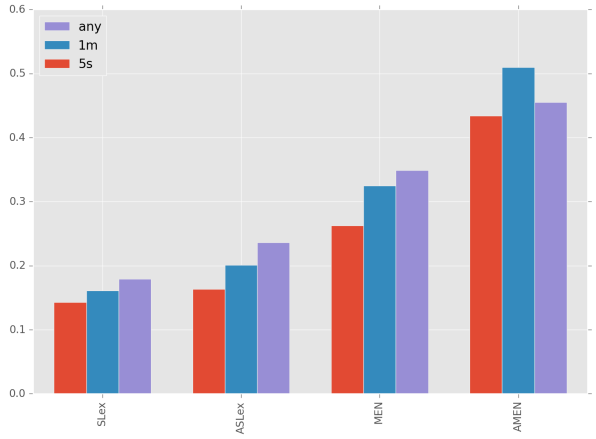


Figure 4: Performance of uni-modal auditory representations on the four datasets when varying the maximum duration.

Model	Auditory	Linguistic	MM-MIDDLE
V-measure	0.39	0.47	0.54

	Linguistic
1	baritone
2	lute, zither, xylophone, lyre, cymbals
3	piano, trombone, clarinet, cello, violin
4	castanets, tambourine, claves, maracas
5	trumpet, horn, bugle, cowbell, carillon

	Multi-modal
1	drum, claves, bongo, bass, conga
2	xylophone, glockenspiel, tambourine, cymbals
3	cello, piano, clarinet, trombone, violin,
4	chimes, bell
5	mandolin, banjo, harmonica, guitar, sitar

Table 6: V-measure performance for clustering musical instruments, together with instruments closest to cluster centroid for linguistic and multi-modal.

tion (is a saxophone a brass or a woodwind instrument?). What is more, how instruments actually sound is rarely described in detail in text, so corpus-based linguistic representations cannot take this information into account. The results are in Table 6, clearly showing that the multi-modal representation which utilizes both linguistic information and auditory input performs much better on this task than the uni-modal representations. It is interesting to observe that the linguistic representations perform better than the auditory ones: a possible explanation for this is that audio files in FreeSound are rarely samples of a single individual instrument, so if a bass is often accompanied by a drum this will affect the overall representation. The table also shows, for the 5 clusters under both models, the nearest instruments to the cluster centroids, qualitatively demonstrating the greater cluster coherence for the multi-modal model.

8 Conclusions

We have studied grounding semantic representations in raw auditory perceptual information, using a bag of audio words model to obtain auditory representations, and combining them into multi-modal representations using a variety of fusion strategies. Following previous work in multi-modal semantics, we evaluated on conceptual similarity and relatedness datasets, and on the cross-modal task of zero-shot learning. We presented a short case study showing that multi-modal representations perform much better than auditory or linguistic representations on a musical instrument clustering task. It may well be the case that the auditory modality is better suited for other evaluations, but we have chosen to follow standard evaluations in multi-modal semantics to allow for a direct comparison.

In future work, it would be interesting to investigate different sampling strategies for the early fusion joint-learning approach and to investigate more sophisticated mixing strategies for the middle and late fusion models, e.g. using the "audio dispersion" of a word to determine how much auditory input should be included in the multi-modal representation (Kiela et al., 2014). Another interesting possibility is to improve auditory representations by training a neural network classifier on the audio files and subsequently transferring the hidden representations to tasks in semantics. Lastly, now that the perceptual modalities of vision, audio and even olfaction (Kiela et al., 2015) have been investigated in the context of distributional semantics, the logical next step for future work is to explore different fusion strategies for multi-modal models that combine various sources of perceptual input into a single grounded model.

Acknowledgments

DK is supported by EPSRC grant EP/I037512/1. SC is supported by ERC Starting Grant DisCoTex (306920) and EPSRC grant EP/I037512/1. We are grateful to Xavier Serra, Frederic Font Corbera, Alessandro Lopopolo and Emiel van Miltenburg

for useful suggestions and thank the anonymous reviewers for their helpful comments.

References

Marco Baroni, Georgiana Dinu, and Germán Kruszewski. 2014. Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors. In *ACL*, pages 238–247.

Shane Bergsma and Randy Goebel. 2011. Using visual information to predict lexical preference. In *Proceedings of RANLP*, pages 399–405.

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

John A. Bullinaria and Joseph P. Levy. 2007. Extracting Semantic Representations from Word Co-occurrence Statistics: A computational study. *Behavior Research Methods*, 39:510–526.

Stephen Clark. 2015. Vector Space Models of Lexical Meaning. In Shalom Lappin and Chris Fox, editors, *Handbook of Contemporary Semantics*, chapter 16. Wiley-Blackwell, Oxford.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *Proceedings of Computer Vision and Pattern Recognition (CVPR)*, pages 248–255.

A. Eronen. 2003. Musical instrument recognition using ICA-based transform of features and discriminatively trained HMMs. In *Proceedings of the Seventh International Symposium on Signal Processing and Its Applications*, volume 2, pages 133–136.

Yansong Feng and Mirella Lapata. 2010. Visual information in semantic representation. In *Proceedings of NAACL*, pages 91–99.

Robert Fergus, Fei-Fei Li, Pietro Perona, and Andrew Zisserman. 2005. Learning object categories from Google's image search. In *Proceedings of ICCV*, pages 1816–1823.

Jonathan T Foote. 1997. Content-based retrieval of music and audio. In *Voice, Video, and Data Communications*, pages 138–147.

Andrea Frome, Gregory S. Corrado, Jonathon Shlens, Samy Bengio, Jeffrey Dean, Marc'Aurelio Ranzato, and Tomas Mikolov. 2013. DeViSE: A Deep Visual-Semantic Embedding Model. In *Proceedings of NIPS*, pages 2121–2129.

Michael S. Gazzaniga, editor. 1995. *The Cognitive Neurosciences*. MIT Press, Cambridge, MA.

Stevan Harnad. 1990. The symbol grounding problem. *Physica D*, 42:335–346.

Felix Hill and Anna Korhonen. 2014. Learning abstract concept embeddings from multi-modal data: Since you probably can't see what I mean. In *Proceedings of EMNLP*, pages 255–265.

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

Douwe Kiela and Léon Bottou. 2014. Learning image embeddings using convolutional neural networks for improved multi-modal semantics. In *Proceedings of EMNLP*, pages 36–45.

Douwe Kiela, Felix Hill, Anna Korhonen, and Stephen Clark. 2014. Improving multi-modal representations using image dispersion: Why less is sometimes more. In *Proceedings of ACL*, pages 835–841.

Douwe Kiela, Luana Bulat, and Stephen Clark. 2015. Grounding semantics in olfactory perception. In *Proceedings of ACL*, pages 231–236, Beijing, China, July.

Angeliki Lazaridou, Elia Bruni, and Marco Baroni. 2014. Is this a wampimuk? Cross-modal mapping between distributional semantics and the visual world. In *Proceedings of ACL*, pages 1403–1414.

Angeliki Lazaridou, Nghia The Pham, and Marco Baroni. 2015. Combining language and vision with a multimodal skipgram model. In *Proceedings of NAACL*.

Chee Wee Leong and Rada Mihalcea. 2011. Going beyond text: A hybrid image-text approach for measuring word relatedness. In *Proceedings of IJCNLP*, pages 1403–1407.

A. Lopopolo and E. van Miltenburg. 2015. Sound-based distributional models. In *Proceedings of the 11th International Conference on Computational Semantics (IWCS 2015)*.

Max M. Louwerse. 2008. Symbol interdependency in symbolic and embodied cognition. *Topics in Cognitive Science*, 59(1):617–645.

David G. Lowe. 2004. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *Proceedings of ICLR*, Scottsdale, Arizona, USA.

D. O'Shaughnessy. 1987. *Speech communication: human and machine*. Addison-Wesley series in electrical engineering: digital signal processing. Universities Press (India) Pvt. Limited.

Stephen Roller and Sabine Schulte im Walde. 2013. A multimodal LDA model integrating textual, cognitive and visual modalities. In *Proceedings of EMNLP*, pages 1146–1157.

Andrew Rosenberg and Julia Hirschberg. 2007. V-measure: A conditional entropy-based external cluster evaluation measure. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 410–420.

David Sculley. 2010. Web-scale k-means clustering. In *Proceedings of WWW*, pages 1177–1178. ACM.

Carina Silberer and Mirella Lapata. 2012. Grounded models of semantic representation. In *Proceedings of EMNLP*, pages 1423–1433.

Carina Silberer and Mirella Lapata. 2014. Learning grounded meaning representations with autoencoders. In *Proceedings of ACL*, pages 721–732.

Josef Sivic and Andrew Zisserman. 2003. Video google: A text retrieval approach to object matching in videos. In *Proceedings of ICCV*, pages 1470–1477.

Richard Socher, Andrej Karpathy, Quoc V. Le, Christopher D. Manning, and Andrew Y. Ng. 2014. Grounded compositional semantics for finding and describing images with sentences. *Transactions of ACL*, 2:207–218.

Stanley Smith Stevens, John Volkmann, and Edwin B. Newman. 1937. A scale for the measurement of the psychological magnitude pitch. *Journal of the Acoustical Society of America*, 8(3):185—190.

Gabriel Synnaeve, Maarten Versteegh, and Emmanuel Dupoux. 2014. Learning words from images and speech. In *NIPS Workshop on Learning Semantics*, Montreal, Canada.

Peter D. Turney and Patrick Pantel. 2010. From Frequency to Meaning: vector space models of semantics. *Journal of Artifical Intelligence Research*, 37(1):141–188, January.

Luis Von Ahn and Laura Dabbish. 2004. Labeling images with a computer game. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 319–326. ACM.

Automatic recognition of habituals: a three-way classification of clausal aspect

Annemarie Friedrich **Manfred Pinkal**
Department of Computational Linguistics
Saarland University, Saarbrücken, Germany
{afried,pinkal}@coli.uni-saarland.de

Abstract

This paper provides the first fully automatic approach for classifying clauses with respect to their aspectual properties as habitual, episodic or static. We bring together two strands of previous work, which address only the related tasks of the episodic-habitual and stative-dynamic distinctions, respectively. Our method combines different sources of information found to be useful for these tasks. We are the first to exhaustively classify *all* clauses of a text, achieving up to 80% accuracy (baseline 58%) for the three-way classification task, and up to 85% accuracy for related subtasks (baselines 50% and 60%), outperforming previous work. In addition, we provide a new large corpus of Wikipedia texts labeled according to our linguistically motivated guidelines.

1 Introduction

In order to understand the function of a clause within a discourse, we need to know the clause's aspectual properties. The distinction between *dynamic* and *stative* lexical aspect, as in examples (1a) and (1b) respectively, is fundamental (Vendler, 1957). Its automatic prediction has been addressed previously (Siegel and McKeown, 2000; Zarcone and Lenci, 2008; Friedrich and Palmer, 2014).

(1) (a) Bill drank a coffee after lunch. (**dynamic**)
(b) Bill likes coffee. (**stative**)

In this work, we focus on *habituality* as another fundamental aspectual property. While example (1a) is an *episodic* sentence, i.e., a sentence expressing information about a particular event, the same dynamic verb can be used to characterize the default behavior of an individual or of a kind in a certain type of situation (2).

(2) (a) Bill usually drinks coffee after lunch. (**habitual**)
(b) Italians drink coffee after lunch. (**habitual**)

The entailment properties of episodic and habitual (or *characterizing*) sentences differ substantially. Also, they have different functions in discourse. The episodic event expressed by (1a) is typically embedded in the temporal structure of a narration. The habitual sentence (2a) can be used, e.g., as an explanation (why Bill is still sitting at the table), or in a contrastive context (today, he ordered a grappa instead). *Generic* sentences with *kind-referring* subjects (2b) can also be habitual, generalizing at the same time over typical members of this kind and over situations in which they typically carry out some action.

Habitual sentences do not move narrative time, similar to stative clauses such as (1b). Carlson (2005) considers habituals to be aspectually stative. Since there are clear differences between ordinary statives such as (1b) and habituals, we apply a three-way distinction for *clausal aspect* in this work. We classify clauses as one of the three categories **habitual**, **episodic** and **static**.[1]

Through its impact on entailment properties and temporal discourse structure, the determination of clausal aspect is relevant to various natural language processing applications requiring text understanding, such as novelty detection (Soboroff and Harman, 2005), knowledge extraction from text (Van Durme, 2010) or question answering (Llorens et al., 2015). Using aspectual information has been shown to improve temporal relation identification (Costa and Branco, 2012).

Some languages (e.g., Czech or Swahili) have systematic morphological markers of habituality

[1]For clarity, we use the label *static* for the clausal aspect of non-episodic and non-habitual sentences. We reserve *stative*, which is more common in the literature, for the lexical aspectual class.

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 2471–2481,
Lisbon, Portugal, 17-21 September 2015.

(Dahl, 1985). In other languages, there are cues for habituality, such as the simple present in English, and the use of certain adverbials (Dahl, 1995). The automatic recognition of habitual sentences for the latter languages is non-trivial. The work in this paper targets the English language; we leave recognition of habituality in other languages to future work.

The only previous work on categorizing sentences as episodic or habitual we know of is by Mathew and Katz (2009). They do not attempt to categorize arbitrary sentences in 'free text', however, but work with a corpus of selected sentences and use gold standard parse information for their experiments. In particular, they consider clauses containing lexically dynamic verbs only.

In this work, we address the task of an exhaustive classification of all clauses of a text into the three aspectual classes **habitual**, **episodic**, and **static**. Static sentences include lexically stative clauses as well as negated or modalized clauses containing a dynamic main verb. A computational model for identifying episodic and habitual clauses clearly needs to address this third class as well if it is to be applied in a realistic setting. Linguistically, the determination of clausal aspect depends on the recognition of the verb's lexical aspectual class (stative or dynamic), and on the recognition of any aspectual markers or transformations, such as use of the perfect tense, negations or modals. Our work builds on results for the related subtasks (Mathew and Katz, 2009; Siegel and McKeown, 2000; Friedrich and Palmer, 2014), using both context-based and verb-type based information.

Our major contributions are: (i) We create a corpus of 102 Wikipedia texts whose about 10,000 clauses are annotated as episodic, static or habitual with substantial agreement. This corpus allows for studying the range of linguistic phenomena related to the clause types as defined above (compared to previous work which uses only a small set of verbs and sentences), and provides a basis for future research. (ii) We provide the first fully automatic approach for this classification task, combining two classification tasks (lexical aspectual class and habituality) that have been treated separately in previous work. For an exhaustive classification of clauses of free text, these two classification tasks need to be addressed jointly. We show two different feature sets (verb-type based features and context-based features) to have different impact on the two subtasks, and to be complementary for our full three-way task.

We reach accuracies of nearly 85% for the two subtasks of identifying static clauses and distinguishing episodic and habitual clauses (majority class baselines are 60% and 50% respectively). A joint model for the three-way classification task reaches an accuracy of 80% (baseline 60%). In addition, we show that the verb-type based linguistic indicator features generalize well across verb types on our tasks: for verbs unseen in the training data, accuracies drop only by 2-5%.

2 Theoretical background and annotation guidelines

In this section, we give an overview of the semantic theory related to habituals, at the same time introducing our annotation guidelines for marking clauses as **habitual**, **episodic** or **static**.

2.1 Habituality

Habitual sentences express regularities in terms of generalizations over events and activities. In semantic theory, habituals are formally represented using a quantifier-like operator GEN (Krifka et al., 1995):

(3) GEN[s](s **is an after-dinner situation** & Bill **is involved in** s; Bill **drinks a coffee in** s)

In the semi-formal representation (3) of sentence (2a) above, GEN binds a variable s ranging over situations, the first argument restricts the situation type, and the second argument provides the activity that is typically carried out by the protagonist in the respective situations. The GEN operator is similar to the universal quantifier of predicate logic. However, habitual sentences tolerate exceptions: (2a) is true even if Bill does not drink a coffee after every lunch. Also note that habituals are not restricted to what one would consider a matter of habit; they can also have inanimate subjects, as illustrated by (4).

(4) Glass breaks easily. (**habitual**)

2.2 Clausal and lexical aspectual class

Clausal aspect is dependent on the lexical aspectual class (stative or dynamic), but the two levels are essentially different. Dynamic verbs express events or activities (e.g., *kill, fix, walk, forget*), while stative verbs express states (e.g., *be,*

like, know, own). The *fundamental aspectual class* (Siegel and McKeown, 2000) of a verb in context describes whether it is used in a stative or dynamic sense before any aspectual markers or transformations (such as use of the past/present perfect or modals) have been applied. It is a function of the main verb and a select group of complements (it may differ per verb which ones are important). For example, the fundamental lexical aspectual class of the verb *make* with the subject *Mary* and the object *cake* in (5) is dynamic. English clauses in past or present perfect such as (5) are static, as they focus on the post-state of an event rather than the event itself (Katz, 2003).

(5) Mary has made a cake. (**static**)

Habituals with verbs of dynamic aspectual class are by far more frequent in our corpus,[2] but there are also instances of stative verbs used in a habitual way, as for example (6).

(6) Sloths sometimes sit on top of branches. (**habitual**, *stative* lexical aspectual class)

2.3 Modality and negation

Modalized (7) and negated sentences (8) tend to be static: they do not express information about a particular event, but refer to actual or possible states of the world.

(7) Mary can swim. (**static**)

(8) Mary didn't go swimming yesterday. (**static**)

The above definitions of habituality and stativity are generally agreed upon in literature. However, the interaction of these phenomena is by no means trivial (Hacquard, 2009), and required making some decisions during the design of our annotation guidelines. Here, we explain these decisions, which are all motivated by a clause's entailment properties.

One difficult issue is how to interpret and mark negated sentences such as (9a) whose positive version (9b) is habitual.

(9) (a) John does not smoke. (**habitual**)
(b) John smokes. (**habitual**)

[2]The distribution of lexical aspectual class of verbs is generally skewed towards dynamic (Friedrich and Palmer, 2014).

Sentence (9a) can be considered either **static** because of the negation (*It is not the case that John smokes*), or as **habitual** because it characterizes John's behavior (*In any relevant situation, John does not smoke*). Both decisions are possible (Garrett, 1998), we decide for the latter possibility. This decision is supported by the observation that (9a) is similar in its entailment properties to (10), which due to the frequency adverbial *never* clearly generalizes over relevant situations (though note that this is not a linguistic test).

(10) John never smokes. (**habitual**)

Likewise, we mark modalized sentences as habitual if they have a strong implicature that an event has actually happened regularly (Hacquard, 2009), as in (11). In contrast, (7) is static as it does not imply that Mary actually swims regularly.

(11) I had to eat an apple every day. (**habitual**)

The above example shows that modality and habituality are interweaved and sometimes hard to identify. Nevertheless, we reach substantial agreement in the annotation of our corpus (see Section 4.2).

Finally, some habituals have a *dispositional* reading, indicating ability/capability (Menéndez-Benito, 2012). Example (12) can be paraphrased by (13), as it does not indicate that the car is actually driven this fast regularly, it only states its maximum speed.

(12) This car goes 200 kph.

(13) This car can go 200 kph.

3 Related work

The task of predicting *fundamental aspectual class* aims to determine whether the verb is used in a stative or dynamic sense. This task predicts the aspectual class of a verb in context before any aspectual markers or transformations (such as use of the perfect or modals) have been applied. Siegel and McKeown (2000) propose the use of linguistic indicators (explained in Section 5.2); Friedrich and Palmer (2014) show the importance of using context-based features in addition. Zarcone and Lenci (2008) classify occurrences of 28 Italian verbs according to Vendlers classes state, process, accomplishment and achievement.

Mathew and Katz (2009) address the problem of 'supervised categorization for habitual versus

episodic sentences' . The authors randomly select 1052 sentences for 57 verbs from the Penn TreeBank (Marcus et al., 1993) and manually mark them with regard to whether they are **habitual** or **episodic**. They focus on verbs that are lexically dynamic and discuss a variety of syntactic features, which they extract from gold standard parse trees. Their aim is to study the ability of syntactic features alone to identify habitual sentences.

Xue and Zhang (2014) annotate verbs with the four event types *habitual event*, *state*, *on-going event* and *episodic event* with the aim of improving tense prediction for Chinese. Recent related work (Williams, 2012; Williams and Katz, 2012) extracts typical durations (in term of actual time measures) for verb lemmas from Twitter. They distinguish episodic and habitual uses of the verbs, using the method of Mathew and Katz (2009).

4 Data

In this section, we describe the data sets used in our experiments.[3]

4.1 Penn TreeBank (M&K) data set

Mathew and Katz (2009) randomly select sentences for several verbs from the WSJ and Brown corpus sections of the Penn Treebank. They require the verb to be lexically dynamic. Sentences are marked as **habitual** or **episodic**, further details on the annotation guidelines are not specified. Their data set contains 2743 annotated sentences for 239 distinct verb types. Mathew and Katz remove verb types with highly skewed distributions of labels, but their filtered data set is not available. We follow their filtering approach, but we could not replicate their filtering step. Our final data set contains 1230 sentences for 54 distinct verb types. Mathew and Katz (2009) state that their data set comprises 1052 examples for 57 verb stems. We aimed at producing a similar distribution of labels: our data set contains 73.3% episodic cases, M&K's version has 73.1%.

4.2 Wikipedia corpus

We select 102 texts from a variety of domains from Wikipedia, as we expect an encyclopedia to contain many habitual sentences. We use the discourse parser SPADE (Soricut and Marcu, 2003) to automatically segment the first 70 sentences of each article into clauses. A clause is approximately defined as a finite verb phrase. Each clause is then labeled as **static**, **episodic** or **habitual**. Details on our annotation scheme have been given in Section 2. Annotators are allowed to skip non-finite clauses (e.g., headlines only containing a noun phrase). This happened in about 14% of all pre-segmented clauses. The final Wikipedia data consists of 10355 labeled clauses. Table 1 gives statistics for the distribution of labels.

Label	#	%
static	6184	59.7
episodic	2114	20.4
habitual	2057	19.9
total	10355	-

Table 1: Wikipedia data, distribution of labels for clausal aspect.

The data set was labeled by three paid annotators, all students of computational linguistics. Annotators were given a written manual and a short training on documents not included in the corpus. Agreement on the Wikipedia data is 0.61 in terms of Fleiss' κ, which indicates substantial agreement (Landis and Koch, 1977). The gold standard that we use in our experiments is constructed via majority voting. The gold standard contains the cases where at least two annotators agreed on the label. We found only 86 cases where all annotators disagree, and manual inspection shows that most of these cases are related to disagreements on the lexical aspectual class that coincide with an attention slip by one of the annotators.

5 Method

In this section, we describe our computational models for determining clausal aspect.

5.1 CONTEXT-BASED features

Table 2 shows the syntactic-semantic features, which we call CONTEXT-BASED as they are extracted from the context of each verb occurrence that we classify. This feature set comprises the features proposed by Mathew and Katz (2009) and the ones proposed by Friedrich and Palmer (2014). In addition, we use the features *modal* and *negated*. We extract these features from syntactic dependency parses created using the Stanford parser (Klein and Manning, 2002). Tense and voice are extracted following the rules pro-

[3] All data sets are freely available from `www.coli.uni-saarland.de/projects/sitent`. We thank Thomas A. Mathew and Graham Katz for allowing us to publish their data set.

Feature		Values
verb	tense*† pos† voice†	past, present, infinitive VB, VBG, VBN, ... active, passive
aspect	progressive*† perfect*†	true, false true, false
subject	bare plural* definite* indefinite*	true, false true, false true, false
object	absent* bare plural* definite* indefinite*	true, false true, false true, false true, false
grammatical dependents†		WordNet lexname/POS
sentence	modal	*would, can,...*
	negated	true, false
	conditionals*	presence of clause starting with if/when/ whenever
	temporal modifiers*	specific, quantificational, including *used to* and *would* (where no if)
	prepositions*	at / in / on (3 features, true/false)

Table 2: CONTEXT-BASED features. Used by: *Mathew and Katz (2009), †Friedrich and Palmer (2014).

vided by Loaiciga et al. (2014). The values of the grammatical dependents' features are the WordNet (Miller, 1995) lexical filename of the dependent's lemma, or, if not available, the dependent's part-of-speech tag. Quantificational adverbs are temporal modifiers such as *always*, *occasionally* or *weekly*.[4] Specific temporal adverbs are, according to a heuristic proposed by Mathew (2009), phrase children marked with the part-of-speech tag TMP and whose child is a prepositional phrase. Noun phrases with one of the determiners *the, this, that, these, those, each, every, all*, as well as possessives, pronouns, proper names and quantified phrases are definite. NPs with determiners *a, an, many, most, some*, and cases of modifying adjectives without determiners (e.g., *few*) or cardinal numbers (part-of-speech tag CD) are indefinite. Mathew (2009) describes their features in detail.

5.2 TYPE-BASED features

This feature set consists of the verb-type based linguistic indicator features of Siegel and McKeown (2000). The computation of these features is based on a large parsed, but otherwise unannotated background corpus. For each verb type (i.e., lemma), these features count how often the verb occurs with each of the linguistic indicators as listed in Table 3. Except for the frequency feature, these values are normalized by the number of occurrences of the verb type. For example, if the verb type *win* occurs 1000 times in the parsed background corpus, of which 100 times with perfect aspect, the value of the linguistic indicator feature `perfect` is 0.1 for the verb type *win*. For any instance whose verb's lemma is *win*, 0.1 will be the value of the feature `perfect`, in other words, all instances of the same verb type receive the same TYPE-BASED feature values. Linguistic indicator features have recently been applied successfully on the related task of classifying the lexical aspectual class of verbs by Friedrich and Palmer (2014), who extract the linguistic indicators from an automatically parsed version of the AFE and XIE parts of Gigaword. We use their database of linguistic indicator values.[5]

Feature	Example
frequency	-
present	*says*
past	*said*
future	*will say*
perfect	*had won*
progressive	*is winning*
negated	*not/never*
particle	*up/in/...*
no subject	-

Feature	Example
continuous adverb	*continually endlessly*
evaluation adverb	*better horribly*
manner adverb	*furiously patiently*
temporal adverb	*again finally*
in-PP	*in an hour*
for-PP	*for an hour*

Table 3: TYPE-BASED feature set (**linguistic indicators**) and examples for lexical items associated with each indicator, following Siegel and McKeown (2000).

5.3 Algorithm

In order to investigate in which circumstances the task of predicting a clause's label (**habitual**, **episodic** or **static**) can be addressed jointly, or whether a pipelined approach is better, we apply the following methods. Our JOINT model learns the decision boundaries for the three classes jointly, i.e., as a three-way classification task. In addition, we present a CASCADED model, which uses two models learned for the two different subtasks: (a) identifying static clauses and (b) distinguishing episodic and habitual clauses.

First, we train a model to distinguish the **static** class from the other two. In this learning step, we simply map all the clauses labeled as **episodic** and

[4] The complete list of quantificational adverbs used is given by Mathew (2009), page 36.

[5] `www.coli.uni-saarland.de/projects/sitent`

habitual to the class **non-static** and learn the decision boundary between the two classes **static** and **non-static**. Second, we train a model to distinguish the **episodic** from the **habitual** class. This model is trained on the subset of examples labeled with either of these two classes.

In the CASCADED model, first, the **static** vs. **non-static** model is applied. The CASCADED model labels all instances automatically labeled as **static** in this first step, and then applies the second model (**episodic** vs. **habitual**) on all remaining instances.

We train Random Forest classifiers (Breiman, 2001) using Weka (Hall et al., 2009) for each step and also for the JOINT model. Besides providing a robust performance, Random Forest classifiers can easily deal with both categorical and numeric features. This is relevant as our CONTEXT-BASED features are categorical while the TYPE-BASED features are numeric. In our experiments, we will compare the impact of the different feature sets on each subtask and on the JOINT model.

5.4 Baseline: Mathew and Katz (2009)

As a baseline, we also report results for the subset of our CONTEXT-BASED features used by Mathew and Katz (2009) and call this subset MK. Mathew and Katz (2009) find a J48 decision tree and a Naive Bayes classifier to work best. We replicate their results for the decision tree in Section 6.2.

6 Experiments and discussion

This section describes our experiments. First, we reproduce the experiments of Mathew and Katz (2009), who use manually created syntactic parses, in a purely automatic setting.

The data set and experiments of Mathew and Katz (2009) focus on the episodic-habitual distinction using a set of sentences selected for a small set of verbs, and their feature design focuses on syntactic properties of the clauses found in this annotated data set. In the further experiments, we turn to the Wikipedia data, which contains annotations for full texts. We expect the Wikipedia data to cover the range of habitual and episodic expressions more fully, and in addition, allows for studying the task of separating static sentences from the other two classes. As we will show, this latter task profits from including features relevant to the stative-dynamic distinction on the lexical level.

We first present experimental results for the two subtasks (described in Section 5.3). Our CASCADED model first identifies static clauses, and then classifies the remaining clauses as episodic or habitual. For reasons of readability, we first report on our experiments for the episodic-habitual distinction using both the M&K and Wikipedia data sets. Using the Wikipedia data, we then report on the results for the static vs. non-static distinction. Finally, we turn to the full task of the three-ways distinction.

6.1 Experimental setting

We report results for 10-fold cross validation (CV) with two different settings: In the RANDOM CV setting, we randomly distribute the instances over the folds, putting all instances of one document into the same fold. In the UNSEEN VERBS CV setting, we simulate the case of not having labeled training data for a particular verb type by putting all instances of one verb type into the same fold.

We compute the information retrieval statistics of precision (P), recall (R) and F1-measure per class, where F1 is the harmonic mean of P and R, $F1 = \frac{2*P*R}{P+R}$. Macro-average P is computed as the (unweighted) average of the P scores of the classes, and macro-average R is computed likewise. Macro-average F1 is the harmonic mean of macro-average P and macro-average R. We use McNemar's test with $p < 0.01$ to compute statistical signficance of differences in accuracies. In our tables, we indicate that two results differ significantly by marking them with the same symbols (we only show this when scores are close).

6.2 M&K data: episodic vs. habitual

We use Weka's 10-fold stratified cross validation and a J48 decision tree in the experiments reported in this section in order to replicate their experimental setting. Results are shown in Table 4. For the sake of completeness, we also show the results as presented in the original paper. F1-scores are computed from P and R as reported in the original paper. Note that their experiments are performed on a different subset of the data and so these numbers are not directly comparable to ours, but our subset has a very similar class distribution (see Section 4.1). Our accuracies based on automatic parses rather than gold standard parses are about 3% lower when using the original feature set (MK). We conclude that our results are in the expected range. Also, we do not find any significant improvements on this data set when using any

other feature sets or combinations thereof (the table shows the results for our CONTEXT-based feature set); the M&K feature set designed for this corpus captures its variation well.

We have used a J48 decision tree in this section for comparability with previous work. In all following sections, we present results using Random Forest classifiers as described in Section 5.3.

	F1-score			Acc.
System	epis.	habit.	*macro*	
majority class*	84.5	0.0	42.2	73.1
MK*	91.1	70.5	80.8	86.1
MK	89.6	63.5	76.5	83.8
CONTEXT	90.0	64.7	77.3	84.4

Table 4: Results for **episodic** vs. **habitual**, J48 decision tree, data from Mathew and Katz (2009). *Numbers from original paper.

6.3 Wikipedia: episodic vs. habitual

We study the classification task of distinguishing **episodic** and **habitual** sentences using the subset of the Wikipedia data having one of these two labels (4171 instances). This task parallels the experiment of Mathew and Katz (2009) described above. We conduct two experiments, once using the RANDOM CV setting and once using the UNSEEN VERBS setting. Table 5 shows the results. The distribution of instances is nearly 50:50 in the gold standard (see Section 4, Table 1), and the majority classes in the respective training folds differ (this is the reason for the different baseline scores). For reasons of space we do not show the other scores here; macro-average F1-scores have (almost) the same values as accuracy, the F1-scores for episodic and habitual are similar to each other in each case.

Our findings are as follows: TYPE-BASED features outperform the majority class baseline,

Features	RANDOM CV	UNSEEN VERBS
majority class	42.1	46.3
lemma	65.4	46.3
TYPE	68.1	53.9
MK	82.3	‡81.4
CONTEXT	*†82.8	‡**83.8**
+ lemma	*84.3	
CONTEXT + TYPE	†**85.1**	83.1
+ lemma	84.0	

Table 5: **Wikipedia**: **Accuracy** of **episodic** vs **habitual**, 4171 instances, 10-fold cross validation, *†‡differences statistically significant.

	RANDOM CV		UNSEEN VERBS	
Features	**F1**	**Acc.**	**F1**	**Acc.**
majority class	37.4	59.7	37.4	59.7‡
MK	67.5	*69.5	59.2	62.7‡
CONTEXT	70.3	*71.7	62.8	64.9‡
+ lemma	81.9	†82.8		
TYPE	78.8	79.3	72.2	73.2‡
CONTEXT + TYPE	83.6	†84.1	**78.4**	**79.2**‡
+ lemma	**83.8**	**84.4**		

Table 6: **Wikipedia**: **static** vs **non-static**. All 10355 instances, 10-fold cross validation.*†‡ differences statistically significant.

which means that some verbs have a preference for being used as either episodic or habitual. The CONTEXT-BASED features work remarkably well. If training data of the same verb type is available, adding the TYPE-BASED features or the lemma to the CONTEXT-BASED features results in improvements; this is not the case in the UNSEEN VERBS setting. The latter setting shows that the additional contextual features (compared to the MK subset) are important: our corpus indeed covers a broader range of phenomena than the M&K data set.

6.4 Wikipedia: static vs. non-static

We evaluate the task of classifying **static** versus **non-static** clauses using all 10355 instances of the Wikipedia data set. Any instance labeled **episodic** or **habitual** receives the label **non-static** both in training and testing. Results of this task are shown in Table 6. For this subtask, the CONTEXT-BASED features are less informative than the TYPE-BASED features. Again, using lemma information approximates the use of type-based information, but this is not an option in the UNSEEN VERBS setting. A combination of the CONTEXT-BASED and TYPE-BASED features achieves the best results. Friedrich and Palmer (2014) find that TYPE-BASED features generalize well across verb types when predicting the aspectual class of verbs in context, the same is true here. They achieve small improvements by adding context-based features. Predicting the lexical aspectual class of the clause's main verb is only part of our classification task, the **static** class includes not only lexically stative clauses but also clauses with lexically dynamic verbs that are stativized, e.g., modals, negation or perfect tense. Hence, as expected, in our task, adding the CONTEXT-BASED features results in a considerable performance improvement (5-7% absolute in accuracy).

	RANDOM CROSS VALIDATION					UNSEEN VERB TYPES EXPERIMENT				
	F1-score				Acc.	F1-score				Acc.
Features	stat.	epis.	habit.	*macro*		stat.	epis.	habit.	*macro*	
majority class baseline	74.8	0	0	24.9	59.7	74.8	0.0	0.0	24.9	‡59.7
JOINT: MK	76.6	65.4	26.1	57.5	*67.0	76.3	41.7	0.8	49.0	‡63.8
JOINT: CONTEXT	77.5	65.8	36.4	60.5	*68.4	74.7	57.1	12.0	51.7	⋆63.9
+ lemma	85.5	75.0	51.6	71.8	†78.0					
JOINT: TYPE	81.9	52.7	49.7	61.5	69.9	74.9	4.2	2.8	40.7	⋆60.0
JOINT: CONTEXT + TYPE	86.1	75.8	58.8	73.8	†79.0	81.2	69.5	31.3	63.6	**72.1
+ lemma	86.8	75.0	59.9	74.2	79.6					
CASCADED	**86.9**	**76.1**	**62.2**	**75.1**	**79.9**	**82.6**	**72.0**	**50.2**	**68.4**	****74.3**

Table 7: **Wikipedia**: **static** vs. **episodic** vs. **habitual**. 10355 instances, 10-fold cross validation. The CASCADED model uses the best models from Table 6 and Table 5. *† ‡ ⋆** differences statistically significant.

It is worth noting that even for verbs not seen in the training data, high accuracies and F1-scores of almost 80% can be reached.

6.5 Wikipedia: combined task

In this section, we describe our experiments for the three-way classification task of **static**, **episodic** and **habitual** clauses, as in a realistic classification setting, a clause may belong to either of these *three* classes. We investigate whether a pipelined CASCADED approach is better, or whether the JOINT model profits from learning the decision boundaries between all three classes jointly. The results for this task are presented in Table 7. Both the CONTEXT-BASED and the TYPE-BASED features when used alone improve over the majority class baseline by about 10% in accuracy in the RANDOM CV setting, and only by about 4% in the UNSEEN VERBS setting. In the latter setting, all feature sets when used alone are ineffective for identifying habituals. This indicates that the CONTEXT-BASED features only 'pick up' on some type-based information in the RANDOM CV case. The best models for this JOINT classification task use both the CONTEXT-BASED and the TYPE-BASED feature sets: F1-scores and accuracy increase remarkably. Again, in the RANDOM CV setting, using the lemma results in a large performance gain, though using the TYPE-BASED features is beneficial, and, in the UNSEEN VERBS setting, essential.

We apply the CASCADED model as described in Section 5.3, training and testing the models for the subtasks in each fold. In the RANDOM CV setting, the accuracy of the CASCADED approach is not significantly better than the one of the JOINT approach, though F1-scores for the less frequent **episodic** and **habitual** classes both increase. In the UNSEEN VERBS setting, however, the difference is remarkable: macro-average F1-score increases by almost 5% (absolute) and accuracy increases by 2.2%. Most notably, the F1-score for the habitual class increases from 0.31 to 0.50 (due to an increase in recall). To conclude, the CASCADED approach is favorable as it works more robustly both for verb types seen or unseen in the training data.

6.6 Feature ablation

In the above sections, we have compared the two major feature groups of CONTEXT-BASED and TYPE-BASED features. In addition, we ablate each single feature from the best results for each experiment. For all classification tasks, we found features reflecting tense and grammatical aspect to be most important, both for the CONTEXT-BASED and TYPE-BASED features. In general, we observe that no single feature has a big impact on the results, accuracy drops only by at most 1-2%. This shows that our feature set is quite robust and some of the features (e.g., part-of-speech tag of the verb and tense) reflect partially redundant information. However, using only the best features results in a significant performance drop by several percentage points in the various settings, which means that though single features may not have a large impact, overall, the models for this classification task profit from including many diverse features.

For the **episodic-habitual** distinction in the UNSEEN VERBS setting, the definiteness of the object was an important CONTEXT-BASED feature. In the **static vs. non-static** task, the subject also plays an important role, as well as the TYPE-BASED feature for *continuous adverbs*. In the UNSEEN VERBS setting, many TYPE-BASED features

are important, including those indicating how often the verb type occurs with *adverbs of manner*, *negation* and *in-PPs* in the background corpus. For the **combined** three-way task, we found the main verb's lemma and the direct object to have most impact. Of the TYPE-BASED features, the *for-PP*, *present* and *temporal adverbial* were most important. In the UNSEEN VERBS setting, many linguistic indicator features (among others *past*, *progressive*, *negation*) play a greater role, as well as information about the object, subject and tense.

7 Conclusion

In this paper, we have presented an approach for classifying the aspect of a clause as **habitual**, **episodic** or **static**. Clearly, when exhaustively classifying all clauses of a text, the **static** class cannot be ignored; we have shown that we can separate these instances from episodic and habitual instances, most of which are lexically dynamic, with high accuracy. Our model for distinguishing episodic and habitual sentences integrates a wide range of contextual information and outperforms previous work. Previous work has only addressed the classification of lexical aspectual class and the automatic distinction of episodic and habitual sentences. Our work is the first bringing together two strands of work relevant to classifying clausal aspect, and we have shown that sources of information relevant to these two underlying aspectual distinctions are relevant for our three-way classification task.

We have shown that for distinguishing static sentences from the other two, TYPE-BASED *and* CONTEXT-BASED information is needed; for distinguishing episodic and habitual clauses, CONTEXT-BASED features are most important. Our experimental results show that the three-way classification task is most effectively approached by combining both contextual and verb-type based information. Especially for verbs unseen in the training data, we found the CASCADED approach to work better. It is hard for the JOINT approach to identify habitual clauses; while in the CASCADED approach, performance for both steps is high and adds up.

We found the overall performance of this task to be about 80% accuracy, and 75% macro-average F1-score. These scores suggest that this method may be usable as a preprocessing step for further temporal processing.

8 Future work

Our models do not yet take discourse information into account. Consider example (14) by Mathew and Katz (2009): The second sentence is habitual, but the only indicator for this is sentence-external.

(14) John rarely ate fruit. He just ate oranges. (**habitual**)

In some preliminary experiments, we tried to leverage the discourse context of a clause for its classification by means of incorporating the gold standard label of the previous clause as a feature. This did not result in significant performance improvements. However, further experiments trying to incorporate discourse information are due, and, due to our new corpus of fully annotated texts, now possible.

Another related research direction is the classification of the different types of static clauses, e.g., the different senses of modality (Ruppenhofer and Rehbein, 2012). As mentioned before, a finer classification of the temporal structure of clauses is needed, among others identifying the lexical aspectual class as well as *viewpoint aspect* as *perfective* vs. *imperfective* (Smith, 1997).

Finally, the next steps in this line of research are to integrate the aspectual information attributed to clauses by our model into models of temporal discourse structure, which in turn are useful for information extraction and text understanding tasks in general. Costa and Branco (2012) are the first to show that aspectual information is relevant here; we hope to show in the future that temporal processing profits from integrating more fine-grained aspectual information.

Acknowledgments

We thank the anonymous reviewers, Alexis Palmer, Alessandra Zarcone and Andrea Horbach for their helpful comments related to this work, Graham Katz for providing us with their data set, and our annotators Christine Bocionek, Kleio-Isidora Mavridou and Melissa Peate Sørensen. This research was supported in part by the Cluster of Excellence "Multimodal Computing and Interaction" of the German Excellence Initiative (DFG), and the first author is supported by an IBM PhD Fellowship.

References

Leo Breiman. 2001. Random forests. *Machine learning*, 45(1):5–32.

Greg Carlson, 2005. *The Encyclopedia of Language and Linguistics*, chapter Generics, Habituals and Iteratives. Elsevier.

Francisco Costa and António Branco. 2012. Aspectual type and temporal relation classification. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 266–275. Association for Computational Linguistics.

Östen Dahl. 1985. *Tense and aspect systems*. Oxford, Blackwell.

Östen Dahl. 1995. The marking of the generic/episodic distinction in tense-aspect systems. *The generic book*, pages 412–425.

Annemarie Friedrich and Alexis Palmer. 2014. Automatic prediction of aspectual class of verbs in context. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL). Baltimore, USA*.

Andrew Garrett. 1998. On the origin of auxiliary do. *English Language and Linguistics*, 2(02):283–330.

Valentine Hacquard. 2009. On the interaction of aspect and modal auxiliaries. *Linguistics and Philosophy*, 32(3):279–315.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. 2009. The WEKA data mining software: an update. *ACM SIGKDD explorations newsletter*, 11(1):10–18.

Graham Katz. 2003. On the stativity of the English perfect. *Perfect explorations*, pages 205–234.

Dan Klein and Christopher D Manning. 2002. Fast exact inference with a factored model for natural language parsing. In *Advances in neural information processing systems*, pages 3–10.

Manfred Krifka, Carlson Gregory N Pelletier, Francis Jeffry, Alice ter Meulen, Gennaro Chierchia, and Godehard Link. 1995. Genericity: An introduction. *The generic book*, pages 1–124.

J Richard Landis and Gary G Koch. 1977. The measurement of observer agreement for categorical data. *biometrics*, pages 159–174.

Hector Llorens, Nathanael Chambers, Naushad UzZaman, Nasrin Mostafazadeh, James Allen, and James Pustejovsky. 2015. SemEval-2015 Task 5: QA TEMPEVAL - Evaluating Temporal Information Understanding with Question Answering. In *9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 46–54, Denver, Colorado.

Sharid Loaiciga, Thomas Meyer, and Andrei Popescu-Belis. 2014. English-French Verb Phrase Alignment in Europarl. In *Proceedings of LREC 2014*.

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

Thomas A. Mathew and Graham E. Katz. 2009. Supervised categorization for habitual versus episodic sentences. In *Sixth Midwest Computational Linguistics Colloquium*, Bloomington, Indiana. Indiana University.

Thomas A. Mathew. 2009. Supervised categorization for habitual versus episodic sentences. Master's thesis, Faculty of the Graduate School of Arts and Sciences of Georgetown University.

Paula Menéndez-Benito. 2012. On dispositional sentences. *Genericity*, 43:276.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Josef Ruppenhofer and Ines Rehbein. 2012. Yes we can!? annotating the senses of english modal verbs. In *Proceedings of the 8th International Conference on Language Resources and Evaluation (LREC)*, pages 24–26.

Eric V Siegel and Kathleen R McKeown. 2000. Learning methods to combine linguistic indicators: Improving aspectual classification and revealing linguistic insights. *Computational Linguistics*, 26(4):595–628.

Carlota S. Smith. 1997. *The Parameter of Aspect*, volume 43 of *Studies in Linguistics and Philosophy*. Springer.

Ian Soboroff and Donna Harman. 2005. Novelty detection: The trec experience. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 105–112, Vancouver, British Columbia, Canada, October. Association for Computational Linguistics.

Radu Soricut and Daniel Marcu. 2003. Sentence level discourse parsing using syntactic and lexical information. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 149–156. Association for Computational Linguistics.

Benjamin D Van Durme. 2010. *Extracting implicit knowledge from text*. Ph.D. thesis, University of Rochester.

Zeno Vendler, 1957. *Linguistics in Philosophy*, chapter Verbs and Times, pages 97–121. Cornell University Press, Ithaca, New York.

Jennifer Williams and Graham Katz. 2012. Extracting and modeling durations for habits and events from twitter. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers - Volume 2*, ACL '12, pages 223–227, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jennifer Williams. 2012. Extracting fine-grained durations for verbs from twitter. In *Proceedings of ACL 2012 Student Research Workshop*, pages 49–54. Association for Computational Linguistics.

Nianwen Xue and Yuchen Zhang. 2014. Buy one get one free: Distant annotation of chinese tense, event type, and modality. *Proceedings of LREC-2014, Reykjavik, Iceland*.

Alessandra Zarcone and Alessandro Lenci. 2008. Computational models for event type classification in context. In *LREC*.

Distributed Representations for Unsupervised Semantic Role Labeling

Kristian Woodsend and **Mirella Lapata**
Institute for Language, Cognition and Computation
School of Informatics, University of Edinburgh
10 Crichton Street, Edinburgh EH8 9AB
k.woodsend@ed.ac.uk, mlap@inf.ed.ac.uk

Abstract

We present a new approach for unsupervised semantic role labeling that leverages distributed representations. We induce embeddings to represent a predicate, its arguments and their complex interdependence. Argument embeddings are learned from surrounding contexts involving the predicate and neighboring arguments, while predicate embeddings are learned from argument contexts. The induced representations are clustered into roles using a linear programming formulation of hierarchical clustering, where we can model task-specific knowledge. Experiments show improved performance over previous unsupervised semantic role labeling approaches and other distributed word representation models.

1 Introduction

In recent years, an increasing body of work has been devoted to learning distributed word representations and their successful usage in numerous tasks and real-world applications. Examples include language modeling (Collobert et al., 2011; Mikolov et al., 2013c; Mnih and Kavukcuoglu, 2013), paraphrase detection (Socher et al., 2011a), sentiment analysis (Socher et al., 2011b; Kalchbrenner et al., 2014), and most notably machine translation (Kalchbrenner and Blunsom, 2013; Cho et al., 2014; Auli et al., 2013). Distributed word representations (also known as word embeddings) are trained by predicting the contexts in which the words or phrases occur.

In this paper, we present a new approach for *unsupervised* semantic role labeling that leverages distributed representations. The goal of semantic role labeling is to discover the relations that hold between a predicate and its arguments in a given input sentence (e.g., "who" did "what" to "whom", "when", "where", and "how").

1. [The burglar]$_{\text{A0}}$ [broke]$_{\text{V}}$ [the window]$_{\text{A1}}$.
2. [The window]$_{\text{A1}}$ [broke]$_{\text{V}}$.

In sentence (1), A0 represents the *Agent* of the breaking event, A1 represents the *Patient* (i.e., the physical object affected by the breaking event) and V determines the boundaries of the predicate. The semantic roles in the example are labeled in the style of PropBank (Palmer et al., 2005), a broad-coverage human-annotated corpus of semantic roles and their syntactic realizations. In the unsupervised case, the model must induce such labels from data without access to a predefined set of semantic roles.

Role induction is commonly treated as a clustering problem (Titov and Klementiev, 2012; Lang and Lapata, 2014). The input to the model are instances of arguments (e.g., *window*, *the burglar* in sentence (1)) and the output is a grouping of these instances into clusters such that each cluster contains arguments corresponding to a specific semantic role and each role corresponds to exactly one cluster. In other words, the syntactic representations of verbal predicates, and argument positions are observable, whereas the associated semantic roles are latent and need to be inferred.

The task is challenging due to its unsupervised nature — it is difficult to define a learning objective function whose optimization will yield an accurate model — but also because each predicate can allow several alternate mappings or linkings between its semantic roles and their syntactic realization. Despite occupying different syntactic positions (subject in sentence (1) and object in sentence (2)), the noun phrase *the window* expresses the same role in both sentences. To learn such linkings, previous work has made use of syntactic and semantic features (e.g., whether two arguments are in the same position in the parse tree,

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 2482–2491,
Lisbon, Portugal, 17-21 September 2015.

whether they have the same POS-tags, whether they are lexically similar). These features are typically defined on argument instances, without taking the predicate into account, and do not interact but instead are sequentially applied.

In this work we propose to learn these features and their complex interactions (e.g., selectional restrictions) automatically from data. Specifically, we induce embeddings to represent a predicate *and* its arguments. Argument embeddings are learned from surrounding contexts involving the predicate and neighboring arguments. Analogously, predicate embeddings are learned from contexts representing their arguments. Our model learns a rich feature space which can serve as input to any clustering algorithm. We use a linear programming formulation of hierarchical clustering which is advantageous for two reasons. Firstly, expressing clustering as a global optimization problem with an explicit objective function can potentially yield higher quality output compared to greedy algorithms (such as agglomerative clustering). Secondly, through the use of constraints, we can model task-specific knowledge (e.g., semantic roles are unique within a frame). Experimental results show improved performance over both previous unsupervised semantic role labeling approaches and other distributed word representation models.

2 Related Work

Our model is inspired by recent work in learning distributed representations of words (Bengio et al., 2006; Mnih and Hinton, 2008; Collobert et al., 2011; Turian et al., 2010; Mikolov et al., 2013a). In this framework, a neural network is used to predict a word taking into account its context. Words are represented by vectors which are concatenated or averaged in order to form a representation of the context. We induce vector representations to represent each predicate and its argument. As a learning objective, vectors are required to contribute to a prediction task about the target argument in the sentence, given the predicate and a small window of surrounding arguments. Similarly, predicate vectors are learned from the contexts of preceding arguments, and are required to contribute to the prediction of upcoming arguments. Our vectors encode the semantics of arguments, predicates, and their interdependence.

Approaches to unsupervised semantic role labeling follow two main modeling paradigms. Under the the first variant, semantic roles are modeled as latent variables in a (directed) graphical model that relates a verb, its semantic roles, and their possible syntactic realizations (Grenager and Manning, 2006; Lang and Lapata, 2010; Garg and Henderson, 2012). Role induction here corresponds to inferring the state of the latent variables representing the semantic roles of arguments. The second approach is similarity-driven and based on clustering. For instance, Lang and Lapata (2014) induce semantic roles via graph partitioning: each vertex in a graph corresponds to an argument instance of a predicate and edges represent features expressing syntactic or semantic similarity. The graph partitioning problem is solved using task-specific adaptations of label propagation and agglomerative clustering. Titov and Klementiev (2012) propose a Bayesian clustering algorithm based on the Chinese Restaurant Process. Their model encourages similar verbs to have similar linking preferences using a distance-dependent Chinese Restaurant Process prior.

More recently, Titov and Khoddam (2015) propose a reconstruction-error minimization framerwork for unsupervised semantic role induction. Their model consists of two componenets: the encoder (implemented as a log-linear model) predicts roles given syntactic and lexical features, whereas the reconstruction component (implemented as a probabilistic tensor factorization model) recovers argument fillers based on the role predictions, the predicate and other arguments. The two components are estimated jointly to minimize errors in argument reconstruction.

Our work follows the similarity-driven modeling paradigm. Rather than engineering relevant features, we learn them using a neural network and a task-appropriate training objective. We are thus able to model complex interactions between arguments and their predicates without making simplifying assumptions (e.g., that arguments are conditionally independent of each other given the predicate). Our embeddings are largely independent of the clustering algorithm used to induce the semantic roles. We advocate the use of linear programming, which supports the incorporation of linguistic and structural constraints during cluster formation. ILP techniques have been previously applied to several *supervised* NLP tasks, including semantic role labeling (Punyakanok et al., 2008), how-

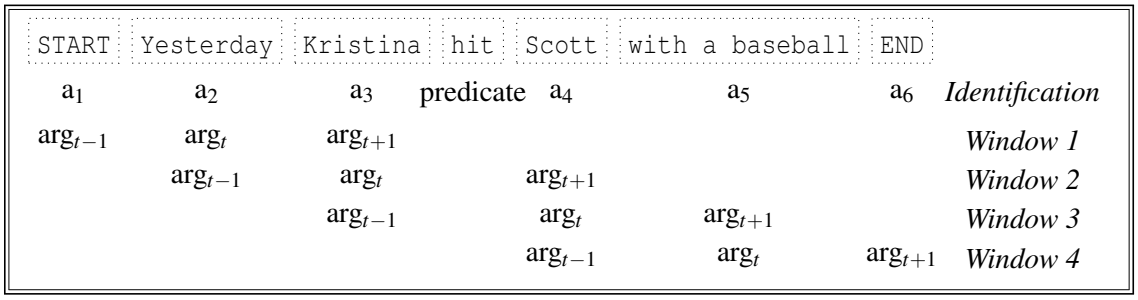


Figure 1: Symmetric context window from the list of arguments

ever their application to *unsupervised* role induction is novel to our knowledge.

3 Model

Unsupervised role induction is commonly modeled after supervised semantic role labeling (Màrquez et al., 2008) and follows a two-stage approach. Given a sentence and a designated verb, the goal is to identify the arguments of the verbal predicate (argument identification) and label them with semantic roles (role induction). The model is first given a syntactically analyzed sentence (e.g., in the form of a dependency parse) with the aim of determining all constitutents that fill a semantic role. Argument identification is performed heuristically using a small number of rules which take into account syntactic relations encountered when traversing the dependency tree from predicate to argument (Lang and Lapata, 2014; Titov and Klementiev, 2012). An alternative which we follow here is to use a supervised classifier trained on a small amount of data using non-lexicalized features.

As mentioned earlier, we treat role induction as a type-level clustering problem: argument instances are assigned to clusters such that these represent semantic roles. We induce a separate set of clusters for each verb, and each cluster thus represents a verb-specific role. Clustering algorithms commonly take a matrix of pairwise similarity scores between instances as input and produce a set of output clusters, often satisfying some optimality criterion. In our case, instances are type-level arguments represented by embeddings whose similarity is quantified using a distance measure such as cosine (see Section 3.1) and clusters are formed using a linear programming formulation of hierarchical clustering (see Section 3.2).

3.1 Predicate and Argument Embeddings

Our approach for learning predicate and argument vectors is inspired by recent methods aimed at learning high-quality vector representations of words from large amounts of unstructured text data (Mikolov et al., 2013a). In this framework, vectors of the surrounding words within a fixed-sized window (the *context*) are summed into a single vector v_c, which is useful in predicting the output vector v_0 representing the current or *target* word. Longer-range context information can also be captured (Le and Mikolov, 2014), specifically words within the current paragraph but outside of the target word context window.

In contrast to previous word-based approaches, our model induces vector representations for each predicate and its semantic arguments. As a learning objective, vectors are required to contribute to a prediction task about the target argument in the sentence, given the predicate and a small window of surrounding arguments. So despite the fact that the argument vectors and weightings are initialized randomly, they can eventually capture semantics as an indirect result of the prediction task. Similarly, predicate vectors are learned from the many contexts sampled from sentences involving that predicate, and are required to contribute to the prediction task of the next argument. One way to consider the role of the predicate token is as another argument. It acts as a memory (similar to the paragraph memory of Le and Mikolov, 2014) that remembers what is missing from the current context, and so captures something of the core nature of the predicate.

Figure 1 illustrates our approach for building the context, for the example sentence *Yesterday, Kristina hit Scott with a baseball.* As a preprocessing step, (verbal) predicates and arguments are identified based on a dependency parse, to give a full list of arguments for the sentence. Boxes show

the span of each argument. In our model, contexts are symmetric and of fixed length ($c = 1$ in the Figure), sampled from a sliding window over the argument list. To enable the first and last arguments within the sentence to be predicted from the context, we augment the argument list with *START* and *END* arguments. Meanwhile, the predicate is associated with all contexts generated from the sliding window approach.

More formally, given a training set comprising a predicate b and a sequence of its semantic arguments $a_1, a_2, a_3, \ldots, a_T$, the objective of the model is to maximize the average log probability:

$$\frac{1}{T}\sum_{t=1}^{T} \log p\left(a_t | b, a_{t+j}, -c \leq j \leq c, j \neq 0\right) \quad (1)$$

where c is the size of the training context around the center argument a_t. We define probability using the softmax function:

$$p\left(a_t | b, a_{\text{context}}\right) \propto \exp\left(v_c^T v_0\right), \quad (2)$$

where v_0 is the target argument vector and v_c the context vector formed from predicate and context arguments vectors. Vectors are trained using stochastic gradient descent where the gradient is obtained via back-propagation. After the training converges, predicates and arguments with similar meaning are mapped to a similar position in the vector space.

Every predicate is mapped to a unique vector v_{pred}, with the vocabulary of vectors shared across the data set. For the arguments, we generate feature vectors f_{-1} and f_{+1} from syntactic information (dependency relations and POS-tags), concatenated with a distributional vector to represent the head word token in each argument. The representation vectors v_{-1} and v_{+1} are calculated from the feature vectors using $v_j = W_{\text{context}} f_j$, where the matrix W_{context} is also updated as part of the learning process. W_{context} is common for all arguments. In a similar manner, the representation vector v_0 for the target argument is calculated using $v_0 = W_{\text{argument}} f_0$. The predicate and argument vectors are concatenated to predict the middle argument in a context. Other ways of dividing the argument window between context and predicted argument, and of combining context vectors, are possible. As the full list of arguments in a sentence is known, we use a symmetric window. The advantage of concatenating the vectors is that information on the sequence of arguments is preserved. An illustration of our model is given in Figure 2.

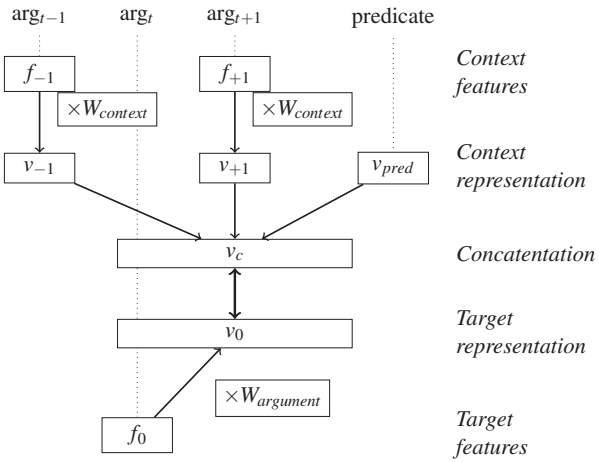


Figure 2: Distributional model for learning representations of predicates and semantic arguments.

Through a context window of arguments rather than neighboring tokens, our model captures a semantic representation of each verbal predicate. Furthermore, the arguments themselves are positioned in vector space as a result of the selectional preferences of the predicates. In the next section, we use the induced semantic space to cluster arguments into semantic roles.

3.2 Argument Clustering

Hierarchical clustering is a method of clustering which seeks to build a hierarchy of clusters, often presented in a dendrogram. In such a representation, all possible pairs of clusters are merged at some level. It is typically implemented as a greedy heuristic algorithm with no explicit objective function. Instead, it requires a measure of dissimilarity between sets of observations, typically through a measure of distance between pairs of observations. An example is the agglomerative clustering technique used in Lang and Lapata (2014). Their algorithm starts from seed clusters based on shared syntactic information, and then repeatedly merges pairs of clusters "bottom up" to form a hierarchy.

It is possible to formalize hierarchical clustering as an integer linear programming (ILP) problem with the dendrogram properties enforced as linear constraints (Gilpin et al., 2013). Although exact solvers exists for ILP, their performance is highly dependent on the number of variables involved, and we found it necessary to develop a linear programming (LP) relaxation to provide approximate solutions faster. Dynamic programming is an al-

ternative approximation technique that could be explored; it has recently been used successfully in the context of supervised semantic role labeling (Täckström et al., 2015).

But first, we consider the exact formalization of agglomerative clustering as an ILP. In order to generate a legal dendrogram, it is necessary for the model to enforce the following partition properties:

Reflexivity A seed cluster is always in the same merged cluster as itself.

Symmetry If seed cluster a is merged into the same cluster as seed cluster b, then b is also in the same cluster as a.

Transitivity If a and b are merged at a certain level, and b and c are also merged at the same level, then a is in the same cluster as c at that level.

To model hierarchical clustering as an ILP problem, we consider all pairs of clusters a and b, and introduce variables $\mathcal{M}_{ab}$ to represent the merge level between clusters a and b. Reflexivity is enforced by the constraint:

$$\mathcal{M}_{aa} = 0, \tag{3}$$

Meanwhile the symmetry requirement is captured by the constraint:

$$\mathcal{M}_{ab} = \mathcal{M}_{ba}. \tag{4}$$

The transitivity requirement and the objective to find the hierarchy that minimizes pair-wise distances are modeled in the objective of the ILP using auxiliary variables O_{abc} that represent the merge order of pairs (a,b) and (a,c), and coefficients w_{abc} that are set equal to the difference between distance metrics D between those pairs:

$$\arg\max_{\mathcal{M},O} \sum_{a,b,c \in \text{Instances}} w_{abc} O_{abc}$$

subject to:

$$\begin{aligned} &\mathcal{M}_{ab} \text{ is a merge function} \\ &O_{abc} = \begin{cases} 1 & \text{if } \mathcal{M}_{ab} < \mathcal{M}_{ac} \\ 0 & \text{otherwise} \end{cases} \\ &w_{abc} = D(a,c) - D(a,b). \end{aligned}$$

Although exact solutions can be found using ILP solvers, for the problems we consider there are typically over 100 seed clusters. This generates in the order of 10^6 transitivity constraints, and it is this in particular that results in combinatorial complexity from off-the-shelf ILP solvers.

An LP relaxation provides approximate solutions faster. A maximum merge level L is first defined as a parameter, although as this is not an integer problem and fractional levels are possible, this does not represent the number of levels. Auxiliary variables $Z_{ab \geq ac}$ capture the merge hierarchy, and T_{abc} rewards transitivity by a factor α:

$$\arg\max_{\mathcal{M},O,Z} \sum_{a,b,c \in \text{Instances}} w_{abc} O_{abc} + \alpha T_{abc}$$

subject to:

$$\begin{aligned} 0 \leq T \leq 1 \\ 0 \leq O \leq 1 \\ 0 \leq Z \leq 1 \\ 0 \leq \mathcal{M} \leq L \\ -L \leq \mathcal{M}_{ac} - \mathcal{M}_{ab} - (L+1) O_{abc} \leq 0 \\ -L \leq \mathcal{M}_{ab} - \mathcal{M}_{ac} - (L+1) Z_{ab \geq ac} + 1 \leq 0 \\ -L \leq \mathcal{M}_{bc} - \mathcal{M}_{ac} - (L+1) Z_{bc \geq ac} + 1 \leq 0 \\ Z_{ab \geq ac} + Z_{bc \geq ac} \geq T_{abc} \end{aligned} \tag{5}$$

To capture the linguistic principles involved in semantic role labeling (Lang and Lapata, 2014), our formulation includes additional constraints. These are expressed explicitly through the construction of the linear programme:

Role Uniqueness Semantic roles are unique within a particular frame. This principle is captured by constraining the merge level of two seed clusters a and b to be at the top level L of the hierarchy, where a and b are roles that occur within the same frame, with the constraint:

$$\mathcal{M}_{ab} = L \qquad \forall (a,b) \text{ in frame.} \tag{6}$$

Syntactic Position Arguments occurring in a specific syntactic position within a specific linking all bear the same semantic role. This is handled by construction of the problem, where all arguments of a particular predicate occurring in a specific syntactic position are collected into a seed cluster at the beginning of the merging problem.

Argument Head Distribution The distribution over argument heads is the same for two clusters that represent the same semantic role. The distribution of arguments is captured in vector space by the model described in Section 3.1. We calculate

centroid vectors from the instances in each cluster. To measure similarity between clusters a and b, we use cosine similarity between centroids:

$$D(a,b) = \frac{v_a^T v_b}{\|v_a\|\|v_b\|} \quad (7)$$

Equations (3)–(7) comprise the LP model.

4 Experimental Setup

In this section we present our experimental setup for assessing the performance of the model presented above. We explain how it was trained and tested, and also briefly introduce the models used for comparison with our approach.

4.1 Training

To obtain distributed representations, we used text from a subset of the English Gigaword corpus (Parker et al., 2011), comprising almost 64 million tokens (2.7 million sentences). The training corpus was pre-processed using `MATE` (Björkelund et al., 2009) to lemmatize the words, provide POS-tags and a dependency parse, identify verbal predicates and the position of arguments.

The neural network model described in Section 3.1 was trained using Matlab. We restricted the predicate vocabulary to use the 5,000 most frequent verbs in the training corpus, and the verbal predicates found in the CoNLL-2008 shared task data set (Surdeanu et al., 2008). Predicates were represented as vectors of size 80, while vectors of length 50 were used for arguments. We used a symmetric context window of size $c = 1$. As the mechanism to prevent all vectors from having the same value, we used "negative-sampling" (Mikolov et al., 2013b), where there are $k = 5$ randomly sampled negative examples of (context, target) pairs for each data sample. This technique has the advantage that we do not need to provide numerical probabilities for the noise distribution. Model parameters were updated during training using stochastic gradient descent over 5 epochs, decreasing the update step size at each epoch.

4.2 Argument clustering

Following common practice in unsupervised role induction (Titov and Klementiev, 2012; Lang and Lapata, 2014), we evaluated our model on the complete CoNLL-2008 shared task data set. We used the clustering metrics of purity, collocation and their harmonic mean F1. In addition, we used V-measure (Rosenberg and Hirschberg, 2007), an entropy-based measure which explicitly evaluates how successfully the criteria of homogeneity and completeness have been satisfied.

In previous work on unsupervised role induction, the results for each predicate were weighted in proportion to the number of times the predicate appeared in the CoNLL-2008 test set. In addition to this measure, we evaluate clustering where predicates are uniformly weighted. In a data set where the top 10 predicates account for almost 20% of the samples, these metrics give a view of performance on the other 3,000-plus predicates where less predicate-specific data is available.

4.3 Comparison Models

We compared our model against a baseline that assigns arguments to clusters based on their syntactic functions (SYNTF; Lang and Lapata, 2014). Specifically, the baseline forms clusters from the syntactic position of an argument using four cues: the verb's voice, the argument's position relative to the predicate, its syntactic relation, and any realizing preposition.[1]

To assess whether our *argument*-based model has any advantages over other *word*-based distributed representations we compared the following variants: (a) the `arg2vec` model presented in Section 3.1 trained on the subset of Gigaword; (b) the continuous bag-of-words model trained using `word2vec` on the same Gigaword corpus; and (c) 300–dimensional vectors pre-trained on part of the Google News dataset[2] (about 100 billion words), again using `word2vec`. In all three instances, we performed argument clustering using the LP of Section 3.2. We also compare against Agglomerative-cosine (AGGLOM), the best performing model of Lang and Lapata (2014).[3] Where applicable, we also refer to the models presented in Titov and Klementiev (2012).

5 Results

Our results on the semantic role induction task are summarized in Tables 1 and 2. Table 1 presents results using the gold standard parses and argu-

[1] Differences in the results compared to Lang and Lapata (2014) are due to our re-implementation of the predicate labeling stage, to be consistent with the preprocessing we used for the other comparison systems.

[2] `http://code.google.com/p/word2vec/`

[3] Differences from published results are again due to changes at the predicate labeling stage.

	Weighted			Unweighted			Weighted			Unweighted		
	PU	CO	F1	PU	CO	F1	HO	CO	V1	HO	CO	V1
SYNTF	81.6	78.1	79.8	90.0	86.8	87.8	71.7	66.2	68.8	85.5	81.7	82.1
AGGLOM	87.4	75.3	**80.9**	95.1	80.7	86.5	79.2	65.5	71.7	93.1	78.1	84.0
word2vec-GIGAWORD	82.8	77.9	80.3	91.4	86.3	88.2	78.8	63.7	70.4	90.2	81.1	84.4
word2vec-GOOGLENEWS	83.4	76.2	79.7	91.6	85.7	87.9	78.7	63.7	70.4	90.2	80.7	84.1
arg2vec-GIGAWORD	87.9	74.7	80.8	94.2	85.4	**88.9**	86.1	64.6	**73.8**	94.6	80.9	**86.2**

Table 1: Purity, collocation and F1 measures (left), and homogeneity, completeness and V1 measures (right) for CoNLL-2008 data set, using gold syntax information.

	Weighted			Unweighted			Weighted			Unweighted		
	PU	CO	F1	PU	CO	F1	HO	CO	V1	HO	CO	V1
SYNTF	68.3	72.1	70.1	80.6	81.3	80.3	55.2	54.9	55.0	74.4	74.3	73.2
AGGLOM	75.5	69.5	72.4	89.3	77.9	82.4	64.9	55.7	60.0	86.1	74.6	78.8
DEPREL+MATE	81.4	77.7	79.5	88.7	84.8	86.2	71.5	65.7	68.5	83.7	79.2	80.3
word2vec-GIGAWORD	83.3	76.1	79.5	91.3	85.7	87.8	78.4	63.4	70.1	89.9	80.3	83.9
word2vec-GOOGLENEWS	82.9	77.4	80.1	91.3	86.0	88.0	78.3	63.6	70.2	89.9	80.9	84.2
arg2vec-GIGAWORD	87.7	74.6	**80.6**	93.9	85.3	**88.8**	85.7	64.4	**73.5**	94.4	80.8	**86.1**

Table 2: Purity, collocation and F1 measures, and homogeneity, completeness and V1 measures for CoNLL-2008 data set using automatic parse syntax information.

ments available in the CoNLL 2008 data set. Notice that our embeddings are still learned using automatically identified arguments. Table 2 uses automatic parses with automatically identified arguments which is a more realistic evaluation setting.

As can be seen in Table 1, when gold standard information is used the syntactic function baseline (SYNTF) is very effective. When considering F1 (weighted by the number of instances), arg2vec performs on the same par with graph-based agglomerative clustering (AGGLOM). Interestingly, word2vec performs worse when trained either on Gigaword or the Google News corpora. According to (weighted) V1, arg2vec outperforms all other comparison models. When predicates are weighted uniformly, arg2vec is the best performing model using F1 or the more information-centric V-measure. This suggests that our model performs well on the the less-frequent predicates and rarer semantic roles. The results also show that our model captures semantic information useful for this task more successfully than the word-based distributional models. Both word2vec models have similar performance, despite significant differences in the size of their training data.

Table 2 shows similar trends. The poorer performance of SYNTF and AGGLOM can partly be ascribed to the heuristics used for argument identification: DEPREL+MATE gives the baseline performance of our dependency parser and argument identification. Nevertheless, when comparing systems that have access to the same preprocessing, our arg2vec model gives the best performance particularly in the information-centric V-measures. Also note, that it seems robust to noise incurred by the automatic parsing and argument identification procedures.

Titov and Klementiev (2012) report a (weighted) F1 of 83.0 on the gold standard CoNLL-2008 dataset, using a coupled model where parameters are shared across verbs and a form of smoothing which replaces argument fillers by lexical cluster ids stemming from Brown et al.'s (1992) algorithm (trained on the RCV1 corpus, about 63 millions words). Our model would presumably benefit from a similar coupling mechanism which we could enforce as a constraint in the ILP. However, we leave this to future work. When tested on automatic parses and gold arguments, their model yields a weighted F1 of 78.8. For comparison, arg2vec obtains an F1 of 80.6 on automatic parses and arguments. Figure 4 shows visualizations of the argument semantic space as captured by the arg2vec-GIGAWORD model, for the predicates *eat* and *win*. Dimensionality reduction was performed by the t-SNE library.[4] The visualization suggests that the model learns similarities beyond simple word contexts.

The evaluation presented so far assesses the quality of the argument representations learned by our model. We also wanted to see whether the predicate embeddings capture meaningful semantic content. Figure 3 shows a visualization

[4] http://lvdmaaten.github.io/tsne/

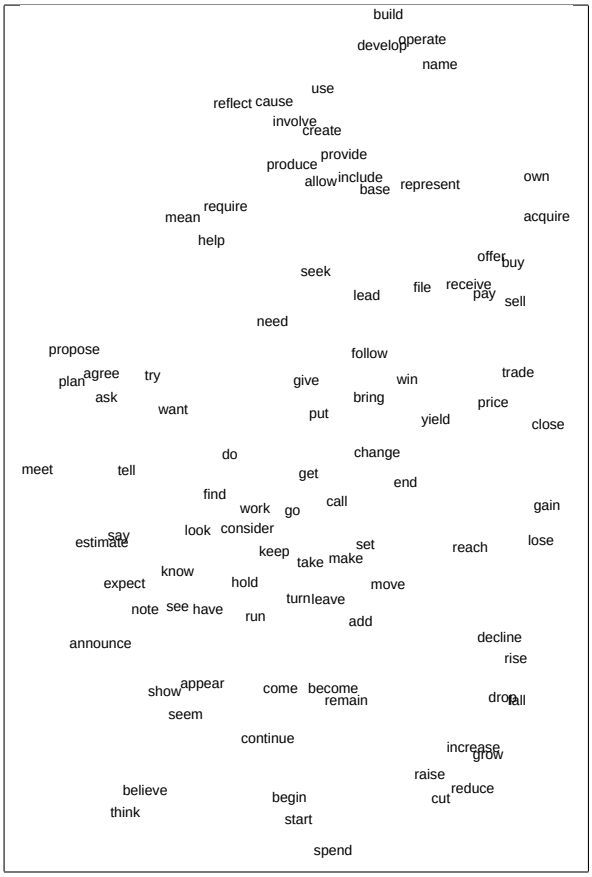


Figure 3: 2-D representation of the induced predicate space for the 100 most frequent predicates in CoNLL-2008.

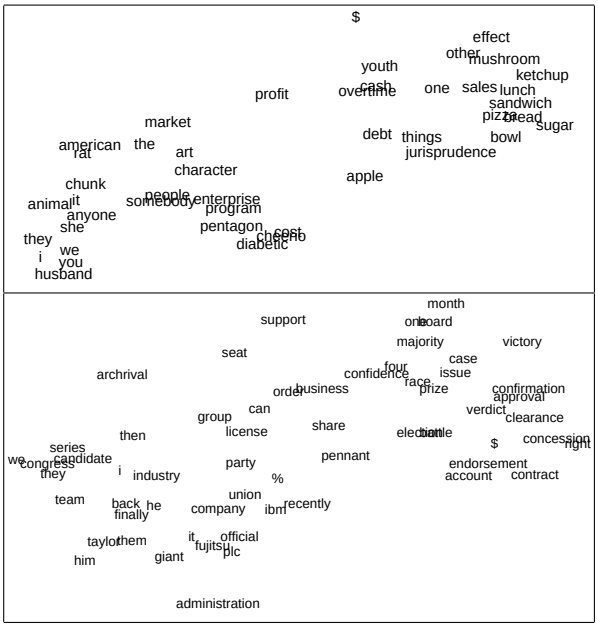


Figure 4: 2-D representation of the induced argument space for the predicates *eat* (top) and *win* (bottom). In both representations, A0 arguments are clustered bottom left, while A1 arguments are found top right.

of the predicate semantic space as captured by `arg2vec` when it is trained on the Gigaword corpus. It shows a projection of the 100 most frequent verbs in CoNLL-2008, with dimensionality reduction again performed by `t-SNE`. The visualization suggests that the model captures non-trivial predicate similarities. Verbs relating to buying and selling lie close together (e.g., *offer, buy, receive, pay, sell*). Verbs denoting growth or decrease are also grouped together (*drop, fall, increase, grow, reduce, cut*). Interestingly, verbs with similar argument structure share regions of the space (e.g., *say, estimate*, or *believe, think* or *seem, appear*). Usefully, verbs are represented in a continuous space rather than discrete clusters (e.g., *acquire* is somewhere between *buy* and *own*).

In order to quantitatively evaluate the quality of the predicate representations induced by our model, we compared the cosine distances between vectors to the hierarchy of VerbNet (Schuler, 2005). VerbNet is a hierarchical domain-independent broad-coverage verb lexicon for English, organizing verbs into classes. The evaluation task was, for all pairs of predicates, to predict whether they would be in the same cluster at the top layer of the hierarchy of VerbNet. To form the top layer of VerbNet, we took the first integer of each VerbNet class number. As an example, the verbs *believe* (VN class conjecture-29.5), *think* (consider-29.9), *expect* (conjecture-29.5-1), and *adopt* (appoint-29.1) would all be in the same class 29. According to this reduction of VerbNet, there are 101 classes. The prediction was based on whether the cosine distance between the pair of vectors was above a threshold value. We measured area under the precision-recall curve (AUC) which captures performance at all thresholds, and F1-score at the best threshold. `arg2vec` does better in both measures than a baseline of random vectors of the same dimension, scoring 0.637 for AUC compared to a baseline of 0.505, and 29.5 against 22.9 for F1.

6 Conclusion

In this paper we presented a new approach for learning distributed representations for predicates and their arguments which we show is useful for unsupervised semantic role labeling. Rather than creating a task-specific algorithm for role induction, we learn a task-specific representation. We

thus decouple feature learning from clustering inference, which results in a conceptually simpler model. Through a formulation of the clustering problem as a linear programme, we are able to perform clustering efficiently and incorporate task-specific constraints. In the future, we would like to investigate how our approach generalizes across languages and tasks.

Acknowledgements We would like to thank Miguel Forte and members of the ILCC at the School of Informatics for their valuable feedback. We acknowledge the support of EPSRC (EP/K017845/1) in the framework of the CHIST-ERA READERS project.

References

Michael Auli, Michel Galley, Chris Quirk, and Geoffrey Zweig. 2013. Joint language and translation modeling with recurrent neural networks. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1044–1054, Seattle, Washington.

Yoshua Bengio, Holger Schwenk, Jean-Sébastien Morin, and Jean-Luc Gauvain, 2006. *Neural Probabilistic Language Models*, pages 137–186. Springer.

Anders Björkelund, Love Hafdell, and Pierre Nugues. 2009. Multilingual Semantic Role Labeling. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL 2009): Shared Task*, pages 43–48, Boulder, Colorado, June. Association for Computational Linguistics.

P. F. Brown, V. J. Della Pietra, P. V. deSouza, J. C. Lai, and R. L. Mercer. 1992. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):283–298.

Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111, Doha, Qatar, October. Association for Computational Linguistics.

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

Nikhil Garg and James Henderson. 2012. Unsupervised Semantic Role Induction with Global Role Ordering. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 145–149, Jeju Island, Korea.

Sean Gilpin, Siegried Nijssen, and Ian Davidson. 2013. Formalizing Hierarchical Clustering as Integer Linear Programming. In *In Proceedings of the 27th AAAI Conference on Artificial Intelligence*, pages 372–378, Bellevue, Washington.

Trond Grenager and Christopher D. Manning. 2006. Unsupervised discovery of a statistical verb lexicon. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 1–8, Sydney, Australia.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1700–1709, Seattle, Washington.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A Convolutional Neural Network for Modelling Sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 655–665, Baltimore, Maryland.

Joel Lang and Mirella Lapata. 2010. Unsupervised induction of semantic roles. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 939–947, Los Angeles, California.

Joel Lang and Mirella Lapata. 2014. Similarity-driven semantic role induction via graph partitioning. *Computational Linguistics*, 40(3):133–164.

Quoc V. Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, pages 1188–1196.

Lluís Màrquez, Xavier Carreras, Kenneth C. Litkowski, and Suzanne Stevenson. 2008. Semantic Role Labeling: An Introduction to the Special Issue. *Computational Linguistics*, 34(2):145–159, June.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient Estimation of Word Representations in Vector Space, January.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeffrey Dean. 2013b. Distributed Representations of Words and Phrases and their Compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc., October.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013c. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751, Atlanta,

Georgia, June. Association for Computational Linguistics.

Andriy Mnih and Geoffrey E. Hinton. 2008. A scalable hierarchical distributed language model. In *Advances in Neural Information Processing Systems*, pages 1081–1088.

Andriy Mnih and Koray Kavukcuoglu. 2013. Learning word embeddings efficiently with noise-contrastive estimation. In *Advances in Neural Information Processing Systems 26*, pages 2265–2273.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The Proposition Bank: An Annotated Corpus of Semantic Roles. *Computational Linguistics*, 31(1):71–106, March.

Robert Parker, David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2011. English Gigaword Fifth Edition. *LDC2011T07*. Linguistic Data Consortium.

Vasin Punyakanok, Dan Roth, and Wen-tau Yih. 2008. The importance of syntactic parsing and inference in semantic role labeling. *Computational Linguistics*, 34(2):257–287, June.

Andrew Rosenberg and Julia Hirschberg. 2007. V-Measure: A Conditional Entropy-Based External Cluster Evaluation Measure. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*.

Karin Kipper Schuler. 2005. *VerbNet: A broad-coverage, comprehensive verb lexicon*. Ph.D. thesis, University of Pennsylvania.

Richard Socher, Eric H. Huang, Jeffrey Pennin, Andrew Y. Ng, and Christopher D. Manning. 2011a. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In J. Shawe-Taylor, R.S. Zemel, P. Bartlett, F.C.N. Pereira, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 801–809.

Richard Socher, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher D. Manning. 2011b. Semi-Supervised Recursive Autoencoders for Predicting Sentiment Distributions. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 151–161, Edinburgh, Scotland, UK.

Mihai Surdeanu, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. The CoNLL 2008 Shared Task on Joint Parsing of Syntactic and Semantic Dependencies. In *CoNLL 2008: Proceedings of the Twelfth Conference on Computational Natural Language Learning*, pages 159–177, Manchester, England, August. Coling 2008 Organizing Committee.

Oscar Täckström, Kuzman Ganchev, and Dipanjan Das. 2015. Efficient inference and structured learning for semantic role labeling. *Transactions of the Association for Computational Linguistics*, 3:29–41.

Ivan Titov and Ehsan Khoddam. 2015. Unsupervised induction of semantic roles within a reconstruction-error minimization framework. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1–10, Denver, Colorado.

Ivan Titov and Alexandre Klementiev. 2012. A Bayesian approach to unsupervised semantic role induction. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 12–22, Avignon, France.

Joseph Turian, Lev-Arie Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 384–394, Uppsala, Sweden.

A Tableau Prover for Natural Logic and Language

Lasha Abzianidze
TiLPS, Tilburg University, the Netherlands
L.Abzianidze@uvt.nl

Abstract

Modeling the entailment relation over sentences is one of the generic problems of natural language understanding. In order to account for this problem, we design a theorem prover for Natural Logic, a logic whose terms resemble natural language expressions. The prover is based on an analytic tableau method and employs syntactically and semantically motivated schematic rules. Pairing the prover with a preprocessor, which generates formulas of Natural Logic from linguistic expressions, results in a proof system for natural language. It is shown that the system obtains a comparable accuracy ($\approx 81\%$) on the unseen SICK data while achieving the state-of-the-art precision ($\approx 98\%$).

1 Introduction

A problem of recognizing textual entailments (RTE)—given two text fragments T (for a text) and H (for a hypothesis), determine whether T entails, contradicts or is neutral to H—is considered as a complex and, at the same time, fundamental problem for several NLP tasks (Dagan et al., 2005). For more than a decade, RTE challenges have been held, where systems are competing to each other with respect to human annotated RTE test data; but there are few systems that try to solve RTE problems by computing meanings of linguistic expressions and employing inference engines similar to proof procedures of formal logics. Moreover, those few systems are usually used in combination with shallow classifiers since the systems' performances alone are poor.

The current paper advocates that purely deductive inference engines over linguistic representations backed up with a simple lexical knowledge base could be solely and successfully used for the RTE task. Our work builds on the theory of an analytic tableau system for Natural Logic (Natural Tableau) introduced by Muskens (2010). The theory offers to employ a tableau method—a proof procedure used for many formal logics—for the version of Natural Logic that employs Lambda Logical Forms (LLFs)—certain terms of simply typed λ-calculus—as Logical Forms (LFs) of linguistic expressions. The merits of the current approach are several and they can be grouped in two categories: virtues attributed to the tableau prover are (i) the high precision for the RTE task characteristic to proof procedures, (ii) the transparency of the reasoning process, and (iii) ability for solving problems with several premises; and those concerning LLFs are (iv) an evidence for LFs that are reminiscent of Surface Forms but still retaining complex semantics, and (v) an automatized way of obtaining LLFs from wide-coverage texts.

The rest of the paper is organized as follows. First, Natural Tableau is introduced, and then a method of obtaining LLFs from raw text is described. We outline the architecture of an implemented theorem prover that is based on the theory of Natural Tableau. The power of the prover is evaluated against the SICK data; the results are analyzed and compared to related RTE systems. The paper concludes with future work.

2 Natural Tableau for Natural Logic

Natural Logic is a vague notion and refers to logics that account for valid inferences of natural languages, where reasoning and the grammar are strongly related to each other and LFs resemble surface forms (Lakoff, 1972). On the other hand, a tableau method (Beth, 1955) is a popular proof procedure and nowadays many formal logics have their own version of it (D'Agostino et al., 1999). A combination of these two devices is offered by Muskens (2010), where the language of Natural Logic is considered to be a part of simply typed

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 2492–2502,
Lisbon, Portugal, 17-21 September 2015.

$$\frac{X\ A\ B:\ [\,]:\ \mathbb{F}}{\begin{array}{c}A:\ [c]:\ \mathbb{T}\\ B:\ [c]:\ \mathbb{F}\\ \text{s.t. } X \in \{\texttt{all}, \texttt{every}\}\\ \text{and } c \text{ is a } \textit{fresh} \text{ term}\end{array}}\forall_F \qquad \frac{A\ B:\ [\vec{C}]:\ \mathbb{X}}{A:\ [B,\vec{C}]:\ \mathbb{X}}\text{PUSH} \qquad \frac{A:\ [B,\vec{C}]:\ \mathbb{X}}{A\ B:\ [\vec{C}]:\ \mathbb{X}}\text{PULL}$$

$$\frac{\begin{array}{c}A:\ [\vec{C}]:\ \mathbb{T}\\ B:\ [\vec{C}]:\ \mathbb{F}\end{array}}{\begin{array}{c}\times\\ \text{s.t. } A \leq B\end{array}}\leq\!\times \qquad \frac{X\ A\ B:\ [\,]:\ \mathbb{F}}{\begin{array}{c}A:\ [d]:\ \mathbb{F} \qquad B:\ [d]:\ \mathbb{F}\\ \text{s.t. } X \in \{\texttt{some}, \texttt{a}\} \text{ and } d \text{ is an } \textit{old} \text{ term}\end{array}}\exists_F$$

$$\frac{\texttt{not}\ A:\ [\vec{C}]:\ \mathbb{X}}{A:\ [\vec{C}]:\ \overline{\mathbb{X}}}\text{NOT}$$

Figure 1: Tableau rules for quantifiers ($\forall_F$ and $\exists_F$), Boolean operators (NOT), formatting (PUSH and PULL) and inconsistency ($\leq\!\times$). The relation $\leq$ stands for entailment, $\vec{C}$ and $\mathbb{X}$ are meta-variables over sequences of terms and truth signs ($\mathbb{T}$ and $\mathbb{F}$), respectively; the bar operator $\overline{\mathbb{X}}$ negates a sign.

λ-terms that are built up from variables and lexical constant terms with the help of application and lambda abstraction. The terms of the language are called LLFs and resemble linguistic surface forms:[1]

$$\texttt{a}_{(et)(et)t}\ \texttt{bird}_{et}\ \texttt{fly}_{et}$$

$$\texttt{some}_{(et)(et)t}\ \texttt{bird}_{et}\ (\texttt{not}_{(et)et}\ \texttt{fly}_{et})$$

$$\texttt{not}_{((et)(et)t)(et)(et)t}\ \texttt{all}_{(et)(et)t}\ \texttt{bird}_{et}\ \texttt{fly}_{et}$$

Note that common nouns and intransitive verbs are typed as properties (i.e. functions from entities to truth values) and quantifiers as binary relations over properties; the latter typing treats quantified noun phrases (QNPs) as generalized quantifiers (GQs)—a term of type properties over properties $(et)t$.

A Natural Tableau entry is a tuple containing a term, a sequence of terms representing an argument list, and a truth sign. The entries are such that when a term is applied to all arguments from an argument list in the order of the list, the resulted term is of type truth value. For example, $A_{eet}c_e : [d_e] : \mathbb{T}$ is a valid tableau entry (i.e. a node) since it consists of a term $A_{eet}c_e$, an argument list $[d_e]$ and a truth sign $\mathbb{T}$ standing for *true*, and additionally, $A_{eet}c_e d_e$ is a term of type t.

A tableau method is a refutation method and it proves an argument by searching a counterexample. The search process is guided by applications of certain set of rules. A tableau rule is a schema with a set of antecedent nodes above a line and a set of precedent branches below a line, where each branch consists of (precedent) nodes. A rule is applicable if all its antecedent nodes match to some nodes in a tableau, and after the rule is applied, precedent nodes of the rule are introduced in the tableau. A tableau consists of branches where each branch models a situation and is either closed (i.e., inconsistent) or open (i.e., consistent) depending whether it contains a closure $\times$ sign (i.e., an obvious contradiction). A tableau is closed if all its branches are closed, otherwise it is open.

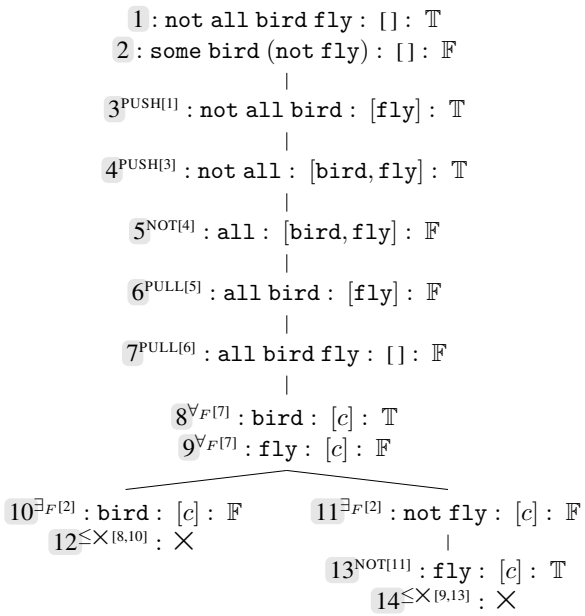


Figure 2: The closed tableau serves as a proof for: *not all birds fly → some bird does not fly*

In Figure 2, a tableau proof, which employs the rules of (Muskens, 2010) from Figure 1, is presented. In order to show a way the tableau is developed, the nodes are enumerated and annotated with a source rule and IDs of nodes from which a current node is obtained. For example, 3 is obtained from 1 by the PUSH rule. In order to prove an argument, the tableau starts with a counterexample of the argument, i.e. a premise being true and a conclusion false. After several rule applications, all the branches of the tableau close meaning

[1]Since modeling intensionality is beyond the scope of the paper, we present LLFs typed with extensional semantic types, i.e. types are built up from basic e (for entities) and t (for truth values) types. We use the comma as a type constructor, e.g., (e, t) stands for a functional type from entities to truth values. The comma is omitted when types are denoted by single letters, e.g., et stands for (e, t). Taking into account right-associativity of the type constructor we often drop parentheses for better readability. Terms are optionally annotated with their types in a subscript.

that none of the situations for the counterexample were consistent.

An advantage of Natural Tableau is that it treats both single and multi-premised arguments in the same fashion and represents a deductive procedure in an intuitive and transparent way.

3 Obtaining LLFs for Natural Tableau

3.1 CCG and the C&C Parser

Combinatory Categorial Grammar (CCG) is a lexicalized grammar formalism that assigns a syntactic category and a semantic interpretation to lexical items, where the items are combined via combinatory rules (Steedman, 2000; Steedman and Baldridge, 2011). The CCG category A/B (or $A\backslash B$) is a category of an item that becomes of category A when it is combined with an item of category B on its right (or left, respectively) side. In the example below, the sentence *every man walks* is analyzed in the CCG formalism, where lexical items are combined via the *forward application* rule and unspecified semantic interpretations are written in a boldface:

$$\dfrac{\dfrac{\dfrac{\text{every}}{(S/(S\backslash NP))/N : \mathbf{every}} \quad \dfrac{\text{man}}{N : \mathbf{man}}}{S/(S\backslash NP) : \mathbf{every\ man}} \quad \dfrac{\text{walks}}{S\backslash NP : \mathbf{walk}}}{S : (\mathbf{every\ man})\ \mathbf{walk}}$$

The CCG derivation trees are suitable structures for obtaining LLFs for at least two reasons. First, the CCG framework is characterized by a transparent interface between syntactic categories and semantic types; second, there exist efficient and robust CCG parsers for wide-coverage texts.

During obtaining LLFs, we employ the C&C CCG parser of Clark and Curran (2007) and EasyCCG of Lewis and Steedman (2014). While the C&C parser is a pipeline of several NLP systems: POS-tagger, chunker, named entity recognizer (NER), lemmatizer (Minnen et al., 2001) supertagger and sub-parser, EasyCCG is an extremely simple but still comparably accurate CCG parser based on A* parsing.[2] These two parsers use different settings for supertagging and parsing; therefore, it is interesting to test both parsers for our application.

In Figure 3, there is a CCG derivation by the C&C parser displayed in a tree style: terminal nodes are annotated with tokens, syntactic categories, lemmas and POS-tags while non-terminal nodes are marked with combinatory rules and resulted categories; some basic categories are subcategorized by features.

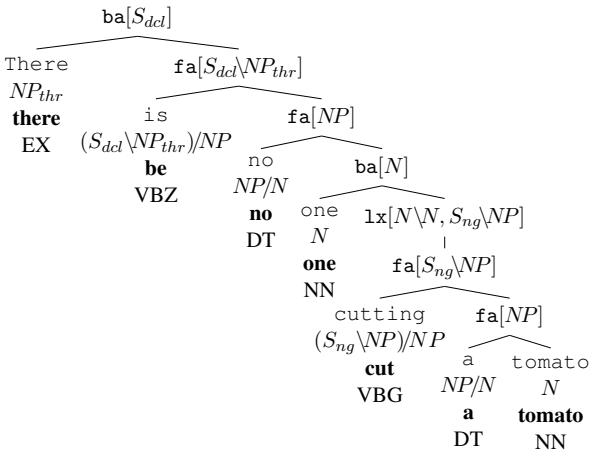


Figure 3: The CCG tree by the C&C parser for *there is no one cutting a tomato* (SICK-2404), where *thr, dcl, ng* category features stand for an expletive *there*, declarative and present participle, respectively.

3.2 From CCG Trees to LLFs

Initially, it may seem easy to obtain fine-grained LLFs from CCG trees of the parsers, but careful observation on the trees reveals several complications. The transparency between the categories and types is violated by the parsers as they employ lexical (i.e. type-changing) rules—combinatory rules, *non-native* ones for CCG, which changes categories. Lexical rules were initially introduced in CCGbank (Hockenmaier and Steedman, 2007) to decrease the total number of categories and rules. In the tree of Figure 3, a lexical rule changes a category $S_{ng}\backslash NP$ of a phrase *cutting a tomato* with $N\backslash N$. In addition to this problem, the trees contain mistakes from supertaggers (and from the other tools, in case of the C&C parser).

The first step in processing CCG trees is to remove directionality from the categories. This step is the same as obtaining unspecified semantic interpretation of a phrase in the CCG framework. While converting categories $A\backslash B$ and A/B into a non-directional type $(\mathtt{b}, \mathtt{a})$, the arrangement of nodes must be changed in a corresponding way. For instance, in case of the top backward application rule ($\mathtt{ba}[S_{dcl}]$ in Figure 3), the order of

[2] The employed C&C parser is trained on *rebanked* CCGbank (Honnibal et al., 2010)—an updated version of CCGbank (Hockenmaier and Steedman, 2007) with improved analyses for predicate-argument structures and nominal modifiers. For EasyCCG, input sentences are already processed by the POS-tagger and the NER of the C&C parser.

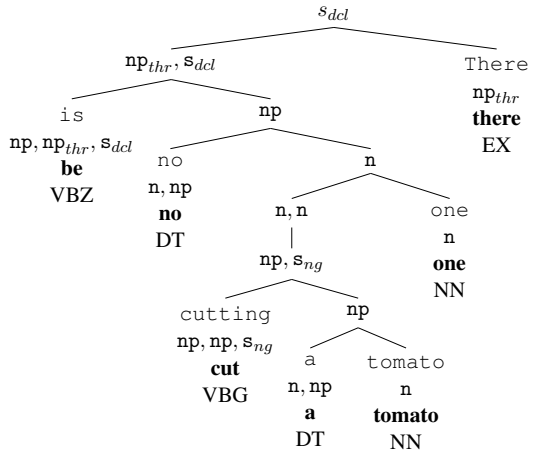


Figure 4: A CCG term obtained from the CCG tree of Figure 3. Categories are converted into types.

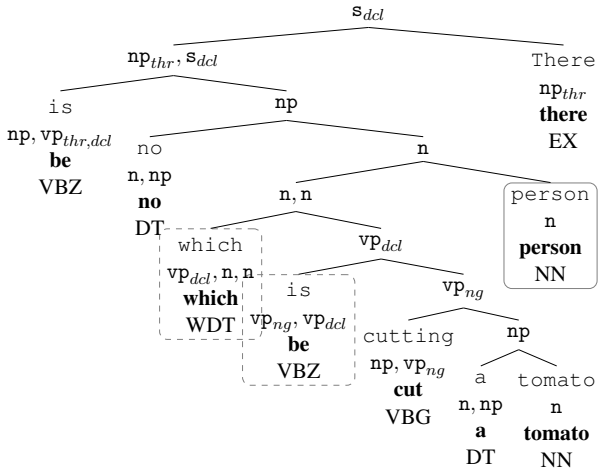


Figure 5: A fixed CCG term that is obtained from the CCG term of Figure 4. A node with a dashed (solid) frame is inserted (substituted, respectively). A type $\mathtt{vp}_{a,b}$ abbreviates $(\mathtt{np}_a, \mathtt{s}_b)$.

nodes is reversed to guarantee that the function category $(\mathtt{s}_{dcl}, \mathtt{np}_{thr})$ precedes its argument category $\mathtt{np}_{thr}$. There are about 20 combinatory rules used by the parsers and for each of them we design a way of reordering subtrees. In the end, the order of nodes coincides with the order according to which semantic interpretations are combined. The reordering recipes for each combinatory rule is quite intuitive and can be found in (Steedman, 2000) and (Bos, 2009), where the latter work also uses the C&C parser to obtain semantic interpretations. Trees obtained after removing the directionality from the categories are called CCG terms since they resemble syntactic trees of typed λ-terms (see Figure 4).

$$\mathtt{on}_{\mathtt{np},\mathtt{pp}}(\mathtt{ice}_{\mathtt{n}})_{\mathtt{np}} \longrightarrow \mathtt{on}_{\mathtt{np},\mathtt{pp}}(\mathtt{a}_{\mathtt{n},\mathtt{np}}\mathtt{ice}_{\mathtt{n}}) \quad (1)$$

$$\mathtt{run}_{\mathtt{np},\mathtt{s}}(\mathtt{dogs}_{\mathtt{n}})_{\mathtt{np}} \longrightarrow \mathtt{run}_{\mathtt{np},\mathtt{s}}(\mathtt{s}_{\mathtt{n},\mathtt{np}}\mathtt{dog}_{\mathtt{n}}) \quad (2)$$

$$(\mathtt{Dow}^{\mathtt{PER}}_{\mathtt{n},\mathtt{n}}\ \mathtt{Jones}^{\mathtt{PER}}_{\mathtt{n}})_{\mathtt{np}} \longrightarrow \mathtt{Dow\_Jones}_{\mathtt{np}} \quad (3)$$

$$(\mathtt{two}_{\mathtt{n},\mathtt{n}}\ \mathtt{dogs}_{\mathtt{n}})_{\mathtt{np}} \longrightarrow \mathtt{two}_{\mathtt{n},\mathtt{np}}\ \mathtt{dogs}_{\mathtt{n}} \quad (4)$$

$$\mathtt{her}_{(\mathtt{pp},\mathtt{n}),\mathtt{np}}\ \mathtt{car}_{\mathtt{pp},\mathtt{n}} \longrightarrow \mathtt{her}_{\mathtt{n},\mathtt{np}}\ \mathtt{car}_{\mathtt{n}} \quad (5)$$

$$\mathtt{who}_{\mathtt{w}}V(Q_{\mathtt{n},\mathtt{np}}N) \longrightarrow Q_{\mathtt{n},\mathtt{np}}(\mathtt{who}_{\mathtt{w}'}VN) \quad (6)$$

$$\mathtt{nobody} \longrightarrow \mathtt{no}_{\mathtt{n},\mathtt{np}}\ \mathtt{person}_{\mathtt{n}} \quad (7)$$

Lexical rules are the third most commonly used combinatory rules (7% of all rules) by the parsers on the SICK data (Marelli et al., 2014b), and therefore, they deserve special attention. In order to compositionally *explain* several category changes made by lexical rules (represented with $(.)_\alpha$ operator in terms), either types of constant terms are set to proper types or lexical entries are inserted in CCG terms. For explaining a lexical rule $\mathtt{n} \rightsquigarrow \mathtt{np}$, mainly used for bare nouns, an indefinite determiner is inserted for singular nouns (1) and a plural morpheme *s* is used as a quantifier for plurals (2). Also identifying proper names with the feature assigned by the C&C NER tool helps to eliminate $\mathtt{n} \rightsquigarrow \mathtt{np}$ change (3). Correcting the type of a quantifier that is treated as a noun modifier is another way of eliminating this lexical rule (4). In case of $(\mathtt{s}, \mathtt{np}) \rightsquigarrow (\mathtt{n}, \mathtt{n})$ change, *which is* phrase is inserted and salvages the category-type transparency of CCG (see Figure 5). As a whole, the designed procedures explain around 99% of lexical rules used in CCG terms of the SICK sentences. Note that explaining lexical rules guarantees a well-formed CCG term in the end.

Apart from the elimination of lexical rules, we also manually design several procedures that fix a CCG term: make it more semantically adequate or simplify it. For example, the C&C parser assigns a category *N/PP* of relational nouns to nouns that are preceded by possessives. In these cases, a type $\mathtt{n}$ is assigned to a noun and a type of possessive is changed accordingly (5). To make a term semantically more adequate, a relative clause is attached to a noun instead of a noun phrase (6), where a type $\mathtt{w} \equiv (\mathtt{vp}, \mathtt{np}, \mathtt{s})$ of a wh-word is changed with $\mathtt{w}' \equiv (\mathtt{vp}, \mathtt{n}, \mathtt{s})$. CCG terms are simplified by substituting terms for *no one, nobody, everyone*, etc. with their synonymous terms (see (7) and Figure 5). These substitutions decrease a vocabulary size, and hence, decrease the number of tableau rules.

The final operation is to convert a fixed CCG term into an LLF, meaning to convert QNPs into GQs of (Montague, 1974; Barwise and Cooper, 1981). In this procedure, a type $(\mathtt{n}, \mathtt{np})$ of a quan-

tifier is replaced with $(\mathtt{n},(\mathtt{np},\mathtt{s}),\mathtt{s})$, and the resulted new NP is applied to the smallest clause it occurs in; but if there are other QNPs too, then it also applies to the clauses where other QNPs are situated. This operation is not deterministic and can return several terms due to multi-options in quantifier scope ordering. As an example, two λ-terms, (9) and (10), are obtained from the CCG term (8) of Figure 5.[3]

$$\mathbf{b}(\mathbf{no}(\mathbf{w}(\mathbf{b}(\mathbf{c}(\mathbf{a}\,\mathbf{t})))\mathbf{p}))\mathbf{th} \quad (8)$$

$$\mathbf{no}\left(\mathbf{w}(\mathbf{b}(\lambda x.\,\mathbf{a}\,\mathbf{t}(\lambda y.\,\mathbf{c}yx)))\mathbf{p}\right)\left(\lambda z.\,\mathbf{b}\,z\,\mathbf{th}\right) \quad (9)$$

$$\mathbf{a}\,\mathbf{t}(\lambda x.\,\mathbf{no}\left(\mathbf{w}(\mathbf{b}(\mathbf{c}x))\mathbf{p}\right)\left(\lambda z.\,\mathbf{b}\,z\,\mathbf{th}\right)) \quad (10)$$

Eventually the final λ-terms, analogous to (9) and (10), obtained from CCG trees will be considered as LLFs that will be used in the wide-coverage theorem prover. It has to be stressed that generated LLFs are theory-independent abstract semantic representation. Any work obtaining semantic representations from CCG derivations can combine its lexicon with (already corrected) LLFs and produce more adequate semantics in this way.

3.3 Extending the Type System

An obvious and simple way to integrate the LLFs, obtained in Subsection 3.2, in Natural Tableau is to translate their types into semantic types built up from e and t.[4] We will not do so, because this means the information loss since the information about syntactic types are erased; for example, usually syntactic types $\mathtt{pp}$, $\mathtt{n}$ and $(\mathtt{np},\mathtt{s})$ are translated as et type. Retaining syntactic types also contributes to fine-grained matching of nodes during rule application in the prover. For instance, without syntactic types it is more complex to determine the context in which a term $\mathbf{game}$ occurs and find an appropriate tableau rule when considering the following LLFs, $\mathbf{game}_{\mathtt{n},\mathtt{n}}\mathbf{theory}$ and $\mathbf{game}_{\mathtt{pp},\mathtt{n}}(\mathbf{of}\ X)$, as both $(\mathtt{n},\mathtt{n})$ and $(\mathtt{pp},\mathtt{n})$ are usually translated into $(et)et$ type, like it is done by (Bos, 2009).

In order to accommodate the LLFs with syntactic types in LLFs of (Muskens, 2010), we extend the semantic type system with $\mathtt{np},\mathtt{n},\mathtt{s},\mathtt{pp}$ basic syntactic types corresponding to basic CCG categories. Thus complex types are now built up from the set $\{e,t,\mathtt{np},\mathtt{n},\mathtt{s},\mathtt{pp}\}$ of types. The extension automatically licenses LLFs with syntactic types as terms of the extended language.

We go further and establish interaction between semantic and syntactic types in terms of a subtyping $\sqsubseteq$ relation. The relation is defined as a *partial order* over types and satisfies the following conditions for any $\alpha_1,\alpha_2,\beta_1,$ and β_2 types:

(a) $e\sqsubseteq\mathtt{np},\quad \mathtt{s}\sqsubseteq t,\quad \mathtt{n}\sqsubseteq et,\quad \mathtt{pp}\sqsubseteq et;$

(b) $(\alpha_1,\alpha_2)\sqsubseteq(\beta_1,\beta_2)$ iff $\beta_1\sqsubseteq\alpha_1$ and $\alpha_2\sqsubseteq\beta_2;$

Moreover, we add an additional typing rule to the calculus: a term is of type β if it is already of type α and $\alpha\sqsubseteq\beta$. According to this typing rule, now a term can be of multiple types. For example, both $\mathbf{walk}_{\mathtt{np},\mathtt{s}}$ and $\mathbf{man}_{\mathtt{n}}$ terms are also of type et, and all terms of type $\mathtt{s}$ are of type t too. From this point on we will use a boldface style for lexical constants of syntactic types.

Initially it may seem that the lexicon of constant terms is doubled in size, but this is not the case as several syntactic constants can mirror their semantic counterparts. This is achieved by *multiple typing* which enables to put semantic and syntactic terms in the same term. For instance, $\mathbf{love}_{\mathtt{np},\mathtt{np},\mathtt{s}}\,c_e\,\mathbf{john}_{\mathtt{np}}$ and $\mathbf{at}_{\mathtt{np},\mathtt{pp}}\,c_e\,d_e$ are well-formed LLFs of type t that combine terms of syntactic and semantic types, and there is no need of introducing semantic terms (e.g., $\mathtt{at}_{\mathtt{eet}}$ or $\mathtt{love}_{\mathtt{eet}}$) in order to have a well-formed term. In the end, the extension of the language is conservative in the sense that LLFs and the tableau proof of Section 2 are preserved. The latter is the case since the tableau rules are naturally extensible to match new LLFs.

4 Implementation of the Prover

In order to further develop and evaluate Natural Tableau, we implement the prover, LangPro, based on the extended theory. Its general architecture is based on the first-order logic (FOL) prover of Fitting (1990). The prover also contains a module for λ-calculus that roughly follows (Blackburn and Bos, 2005).

Setup of the inventory of rules is a crucial for efficiency of the prover. There is a priority order for the categories of rules according to their computational efficiency. The prover most prefers to employ non-branching rules that introduce no fresh terms and antecedents of which can be ignored af-

[3] We use initial letters of lemmas to abbreviate a term corresponding to a lexical entry. Note that (9) represents a reading with *no one* having a wide scope while, in (10), *a tomato* has a wide scope.

[4] The similar translation is carried out in (Bos, 2009) for Boxer (Bos, 2008), where basic CCG categories are mapped to semantic types and the mapping is isomorphically extended to complex categories.

ter the application (e.g., NOT). Less preferred and inefficient rules are the ones that branch, produce new terms or antecedents of which are kept after the application (e.g., $\forall_F$ and $\exists_F$). In order to encourage finding short proofs, *admissible* rules representing shortcuts of several rule applications are also introduced (e.g., FUN$\uparrow$ and ARG in Figure 9). The inventory consists of about 50 rules, where most of them are manually designed based on RTE problems (see Section 5.1) and the rest represents the essential part of the rules found in (Muskens, 2010).

The LLF generator (LLFgen) is a procedure that generates LLFs from a CCG derivation in the way described in Subsection 3.2. We also implement an LLF-aligner that serves as an optional preprocessor between LLFgen and the prover itself; it aligns identical chunks of LLFs and treats them as a constant (i.e. having no internal structure). This treatment often leads to smaller tableau proofs. The example of aligned LLFs is given in Figure 8.

LangPro uses only the antonymy relation and a transitive closure of the hyponymy/hypernymy relations from WordNet 3.0 (Fellbaum, 1998) as its knowledge base (KB). The entailment $\leq$ (contradiction $\perp$) relation between lexical constants of the same type $A \leq B$ ($A \perp B$) holds if there exists a WordNet sense of A that is a transitive hyponym (an antonym) of some WordNet sense of B. Note that there is no word sense disambiguation (WSD) used by the prover; therefore, adopting these interpretations of entailment and contradiction amounts to considering all senses of the words. For example, *a man is crying* entails *a man is screaming* as there are senses of *cry* and *scream* that are in the entailment relation.

All in all, chaining a CCG parser, LLFgen, the LLF-aligner, the prover and KB results in an automatized tableau prover LangPro which operates directly over natural language text.

5 Learning and Evaluation

5.1 Learning

For learning and evaluation purposes, we use the SICK data (Marelli et al., 2014b). The data consists of problems that are rich in the lexical, syntactic and semantic phenomena that compositional distributional semantic models (Mitchell and Lapata, 2010) are expected to account for.[5] The SICK data contains around 10K text-hypothesis pairs that are classified in three categories: entailment, contradiction, and neutral.

During learning we used only the trial portion of the data, SICK-trial, including 500 problems. The learning process consists of improving the components of the prover while solving the RTE problems: designing fixing procedures of LLFgen, adding new sound rules to the inventory, and introducing valid relations in KB that were not found in WordNet (e.g., *woman*$\leq$*lady*, *note*$\leq$*paper* and *food*$\leq$*meal*). During learning, each RTE problem is processed as follows:

input: $(T, H, \texttt{answer})$;
1: $t =$ the first LLF of $\texttt{llf}(T)$;
2: $h =$ the first LLF of $\texttt{llf}(H)$;
3: **case** answer, $\texttt{tab}\{t{:}\mathbb{T}, h{:}\mathbb{F}\}$, $\texttt{tab}\{t{:}\mathbb{T}, h{:}\mathbb{T}\}$
ENTAILMENT, CLOSED, OPEN: HALT;
CONTRADICTION, OPEN, CLOSED: HALT;
NEUTRAL, OPEN, OPEN: HALT;
4: **otherwise**
5: **if** t or h is incorrect **then** try to amend llf; go to 1
6: **else if** a rule is missing **then** add it; go to 3
7: **else if** a relation is missing **then** add it; go to 3
8: **else** HALT;

A function llf denotes the combination of LLFgen and a CCG parser; for learning we use only the C&C parser. A function $\texttt{tab} : S \rightarrow \{\text{CLOSED}, \text{OPEN}\}$ returns CLOSED if one of the tableaux initiated with aligned or non-aligned set S of nodes closes; otherwise it returns OPEN. For instance, while checking a problem (T, H) on entailment (contradiction), tableau starts with a counterexample: T being true and H false (true, respectively). Note that 5-7 procedures are carried out manually while the phase is significantly facilitated by graphical proofs produced by LangPro.[6]

As a result, there were collected around 30 new rules where about a third of them are admissible ones; the new rules cover phenomena like noun and adverbial modifiers, prepositional phrases, passive constructions, expletive sentences, verb-particle constructions, auxiliaries, light verb constructions, etc. Most of the new rules are discussed in more details in (Abzianidze, 2015).

[5] SICK is partitioned in three parts (trail, train and test) and used as a benchmark for RTE14 (Marelli et al., 2014a). The data and the system results of RTE14 are available at http://alt.qcri.org/semeval2014/task1/

[6] Automating a tableau rule extraction is quite hard for the following reasons: it is unclear how to determine automatically whether a CCG derivation is wrong, a tableau rule is missing, or lexical knowledge is lacking; and the general format of a rule makes search procedure extremely inefficient.

ID	Gold/LP	Problem (premise ? conclusion)
3670	E/N	It is raining on a walking man ? A man is walking in the rain
219	E/N	There is no girl in white dancing ? A girl in white is dancing
5248	N/E	Someone is playing with a toad ? Someone is playing with a frog
8490	N/C	A man with a shirt is holding a football ? A man with no shirt is holding a football
7402	N/C	There is no man and child kayaking through gentle waters ? A man and a young boy are riding in a yellow kayak
1431	C/C	A man is playing a guitar ? A man is not playing a guitar
8913	N/C	A couple is not looking at a map ? A couple is looking at a map

Table 1: Problems from SICK-trial and SICK-train with gold and LangPro judgments.

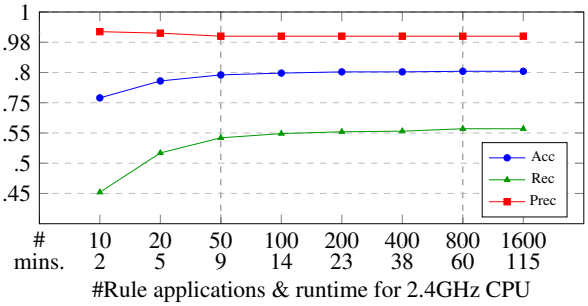


Figure 6: Performance of LangPro on SICK-train (4500) using CCG derivations of the C&C parser.

LangPro \ SICK	test (4927 problems)		
	Prec%	Rec%	Acc%
Baseline (majority)	-	-	56.36
+C&C+50	98.03	53.75	79.52
+EasyCCG+50	98.03	51.41	78.53
LangPro Hybrid-50	97.99	57.03	80.90
+C&C+800	97.99	54.73	79.93
+EasyCCG+800	98.00	52.67	79.05
LangPro Hybrid-800	97.95	58.11	**81.35**

Table 2: Evaluation of the versions of LangPro

LangPro was unable to prove several problems requiring complex background knowledge (e.g., SICK-3670 in Table 1) or having wrong CCG derivations from the C&C parser (e.g., in SICK-219, *white dancing* is a noun constituent).

5.2 Development

The part of the SICK data, SICK-train, issued for training at RTE14 was used for development. After running LangPro on SICK-train, we only analyzed false positives, i.e. neutral problems that were identified either as entailment or contradiction by the prover. The analysis reveals that the parsers and WordNet are responsible for almost all these errors. For example, in Table 1, SICK-5248 is classified as entailment since *toad* and *frog* might have synonymous senses; this problem shows the advantage of not using WSD, where a proof search also searches for word senses that might give rise to a logical relation. SICK-7402 was falsely identified as contradiction because of the wrong analyses of the premise by both CCG parsers: *no man* and *child...* are in coordination, which implies *there is no man*, and hence, contradicts the conclusion. SICK-8490 is proved as contradiction since the prover considers LLFs where *shirt* takes a wide scope. With the help of LangPro, we also identified inconsistency in the annotations of problems, e.g., SICK-1431, 8913 are similar problems but classified differently; it is also surprising that SICK-5248 is classified as neutral.

During this phase, also the effective (800) and efficient (50) upper bounds for the rule application number were determined (see Figure 6). Moreover, 97.4% of proofs found in 1600 rule applications are actually attainable in at most 50 rule applications; this shows that the rule application strategy of LangPro is quite efficient.

5.3 Evaluation

We evaluate LangPro on the unseen portion of the SICK data, SICK-test, which was used as a benchmark at RTE14; the data was also held out from the process of designing LLFgen. The prover classifies each SICK problem as follows:

input: (T, H);
try t = the first LLF of `llf`(T);
h = the first LLF of `llf`(H)
if no error **then**
case `tab`$\{t{:}\mathbb{T}, h{:}\mathbb{F}\}$, `tab`$\{t{:}\mathbb{T}, h{:}\mathbb{T}\}$
CLOSED, OPEN: classify as `ENTAILMENT`;
OPEN, CLOSED: classify as `CONTRADICTION`;
OPEN, OPEN: classify as `NEUTRAL`;
CLOSED, CLOSED: classify as `ENTAILMENT`; report it;
else classify as `NEUTRAL`; report it;

The results, in Table 2, show evaluation of LangPro on SICK-test using both parsers separately with the efficient and effective rule application upper bounds. Slightly better results with the C&C parser is explained by employing the parser in the learning phase. The difference of .5% in

System \ Measure+	Prec%	Rec%	Acc% (+LP)
Illinois-LH	81.56	**81.87**	**84.57** (+0.55)
ECNU	84.37	74.37	83.64 (+1.69)
UNAL-NLP	81.99	76.80	83.05 (+1.44)
SemantiKLUE	85.40	69.63	82.32 (+**2.78**)
The Meaning Factory	93.63	60.64	81.59 (+2.72)
LangPro Hybrid-800	**97.95**	58.11	81.35
UTexas	97.87	38.71	73.23 (+8.97)
Prob-FOL	-	-	76.52
Nutcracker	-	-	78.40
Baseline (majority)	-	-	56.69

Table 3: Comparing LangPro to the top or related RTE systems and combining their answers[7]

accuracy between the C&C-based and EasyCCG-based provers show that LLFgen was not fitted to the C&C parser's output during the learning phase.

In order to eliminate at some extent errors coming from the parsers, hybrid provers are designed that simply combine answers of two systems—if one of the systems proves a relation then it is an answer. Both hybrid versions of LangPro show more than 80% of accuracy while only 5 systems were able to do so at RTE14, where 77.1% was a median accuracy. The prover turns out to be extremely reliable with its state-of-the-art precision being almost 98%. A high precision is conditioned by the formal deductive proof nature of LangPro and by the sound rules it employs.

In Table 3, we compare the best version of hybrid LangPro to the top 5 systems of RTE14 on SICK-test and show the improvement it gives to each system when blindly adopting its positive answers (i.e. entailment and contradiction).

The decision procedure of the prover is completely rule-based and easy to comprehend since it follows the intuitive deductive rules. Tableaux proofs by LangPro for SICK-247 (in Figure 7) and SICK-2895 (in Figure 8) show step by step how T contradicts and entails, respectively, H.[8] Several new rules employed in these tableaux are given in Figure 9. Note that the both problems, SICK-247, 2895, were wrongly classified by all the top 7 systems of the RTE14. Taking into account that solving SICK-247 requires a sort of De Morgan's law for negation and disjunction, this demonstrates where LangPro, a purely logic-based system, outperforms non-logic-based systems.[9] The another problem, SICK-2895, is an evidence how unreliable the state-of-the-art and non-logic-based RTE systems might be since solving the problem only requires a lexical knowledge $barbell \leq weight$, which is available in WordNet.

6 Related Work

Using formal logic tools for a wide-coverage RTE task goes back to the Nutcracker system (Bos and Markert, 2005), where a wide-coverage semantic processing tool Boxer (Bos, 2008), in combination with the C&C tools, first produces discourse representation structures of (Kamp and Reyle, 1993) and then FOL semantic representations (Curran et al., 2007). Reasoning over FOL formulas is carried by off-the-shelf theorem provers and model builders for FOL.[10] Our approach differs from the latter in several main aspects: (i) the underling logic of LLFs (i.e. higher-order logic) is more expressive than FOL (e.g., it can properly model GQs and subsective adjectives), (ii) LLFs are cheap to get as they are easily obtained from CCG derivations, and (iii) we develop a completely new proof procedure and a prover for a version of Natural Logic.

The other related works are (MacCartney and Manning, 2008) and (Angeli and Manning, 2014). Both works contribute to Natural Logic and are based on the same methodology.[11] The approach has two main shortcomings compared to Natural Tableau; namely, it is unable to process multi-premised problems, and its underling logic is weaker (e.g., according to (MacCartney, 2009), it cannot capture the entailment in Figure 2).

[7]The top 5 systems of RTE14 are Illinois-LH (Lai and Hockenmaier, 2014), ECNU (Zhao et al., 2014), UNAL-NLP (Jimenez et al., 2014), SemantiKLUE (Proisl et al., 2014) and The Meaning Factory (Bjerva et al., 2014).

[8]In the tableaux, due to lack of space, several constants are denoted with initial characters of their lemmas and some intermediate nodes are omitted. Some of the nodes are annotated with a sequence of source rule applications.

[9]Even a shallow heuristic—if H has a named entity that does not appear in T, then there is no entailment—is not sufficient for showing that SICK-247 is contradiction. We thank our reviewer for mentioning this heuristic w.r.t. SICK-247.

[10]Nutcracker obtains 3% lower accuracy on SICK than our prover (Pavlick et al., 2015). The Meaning Factory (Bjerva et al., 2014) that is a brother system of Nutcracker, instead of solely relying on decisions of theorem provers and model builders, uses machine learning methods over the features extracted from these tools; this method results in a more robust system. RTE systems UTexas (Beltagy et al., 2014) and Prob-FOL (Beltagy and Erk, 2015) also use Boxer FOL representations but employ probabilistic FOL. For comparison purposes, the results of these systems on the SICK data are given in Table 3.

[11]They relate two sentences by a sequence of string edits; the final logical relation between the sentences is computed by composing logical relations associated with these edits.

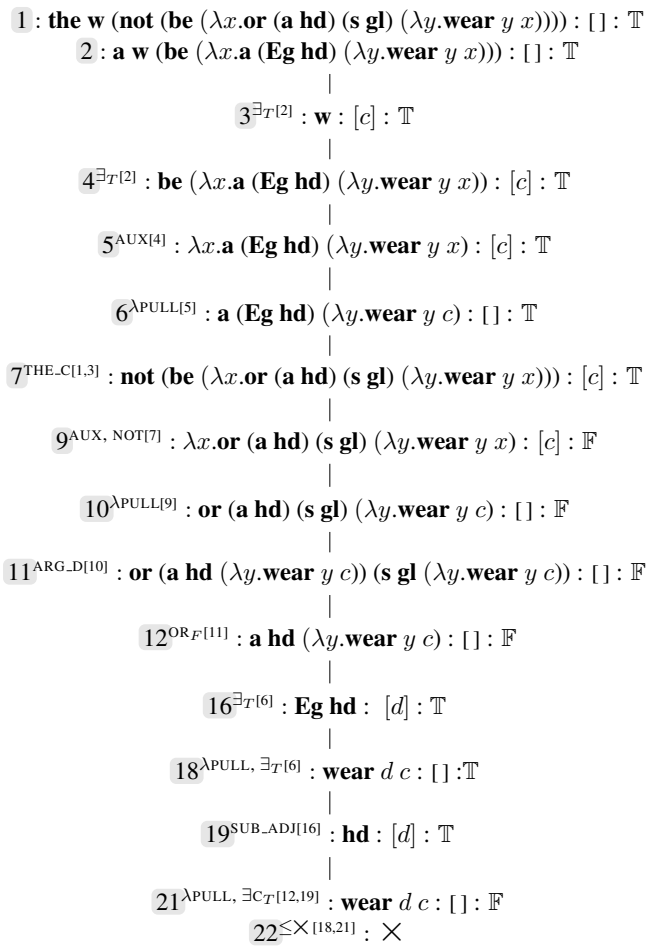


Figure 7: A closed tableau for SICK-247: *The woman is not wearing glasses or a headdress* $\perp$ *A woman is wearing an Egyptian headdress*

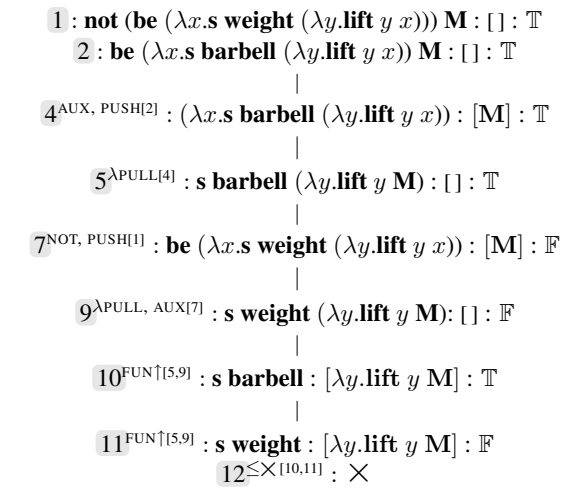


Figure 8: A closed tableau for SICK-2895: *The man isn't lifting weights* $\perp$ *The man is lifting barbells*, where M abbreviates a shared term **the man** aligned by the LLF-aligner.

$$\frac{G\ A : [\vec{C}] : \mathbb{T} \quad H\ A : [\vec{C}] : \mathbb{F}}{\begin{array}{l} G : [A, \vec{C}] : \mathbb{T} \\ H : [A, \vec{C}] : \mathbb{F} \end{array}}\text{ARG} \qquad \frac{F\ A : [\vec{C}] : \mathbb{T} \quad F\ B : [\vec{C}] : \mathbb{F}}{\begin{array}{l} A : [\vec{D}] : \mathbb{T} \\ B : [\vec{D}] : \mathbb{F} \\ \text{s.t. } \vec{D} \text{ is fresh and} \\ F \text{ is upward monotone} \end{array}}\text{FUN}\uparrow$$

$$\frac{\mathbf{the}\ N\ V : [\,] : \mathbb{X} \quad N : [c] : \mathbb{T}}{V : [c] : \mathbb{X}}\text{THE\_C} \qquad \frac{A\ N : [c] : \mathbb{T}}{\begin{array}{l} N : [c] : \mathbb{T} \\ \text{s.t. } A \text{ is subsective} \end{array}}\text{SUB\_ADJ}$$

$$\frac{X\ V : [\vec{C}] : \mathbb{X}}{\begin{array}{l} V : [\vec{C}] : \mathbb{X} \\ \text{s.t. } X \in \{\mathbf{be}, \mathbf{do}\} \end{array}}\text{AUX} \qquad \frac{\lambda x.\ A : [d\vec{C}] : \mathbb{X}}{A[x := d] : [\vec{C}] : \mathbb{X}}\lambda\text{PULL}$$

Figure 9: Several rules learned from SICK-trial

7 Conclusion and Future Work

We made Natural Tableau of Muskens (2010) suitable for the wide-coverage RTE task by extending it both in terms of rules and language. Based on the extended Natural Tableau, the prover LangPro was implemented, which has a modular architecture consisting of the inventory of rules, KB and the LLF generator. As a whole, the prover represents a deductive model of natural reasoning with the transparent and naturally interpretable decision procedure. While learning only from the SICK-trial data, LangPro showed the comparable accuracy and the state-of-the-art precision on the unseen SICK data.

For future work, we plan to explore the FraCaS (Consortium et al., 1996) and newswire RTE (Dagan et al., 2005) data sets to further improve the LLF generator and enrich the inventory of tableau rules. These tasks are also interesting for two reasons: to find out how much effort is required for adapting the LLF generator to different data, and which rules are to be added to the inventory for tackling the new RTE problems. Incorporating more WordNet relations (e.g., similarity, derivation and verb-group) and the paraphrase database (Ganitkevitch et al., 2013) in KB is also a part of our future plans.

Acknowledgments

I thank Reinhard Muskens for helpful discussions on this work, Jan Sprenger for his feedback on an early version of this paper, the authors of (Honnibal et al., 2010) for sharing a recent parser model, and anonymous reviewers for useful comments. This work is a part of the project "Towards Logics that Model Natural Reasoning" and supported by the NWO grant (project number 360-80-050).

References

Lasha Abzianidze. 2015. Towards a wide-coverage tableau method for natural logic. In Tsuyoshi Murata, Koji Mineshima, and Daisuke Bekki, editors, *New Frontiers in Artificial Intelligence*, volume 9067 of *Lecture Notes in Artificial Intelligence*, page Forthcoming. Springer-Verlag Berlin Heidelberg.

Gabor Angeli and Christopher D. Manning. 2014. Naturalli: Natural logic inference for common sense reasoning. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 534–545, Doha, Qatar, October. Association for Computational Linguistics.

Jon Barwise and Robin Cooper. 1981. Generalized quantifiers and natural language. *Linguistics and Philosophy*, 4(2):159–219.

Islam Beltagy and Katrin Erk. 2015. On the proper treatment of quantifiers in probabilistic logic semantics. In *Proceedings of the 11th International Conference on Computational Semantics (IWCS-2015)*, London, UK, April.

Islam Beltagy, Stephen Roller, Gemma Boleda, Katrin Erk, and Raymond Mooney. 2014. Utexas: Natural language semantics using distributional semantics and probabilistic logic. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 796–801, Dublin, Ireland, August. Association for Computational Linguistics and Dublin City University.

Evert W. Beth. 1955. Semantic Entailment and Formal Derivability. *Koninklijke Nederlandse Akademie van Wentenschappen, Proceedings of the Section of Sciences*, 18:309–342.

Johannes Bjerva, Johan Bos, Rob Van der Goot, and Malvina Nissim. 2014. The meaning factory: Formal semantics for recognizing textual entailment and determining semantic similarity. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 642–646, Dublin, Ireland.

Patrick Blackburn and Johan Bos. 2005. *Representation and Inference for Natural Language: A First Course in Computational Semantics*. CSLI Press.

Johan Bos and Katja Markert. 2005. Recognising textual entailment with logical inference. In *Proceedings of the 2005 Conference on Empirical Methods in Natural Language Processing (EMNLP 2005)*, pages 628–635.

Johan Bos. 2008. Wide-coverage semantic analysis with boxer. In Johan Bos and Rodolfo Delmonte, editors, *Semantics in Text Processing. STEP 2008 Conference Proceedings*, Research in Computational Semantics, pages 277–286. College Publications.

Johan Bos. 2009. Towards a large-scale formal semantic lexicon for text processing. In C. Chiarcos, R. Eckart de Castilho, and Manfred Stede, editors, *From Form to Meaning: Processing Texts Automatically. Proceedings of the Biennal GSCL Conference 2009*, pages 3–14.

Stephen Clark and James R. Curran. 2007. Wide-coverage efficient statistical parsing with ccg and log-linear models. *Computational Linguistics*, 33.

The Fracas Consortium, Robin Cooper, Dick Crouch, Jan Van Eijck, Chris Fox, Josef Van Genabith, Jan Jaspars, Hans Kamp, David Milward, Manfred Pinkal, Massimo Poesio, Steve Pulman, Ted Briscoe, Holger Maier, and Karsten Konrad. 1996. Using the framework.

James Curran, Stephen Clark, and Johan Bos. 2007. Linguistically motivated large-scale nlp with c&c and boxer. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 33–36, Prague, Czech Republic, June. Association for Computational Linguistics.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The pascal recognising textual entailment challenge. In *Proceedings of the PASCAL Challenges Workshop on Recognising Textual Entailment*.

Marcello D'Agostino, Dov M. Gabbay, Reiner Hhnle, and Joachim Posegga, editors. 1999. *Handbook of Tableau Methods*. Springer.

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

Melvin Fitting. 1990. *First-order Logic and Automated Theorem Proving*. Springer-Verlag New York, Inc., New York, NY, USA.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: The paraphrase database. In *Proceedings of NAACL-HLT*, pages 758–764, Atlanta, Georgia, June. Association for Computational Linguistics.

Julia Hockenmaier and Mark Steedman. 2007. Ccgbank: A corpus of ccg derivations and dependency structures extracted from the penn treebank. *Comput. Linguist.*, 33(3):355–396, September.

Matthew Honnibal, James R. Curran, and Johan Bos. 2010. Rebanking ccgbank for improved np interpretation. In *Proceedings of the 48th Meeting of the Association for Computational Linguistics (ACL 2010)*, pages 207–215, Uppsala, Sweden.

Sergio Jimenez, George Dueñas, Julia Baquero, and Alexander Gelbukh. 2014. Unal-nlp: Combining soft cardinality features for semantic textual similarity, relatedness and entailment. In *Proceedings of*

the 8th International Workshop on Semantic Evaluation (SemEval 2014), pages 732–742, Dublin, Ireland, August. Association for Computational Linguistics and Dublin City University.

Hans Kamp and Uwe Reyle. 1993. *From discourse to logic; an introduction to modeltheoretic semantics of natural language, formal logic and DRT*. Dordrecht: Kluwer.

Alice Lai and Julia Hockenmaier. 2014. Illinois-lh: A denotational and distributional approach to semantics. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 329–334, Dublin, Ireland, August. Association for Computational Linguistics and Dublin City University.

George Lakoff. 1972. Linguistics and natural logic. In Donald Davidson and Gilbert Harman, editors, *Semantics of Natural Language*, volume 40 of *Synthese Library*, pages 545–665. Springer Netherlands.

Mike Lewis and Mark Steedman. 2014. A* ccg parsing with a supertag-factored model. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 990–1000, Doha, Qatar, October. Association for Computational Linguistics.

Bill MacCartney and Christopher D. Manning. 2008. Modeling semantic containment and exclusion in natural language inference. In Donia Scott and Hans Uszkoreit, editors, *COLING*, pages 521–528.

Bill MacCartney. 2009. *Natural language inference*. Phd thesis, Stanford University.

Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, and Roberto Zamparelli. 2014a. Semeval-2014 task 1: Evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. In *Proceedings of SemEval 2014 (International Workshop on Semantic Evaluation)*, pages 1–8, East Stroudsburg PA. ACL.

Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, and Roberto Zamparelli. 2014b. A sick cure for the evaluation of compositional distributional semantic models. In Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Hrafn Loftsson, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, Reykjavik, Iceland. European Language Resources Association (ELRA).

Guido Minnen, John Carroll, and Darren Pearce. 2001. Applied morphological processing of english. *Natural Language Engineering*, 7(3):207–223, September.

Jeff Mitchell and Mirella Lapata. 2010. Composition in distributional models of semantics. *Cognitive Science*, 34(8):1388–1439.

Richard Montague. 1974. English as a Formal Language. In Richmond H. Thomason, editor, *Formal philosophy: Selected Papers of Richard Montague*, chapter 6, pages 188–221. Yale University Press.

Reinhard Muskens. 2010. An analytic tableau system for natural logic. In Maria Aloni, Harald Bastiaanse, Tikitu de Jager, and Katrin Schulz, editors, *Logic, Language and Meaning*, volume 6042 of *Lecture Notes in Computer Science*, pages 104–113. Springer Berlin Heidelberg.

Ellie Pavlick, Johan Bos, Malvina Nissim, Charley Beller, Benjamin Van Durme, and Chris Callison-Burch. 2015. Adding semantics to data-driven paraphrasing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL 2015)*, pages 1512–1522.

Thomas Proisl, Stefan Evert, Paul Greiner, and Besim Kabashi. 2014. Semantiklue: Robust semantic similarity at multiple levels using maximum weight matching. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 532–540, Dublin, Ireland, August. Association for Computational Linguistics and Dublin City University.

Mark Steedman and Jason Baldridge. 2011. Combinatory categorial grammar. In D. Borsley, Robert and Kersti Brjars, editors, *Non-Transformational Syntax: Formal and Explicit Models of Grammar*, pages 181–224. Wiley-Blackwell.

Mark Steedman. 2000. *The Syntactic Process*. MIT Press, Cambridge, MA, USA.

Jiang Zhao, Tiantian Zhu, and Man Lan. 2014. Ecnu: One stone two birds: Ensemble of heterogenous measures for semantic relatedness and textual entailment. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 271–277, Dublin, Ireland, August. Association for Computational Linguistics and Dublin City University.

JEAM: A Novel Model for Cross-Domain Sentiment Classification Based on Emotion Analysis

Kun-Hu Luo, Zhi-Hong Deng*, Liang-Chen Wei, Hongliang Yu
School of Electronic Engineering and Computer Science
Peking University, Beijing, China
{dr.tiger126@gmail.com, zhdeng@cis.pku.edu.cn, pkuhaywire@gmail.com,
yuhongliang324@gmail.com}

Abstract

Cross-domain sentiment classification (CSC) aims at learning a sentiment classifier for unlabeled data in the target domain based on the labeled data from a different source domain. Due to the differences of data distribution of two domains in terms of the raw features, the CSC problem is difficult and challenging. Previous researches mainly focused on concepts mining by clustering words across data domains, which ignored the importance of authors' emotion contained in data, or the different representations of the emotion between domains. In this paper, we propose a novel framework to solve the CSC problem, by modelling the emotion across domains. We first develop a probabilistic model named JEAM to model author's emotion state when writing. Then, an EM algorithm is introduced to solve the likelihood maximum problem and to obtain the latent emotion distribution of the author. Finally, a supervised learning method is utilized to assign the sentiment polarity to a given online review. Experiments show that our approach is effective and outperforms state-of-the-art approaches.

1 Introduction

Cross-domain sentiment classification (CSC) is the task that learns a sentiment classifier for unlabeled data in the target domain based on the labeled data from the source domain. With the increasing amount of opinion information available on the Internet, CSC has become a hot spot in recent years. Traditional machine learning algorithms often train a classifier utilizing the labeled data for CSC. However, in some practical cases, we may have many labeled data for some domains (source domains) but very few or no labeled data for other domains (target domains). Due to the differences of the distribution of two domains in terms of raw features, e.g. raw term frequency, the classifier trained from the source domain often performs badly on the target domain. To overcome this issue, several feature-based studies have been proposed to improve the sentiment classification domain adaptation [Zhuang et al., 2013; He et al., 2011; Gao and Li, 2011; Li et al., 2012; Dai et al., 2007; Zhuang et al., 2010; Pan et al., 2010; Wang et al., 2011; Long et al., 2012; Lin and He, 2009].

Existing studies build various generative models to solve the domain adaptation problems for CSC. In most cases, the models are trained by using the whole corpora without specifying on the sentiment of the texts. For example, [Zhuang et al., 2013] propose a general framework HIDC to mine high-level concepts (e.g. word clusters) across various domains. However, their learned concepts contain many topics not restricted to the sentiment. On the other hand, some researchers focus on the usage of the sentiment in CSC study [Mitra et al., 2013a; Mitra et al., 2013b; He et al., 2011]. [He et al., 2011] modify JST model [Lin and He, 2009] by incorporating word polarity priors through adjusting the topic-word Dirichlet priors. However, they fail to consider the expression differences among various domains.

To overcome the above issues, we employ "emotion", for its ubiquity among domains. The sentiment words in different domains might vary

* Corresponding author

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 2503–2508,
Lisbon, Portugal, 17-21 September 2015.

significantly, but the emotion can be effectively transferred. For example, when expressing the emotion "happiness", one uses "bravo" in the domain of sport, while "yummy" in the domain of food. Therefore, we propose an EA framework to model the latent emotions which are commonly contained in subjective articles and expressed by "emotional words". We infer the sentiment polarity of a document based on the emotion state. The hierarchy of EA is composed by four layers:

(1) Sentiment Layer

Normally, the sentiment of a document is the general opinion towards a certain event or object. For example, a movie review in IMDB might voice the feeling about the movie by a reviewer [Yu et al., 2013].

(2) Emotion Layer

Based on the emotion classification theories in psychology [Plutchik, 2002], the emotion can be classified into the basic ones influenced by the physiological factors, e.g. happiness, sadness, anger, etc., and dozens of complicated ones formed under some specific social conditions, e.g. shame, guilt, abashment, etc. Additionally, the emotion can be classified as positive and negative (similar to the sentiment classification) based on dimensional models of emotion [Schlosberg, 1954; Plutchik, 2002; Rubin and Talerico, 2009]. Intuitively, we assume that a document tends to contain the emotions of similar polarity.

(3) Lexicon Layer

To build the connection between words and the emotion, we introduce emotional words instead of raw word features into our model. By utilizing the emotional lexicon MPQA [Wiebe et al., 2005], we select groups of strong polar words, which get high scores in the emotional lexicon. These words are considered highly correlated to the certain emotion of the same polarity. And these strong polar words have invariant polarity across domains. Therefore, the emotion can be substantialized by a series of emotional words drawn from corresponding probability distribution.

(4) Expression Layer

In many practical cases, data come from different domains. We suppose that the correlation between emotion state and sentiment orientation is stable over domains, but one emotion may have different expressions when domain varies. E.g., "satisfaction" may be expressed as "interesting" or "attractive" for a book; meanwhile, it may be expressed "efficient" for an electronics device. Formally, we have

$$p(e|y,r1) = p(e|y,r2) = p(e|y) \quad (1)$$

$$p(w_e|e,r1) \neq p(w_e|e,r2) \quad (2)$$

where e denotes the emotion, y denotes the author's sentiment orientation, $r1$ and $r2$ denotes two different domains, and w_e denotes the emotional words.

Along this line, we propose the Joint Emotion Analysis Model (named JEAM for abbreviation) utilizing the probabilistic methods. See details in the next section.

2 Proposed Model

2.1 Problem Formulation

The CSC problem can be formulated as follows: Suppose we have two sets of data, denoted as D_s and D_t, which represent the source domain data and the target domain data respectively. In the CSC problem, the source domain data consist of labeled instances, denoted by $D_s = \{\left(x_i^{(s)}, y_i^{(s)}\right)\}|_{i=1}^{n_s}$, where $x_i^{(s)} \in \mathbb{R}^k$ is an input vector, $y_i^{(s)} \in \{0,1\}$ is the output label, and n_s is the number of documents in D_s. Unlike that of the source domain, the target domain data consists of samples without any label information, denoted by $D_t = \{x_i^{(t)}\}|_{i=1}^{n_t}$, where $x_i^{(t)} \in \mathbb{R}^k$ is an input vector, and n_t is the number of documents in D_t. The task of CSC is to leverage the training data of source domain D_s to predict the label $y_i^{(t)}$ corresponding to input vector $x_i^{(t)}$ of target domain D_t.

2.2 The JEAM Model

To model the author's emotion state contained in the document, we propose the JEAM model based on the probabilistic graphical principle. Note that all the factors and edges in JEAM are derived from the specific concepts and relations in EA, e.g., Eq(1) and Eq(2). We draw the graphical representation of JEAM in Figure 1, and show the notations of this paper in Table 1.

In Figure 1, y denotes the sentiment orientation of the author, which is a latent variable in this model. e denotes any emotion (topic) generated by y from a conditional probability $p(e|y)$. e is also a latent variable in this model. r denotes any data domain, e.g., books, dvd, kitchen, and electronics etc. d denotes any document chosen from domain r with label y. For documents from the source domain, the conditional probability $p(d|r,y)$ is known, which can be used to supervise the modeling process. u denotes the prior sentiment polarity of the corresponding emotional word. In practice, u can be obtained from the emotional lexicon,

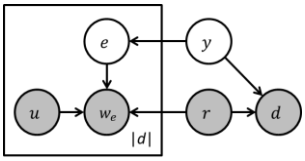


Figure 1. The Graphical representation of JEAM. All the latent variables are marked in white, and all the observed variables are marked in gray.

e	Emotion
w_e	Emotional word
r	Domain
d	Document
u	Prior sentiment polarity of the emotional word
y	Sentiment polarity of the document
X	All the observed variables
θ	All the model parameters

Table 1. Means of Symbols

which classifies a series of words into positive and negative categories. w_e denotes any emotional word with polarity u, which is chosen over words conditioned on emotion e and domain r from conditional probability $p(w_e|e,r,u)$. In this paper, we only select emotional words with strong sentiment polarities to represent the vector of the document. Therefore, we rebuild the data with the help of emotional lexicon cutting out the non-emotional words. As a result, any word chosen from the rebuilt data will be an emotional word, which is supposed not to change its polarity in different domains. Additionally, the joint probability over all the observed variables can be defined as follows based on the hidden variables:

$$p(w_e,d,r,u) = \sum_{e,y} p(e,y,w_e,d,r,u) \quad (3)$$

Based on the graphical model, we have:

$$p(e,y,w_e,d,r,u) = p(w_e|e,r,u)p(d|r,y)p(e|y)p(r)p(y)p(u) \quad (4)$$

We need to learn the unobservable probabilities (e.g., $p(w_e|e,r,u), p(d|r,y), p(e|y), p(y)$) to infer the hidden emotion distribution. Therefore, we develop an Expectation-Maximization (EM) algorithm to maximize the log likelihood of generating the whole dataset and obtain the iterative formula in E-step as follows:

$$p(e,y|w_e,d,r,u) = \frac{p(w_e|e,r,u)p(d|r,y)p(e|y)p(r)p(y)p(u)}{\sum_{e,y} p(w_e|e,r,u)p(d|r,y)p(e|y)p(r)p(y)p(u)} \quad (5)$$

where all the factors are calculated in M-step similar to PLSA and HIDC (Hoffman, 1999; Zhuang et al., 2013).

2.3 CSC via JEAM

To use JEAM to solve CSC problems, we adopt two optimizations:

First, we supervise the EM optimization with the polarity information of emotional words and instances respectively in the source domain. On the one hand, we estimate $p(e,y|w_e,d,r,u)$ utilizing the polarity label of the emotional words. Let the emotion set E be divided into positive set E_p and negative set E_n. We set $p(e_i,y|w_e,d,r,u_{w_e}) = 0$ during the whole EM process when the polarities of the emotion and current emotional word are different. On the other hand, we estimate the probability $p(d|r,y)$ with the label information of instances in the source domain. When the document is from the source domain, we set $p(d|r,y) = 0$ if y is different with the ground truth.

Second, we reconstruct the document as follows,

$$d^* = [e_1,e_2,\dots e_m], e_i = \begin{cases} 1 + \frac{|W_{e_i}^r|}{\sum_{j=1}^m |W_{e_j}^r|}, if\ W_{e_i}^r \neq \emptyset \\ 0, others \end{cases} \quad (6)$$

where $[e_1,e_2,\dots e_m]$ is the distribution over emotions, $W_{e_i}^r = \{w_e|e_{w_e}^r = i\}$, $e_{w_e}^r = argmax_e\, p(e|w_e,r)$, and $p(e|w_e,r)$ can be computed based on $p(w_e|e,r,u)$, $p(e|y), p(u)$ and $p(r)$ obtained after EM algorithm. The main function of this step is to process a new given document faster, avoiding training JEAM again with the new input. Finally, a machine learning method Support Vector Machine (SVM) is introduced to train a classifier with the labeled data from the source domain and assign polarities to documents from the target domain based on our reconstructed data.

3 Experiments

3.1 Experimental Setup

We demonstrate the effectiveness of JEAM on the Multi-Domain sentiment data set [Blitzer et al., 2007] which contains four types (domains) of real-world product documents taken from Amazon.com, which are books, dvd, electronics and kitchen. We randomly select 1800 documents from the one domain (source domain) and 200 documents from another domain (target domain). Then, we train a sentiment classifier using documents selected from the source domain and

assign labels to documents selected from the target domain, which generates 12 classification tasks. We preform 10 random selections and report the average results over 10 different runs. We use MPQA subjective lexicon [1] as the emotional lexicon. In our experiments, only strongly subjective clues are considered as emotional words, consisting of 1717 positive and 3621 negative words. We rebuild the dataset by cutting out the non-emotional words. For experiment parameters, we set $p = 25$, $n = 25$, and $T = 100$ after plenty of experiments. Considering the data in practice, the sentiment orientation y has only two forms, positive or negative. Note that we do neither instance selection nor complicated feature selection (only filter the low-frequency words) to our proposed method and other methods in comparison.

3.2 Experimental Result

Performance of Emotional Words

We show the effectiveness of introducing emotional words to solve the CSC problem. In JEAM, we reconstruct the documents by cutting out the non-emotional words. To compare the classification accuracy on the original documents and the reconstructed (emotional) documents, we choose two common classification algorithms, linear SVM and PLSA (topic size=10) for experiment respectively. The experiment results shows that both SVM and PLSA perform better on the emotional documents (60.43% and 60.48%) than on the original documents (57.73% and 56.69%) for the average accuracy over 12 classification tasks.

Effectiveness of using domain information and word polarity

We show the effectiveness of using domain information and word polarity, which are employed in our approach. For this purpose, we repeat the experiment without introducing domain and word polarity (node u and node r) into the model. Figure 2 shows the results. As it is clear, the highest performance can be achieved when domain information and word polarity are both used, while the lowest performance is obtained when neither of them is used.

Comparison with the Baselines

We compare our proposed approach with PLSA, SVM, SFA [Pan et al., 2010], JST [He et al., 2011] and HIDC [Zhuang et al., 2013]. The experimental results of the 12 classification tasks are shown in Figure 3. It can be observed that our proposed approach outperforms all the other approaches in general. Note that in order to obtain a more precise comparison of the algorithms, we do neither the instance selection nor the complicated feature selection. The result of our proposed approach can possibly be improved with the help of these selection strategies.

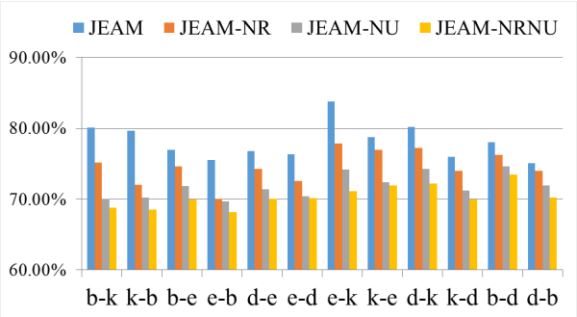


Figure 2: Effectiveness of using domain information and word polarity

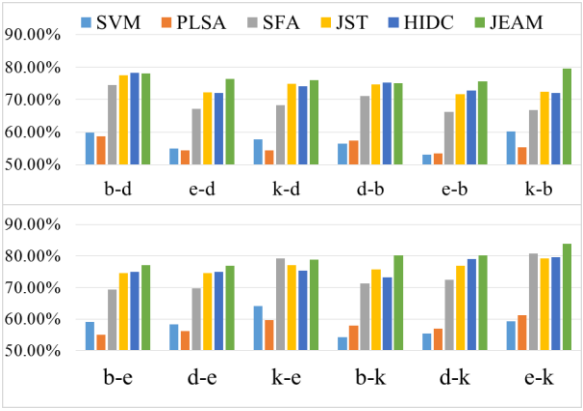


Figure 3: Comparison with the Baselines

4 Conclusion

In this paper, we propose a novel framework to solve the CSC problem, by modelling emotions across domains. We deeply analyze the relation between the author and the document based on the emotion theories in the field of psychology. Along this line, we propose a framework named EA, which takes the emotions and domains into account. Based on EA, we propose a novel model named JEAM to model the author's emotion state for Cross-domain sentiment classification. We conduct extensive experiments on real datasets to evaluate JEAM. The experiment results show that emotion plays an important role in CSC and JEAM outperforms existing state-of-the-art methods on the task of CSC.

[1] http://www.cs.pitt.edu/mpqa

5 Acknowledgement

This work is partially supported by Project 61170091 supported by National Natural Science Foundation of China and Project 2015AA015403 supported by the National High Technology Research and Development Program of China (863 Program). We would also like to thank the anonymous reviewers for their helpful comments.

References

John Blitzer, Mark Dredze, Fernando Pereira, et al.2007. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *ACL*, volume 7, pages 440–447.

Wenyuan Dai, Gui-Rong Xue, Qiang Yang, and Yong Yu. 2007. Co-clustering based classification for out-of-domain documents. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 210–219. ACM.

Hal Daum′e III. 2009. Frustratingly easy domain adaptation. *arXiv preprint arXiv:0907.1815*.

Sheng Gao and Haizhou Li. 2011. A cross-domain adaptation method for sentiment classification using probabilistic latent analysis. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 1047–1052. ACM.

Boqing Gong, Kristen Grauman, and Fei Sha. 2013. Connecting the dots with landmarks: Discriminatively learning domain-invariant features for unsupervised domain adaptation. In *Proceedings of The 30th International Conference on Machine Learning*, pages 222–230.

Yulan He, Chenghua Lin, and Harith Alani. 2011. Automatically extracting polarity-bearing topics for cross-domain sentiment classification. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 123–131. Association for Computational Linguistics.

Thomas Hofmann. 1999. Probabilistic latent semantic indexing. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 50–57. ACM.

Shoushan Li, Chu-Ren Huang, Guodong Zhou, and Sophia Yat Mei Lee. 2010. Employing personal/ impersonal views in supervised and semisupervised sentiment classification. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 414–423. Association for Computational Linguistics.

Fangtao Li, Sinno Jialin Pan, Ou Jin, Qiang Yang, and Xiaoyan Zhu. 2012. Cross-domain co-extraction of sentiment and topic lexicons. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 410–419. Association for Computational Linguistics.

Chenghua Lin and Yulan He. 2009. Joint sentiment/ topic model for sentiment analysis. In *Proceedings of the 18th ACMconference on Information and knowledge management*, pages 375–384. ACM.

Mingsheng Long, Jianmin Wang, Guiguang Ding, Wei Cheng, Xiang Zhang, and Wei Wang. 2012. Dual transfer learning. In *SDM*, pages 540–551. SIAM.

Mitra Mohtarami. Man lan, and chew lim tan. 2013a. from semantic to emotional space in probabilistic sense sentiment analysis. In *the 27th AAAI Conference on Artificial Intelligence*.

Sinno Jialin Pan, Xiaochuan Ni, Jian-Tao Sun, Qiang Yang, and Zheng Chen. 2010. Cross-domain sentiment classification via spectral feature alignment. In *Proceedings of the 19th international conference on World wide web*, pages 751–760. ACM.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Foundations and trends in information retrieval*, 2(1-2):1–135.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 79–86. Association for Computational Linguistics.

Robert Plutchik. 1980. *Emotion: A psychoevolutionary synthesis*. Harpercollins College Division.

Robert Plutchik. 2002. Emotion and life.

David C Rubin and Jennifer M Talarico. 2009. A comparison of dimensional models of emotion: Evidence from emotions, prototypical events, autobiographical memories, and words. *Memory*, 17(8):802–808.

Harold Schlosberg. 1954. Three dimensions of emotion. *Psychological review*, 61(2):81.

Hua Wang, Heng Huang, Feiping Nie, and Chris Ding. 2011. Cross-language web page classification via dual knowledge transfer using nonnegative matrix trifactorization. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 933–942. ACM.

Janyce Wiebe, Theresa Wilson, and Claire Cardie. 2005. Annotating expressions of opinions and emotions in language. *Language resources and evaluation*, 39(2-3):165–210.

Hongliang Yu, Zhi-Hong Deng, and Shiyingxue Li. 2013. Identifying sentiment words using an optimization-based model without seed words. In *ACL (2)*, pages 855–859.

Fuzhen Zhuang, Ping Luo, Peifeng Yin, Qing He, and Zhongzhi Shi. 2013. Concept learning for crossdomain text classification: A general probabilistic framework. In *Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*, pages 1960–1966. AAAI Press.

Fuzhen Zhuang, Ping Luo, Changying Du, Qing He, Zhongzhi Shi, and Hui Xiong. 2014. Triplex transfer learning: exploiting both shared and distinct concepts for text classification. *Cybernetics, IEEE Transactions on*, 44(7):1191–1203.

PhraseRNN: Phrase Recursive Neural Network for Aspect-based Sentiment Analysis

Thien Hai Nguyen **Kiyoaki Shirai**
School of Information Science
Japan Advanced Institute of Science and Technology
1-1 Asahidai, Nomi, Ishikawa 923-1292, Japan
{nhthien, kshirai}@jaist.ac.jp

Abstract

This paper presents a new method to identify sentiment of an aspect of an entity. It is an extension of RNN (Recursive Neural Network) that takes both dependency and constituent trees of a sentence into account. Results of an experiment show that our method significantly outperforms previous methods.

1 Introduction

Aspect-based sentiment analysis (ABSA) has been found to play a significant role in many applications such as opinion mining on product or restaurant reviews. It is a task to determine an attitude, opinion and emotions of people toward aspects in a sentence. For example, given a sentence "Except the design, the phone is bad for me", the system should classify positive and negative as the sentiments for the aspects 'design' and 'phone', respectively.

The simple approach is to calculate a sentiment score of a given aspect as the weighted sum of opinion scores, which are defined by a sentiment lexicon, of all words in the sentence (Liu and Zhang, 2012; Pang and Lee, 2008). This method is further improved by identifying the aspect-opinion relations using tree kernel method (Nguyen and Shirai, 2015a).

Other researches have attempted to use unsupervised topic modeling methods. To identify the sentiment category of the aspect, topic models which can simultaneously exploit aspect and sentiment have been proposed, such as TSLDA (Nguyen and Shirai, 2015b), ASUM (Jo and Oh, 2011), JST (Lin and He, 2009) and FACTS model (Lakkaraju et al., 2011).

Recursive Neural Network (RNN) is a kind of deep neural network. Using distributed representations of words (aka word embedding) (Bengio et al., 2003; Hinton, 1986), RNN merges word representations to represent phrases or sentences. It is one of the best methods to predict sentiment labels for the phrases (Socher et al., 2011; Socher et al., 2012; Socher et al., 2013). AdaRNN (Adaptive Recursive Neural Network) is an extension of RNN for Twitter sentiment classification (Dong et al., 2014a; Dong et al., 2014b).

This paper proposes a new method PhraseRNN for ABSA. It is an extended model of RNN and AdaRNN, which are briefly introduced in Section 2. The basic idea is to make the representation of the target aspect richer by using syntactic information from both the dependency and constituent trees of the sentence.

2 Recursive Neural Networks for ABSA

In RNN and AdaRNN, given a sentence containing a target aspect, "binary dependency tree" is built from a dependency tree of the sentence. Intuitively, it represents syntactic relations associated with the aspect. Each word (leaf) or phrase (internal node) in the binary dependency tree is represented as a d-dimensional vector. From bottom to up, the representations of a parent node v is calculated by combination of left and right child vector representations (v_l and v_r) using a global function g in RNN:

$$g(v_l, v_r) = W \begin{bmatrix} v_l \\ v_r \end{bmatrix} + b \quad (1)$$

where $W \in \Re^{d \times 2d}$ is the composition matrix and $b \in \Re^d$ is the bias vector. Then $v = f(g(v_l, v_r))$ where f is a nonlinear function such as $tanh$.

Instead of using only a global function g, AdaRNN employed n compositional functions $G = \{g_1, \cdots, g_n\}$ and selected them depending on the linguistic tags and combined vectors as follows:

$$v = f\left(\sum_{i=1}^{n} P(g_i|v_l, v_r, e) g_i(v_l, v_r)\right) \quad (2)$$

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 2509–2514,
Lisbon, Portugal, 17-21 September 2015.

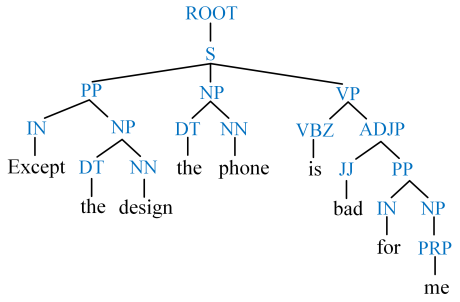


Figure 1: Example of a Constituent Tree

where $P(g_i|v_l, v_r, e)$ is the probability of function g_i given the child vectors v_l, v_r and external feature vector e. The probabilities are estimated as Equation (3).

$$\begin{bmatrix} P(g_1|v_l, v_r, e) \\ \cdots \\ P(g_n|v_l, v_r, e) \end{bmatrix} = softmax\left(\beta R \begin{bmatrix} v_l \\ v_r \\ e \end{bmatrix}\right) \quad (3)$$

where $\beta \in \Re$ is a hyper-parameter, and $R \in \Re^{n\times(2d+|e|)}$ is the parameter matrix.

The vector of the root node of the binary dependency tree is regarded as a representation of the target aspect. It is fed to a logistic regression to predict the sentiment category of the aspect.

3 PhraseRNN: Phrase Recursive Neural Network

In this model, a representation of an aspect will be obtained from a "target dependent binary phrase dependency tree" constructed by combining the constituent and dependency trees. In addition, instead of using a list of global functions G as in AdaRNN, two kinds of composition functions G in inner-phrase and H in outer-phrase are used.

3.1 Building Hierarchical Structure

First, the basic phrases (noun phrases, verb phrases, preposition phrases and so on) are extracted from the constituent tree of the sentence. For example, a list of phrases P = {PP[Except the design], NP[the phone], VP[is bad for me]} is extracted from the constituent tree in Figure 1.

Given a dependency tree and a list of phrases, a phrase dependency tree is created by Algorithm 1. The input is a dependency tree $T = (V, E)$ consisting of a set of vertices $V = \{v_1, \cdots, v_{|V|}\}$ and a set of relation edges $E = \{(r_{ji}, v_i, v_j)\}$ between two vertices, and a list of phrases $P = \{p_1, \cdots, p_K\}$ extracted from the constituent tree. The output is a phrase dependency tree $pT = (pV, pE)$ where $pV = \{T_1, \cdots, T_K\}$ ($T_i = (V_i, E_i)$ is a subtree) and $pE = \{(r_{ji}, T_i, T_j)\}$ (a set of relations between two subtrees). With the dependency tree and the phrase list in Figure 2(a), the algorithm will output a phrase dependency tree in Figure 2(b).

Algorithm 1: Convert to Phrase Dependency Tree

Input: dependency tree $T = (V, E)$, phrase list $P = \{p_1, \cdots, p_K\}$
Output: phrase dependency tree: $pT = (pV, pE)$ where $pV = \{T_1, \cdots, T_K\}, T_i = (V_i, E_i)$ and $pE = \{(r_{ji}, T_i, T_j)\}$

```
for each phrase p_i ∈ P do
    V_i ← {v_j | v_j ∈ p_i}
end
for each edge (r_nm, v_m, v_n) ∈ E do
    v_m ∈ p_k, v_n ∈ p_l
    if k = l then
        E_k ← E_k ∪ {(r_nm, v_m, v_n)}
    else
        pE ← pE ∪ {(r_nm, T_k, T_l)}
    end
end
```

The phrase dependency tree is transformed into a target dependent binary phrase dependency tree bpT by Algorithm 2. The input of the algorithm is a phrase dependency tree $pT = (pV, pE)$ and a target word v_t (the aspect word we want to predict the sentiment category). The output is the binary tree bpT. Note that the leaves of the binary tree bpT are binary subtrees $bT_1, \cdots, bT_K$ which are the binary versions of subtrees $T_1, \cdots, T_K$. On the other hand, the leaves of binary subtree bT_i are the words in phrase p_i. bpT and bT_i are obtained by *convert* function defined as Algorithm 3. It can convert an arbitrary tree to a binary tree [1]. Figure 2(c) and Figure 3 show the outputs for the aspect 'design' and 'phone', respectively.

3.2 Constructing the Aspect Representation

Each node in the binary tree is represented as a d-dimensional vector. In this research, we use the pre-trained Google News dataset [2] by word2vec algorithms (Mikolov et al., 2013). Each word is

[1] Note that *convert* function returns a tree represented by nested brackets such as [PP,[NP,VP]].

[2] https://code.google.com/p/word2vec/

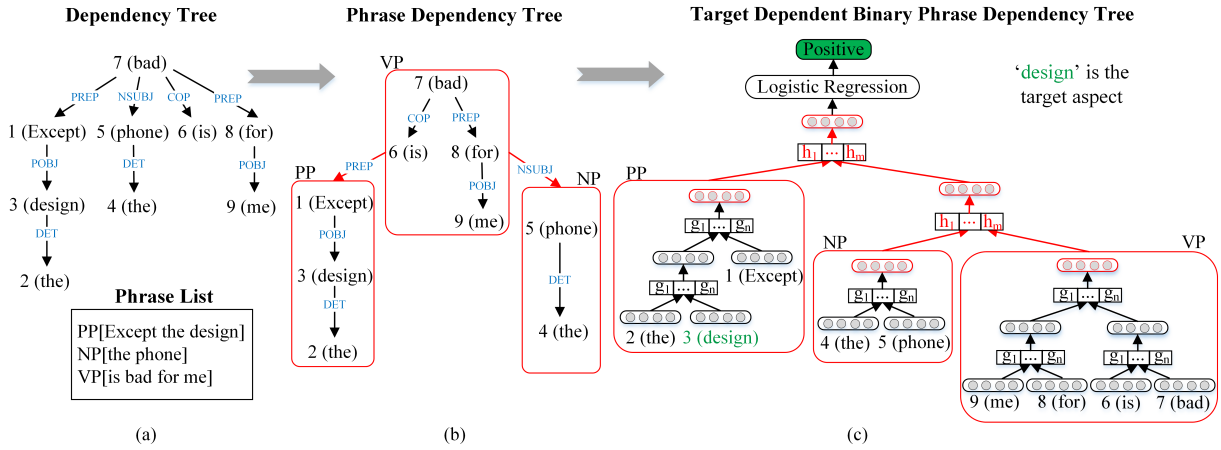


Figure 2: Hierarchical Structures in PhraseRNN: (a) Dependency Tree, (b) Phrase Dependency Tree and (c) Target Dependent Binary Phrase Dependency Tree

Algorithm 2: Convert to Target Dependent Binary Phrase Dependency Tree

Input: phrase dependency tree: $pT = (pV, pE)$, target v_t
Output: target dependent binary phrase dependency tree: bpT

for $T_i = (V_i, E_i) \in pV$ **do**
  if $v_t \in V_i$ **then**
    $h \leftarrow v_t$
  else
    $h \leftarrow$ vertex having no head in E_i
  end
  $bT_i \leftarrow convert(E_i, h)$
end
$T_{v_t} \leftarrow T_i$ that contains v_t
$bpT \leftarrow convert(pE, T_{v_t})$
Replace all T_i in bpT with bT_i

Algorithm 3: Convert to a Binary Tree

Function *convert*(E, v_t)**:**
  $v \leftarrow v_t$
  for $v_i \rightarrow v_t$, $v_t \rightarrow v_i$ *in* E **do**
    if $v_t \rightarrow v_i$ **then**
      $E' \leftarrow E \setminus \{v_t \rightarrow v_i\}$
      $w \leftarrow [convert(E, v_i), v]$
    else
      $E' \leftarrow E \setminus \{v_i \rightarrow v_t\}$
      $w \leftarrow [v, convert(E, v_i)]$
    end
    $v \leftarrow w$
  end
  return v
end

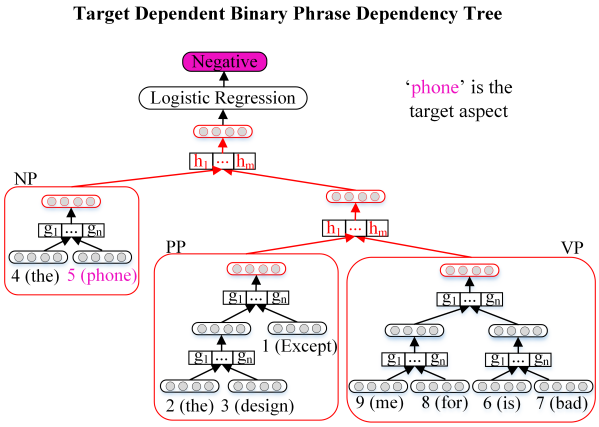


Figure 3: Another Target Dependent Binary Phrase Dependency Tree (Target Aspect 'phone')

represented as a 300-dimensional vector in this pre-trained dataset.

PhraseRNN uses two kinds of composition function $G = \{g_1, \cdots, g_n\}$ for inner-phrase and $H = \{h_1, \cdots, h_m\}$ for outer-phrase. n and m are the number of functions in G and H, respectively.

The vector of the parent node v_{in} in the binary subtree bT_i, where v_l and v_r are the vectors of the left and right children, is computed as:

$$v_{in} = f\left(\sum_{i=1}^{n} P(g_i|v_l, v_r, e_{in}) g_i(v_l, v_r)\right) \quad (4)$$

where e_{in} is the external feature vector. $P(g_i|v_l, v_r, e_{in})$ is the probability of function g_i given the child vectors v_l, v_r and e_{in}. It is

defined as Equation (5).

$$\begin{bmatrix} P(g_1|v_l, v_r, e_{in}) \\ \cdots \\ P(g_n|v_l, v_r, e_{in}) \end{bmatrix} = softmax\left(\beta R \begin{bmatrix} v_l \\ v_r \\ e_{in} \end{bmatrix}\right) \quad (5)$$

where $\beta \in \Re$ is a hyper-parameter, and $R \in \Re^{n\times(2d+|e_{in}|)}$ is the parameter matrix.

In the target dependent binary phrase dependency tree bpT, the vector of the parent node v_{out}, where the vectors of the left and right children are v_l and v_r, is computed as:

$$v_{out} = f\left(\sum_{i=1}^{m} P(h_i|v_l, v_r, e_{out}) h_i(v_l, v_r)\right) \quad (6)$$

$P(h_i|v_l, v_r, e_{out})$ is the probability of function h_i given the child vectors v_l, v_r and external feature vector e_{out} as shown in Equation (7).

$$\begin{bmatrix} P(h_1|v_l, v_r, e_{out}) \\ \cdots \\ P(h_m|v_l, v_r, e_{out}) \end{bmatrix} = softmax\left(\beta S \begin{bmatrix} v_l \\ v_r \\ e_{out} \end{bmatrix}\right) \quad (7)$$

where $S \in \Re^{m\times(2d+|e_{out}|)}$ is the parameter matrix.

The external features e_i (e_{in} and e_{out}) of the node v_i consists of three types of features: $Label_l$, $Label_r$ and $DepType_i$. $Label_l$ and $Label_r$ are the labels of the left and right child nodes, respectively. If node v_l is a leaf word, $Label_l$ is the POS of the word v_l. Otherwise, it is the non-terminal symbol of the lowest common parent of descendants of v_l in the constituent tree. For example, the $Label$ of the node combined from 'the' and 'design' in Figure 2(c) is 'NP' which is the lowest common parent of these two words in the constituent tree in Figure 1. $DepType_i$ is the dependency relation for node v_i. If the left and right children of v_i are leaf nodes, it is the direct relation in the dependency tree between them. Otherwise, $DepType_i$ is the relation between head words of the left and right nodes. For instance, in Figure 2(c), let a be the parent of 'is' and 'bad', b is the parent of 'for' and 'me', c is the parent of a and b. $DepType$ of a and b are 'COP' and 'POBJ' that are direct relations between two child nodes in the dependency tree in Figure 2(a). While, $DepType$ of c is 'PREP' that is the dependency relation between two head words 'bad' and 'for'. e_i is a binary vector where the weight of the vector represents the presence of each feature.

We suppose a batch training data consisting of B instances $\{(x^{(1)}, t^{(1)}), \cdots, (x^{(B)}, t^{(B)})\}$, where $x^{(b)}$ and $t^{(b)}$ are the aspect and its sentiment category of b-th instance. Let $y^{(b)}$ be the predicted sentiment category for aspect $x^{(b)}$ by PhraseRNN. The goal is to minimize the loss function which is the sum of the mean of negative log likelihood and L2 regularization penalty in a batch training set as in Equation (8).

$$L = -\frac{1}{B}\sum_{b=1}^{B} log(P(y^{(b)} = t^{(b)}|x^{(b)}, \theta)) + \lambda \sum_{\theta_i \in \theta} \| \theta_i \|^2 \quad (8)$$

where λ is a constant controlling the degree of penalty, θ is all the parameters in the model.

Stochastic gradient descent is used to optimize the loss function. Backpropagation is employed to propagate the errors from the top node to the leaf nodes. The derivatives of parameters are used to update the parameters.

4 Evaluation

We use the restaurant reviews dataset in SemEval2014 Task 4 consisting of over 3000 English sentences. For each aspect, "positive", "negative" or "neutral" is annotated as its polarity. Dataset is divided into three parts: 70% training, 10% development and 20% test.

We compare the following methods:

ASA w/o RE: It defines a sentiment score of a given aspect as the weighted sum of opinion scores of all words in the sentence, where the weight is defined by the distance from the aspect (Liu and Zhang, 2012; Pang and Lee, 2008).

ASA with RE: It improves "ASA w/o RE" by firstly identifying the aspect-opinion relations using tree kernel, then integrating them to the sentiment calculation (Nguyen and Shirai, 2015a).

RNN: It uses only one global function g_1 over the binary dependency tree.

AdaRNN: It uses multi-composition functions $G = \{g_1, \cdots, g_n\}$ over a binary dependency tree (Dong et al., 2014a).

PhraseRNN-1: our PhraseRNN with only one global function: $G = H = g_1$

PhraseRNN-2: our PhraseRNN with two global functions. One for inner-phrase, the other for outer-phrase: $G = g_1$ and $H = h_1$

PhraseRNN-3: our PhraseRNN with multiple global functions: $G = H = \{g_1, \cdots, g_n\}$

PhraseRNN-4: our PhraseRNN with two lists of global functions. One for inner-phrase, the other for outer-phrase: $G = \{g_1, \cdots, g_n\}$ and $H = \{h_1, \cdots, h_m\}$

Stanford CoreNLP (Manning et al., 2014) is used to parse the sentence and obtain constituent and dependency trees. For RNN, AdaRNN and PhraseRNN, the optimal parameters, which minimize the error in the development set, are used for the sentiment classification of the test set. We set $\beta = 1$ for AdaRNN and PhraseRNN since it is reported that $\beta = 1$ is the best parameter (Dong et al., 2014a). The optimized number of composition functions n and $m = \frac{n}{2}$ are selected by grid search with $n = \{2, 4, 6, 8, 10\}$ on the development set. $\lambda = 0.0001$ is employed. Accuracy (**A**), Precision (**P**), Recall (**R**) and F-measure (**F**) are used as evaluation metrics [3].

Table 1 shows the results of the methods. Differences of PhraseRNN and RNN are verified by statistical significance tests. We use the paired randomization test because it does not require additional assumption about distribution of outputs (Smucker et al., 2007). The results indicate that four variations of our PhraseRNN outperform "ASA w/o RE", "ASA with RE", RNN and AdaRNN methods from 5.35% to 19.44% accuracy and 8% to 16.48% F-measure. Among four variations, PhraseRNN-2 and PhraseRNN-3 achieved the best performance. By using different global functions in the inner and outer phrases, PhraseRNN-2 improves PhraseRNN-1 by 2.54% F-measure while keeping the comparable accuracy. Using multi-composition functions is also effective since PhraseRNN-3 was better than PhraseRNN-1 by 1.55% accuracy. PhraseRNN-4 improved PhraseRNN-3 by 6.38% precision while keeping comparable in other metrics.

Since our PhraseRNN-1 and PhraseRNN-3 outperform RNN and AdaRNN (the models relying on the binary dependency tree) respectively, we can conclude that our target dependent binary phrase dependency tree is much effective than binary dependency tree for ABSA.

In the data used in (Dong et al., 2014a), one sentence contains only one aspect. On the other hand, two or more aspects can be appeared in one sentence in SemEval 2014 data. It is common in the real text. To examine in which cases our method is better than the others, we conduct an additional experiment by dividing the test set into three disjoint subsets. The first subset (**S1**) contains sentences having only one aspect. The second subset (**S2**) and third subset (**S3**) have two or more aspects in each sentence. All aspects in a sentence in S2 have the same sentiment category, while different sentiment categories in S3. The number of aspects in S1, S2 and S3 are 200, 323 and 187, respectively.

[3] Precision, Recall and F-measure are the average for three polarity categories weighted by the number of true instances.

Table 1: Results of ABSA

Methods	A	P	R	F
ASA w/o RE	46.76	54.63	46.76	48.06
ASA with RE	52.39	53.91	52.39	52.54
RNN	60.85	53.59	60.85	54.21
AdaRNN	60.42	36.78	60.42	45.73
PhraseRNN-1	64.65†	58.59†	64.65†	59.67*
PhraseRNN-2	63.94†	**62.40***	63.94†	**62.21***
PhraseRNN-3	**66.20***	53.88	**66.20***	59.32*
PhraseRNN-4	65.92*	60.26†	65.92*	59.80*

Notes: Statistical significance test of PhraseRNN comparing to RNN.
* Significant at the 1 percent level.
† Significant at the 5 percent level.

Table 2: The Number of Correctly Identified Aspects in Subsets S1, S2 and S3

Methods	S1	S2	S3
ASA w/o RE	98 (49.00)	156 (48.30)	78 (41.71)
ASA with RE	111 (55.50)	176 (54.49)	85 (45.45)
RNN	123 (61.50)	226 (69.97)	83 (44.39)
AdaRNN	117 (58.50)	234 (72.45)	78 (41.71)
PhraseRNN-1	**129** (**64.50**)	248 (76.78)	82 (43.85)
PhraseRNN-2	125 (62.50)	247 (76.47)	82 (43.85)
PhraseRNN-3	125 (62.50)	**257** (**79.57**)	88 (47.06)
PhraseRNN-4	128 (64.00)	250 (77.40)	**90** (**48.13**)

Table 2 shows the number of aspects where their sentiments are correctly identified by the methods in the subsets S1, S2 and S3. The accuracies are also shown in parentheses. Among three subsets, S3 is the most difficult and ambiguous case. In all methods, the performance in S3 is worse than S1 and S2. Comparing with other methods in each subset, PhraseRNN improves the accuracy in S2 more than in S1 and S3.

5 Conclusion

We proposed PhraseRNN to identify the sentiment of the aspect in the sentence. Propagating the semantics through the binary dependency tree in RNN and AdaRNN could not be enough to represent the sentiment of the aspect. A new hierarchical structure was constructed by integrating the dependency relations and phrases. The results indicated that our PhraseRNN outperformed "ASA w/o RE", "ASA with RE", RNN and AdaRNN.

References

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *The Journal of Machine Learning Research (JMLR)*, 3:1137–1155.

Li Dong, Furu Wei, Chuanqi Tan, Duyu Tang, Ming Zhou, and Ke Xu. 2014a. Adaptive recursive neural network for target-dependent twitter sentiment classification. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 49–54.

Li Dong, Furu Wei, Ming Zhou, and Ke Xu. 2014b. Adaptive multi-compositionality for recursive neural models with applications to sentiment analysis. In *Twenty-Eighth AAAI Conference on Artificial Intelligence (AAAI)*, pages 1537–1543.

Geoffrey E Hinton. 1986. Learning distributed representations of concepts. In *Proceedings of the eighth annual conference of the cognitive science society*, volume 1, page 12. Amherst, MA.

Yohan Jo and Alice H Oh. 2011. Aspect and sentiment unification model for online review analysis. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 815–824. ACM.

Himabindu Lakkaraju, Chiranjib Bhattacharyya, Indrajit Bhattacharya, and Srujana Merugu. 2011. Exploiting coherence for the simultaneous discovery of latent facets and associated sentiments. In *Proceedings of the Eleventh SIAM International Conference on Data Mining (SDM)*, pages 498–509. SIAM / Omnipress.

Chenghua Lin and Yulan He. 2009. Joint sentiment/topic model for sentiment analysis. In *Proceedings of the 18th ACM conference on Information and knowledge management (CIKM)*, pages 375–384. ACM.

Bing Liu and Lei Zhang. 2012. A survey of opinion mining and sentiment analysis. In *Mining Text Data*, pages 415–463. Springer.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations (ACL)*, pages 55–60.

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

Thien Hai Nguyen and Kiyoaki Shirai. 2015a. Aspect-based sentiment analysis using tree kernel based relation extraction. In Alexander Gelbukh, editor, *Computational Linguistics and Intelligent Text Processing (CICLing)*, volume 9042 of *Lecture Notes in Computer Science*, pages 114–125. Springer International Publishing.

Thien Hai Nguyen and Kiyoaki Shirai. 2015b. Topic modeling based sentiment analysis on social media for stock market prediction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL), Volume 1: Long Papers*, pages 1354–1364. The Association for Computer Linguistics.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Foundations and trends in information retrieval*, 2(1-2):1–135.

Mark D Smucker, James Allan, and Ben Carterette. 2007. A comparison of statistical significance tests for information retrieval evaluation. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management (CIKM)*, pages 623–632. ACM.

Richard Socher, Cliff C Lin, Chris Manning, and Andrew Y Ng. 2011. Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of the 28th international conference on machine learning (ICML)*, pages 129–136.

Richard Socher, Brody Huval, Christopher D Manning, and Andrew Y Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 1201–1211. Association for Computational Linguistics.

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, volume 1631, page 1642. Citeseer.

ASTD: Arabic Sentiment Tweets Dataset

Mahmoud Nabil
Computer Engineering
Cairo University
Giza, Egypt
mah.nabil@cu.edu.eg

Mohamed Aly
Computer Engineering
Cairo University
Giza, Egypt
mohamed@mohamedaly.info

Amir F. Atiya
Computer Engineering
Cairo University
Giza, Egypt
amir@alumni.caltech.edu

Abstract

This paper introduces **ASTD**, an Arabic social sentiment analysis dataset gathered from Twitter. It consists of about 10,000 tweets which are classified as objective, subjective positive, subjective negative, and subjective mixed. We present the properties and the statistics of the dataset, and run experiments using standard partitioning of the dataset. Our experiments provide benchmark results for 4 way sentiment classification on the dataset.

1 Introduction

Arabic sentiment analysis work is gaining large attention nowadays. This is mainly due to the need of a product that can utilize natural language processing technology to track and analyze the public mood through processing social data streams. This calls for using standard social sentiment analysis datasets. In this work we present **ASTD** (**A**rabic **S**entiment **T**weets **D**ataset) an Arabic social sentiment analysis dataset gathered from Twitter. We discuss our method for gathering and annotating the dataset, and present its properties and statistics through the following tasks: (1) 4 way sentiment classification (2) Two stage class classification; and (3) sentiment lexicon generation. The contributions in this work can be summarized as:

1. We present an Arabic social dataset of about 10k tweets for subjectivity and sentiment analysis gathered from.
2. We investigate the properties and the statistics of the dataset and provide standard splits for balanced and unbalanced settings of the dataset.
3. We present a set of benchmark experiments to the dataset to establish a baseline for future comparisons.
4. We make the dataset and the used experiments publicly available[1].

2 Related Work

The detection of user sentiment in texts is a recent task in natural language processing. This task is gaining a large attention nowadays due to the explosion in the number of social media platforms and the number of people using them. Some Arabic sentiment datasets have been collected (see Table 1). (Abdul-Mageed et al., 2014) proposed the SAMAR system that perform subjectivity and sentiment analysis for Arabic social media where they used different multi-domain datasets collected from Wikipedia TalkPages, Twitter, and Arabic forums. (Aly and Atiya, 2013) proposed LABR, a book reviews dataset collected from GoodReads. (Rushdi-Saleh et al., 2011) presented an Arabic corpus of 500 movie reviews collected from different web pages. (Refaee and Rieser, 2014) presented a manually annotated Arabic social corpus of 8,868 Tweets and they discussed the method of collecting and annotating the corpus. (Abdul-Mageed and Diab, 2014) proposed SANA, a large-scale, multi-domain, and multi-genre Arabic sentiment lexicon. The lexicon automatically extends two manually collected lexicons HUDA (4,905 entries) and SIFFAT (3,325 entries). (Ibrahim et al., 2015) built a manual corpus of 1,000 tweets and 1000 microblogs and used it for sentiment analysis task. (ElSahar and El-Beltagy, 2015) introduced four datasets in their work to build a multi-domain Arabic resource (sentiment lexicon). (Nabil et al., 2014) and (El-Sahar and El-Beltagy, 2015) proposed a semi-supervised method for building a sentiment lexicon that can be used efficiently in sentiment analysis.

[1] https://github.com/mahmoudnabil/ASTD

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 2515–2519,
Lisbon, Portugal, 17-21 September 2015.

Data Set Name	Size	Source	Type	Cite
TAGREED (TGRD)	3,015	Tweets	MSA/Dialectal	(Abdul-Mageed et al., 2014)
TAHRIR (THR)	3,008	Wikipedia TalkPages	MSA	(Abdul-Mageed et al., 2014)
MONTADA (MONT)	3,097	Forums	MSA/Dialectal	(Abdul-Mageed et al., 2014)
OCA(Opinion Corpus for Arabic)	500	Movie reviews	Dialectal	(Rushdi-Saleh et al., 2011)
AWATIF	2,855	Wikipedia TalkPages/Forums	MSA/Dialectal	(Abdul-Mageed and Diab, 2012)
LABR(Large Scale Arabic Book Reviews)	63,257	GoodReads.com	MSA/Dialectal	(Aly and Atiya, 2013)
Hotel Reviews (HTL)	15,572	TripAdvisor.com	MSA/Dialectal	(ElSahar and El-Beltagy, 2015)
Restaurant Reviews (RES)	10,970	Qaym.com	MSA/Dialectal	(ElSahar and El-Beltagy, 2015)
Movie Reviews (MOV)	1,524	Elcinemas.com	MSA/Dialectal	(ElSahar and El-Beltagy, 2015)
Product Reviews (PROD)	4,272	Souq.com	MSA/Dialectal	(ElSahar and El-Beltagy, 2015)
Arabic Twitter Corpus	8,868	Tweets	MSA/Dialectal	(Refaee and Rieser, 2014)

Table 1: Arabic sentiment data sets

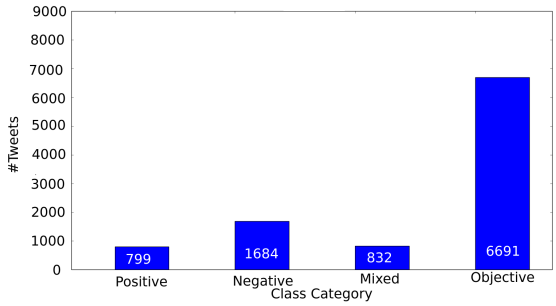


Figure 1: **Tweets Histogram**: The number of tweets for each class category. Notice the unbalance in the dataset, with much more objective tweets than positive, negative, or mixed.

Total Number of conflict free tweets	10,006
Subjective positive tweets	799
Subjective negative tweets	1,684
Subjective mixed tweets	832
Objective tweets	6,691

Table 2: Twitter dataset statistics

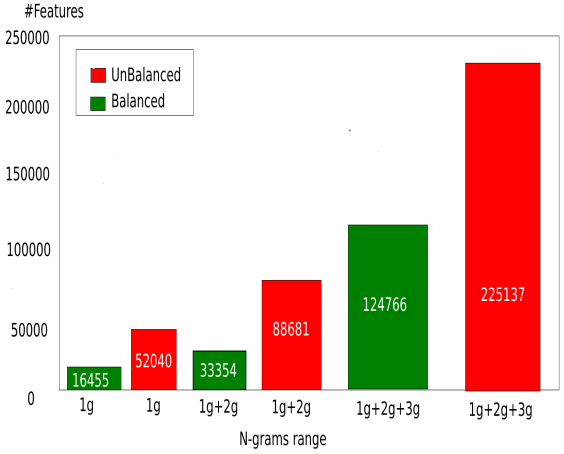


Figure 3: **Feature Counts.** Number of unigram, bigram, and trigram features per each class category.

3 Twitter Dataset

3.1 Dataset Collection

We have collected over 84,000 Arabic tweets. We downloaded the tweets over two stages: In the first stage we used SocialBakers [2] to determine the most active Egyptian Twitter accounts. This gave us a list of 30 names. We got the recent tweets of these accounts till November 2013, and this amounted to about 36,000. In the second stage we crawled EgyptTrends [3], a Twitter page for the top trending hash tags in Egypt. We got about 2500 distinct hash tags which are used again to download the tweets. We ended up obtaining about 48,000 tweets. After filtering out the non-Arabic tweets, and performing some pre-processing steps to clean up unwanted content like HTML, we ended up with 54,716 Arabic tweets.

3.2 Dataset Annotation

We used Amazon Mechanical Turk (**AMT**) service to manually annotate the data set through an API called Boto[4]. We used four tags: objective, subjective positive, subjective negative, and subjective mixed. The tweets that are assigned the same rating from at least two raters were considered as conflict free and are accepted for further processing. Other tweets that have conflict from all the three raters were ignored. We were able to label around 10k tweets. Table 2 summarizes the statistics for the conflict free ratings tweets.

3.3 Dataset Properties

The dataset has 10,006 tweets. Table 2 contains some statistics gathered from the dataset. The histogram of the class categories is shown in Fig. 1,

[2] http://www.socialbakers.com/twitter/country/egypt/

[3] https://twitter.com/EgyptTrends

[4] https://github.com/boto/boto

	Tweet	Translation	Rate
1	اكثر شعور يوجع ! ^ #لما تجوع في بيت مو بيتكم ☺**	Feeling that hurts ^ ! #To starve ina house not yours	Negative
2	محبين البرنامج يزيدوا :(	Fans of El-Bernameg are increasing :)	Positive
3	#كفاية اسفاف	#stop smallness	Negative
4	الطاقة البشرية اذا ما احسن استغلالها هى رصيدا و ليست عبئا قوتنا فى عددنا	Human energy if properly exploited is an asset and not a burden our strength in our numbers	Positive
5	احيي الشيخ حسن عبد البصير امام مسجد سيدى جابر الذي رفض تعليمات الأوقاف بنفاق مرسي في خطبة الجمعة تعلموا الاستقامة أيها #الاخوان الكاذبون	I greet Sheikh Hassan AbdelBassir Imam Sidi Gaber mosque,who refused the instructions of the endowments to hypocrite Morsi in his Friday sermon learn the integrity liars brotherhood	Mixed
6	هل تُتوج ايتكو مدريد بلقب الليجا الأحد القادم؟ #برشلونة	Is Atletico Madrid going to be crowned La Liga next Sunday? # Barcelona	Objective

Figure 2: **ASTD tweets examples**. The English translation is in the second column, the original Arabic review on the middle column, and the rating shown in right.

Number of tweets	10,006
Median tokens per tweet	16
Max tokens per tweet	45
Avg. tokens per tweet	16
Number of tokens	160,206
Number of vocabularies	38,743

Table 3: **Twitter Dataset Statistics**..

where we notice the unbalance in the dataset, with much more objective tweets than positive, negative, or mixed. Fig. 2 shows some examples from the data set, including positive, negative, mixed ,and objective tweets.

4 Dataset Experiments

In this work, we performed a standard partitioning to the dataset then we used it for the sentiment polarity classification problem using a wide range of standard classifiers to perform 4 way sentiment classification.

4.1 Data Preparation

We partitioned the data into training, validation and test sets. The validation set is used as a mini-test for evaluating and comparing models for possible inclusion into the final model. The ratio of the data among these three sets is 6:2:2 respectively.

Fig. 4 and Table 4 show the number of tweets for each class category in the training, test, and validation sets for both the balanced and unbalanced settings. Fig. 3 also shows the number of n-gram counts for both the balanced and unbalanced settings.

4.2 4 Way Sentiment Classification

We explore using the dataset for the same set of experiments presented in (Nabil et al., 2014) by applying a wide range of standard classifiers on the balanced and unbalanced settings of the dataset. The experiment is applied on both the token counts and the Tf-Idf (token frequency inverse document frequency) of the n-grams. Also we used the same accuracy measures for evaluating our results which are the weighted accuracy and the weighted F1 measure.

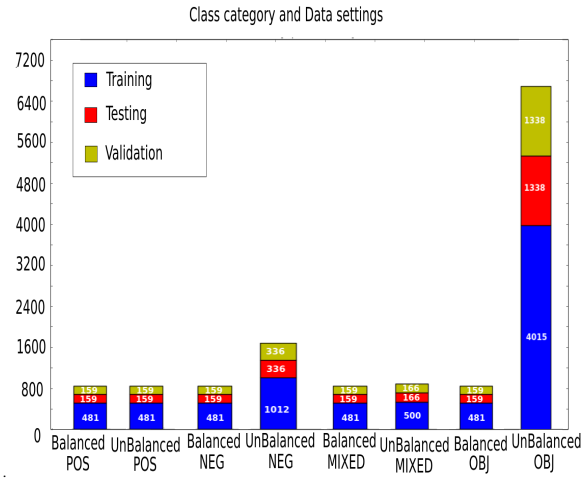


Figure 4: **Dataset Splits.** Number of tweets for each class category for training, validation, and test sets for both balanced and unbalanced settings.

Table 5 shows the result for each classifier after training on both the training and the validation set and evaluating the result on the test set (i.e. the train:test ratio is 8:2). Each cell has numbers that represent **weighted accuracy / F1 measure** where the evaluation is performed on the test set. All the experiments were implemented in Python using Scikit Learn[5]. Also the experiments were performed on a machine with Intel® Core™ i5-4440

[5] http://scikit-learn.org/

		Balanced				Unbalanced			
		Positive	Negative	Mixed	Objective	Positive	Negative	Mixed	Objective
Tweets Count	Train Set	481	481	481	481	481	1012	500	4015
	Test Set	159	159	159	159	159	336	166	1338
	Validation Set	159	159	159	159	159	336	166	1338
Features Count	unigrams	16,455				52,040			
	unigrams+bigrams	33,354				88,681			
	unigrams+bigrams+trigrams	124,766				225,137			

Table 4: **Dataset Preparation Statistics**. The top part shows the number of reviews for the training, validation, and test sets for each class category in both the balanced and unbalanced settings. The bottom part shows the number of features.

Features	Tf-Idf	Balanced			Unbalanced		
		1g	1g+2g	1g+2g+3g	1g	1g+2g	1g+2g+3g
MNB	No	0.467/0.470	0.487/0.491	**0.491/0.493**	0.686/0.604	0.684/0.590	0.682/0.584
	Yes	**0.481/0.484**	**0.491/0.492**	0.484/0.485	0.669/0.537	0.670/0.539	0.669/0.538
BNB	No	0.465/0.446	0.431/0.391	0.392/0.334	0.670/0.540	0.669/0.537	0.669/0.537
	Yes	0.289/0.184	0.255/0.110	0.253/0.107	0.669/0.537	0.669/0.537	0.669/0.537
SVM	No	0.425/0.421	0.443/0.440	0.431/0.425	0.644/0.611	0.679/0.625	0.679/0.616
	Yes	0.451/0.450	0.469/0.467	0.461/0.460	**0.687/0.620**	**0.689/0.624**	**0.691/0.626**
Passive Aggressive	No	0.421/0.422	0.447/0.443	0.439/0.435	0.639/0.609	0.664/0.621	0.671/0.616
	Yes	0.448/0.449	0.469/0.469	0.459/0.458	0.641/0.616	0.671/0.633	0.677/0.632
SGD	No	0.282/0.321	0.324/0.276	0.311/0.261	0.318/0.276	0.360/0.398	0.386/0.423
	Yes	0.340/0.295	0.409/0.382	0.415/0.388	0.664/0.557	0.671/0.557	0.669/0.551
Logistic Regression	No	0.451/0.447	0.448/0.444	0.440/0.435	0.682/0.621	0.694/0.620	0.693/0.614
	Yes	0.456/0.456	0.454/0.454	0.451/0.449	0.680/0.576	0.676/0.562	0.675/0.557
Linear Perceptron	No	0.395/0.399	0.428/0.426	0.429/0.425	0.480/0.517	0.656/0.622	0.649/0.618
	Yes	0.437/0.436	0.456/0.455	0.440/0.439	0.617/0.602	0.650/0.625	0.648/0.629
KNN	No	0.288/0.260	0.283/0.251	0.285/0.244	0.653/0.549	0.654/0.547	0.651/0.540
	Yes	0.371/0.370	0.406/0.406	0.409/0.409	0.665/0.606	0.663/0.611	0.666/0.615

Table 5: **Experiment 1: 4 way Classification Experimental Results**. *Tf-Idf* indicates whether tf-idf weighting was used or not. *MNB* is Multinomial Naive Bayes, *BNB* is Bernoulli Naive Bayes, *SVM* is the Support Vector Machine, SGD is the stochastic gradient descent and KNN is the K-nearest neighbor. The numbers represent weighted accuracy / F1 measure where the evaluation is performed on the test set. For example, 0.558/0.560 means a weighted accuracy of 0.558 and an F1 score of 0.560.

CPU @ 3.10GHz (4 cores) and 16GB of RAM.

From table 5 we can make the following observations:

1. The 4 way sentiment classification task is more challenging than the 3 way sentiment classification task. This is to be expected, since we are dealing with four classes in the former, as opposed to only three in the latter.

2. The balanced set is more challenging than the unbalanced set for the classification task. We believe that this because the the balanced set contains much fewer tweets compared to the unbalanced set. Since having fewer training examples create data sparsity for many n-grams and may therefore leads to less reliable classification.

3. SVM is the best classifier and this is consistent with previous results in (Aly and Atiya, 2013) suggesting that the SVM is reliable choice.

5 Conclusion and Future Work

In this paper we presented **ASTD** an Arabic social sentiment analysis dataset gathered from twitter. We presented our method of collecting and annotating the dataset. We investigated the properties and the statistics of the dataset and performed two set of benchmark experiments: (1) 4 way sentiment classification; (2) Two stage classification. Also we constructed a seed sentiment lexicon from the dataset. Our planned next steps include:

1. Increase the size of the dataset.

2. Discuss the issue of unbalanced dataset and text classification.

3. Extend the generated method either automated or manually.

Acknowledgments

This work has been funded by ITIDA's ITAC project number CFP-65.

References

Muhammad Abdul-Mageed and Mona T Diab. 2012. Awatif: A multi-genre corpus for modern standard arabic subjectivity and sentiment analysis. In *LREC*, pages 3907–3914.

Muhammad Abdul-Mageed and Mona Diab. 2014. Sana: A large scale multi-genre, multi-dialect lexicon for arabic subjectivity and sentiment analysis. In *Proceedings of the Language Resources and Evaluation Conference (LREC)*.

Muhammad Abdul-Mageed, Mona Diab, and Sandra Kübler. 2014. Samar: Subjectivity and sentiment analysis for arabic social media. *Computer Speech & Language*, 28(1):20–37.

Mohammed Aly and Amir Atiya. 2013. Labr: Large scale arabic book reviews dataset. In *Meetings of the Association for Computational Linguistics (ACL), Sofia, Bulgaria*.

Hady ElSahar and Samhaa R El-Beltagy. 2015. Building large arabic multi-domain resources for sentiment analysis. In *Computational Linguistics and Intelligent Text Processing*, pages 23–34. Springer.

Hossam S Ibrahim, Sherif M Abdou, and Mervat Gheith. 2015. Sentiment analysis for modern standard arabic and colloquial. *arXiv preprint arXiv:1505.03105*.

Mahmoud Nabil, Mohamed A. Aly, and Amir F. Atiya. 2014. LABR: A large scale arabic book reviews dataset. *CoRR*, abs/1411.6718.

Eshrag Refaee and Verena Rieser. 2014. An arabic twitter corpus for subjectivity and sentiment analysis. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC 14), Reykjavik, Iceland, may. European Language Resources Association (ELRA)*.

Mohammed Rushdi-Saleh, M Teresa Martín-Valdivia, L Alfonso Ureña-López, and José M Perea-Ortega. 2011. Oca: Opinion corpus for arabic. *Journal of the American Society for Information Science and Technology*, 62(10):2045–2054, October.

Adjective Intensity and Sentiment Analysis

Raksha Sharma, Mohit Gupta, Astha Agarwal, Pushpak Bhattacharyya
Dept. of Computer Science and Engineering
IIT Bombay, Mumbai, India
{raksha,mohitgupta,astha11,pb}@cse.iitb.ac.in

Abstract

For fine-grained sentiment analysis, we need to go beyond zero-one polarity and find a way to compare adjectives that share a common semantic property. In this paper, we present a semi-supervised approach to assign intensity levels to adjectives, *viz. high, medium* and *low*, where adjectives are compared when they belong to the same semantic category. For example, in the semantic category of EXPERTISE, *expert, experienced* and *familiar* are respectively of level *high, medium* and *low*. We obtain an overall accuracy of 77% for intensity assignment. We show the significance of considering intensity information of adjectives in predicting star-rating of reviews. Our intensity based prediction system results in an accuracy of 59% for a 5-star rated movie review corpus.

1 Introduction

Sentence intensity becomes crucial when we need to compare sentences having the same polarity orientation. In such scenarios, we can use intensity of words to judge the intensity of a sentence. Words that bear the same semantic property can be used interchangeably to upgrade or downgrade the intensity of the expression. For example, *good* and *outstanding* both are positive words from the QUALITY category, but the latter can be used to intensify positive expression in a sentence.

There are several manually or automatically created lexical resources (Liu, 2010; Wilson et al., 2005b; Wilson et al., 2005a; Taboada and Grieve, 2004) that assign a fixed positive (+1) or negative (−1) polarity to words, making no distinction among them in terms of their intensity. This paper presents a semi-supervised approach to assign intensity levels to adjectives, *viz. high, medium* and *low*, which share the same semantic property. We have used the semantic frames of FrameNet-1.5 (Baker et al., 1998) to obtain these semantic categories. Our approach is based on the idea that the most intense word has higher contextual similarity with *high* intensity words than with *medium* or *low* intensity words. We use the intensity annotated movie review corpus to obtain the most intense word for a semantic category. Then, cosine similarity between word vectors of the most intense word and other words of the category is used to assign intensity levels to those words. Our approach with the used resources is shown in figure 1.

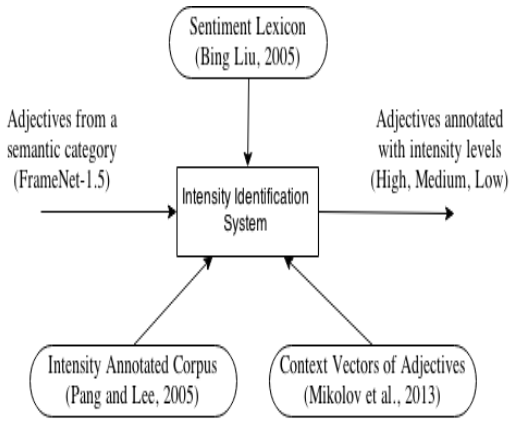


Figure 1: Intensity Analysis System

Our Contribution: Corpus based approaches suffer from the data sparsity problem. Our approach tackles this problem by using word vectors for intensity assignment (Section 2.3). It also provides a better overall accuracy (77%) than current state of the art when compared with gold-standard intensity levels (Section 6.2). In addition to this, we show that accuracy of the star rating prediction task improves when we incorporate our intensity levels as features in addition to standard features such as unigrams (Section 6.3).

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 2520–2526,
Lisbon, Portugal, 17-21 September 2015.

2 Idea Used for Deriving Adjectival Scale

In this paper, we dealt with 52 semantic (polar) categories of the FrameNet data and derived the polarity-intensity ordering among adjectives for each category. Examples of these semantic categories with a few words that belong to the category are as follows.

- INTELLIGENCE: *Brainy, brainless, intelligent, smart, dim etc*.
- CANDIDNESS: *Honest, dishonest, trustworthy, reliable, gullible etc*.
- EMOTION: *Sad, upset, appalled, tormented, gleeful, happy, pleased etc*.

Our algorithm to assign intensity levels to adjectives is based on the following ideas:

2.1 What Does Intensity Annotated Corpus Tell About The Intensity of Words?

Rill et al. (2012) showed that an intensity annotated polar corpus can be used to derive the intensity of the adjectives. A high intensity word will occur more frequently in high intensity reviews. For example, the word *excellent* is found 118 times, while *average* is found only 16 times in 5-star rated movie reviews (Section 3). Based on this distribution, we use a weighted mean formula to find intensity of the words from the corpus. We call it **Weighted Normalized Polarity Intensity (WNPI)** formula. For a 5-star intensity rating corpus, the $WNPI$ formula is as follows:

$$WNPI(word) = \frac{\sum_{i=1}^{5} i * C_i}{5 * \sum_{i=1}^{5} C_i} \quad (1)$$

where C_i is the count of the $word$ in i-star reviews.

2.2 Need Significant Occurrence of A Word

The $WNPI$ formula gives a corpus based result, hence can give biased scores for words which occur less frequently in the corpus. For example, in our movie review data-set, the word *substandard* occurs only 3 times in the corpus, and these occurrences happen to be in 1-star and 2-star reviews only. Hence, the $WNPI$ formula assigns a higher score to *substandard*. To avoid such a bias, we integrate $WNPI$ formula with Chi-Square test. Sharma and Bhattacharyya (2013) used Chi-Square test to find significant polar words in a domain. We use the same categorical Chi-Square test in our work.

2.3 How to Get Intensity Clue for All Words?

A combination of $WNPI$ formula and Chi-Square test cannot assign intensity scores to adjectives, which are not present in the corpus. To overcome this data sparsity problem, we restrict the use of $WNPI$ formula to identify the most intense word in each category. We explore precomputed context vectors of words, presented by Mikolov et al. (2011) (Section 3), to assign intensity levels to remaining words of the semantic category:

Case-1 Words which have less number of senses: These words will have a limited set of context words. Hence, their context vectors will also be based on these limited words. Example: *excellent, extraordinary, amazing, superb, great etc.*

Case-2 Words which have many senses: These words will have a large set of context words. Hence their context vectors will be based on a set of large number of words. Example: *good, fair, fine, average etc.*

Inferences:

1. Two words expressing similar meaning, and satisfying case-1 will have similar context. Hence, their word vectors will exhibit high cosine similarity. Whereas a word satisfying case-2 will be less similar to a word satisfying case-1.

2. The classical *semantic bleaching theory*[1] states that a word which has less number of senses (possibly one) tends to have higher intensity in comparison to a word having more senses. Considering *semantic bleaching* phenomenon as a base, we deduce that words which satisfy case-1 tend to be *high* intensity words while words satisfying case-2 are *low* intensity words.

Hence, we conclude that *high* intensity words (case-1) have higher *cosine similarity* with each other than with *low* or *medium* intensity words (case-2). Therefore, cosine similarity with a *high* intensity word can be used to obtain intensity ordering for remaining words of the category.

3 Data and Resources

This section gives an overview of the corpus and lexical resources used in our approach.

Semantic Categories: We worked with frames of FrameNet-1.5 (Baker et al., 1998). A frame

[1]The *semantic bleaching phenomenon* in words was reported in US edition of *New York Times*: `http://www.nytimes.com/2010/07/18/magazine/18onlanguage-anniversary.html?\_r=0`

Rating	Definition	Size
0	Totally painful, unbearable picture	179
1	Poor Show (dont waste your money)	1057
2	Average Movie	888
3	Excellent show, look for it	1977
4	A must see film	905

Table 1: Review ratings with their definitions and number of reviews.

represents a semantic property and contains words bearing the property. We explored the FrameNet data manually and found 52 frames (semantic categories) with polar semantic properties.

Intensity Annotated Corpus: To identify a *high* intensity word for a semantic category, we use a movie review corpus[2] (Pang and Lee, 2005) of 5006 files. Each review is rated on a scale of 0 to 4, where 0 indicates *an unbearable movie* and 4 represents *a must see film*. Table 1 describes the meanings of the rating scores with the count of reviews in each rating. We can infer that increase in rating corresponds to increase in positive intensity and decrease in negative intensity.

Sentiment Lexicon: To identify the polarity orientation of words, we use a list of positive (2006) and negative (4783) words[3] (Liu, 2010). We manually assign polarities to universally polar words like *enduring, creditable* and *nonsensical*, which are missing in this lexicon, using other standard lexicons. We found a total of 218 such missing words.

Context Vectors: We use the precomputed context vectors of words generated using Recurrent Neural Network Language Model (RNNLM) (Mikolov et al., 2013). The RNN is trained with $320M$ words from the broadcast news data.

4 Gold Standard Data Preparation

We asked five annotators to assign words to different intensity levels: *high, medium*, and *low*. Annotators were given positive and negative words of each category separately. The level chosen by a majority of annotators is selected as the gold standard intensity level for the word. To compute agreement among five annotators, we used *fleiss' kappa*, and obtained a score of **0.61**.

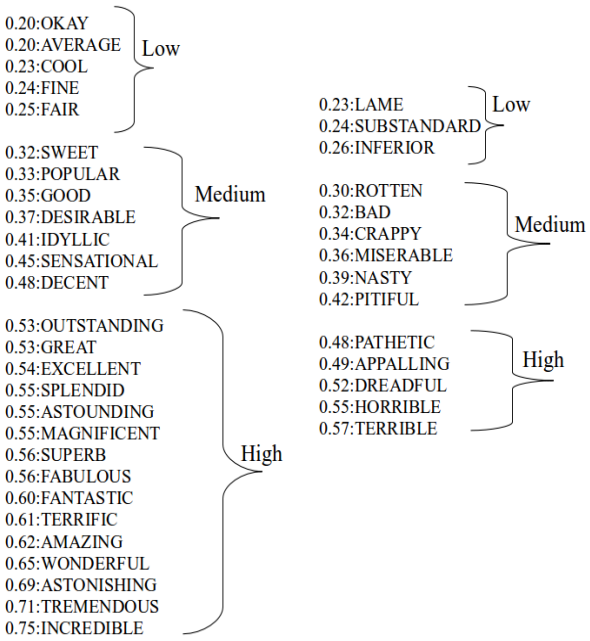


Figure 2: Intensity scale for QUALITY category, where *extraordinary* was found as *Pos-pivot* and *awful* as *Neg-pivot*.

5 Identification of Intensity of Adjectives

In this section, we give a step-by-step description of our approach.

Step 1: Find Intensity of Words

We calculate polarity-intensity of each word of a semantic category using $WNPI$ formula (eq. 1). Based on the polarity orientation of a word, the $WNPI$ formula uses intensity interpretation of star-rating as shown in table 2. The variable i of the $WNPI$ formula refers to these star ratings (intensity levels). The polarity orientation of an observed word is obtained using Bing Liu's lexicon.

Star-Rating / Word-Orientation	0	1	2	3	4
Positive	1	2	3	4	5
Negative	5	4	3	2	1

Table 2: Interpretation of star rating as intensity scores of reviews for positive and negative words.

Step 2: Find Pivot Using Chi-Square Test

The word which gives the highest Chi-Square

[2]Written and rated by four authorized movie critics. Available at: `http://www.cs.cornell.edu/people/pabo/movie-review-data/`

[3]Available at: `http://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html\#datasets`

score with the highest intensity score as per $WNPI$ is set as *pivot* (*Pos-pivot* and *Neg-pivot*). The Chi-Square test helps us to exclude the biased words, which are getting high intensity scores by the $WNPI$ formula, just by chance (Section 2.2).

Step 3: Obtain Similarity Scores with Pivot Further, we compute the *cosine similarity* between the context vectors of the pivot and the other words of the category. We use *Pos-pivot*, if the observed word is positive and *Neg-pivot*, if the observed word is negative.

Step 4: Assign Intensity Level to Words Finally, we arrange similarity scores obtained above in decreasing order, and place 2 break points in the sequence where consecutive similarity scores differ the most. We set these breakpoints as the thresholds for intensity levels.

Figure 2 shows the intensity scale obtained by our approach for the QUALITY category, where *extraordinary* was found as *Pos-pivot* and *awful* as *Neg-pivot*.

6 Experiments And Results

To evaluate the performance of our approach, we consider three measures: accuracy with the gold-standard data, comparison with state of the art and accuracy for the *star rating prediction task*.

6.1 Evaluation Using Gold Standard Data

We compute accuracy as the fraction of adjectives for which the predicted intensity level is the same as the gold standard level. We obtained an overall accuracy of 77% across 52 polar categories, containing a total of 697 adjectives.

6.2 Comparison with State of The Art

Ruppenhofer et al. (2014) showed that a corpus based method called MeanStar approach performs the best for intensity ordering task among existing approaches (De Melo and Bansal, 2013; Kim and de Marneffe, 2013; Fahrni and Klenner, 2008; Dragut et al., 2010) for polar semantic categories. Figure 3 shows the comparison between MeanStar and our approach for four semantic categories[4]. For first three categories, our approach performs better than MeanStar and for EXPERTISE we obtain the same level of accuracy. MeanStar approach gives an overall accuracy of 73% across 52 polar categories, which is significantly lesser than the accuracy obtained with our approach. MeanStar approach does not assign intensity score to words missing from the corpus. While, 88 out of 122 missing words are assigned correct intensity levels by our approach.

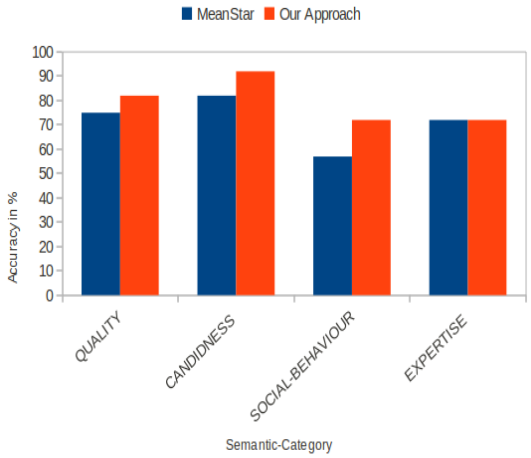


Figure 3: Accuracy obtained with MeanStar and our approach

6.3 Evaluation Using Star Rating Prediction

There have been several successful attempts at sentiment polarity detection in the past (Turney, 2002; Pang et al., 2002; Pang and Lee, 2004; Mohammad et al., 2013; Svetlana Kiritchenko and Mohammad, 2014). However, prediction of star ratings still considered as a challenging task (Qu et al., 2010; Gupta et al., 2010; Boteanu and Chernova, 2013). We implemented three systems to evaluate the significance of intensity annotated adjectives in star rating prediction task.

System 1: A rule based system based on the concept that *negatively high intense* words will occur more frequently in the *low star reviews* and *positively high intense* words will occur more frequently in the *high star reviews*. This system uses the following function I to assign intensity score to a review r:

$$I(r) = \frac{\sum_{i=1}^{3} i * C_i^P - \sum_{i=1}^{3} i * C_i^N}{3 * (\sum_{i=1}^{3} C_i^P + \sum_{i=1}^{3} C_i^N)} \quad (2)$$

where C_i^P and C_i^N respectively represent sum of the term-frequencies of positive and negative adjectives with intensity i.

Eq. 2 gives us an intensity score between -1 and $+1$ for each review. We need four breakpoints on these intensity scores to map intensity scores

[4]We have used the same semantic categories and intensity annotated movie review corpus in our work as used by Ruppenhofer et al. (2014).

into 5-star ratings. We learn these breakpoints by maximizing accuracy for the training data[5] over all possible breakpoints.

System 2: In this system, we consider intensity of each adjective as $+1$ or -1 as per its polarity, and then uses eq. 2 to find review intensity.

System 3: This is an SVM based system which uses four different types of features: (a) unigrams, (b) unigrams with the modification that if adjective belongs to our intensity annotated adjective list, then feature value is intensity of the adjective, (c) and (d) use the scores coming from eq. 2 as an additional feature over those in (a) and (b) respectively.

System	Accuracy(%)	MSE	MAE
1	42.28	0.94	0.69
2	27.33	1.12	0.86
3(a)	55.81	0.63	0.50
3(b)	57.21	0.56	0.47
3(c)	58.71	0.57	0.46
3(d)	**59.21**	**0.54**	**0.45**

Table 3: Comparison of rating prediction systems, where MSE is the Mean Squared Error and MAE is the Mean Absolute Error

Table 3 shows the results obtained with the above systems. System 3(d) achieves the maximum accuracy depicting that inclusion of intensity information with the standard features improves the star rating prediction significantly.

7 Related Work

Sentiment analysis on adjectives has been extensively explored in NLP literature. However, most of the works addressed the problem of finding polarity orientation of adjectives (Hatzivassiloglou and McKeown, 1997; Wiebe, 2000; Fahrni and Klenner, 2008; Dragut et al., 2010). The first work in the direction of adjectival scale was done by Hatzivassiloglou and McKeown (1993). They exploited linguistic knowledge available in the corpora to compute similarity between adjectives. However, their approach did not consider polarity orientation of adjectives, they provided ordering among non-polar adjectives like, *cold, lukewarm, warm, hot*.

[5]We use 80% of the star-rated movie review corpus as training data and 20% as test data. The results reported in table 3 are based on the 20% test data.

The task of ordering adjectives according to their polarity-intensity has recently received much attention due to the vital role of intensity analysis in several real world tasks. Kim et al. (2013) interpreted the continuous space word representation by demonstrating that vector off-set can be used to derive scalar relationship amongst adjectives. Their approach provided relationship among all the adjectives independent of their semantic property. De Melo and Bansal (2013) used a pattern based approach to identify intensity relation among adjectives, but their approach had a severe coverage problem. They also did not consider the semantic property of adjectives, assuming one single intensity-scale for all adjectives.

Ruppenhofer et al. (2014) provided ordering among polar adjectives that bear the same semantic property. Their approach was completely corpus dependent, it was not able to derive intensity of those adjectives which were not found in the corpus. We have used the same star-rated movie review corpus in our work as used by Ruppenhofer et al. (2014) and found 122 polar adjectives which are absent from the corpus. Our system is able to identify intensity levels for these missing adjectives. Moreover, we obtained an improvement of 4% in overall accuracy.

8 Conclusion

In this paper, we have proposed an approach that assigns intensity levels to domain independent adjectives, *viz. high, medium* and *low*. The important feature of our approach is that it is not fully corpus dependent, hence is able to assign intensity to adjectives that are absent in the corpus. We have reported that the overall results are better than the recently reported corpus based approach and fairly close to human agreement on this challenging task.

The use of adjectives with their intensity information can enrich existing sentiment analysis systems. We have shown the significance of considering intensity information of adjectives in predicting the intensity of movie reviews.

9 Acknowledgment

We heartily thank English linguists Rajita Shukla and Jaya Saraswati from CFILT Lab, IIT Bombay for giving their valuable contribution in gold standard data creation.

References

Collin F Baker, Charles J Fillmore, and John B Lowe. 1998. The berkeley framenet project. In *Proceedings of the 17th international conference on Computational linguistics-Volume 1*. Association for Computational Linguistics.

Adrian Boteanu and Sonia Chernova. 2013. Unsupervised rating prediction based on local and global semantic models. In *2013 AAAI Fall Symposium Series*.

Gerard De Melo and Mohit Bansal. 2013. Good, great, excellent: Global inference of semantic intensities. *Transactions of the Association for Computational Linguistics*, 1.

Eduard C Dragut, Clement Yu, Prasad Sistla, and Weiyi Meng. 2010. Construction of a sentimental word dictionary. In *Proceedings of the 19th ACM international conference on Information and knowledge management*. ACM.

Angela Fahrni and Manfred Klenner. 2008. Old wine or warm beer: Target-specific sentiment analysis of adjectives. In *Proc. of the Symposium on Affective Language in Human and Machine, AISB*.

Narendra Gupta, Giuseppe Di Fabbrizio, and Patrick Haffner. 2010. Capturing the stars: predicting ratings for service and product reviews. In *Proceedings of the NAACL HLT 2010 Workshop on Semantic Search*. Association for Computational Linguistics.

Vasileios Hatzivassiloglou and Kathleen R McKeown. 1993. Towards the automatic identification of adjectival scales: Clustering adjectives according to meaning. In *Proceedings of the 31st annual meeting on Association for Computational Linguistics*. Association for Computational Linguistics.

Vasileios Hatzivassiloglou and Kathleen R McKeown. 1997. Predicting the semantic orientation of adjectives. In *Proceedings of the 35th annual meeting of the association for computational linguistics and eighth conference of the european chapter of the association for computational linguistics*. Association for Computational Linguistics.

Joo-Kyung Kim and Marie-Catherine de Marneffe. 2013. Deriving adjectival scales from continuous space word representations. In *EMNLP*.

Bing Liu. 2010. Sentiment analysis and subjectivity. *Handbook of natural language processing*, 2.

Tomas Mikolov, Anoop Deoras, Daniel Povey, Lukas Burget, and Jan Cernocky. 2011. Strategies for training large scale neural network language models. In *Automatic Speech Recognition and Understanding (ASRU), 2011 IEEE Workshop on*. IEEE.

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

Saif M. Mohammad, Svetlana Kiritchenko, and Xiaodan Zhu. 2013. Nrc-canada: Building the state-of-the-art in sentiment analysis of tweets. In *Proceedings of the seventh international workshop on Semantic Evaluation Exercises*.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd annual meeting on Association for Computational Linguistics*, page 271. Association for Computational Linguistics.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*. Association for Computational Linguistics.

Lizhen Qu, Georgiana Ifrim, and Gerhard Weikum. 2010. The bag-of-opinions method for review rating prediction from sparse text patterns. In *Proceedings of the 23rd International Conference on Computational Linguistics*. Association for Computational Linguistics.

Sven Rill, Jörg Scheidt, Johannes Drescher, Oliver Schütz, Dirk Reinel, and Florian Wogenstein. 2012. A generic approach to generate opinion lists of phrases for opinion mining applications. In *Proceedings of the First International Workshop on Issues of Sentiment Discovery and Opinion Mining*. ACM.

Josef Ruppenhofer, Michael Wiegand, and Jasper Brandes. 2014. Comparing methods for deriving intensity scores for adjectives. *EACL 2014*, 117.

Raksha Sharma and Pushpak Bhattacharyya. 2013. Detecting domain dedicated polar words. *IJCNP 2013*, pages 661–666.

Xiaodan Zhu Svetlana Kiritchenko and Saif M. Mohammad. 2014. Sentiment analysis of short informal texts. 50.

Maite Taboada and Jack Grieve. 2004. Analyzing appraisal automatically. In *Proceedings of AAAI Spring Symposium on Exploring Attitude and Affect in Text (AAAI Technical Re# port SS# 04# 07), Stanford University, CA, pp. 158q161. AAAI Press*.

Peter D Turney. 2002. Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 417–424. Association for Computational Linguistics.

Janyce Wiebe. 2000. Learning subjective adjectives from corpora. In *AAAI/IAAI*.

Theresa Wilson, Paul Hoffmann, Swapna Somasundaran, Jason Kessler, Janyce Wiebe, Yejin Choi, Claire Cardie, Ellen Riloff, and Siddharth Patwardhan. 2005a. Opinionfinder: A system for subjectivity analysis. In *Proceedings of hlt/emnlp on interactive demonstrations*. Association for Computational Linguistics.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005b. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the conference on human language technology and empirical methods in natural language processing*. Association for Computational Linguistics.

The Rating Game: Sentiment Rating Reproducibility from Text

Lasse Borgholt, Peter Simonsen, Dirk Hovy
Center for Language Technology
University of Copenhagen
{fkr838|cbl123}@alumni.ku.dk, dirk.hovy@hum.ku.dk

Abstract

Sentiment analysis models often use ratings as labels, assuming that these ratings reflect the sentiment of the accompanying text. We investigate *(i)* whether human readers can infer ratings from review text, *(ii)* how human performance compares to a regression model, and *(iii)* whether model performance is affected by the rating "source" (i.e. original author vs. annotator). We collect IMDb movie reviews with author-provided ratings, and have them re-annotated by crowdsourced and trained annotators. Annotators reproduce the original ratings better than a model, but are still far off in more than 5% of the cases. Models trained on annotator-labels outperform those trained on author-labels, questioning the usefulness of author-rated reviews as training data for sentiment analysis.

1 Introduction

Machine learning approaches have become the dominant paradigm for sentiment analysis since introduced by Pang et al. (2002). While these approaches produce good results, they need to be trained on sufficiently large labeled data sets. Since human annotation can be both slow and expensive, many studies use data with an inherent subjectivity indicator, such as movie or product reviews with user ratings (Dave et al., 2003; Pang and Lee, 2005; Snyder and Barzilay, 2007; Elming et al., 2014, i.a.). While it is a fair assumption that the *rating* expresses the author's attitude towards the subject, it is less obvious to what extent the review *text* reflects this attitude, and hence what the relation between text and rating is. In this study, we ask

(i) whether readers are able to infer the author's numerical rating based on the author's review text,

(ii) how well learning algorithms perform on the task compared to human readers, and

(iii) whether model performance is affected by the rating source used for labeling (i.e. how the numerical rating is obtained) .

In order to investigate these questions, we compile a data set of user-generated movie reviews with *author ratings* and collect both *crowdsourced annotator ratings* and *trained annotator ratings*. This setup allows us to evaluate the reproducibility of ratings for both humans and models.

We address *(i)* by comparing author ratings to crowdsourced and trained annotator ratings. Author ratings supposedly capture the essence of the author's sentiment, but we do not expect annotators to perfectly reproduce these ratings based on text alone.

We investigate *(ii)* by evaluating a linear regression model on author-labeled data. Sentiment analysis models supposedly emulate the cognitive process of text-based rating inference. The gap between human and model performance is interesting, because if human annotators are unable to consistently infer author ratings, we cannot expect learning algorithms to achieve this goal.

Finally, we address *(iii)* by comparing regression models trained on data labeled with crowdsourced and author ratings. Existing work treats both labeling sources as ontologically interchangeable. That is, it does not matter whether a text was labeled by the author in the process of writing said text, or by an annotator who has been paid to label the text a posteriori. This is not at all self-evident.

To the best of our knowledge, no previous study has investigated the assumption that the sentiment of a text can be objectively inferred. Since sentiment analysis is still far from being solved, investigating this core bias can help address current limitations.

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 2527–2532,
Lisbon, Portugal, 17-21 September 2015.

2 Data

We collect 2,000 user-generated IMDb movie reviews and randomly sample 200 authors, each contributing 10 reviews of a length between 800 and 2,000 characters. All reviews are rated on a 10-point scale. Some authors mention their rating in the review text. This mention is of course an unwanted clue for the annotators, why we remove these reviews.

We pay annotators on CrowdFlower to rate the semantic orientation of reviews on a scale from 1 (negative) to 10 (positive). Each review is labeled by five experienced annotators. Overall, 171 annotators participated in the task. We incorporate control items in the annotation task, and each annotator starts by completing eight of these test questions. Further test questions are inserted randomly throughout the annotation tasks. We define a range of permitted ratings (within two steps of the original author rating). If annotators fall below an accuracy of 70%, they are removed from the project. Reviews used as test questions (10% of the initial data) are not part of the final data set.

We use three trained annotators to rate a 20% subset of the reviews: two authors of this study, and a student. All three annotate the full subset. We use stratified sampling to select the subset, considering each rating as a stratum. The distribution of author ratings in our subset thus matches the distribution of author ratings in the full data set. The subset contains 317 reviews, the full data set 1,629 reviews. Notice that only the subset is used to answer *(i)*, whereas the full data set is used for the regression-based tasks *(ii)* and *(iii)*.

3 Experiments

We want to establish the reproducibility of author ratings from text by human annotators and statistical models. In order to measure performance of the different methods, we use *mean absolute error (MAE)* and *root mean squared error (RMSE)*. While RMSE is more common, MAE is more directly interpretable, as it does not emphasize outliers. For this reason, we focus on MAE in our analysis.

MAE and RMSE measure the proximity between two sets of observations, but we also need a measure of the *relative* movement between observations. For this purpose, we use mainly Spearman's ρ, but also report Krippendorff's α and Cohen's κ. The latter is a standard agreement measure, but does not work as well for ordinal ratings such as these, since it assumes a uniform distribution to compute chance agreement.

We have two sources of human annotations, namely three trained annotators and five crowdsource annotators per review. In order to obtain our final ratings, we average over each of those annotation sources.[1] This result is more robust towards individual biases and misinterpretations. This effect is known as *wisdom of the crowd* and well-documented in the literature, e.g. Steyvers et al. (2009). However, we also wish to investigate how well individual annotators perform. Therefore, we also compute error and pairwise correlation for each individual annotator with the authors or other annotators, and then average over the pairwise comparisons for each annotator type.

This measure is equivalent to a *macro*-score and captures the average influence of individual annotators. When comparing across the two groups of annotators, we use all possible 3x5 combinations.

We use the same measures as outlined above to compare the different annotators to each other within the two groups. Hence, we compute both MAE, RMSE and correlation calculated between the individual crowdsource and trained annotators, respectively.

In order to control for different levels variance in the rating distributions, we *align* the crowdsource annotator and author distribution by sub-sampling. The number of reviews per rating is determined by the distribution with fewer reviews for the given rating. The resulting two data sets contain the same number of reviews per rating, and a total of 1,319 reviews. The main implication of aligning the distributions, is that variance for both distributions will be identical, thus making the comparison more appropriate.

3.1 Model

We use a linear least-squares model with L_2 regularization *(ridge regression)* to reduce overfitting.[2] L_2 imposes a term α, which penalizes the parameters w of the model if they grow too large. Formally, w can be calculated by

$$\min_w \|Xw - y\|_2{}^2 + \alpha \|w\|_2{}^2$$

We also experiment with incorporating a prior,

[1] Aggregating with an item-response model like MACE (Hovy et al., 2013) results in worse estimates, since it requires nominal data.

[2] Experimenting with support vector regression did not yield better results, so we chose the simpler model.

to model the tendency of authors to use the extremes more than predicted by a Gaussian distribution. We use a beta distribution with shape parameters $(0.8, 0.8)$.

We use 10-fold cross validation for robust results, and 5-fold cross validation on each of the then training folds in order to determine the optimal α.

We use bag-of-words features, including all unigrams appearing more than twice in the training data. [3]

4 Results

As baselines, we use the average rating over each of the *entire* rating distributions. Since the distributions differ between author and annotator ratings, the baseline differs from task to task.

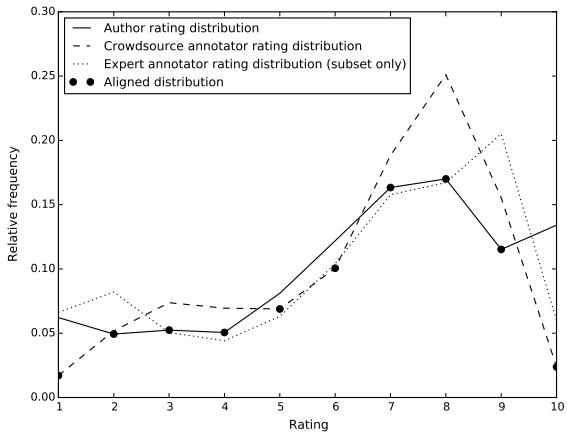


Figure 1. Rating distributions for authors, crowdsourced, and trained annotator ratings. Dots indicate aligned distribution.

Human Rating Inference *(i)* Figure 1 shows the rating distributions of the three human sources. Note the more peaked distributions of both annotator types, as compared to the author distribution. Especially crowdsource annotators have a smaller variance. Furthermore, the author distribution includes more extreme ratings, while the annotator distributions show no such "flaps".

[3]To rule out that the lack of any syntactic information (which human annotators use) disadvantages the model, we also experimented with including dependency triples (*dobj* and *nsubj*, the most frequent dependencies) using the Stanford Parser (Klein and Manning, 2003). However, performance did not improve, so due to limited space, we did not further explore this option.

Trained annotators are more correlated with one another ($\rho = 0.90, \alpha = 0.67, \kappa = 0.34$) than the crowdsource annotators ($\rho = 0.75, \alpha = 0.52, \kappa = 0.09$). Likewise, we see a lower MAE among trained annotators (0.75) than among crowdsource annotators (1.13), indicating a more diverse set of ratings for the latter.

		aut - cs	aut - tr	tr - cs
Corr. (ρ)	Mean	0.84	0.85	0.94
	Ind.	0.71	0.83	0.80
MAE	Mean	0.96	0.96	0.71
	Ind.	1.15	1.05	1.07
RMSE	Mean	1.31	1.30	1.01
	Ind.	1.61	1.44	1.48

Table 1. Pairwise comparisons between author (aut), trained (tr), and crowdsource (cs) ratings.

Table 1 compares the different rating sources. We find a higher correlation and lower error between the two sets of annotator ratings than between the author ratings and any of the annotator ratings. However, when comparing the individual rating correlations, author ratings are highly correlated with trained, but not with crowdsource annnotators, showing the uncertain nature of crowdsource annotators.

There is no discernible difference between the two annotator groups in terms of error margins. 80% of mean annotator ratings, regardless of source, are correctly inferred or one step off. Slightly more than 5% of ratings are more than two steps off. However, comparing *individual* annotator ratings instead of mean ratings, some crowdsource annotators are a full nine steps off, and in a single case, even one of the trained annotators was eight steps off.

	Ridge	+Prior	Aligned
Aut	1.66 / 2.14	1.70 / 2.21	1.52 / 1.95
Base	2.15 / 2.62	2.15 / 2.62	1.85 / 2.24
Ann	1.31 / 1.69	-	1.34 / 1.72
Base	1.85 / 2.23	-	1.85 / 2.24
Ann/Aut	1.60 / 2.05	-	-
Base	2.15 / 2.62	-	-

Table 2. MAEs/RMSEs for baselines and regressors trained and tested on Aut=authors; Ann=crowdsource annotators; Ann/Aut=trained on annotators and tested on authors.

In the next section, we compare the MAE be-

tween author ratings and the two sets of annotator ratings with the performance of the linear model. Conveniently, the numbers for the two types of annotators are equal (0.96), making it unnecessary to distinguish between them.

Human vs. Machine ***(ii)*** Table 2 shows the regression results. Since we want to compare the ability of people and regressors to infer *author ratings* from text, we only look at the *Author* row. Both the full rating distributions and the aligned distribution(s) are presented in Figure 1. All settings easily outperform the baseline.

The regression model achieves an MAE of 1.66, whereas both sets of human annotators achieve a MAE of 0.96 (see Table 1). This is an absolute difference of 0.70 in favor of the annotators. Or, in relative terms: the MAE of the learning algorithm is 72.6% larger than the human MAE.

Author vs. Annotator Labels ***(iii)*** In order to test whether the model is influenced by the label source, we compare the results in the *Author* and *Annotator* rows of Table 2. The regressor performs noticeably better on the annotator ratings than on the author ratings when using the full data set. Just as human annotators, the regressor under-estimates the extreme ratings (i.e., the "flaps"). Even incorporating a prior to address this shortcoming does not increase performance.

The performance difference between the models trained on the aligned distributions is smaller, but still noticeable. This is an important result, indicating that the model's performance drop when trained on authors is not solely due to the variance in the underlying distribution, but to the *quality* of the ratings.

The Ann/Aut row indicates that even if the goal is to predict author ratings, it could still be advantageous to train on annotator-labeled data.

5 Related Work

Since Pang et al. (2002) used author-labeled IMDb user reviews in their seminal study, author-labeled data has been used for a wide range of domains, like user-generated product reviews (Dave et al., 2003), restaurant reviews with several aspect ratings (Snyder and Barzilay, 2007), movie reviews from experienced film critics (Pang and Lee, 2005), business reviews (Hardt and Wulff, 2012; Elming et al., 2014; Hovy, 2015), and many more.

Pang and Lee (2005) also argue that it is unreasonable to expect a learning algorithm to predict ratings on a fine-grained scale if humans are not able to do so. To test this, they presented pairs of movie reviews from a single author rated on a 10-point Likert scale to two subjects (the authors themselves). Subjects had to decide whether one review was more, less, or equally positive than the other. Subjects correctly discerned reviews separated by more than three steps, but accuracy dropped when relative difference decreases. Pang and Lee (2005) also identify three obstacles for humans to accurately infer author ratings, namely *lack of calibration*, *author inconsistency* and *textually unsupported ratings*.

While suitable for their purposes, the study does not answer our research questions. First of all, the experiment is rather small (178 instances), which limits general validity and reliability. Second, the study tests the human ability to discern *relative*, not *absolute differences*. If two reviews rated 7 and 8 are judged a 3 and a 4, the *relative* difference will be correctly identified, even though the guess is far off in absolute terms. Furthermore, single-author reviews dilute the effects of the three aforementioned obstacles. Inconsistencies within a single author are undoubtedly smaller than inconsistencies between multiple authors. Single-author use also affects lack of calibration, since subjects can adjust to the writing style of one author better than that of several. Finally, we expect experienced authors to be less prone to producing reviews that do not support their ratings.

Annotator labels are typically used for phrase-level semantics (Wilson et al., 2005; Wiebe et al., 2005; Socher et al., 2013). Alternatively, labels can be induced from salient sentiment-related features like emoticons (Pak and Paroubek, 2010; Go et al., 2009; Tang et al., 2014) or hashtags (Kouloumpis et al., 2011). Often, the label source tends to be a matter of convenience, rather than theoretical reflection. The lack of considerations regarding potential differences between author and annotator labels implies that these are often perceived as ontologically equivalent. We do not believe this to be the case.

6 Discussion

Human rating inference ***(i)*** We observe some interesting differences between the three rating distributions. First, the "flaps" in the extreme

ratings in the author ratings are not present in the annotator rating distributions. This phenomenon might be explained by the observation that *"the propensity to post online reviews is higher for movies that are perceived by consumers to be exceptionally good or exceptionally bad"* (Dellarocas and Narayan, 2006). However, this tendency does not explain why the flaps are not present in the annotator distributions. One possible explanation is *risk aversion*. An annotator might estimate a review to be between 6 and 10. She might also estimate 10 to be the most likely rating and 6 the least. However, in order to minimize the margin of error, picking 8 is a better option than 6 or 10, since it will ensure the annotator is within two steps of the author's rating. This behavior is especially prevalent with crowdsourced annotators, who have a monetary incentive to minimize their error, which could explain the lack of flaps in *their* distribution. Indeed, the trained annotators show *some* evidence of flaps, but are still less extreme than the authors (i.e. their mean is closer to the center of the scale).

We also want to stress the role of *wisdom of the crowd*. Individual annotators perform worse (with regard to both correlation and MAE) than the mean over all annotators. This holds for both annotator types. Human ability to infer author ratings should thus be seen in light of these results. No individual annotator performed better than the mean of all annotators. The wisdom-of-the-crowd effect might also explain why crowdsource annotators perform as well as trained annotators: using five crowdsourced (vs. three trained) annotators provides more robust estimates to counter sloppy annotators.

We might expect a simple answer to our initial research question whether humans are able to infer author ratings. Of course, this is not the case. Most annotator ratings were within two steps of the original author rating. Only slightly more than 5% were further off. These results indicate that humans in *most* cases are able to infer the original author rating with decent accuracy, if allowed to "work together".

Human vs. Machine ***(ii)*** Based on our results, learning algorithms are *still* worse than humans in detecting semantic orientation of text. This difference holds even though humans, too, fail in a considerable number of cases. Overall, our results provide an upper bound for the performance we can expect from learning algorithms.

Author vs. Annotator Labels ***(iii)*** As hypothesized, using annotator labels lowered the MAE more than using author labels. Presumably, annotator labels follow a more regular, and thus predictable, pattern than author labels, since the former are generated by the reader's interaction with the text.

The aligned-distribution results support this theory. Aligning the distributions controls for different levels of rating variation in the distributions, thus ruling it out as confounder for the MAE difference. The aligned-distribution results also indicate that the model is biased towards mean ratings: MAE improves for author labels, since the relatively high variation is eliminated, but worsens for the annotator labels, as variance increases.

However, alignment also creates problems. First, the reviews contained in the author and annotator data sets differ in 18.6 % of the cases, although this should not be of significant advantage to either set. Second, aligned distributions do not evaluate the natural rating distributions. However, results follow the same trend as when using unmodified distributions (and hence the exact same reviews): annotator labels outperform author labels. All this suggests that annotator labels are more aligned with the text than author labels.

7 Conclusion

We find that readers infer author ratings from the review text fairly accurately (on average less than one step off on a 10-point scale). However, in more than 5% of the cases, the annotators were off by at least three points.

Human annotators outperform a linear regression model, even when adding a prior. We believe that no trivial adjustments can bridge this gap. However, the model achieves better results using annotator rather than author ratings, even when controlling for rating variance as a confounding factor. This suggests that author ratings are *not* optimal data labels for text-based sentiment analysis models.

Acknowledgements

The authors would like to thank the anonymous reviewers and the members of the CoAStAL group for their helpful comments. This research was funded in part by the ERC Starting Grant LOWLANDS No. 313695.

References

Kushal Dave, Steve Lawrence, and David M Pennock. 2003. Mining the peanut gallery: Opinion extraction and semantic classification of product reviews. In *Proceedings of the 12th international conference on World Wide Web*, pages 519–528. ACM.

Chrysanthos Dellarocas and Ritu Narayan. 2006. What motivates consumers to review a product online? a study of the product-specific antecedents of online movie reviews. In *WISE*.

Jakob Elming, Barbara Plank, and Dirk Hovy. 2014. Robust cross-domain sentiment analysis for low-resource languages. In *Proceedings of the 5th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 2–7, Baltimore, Maryland, June. Association for Computational Linguistics.

Alec Go, Richa Bhayani, and Lei Huang. 2009. Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford*, pages 1–12.

Daniel Hardt and Julie Wulff. 2012. What is the meaning of 5*s? an investigation of the expression and rating of sentiment. In *Proceedings of EMNLP*, pages 319–326.

Dirk Hovy, Taylor Berg-Kirkpatrick, Ashish Vaswani, and Eduard Hovy. 2013. Learning whom to trust with MACE. In *NAACL*.

Dirk Hovy. 2015. Demographic factors improve classification performance. In *Proceedings of ACL*.

Dan Klein and Christopher D Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 423–430. Association for Computational Linguistics.

Efthymios Kouloumpis, Theresa Wilson, and Johanna Moore. 2011. Twitter sentiment analysis: The good the bad and the omg! *ICWSM*, 11:538–541.

Alexander Pak and Patrick Paroubek. 2010. Twitter as a corpus for sentiment analysis and opinion mining. In *LREC*, volume 10, pages 1320–1326.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 115–124. Association for Computational Linguistics.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 79–86. Association for Computational Linguistics.

Benjamin Snyder and Regina Barzilay. 2007. Multiple aspect ranking using the good grief algorithm. In *HLT-NAACL*, pages 300–307.

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, volume 1631, page 1642. Citeseer.

Mark Steyvers, Brent Miller, Pernille Hemmer, and Michael D Lee. 2009. The wisdom of crowds in the recollection of order information. In *Advances in neural information processing systems*, pages 1785–1793.

Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. 2014. Learning sentiment-specific word embedding for twitter sentiment classification. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 1555–1565.

Janyce Wiebe, Theresa Wilson, and Claire Cardie. 2005. Annotating expressions of opinions and emotions in language. *Language resources and evaluation*, 39(2-3):165–210.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the conference on human language technology and empirical methods in natural language processing*, pages 347–354. Association for Computational Linguistics.

A Multi-lingual Annotated Dataset for Aspect-Oriented Opinion Mining

Salud María Jiménez Zafra[1], **Giacomo Berardi**[2], **Andrea Esuli**[2], **Diego Marcheggiani**[2], **María Teresa Martín-Valdivia**[1], **Alejandro Moreo Fernández**[2]
[1]Departamento de Informática, Escuela Politécnica Superior de Jaén
Universidad de Jaén, E-23071 - Jaén, Spain
{sjzafra, maite}@ujaen.es
[2]Istituto di Scienza e Tecnologie dell'Informazione "A. Faedo"
Consiglio Nazionale delle Ricerche, I-56124 - Pisa, Italy
{firstname.lastname}@isti.cnr.it

Abstract

We present the Trip-MAML dataset, a Multi-Lingual dataset of hotel reviews that have been manually annotated at the sentence-level with Multi-Aspect sentiment labels. This dataset has been built as an extension of an existent English-only dataset, adding documents written in Italian and Spanish. We detail the dataset construction process, covering the data gathering, selection, and annotation. We present inter-annotator agreement figures and baseline experimental results, comparing the three languages. Trip-MAML is a multi-lingual dataset for aspect-oriented opinion mining that enables researchers (i) to face the problem on languages other than English and (ii) to the experiment the application of cross-lingual learning methods to the task.

1 Introduction

Reviews of products and services that are spontaneously produced by customers represent a source of unquestionable value not only for marketing strategies of private companies and organizations, but also for other users since their purchasing decisions are likely influenced by other customers' opinions (Chevalier and Mayzlin, 2006).

Overall ratings (e.g., in terms of a five stars rating scale), and also aspect-specific ratings (e.g., the Cleanliness or Location of a hotel), are the typical additional information expressed by customers in their reviews. Those ratings help to derive a number of global scores to facilitate a first screening of the product or service at hand. Notwithstanding, users who pay more attention to a particular aspect (e.g., the Rooms of a hotel) remain constrained to manually inspect the entire text of reviews in order to find out the reasons other users argued in that respect. Methods for automatic analysis of the aspect-oriented sentiment expressed in reviews would enable highlighting aspect-relevant parts of the document, so as to allow users to perform a faster and focused inspection of them.

Previous work on opinion mining (Pang and Lee, 2008) has already faced the overall sentiment prediction (Pang et al., 2002), multiple aspect-oriented analysis (Hu and Liu, 2004), and fine-grained phrase-level analysis (Wilson et al., 2009). Most of the available opinion mining datasets contain only documents written in English, as this language is the most used on the Internet and the one for which more NLP tools and resources are available. (Hu and Liu, 2004) worked on the summarization of reviews by means of weakly supervised feature mining. (Täckström and McDonald, 2011) used a finer-grained dataset in which global polarity annotation is applied also to each sentence composing the document. Similarly did (Socher et al., 2013) with the Stanford Sentiment Treebank, which annotates each syntactically plausible phrase in thousands of sentences using annotators from Amazon's Mechanical Turk, annotating the polarity of phrases on a five-level scale. (Lazaridou et al., 2013) performed a single-label polarity annotation of elementary discourse units of TripAdvisor reviews, adopting ten aspect labels. (Marcheggiani et al., 2014) did a similar annotation work, using sentences as the annotation elements and adopting a multi-label polarity annotation, i.e., each sentence can be assigned to zero, one, or more than one aspect.

Cross-lingual sentiment classification (Wan, 2009; Prettenhofer and Stein, 2011) explores the scenario in which training data are available for

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 2533–2538,
Lisbon, Portugal, 17-21 September 2015.

a language that is different from the language of the test documents. Cross-lingual learning methods have important practical applications, since they allow to build classifiers for many languages reusing the training data produced for a single language (typically English), probably giving up a bit of accuracy, but compensating it with a large save in terms of human annotation costs.

Multi-lingual datasets are beneficial to the research community both as a benchmark to explore cross-lingual learning and also as resources on which to develop and test new NLP tools for languages other than English. Prettenhofer and Stein (2010) used a multi-lingual dataset focused on full-document classification at the global polarity level. Denecke (2008) used a dataset of 200 Amazon reviews in German to test cross-lingual document polarity classification using an English training set. Klinger and Cimiano (2014) produced a bi-lingual dataset (English and German), named USAGE, in which aspect expressions and subjective expressions are annotated in Amazon product reviews. In (Klinger and Cimiano, 2014) aspect expressions can be any piece of text that mentions a relevant property of the reviewed entity (e.g., *washer, hose, looks*) and are not categorical label, as in our dataset. The USAGE dataset is thus more oriented at information extraction rather than at text classification applications. Banea et al. (2010) used machine translation to create a multilingual version of the information-extraction oriented MPQA dataset (Wiebe et al., 2005) on six languages (English, Arabic, French, German, Romanian and Spanish).

In this paper we present Trip-MAML, which extends the Trip-MA[1] dataset of Marcheggiani et al. (2014) with Italian and Spanish annotated reviews. We describe Trip-MAML and report experiments aimed at defining a first baseline. Both the dataset and the software used in experiments are publicly available at `http://hlt.isti.cnr.it/trip-maml/`.

2 Annotation Process

We recall the annotation process adopted by Marcheggiani et al. (2014) for Trip-MA and the procedure we employed to extend it into Trip-MAML. We will use the national codes EN, ES, and IT, to denote the English, Spanish, and Italian parts of the Trip-MAML dataset, respectively. Note that EN coincides with Trip-MA.

2.1 English Reviews

The Trip-MA dataset was created by Marcheggiani et al. (2014) by annotating a set of 442 reviews, written in English, randomly sampled from the publicly available TripAdvisor dataset of Wang et al. (2010), composed by 235,793 reviews. Each review comes with an overall rating on a discrete ordinal scale from 1 to 5 "stars". The dataset was annotated according to 9 recurrent aspects frequently involved in hotel reviews: Rooms, Cleanliness, Value, Service, Location, Check-in, Business, Food, and Building. The last two are not officially rated by TripAdvisor but were added because they are frequently commented in reviews. Two "catch-all" aspects, Other and NotRelated, were also added, for a total of 11 aspect. Aspect Other denotes opinions that are pertinent to the hotel being reviewed, but not relevant to any of the former nine aspects (e.g., generic evaluations like *Pulitzer exceeded our expectations*). Aspect NotRelated denotes opinions that are not related to the hotel (e.g., *Tour Eiffel is amazing*).

If a sentence is relevant to an aspect, the possible sentiment label values are three: Positive, Negative, and Neutral/Mixed[2]. Neutral/Mixed annotates subjective evaluations that are not clearly polarized (e.g., *The hotel was fine with some exceptions*).

2.1.1 Annotation protocol

Marcheggiani et al. (2014) relied on three human annotators to annotate each sentence of the 442 reviews with respect to polarities of opinions that are relevant to any of the 11 aspects. 73 reviews, out of 442, were independently annotated by all the annotators in order to measure the inter-annotator agreement, while the remaining 369 reviews were partitioned into 3 equally-sized sets, one for each annotator. Bias in the estimation of inter-annotator agreement was minimized by sorting the list of reviews of each annotator so that every eighth review was common to all annotators; this ensured that each annotator had the same amount of coding experience when labeling the same shared review.

[1] Marcheggiani et al. (2014) gave no name to their dataset, here we name it Trip-MA to identify its source and its multi-aspect nature.

[2] Marcheggiani et al. (2014) initially distinguished between implicit and explicit opinions but the human agreement was so low they removed this distinction from the schema.

	# Reviews	# Sentences	# Opinion-laden sentences
EN	442	5799	4810
ES	500	2620	2400
IT	500	2593	2392

Table 1: Number of reviews, sentences, and sentences with at least one opinion annotation.

2.2 Spanish and Italian Reviews

For the creation of ES and IT parts of the Trip-MAML dataset we followed the same annotation protocol of Marcheggiani et al. (2014), employing teams of three native speakers as annotators for each language. We crawled the Spanish and Italian reviews from TripAdvisor by accessing its websites with the '.es' and '.it' domains, which mostly contains reviews in the national language. From that domains we downloaded the reviews for the 10 most visited cities in Spain and Italy, respectively. We downloaded 10 reviews for every hotel of each city, obtaining a total of 17,020 reviews for Spanish and 33,325 for Italian. For each dataset, 500 reviews were selected by randomly sampling 50 reviews for each city. We thus obtained 139 unique reviews for each annotator, plus 83 reviews which all three annotators independently annotated.

We decided to annotate the aspects that were ratable on TripAdvisor at the time of our crawl (April 2015: Rooms, Cleanliness, Value, Service, Location, and Sleep Quality). Differently from the aspect schema in EN, we included the new aspect Sleep Quality, and we did not consider the missing aspects Check-in and Businnes, which are, in any case, the least frequent aspects in the Trip-MA dataset (see Table 2). We kept the additional aspects Food, Building, Other, and NotRelated, as they still appear frequently in the reviews. We adopted the same 3-values sentiment label schema of EN, i.e., Positive, Negative, or Neutral/Mixed.

Following the same procedure adopted by Marcheggiani et al. (2014), the Spanish and Italian annotator teams performed a preliminary annotation session on reviews not included in the final dataset. This preliminary activity was aimed at aligning the annotators' understanding about the labeling process for the different aspects, by sharing and solving any doubt that might arise during the annotation of some examples.

2.3 Statistics

Table 1 shows that English reviews have, on average, about double the number of sentences of Spanish and Italian reviews. This can be in part motivated by observing that the sentences in EN are, on average, 25% shorter than in ES and IT. Also, after a manual inspection of the data, we found that the EN part contains some reviews related to long vacations in resorts, thus describing in longer details the experience, while IT and ES reviews are mainly related to relatively short visits to classic hotels. However, the portion of opinionated sentences is similar across the three parts, indicating homogeneity in content, which is confirmed by the detailed aspect-level statistics reported in Table 2.

Both aspect and sentiment labels show imbalanced distributions that follow similar distributions across the three parts. The most frequent aspect in all collections is Other, followed by Rooms, Service, and Location. Building and Value are among the least frequent ones. The average value of the Pearson correlation between the lists of the shared aspects ranked by their relative frequency, measured pairwise among the three parts, is 0.795, which indicates a good uniformity of content among the parts. In all the three parts, Positive is the most frequent sentiment label, followed by Negative. Location is always the aspect with the highest frequency of positive labels.

3 Inter-annotator Agreement

We measured the inter-annotator agreement in two steps. The F_1 score measures the agreement on aspect identification, regardless of the sentiment label assigned. Then symmetric Macro-averaged Mean Absolute Error (sMAE^M) (Baccianella et al., 2009) measures the agreement on sentiment labels on the annotations for which the annotators agreed at the aspect level. Aspect NotRelated is not included in agreement evaluation, nor in the experiments of Section 4. sMAE^M is computed between each of the three possible pairs of annotators and then averaged to determine the agreement values reported in Table 3.

Agreement on aspect detection is higher for ES and IT than for EN. This difference is in part motivated by the fact that the two aspects that are missing in ES and IT have low agreement on EN, and the novel Sleep Quality aspect has instead a high agreement. However, also on the other aspects

		Other	Service	Rooms	Clean.	Food	Loc.	Check-in	Sleep-q.	Value	Build.	Busin.	NotRelated	*Total*
EN	Pos	893	513	484	180	287	435	93	-	188	185	23	63	*3344*
	Neg	353	248	287	66	127	51	56	-	87	62	3	40	*1377*
	Neu	167	40	111	5	82	38	12	-	35	22	4	350	*866*
	Total	*1413*	*801*	*882*	*251*	*496*	*524*	*161*	-	*310*	*269*	*30*	*453*	5587
ES	Pos	634	382	275	181	128	452	-	126	114	71	-	39	*2402*
	Neg	244	85	159	40	37	38	-	75	48	28	-	38	*792*
	Neu	46	19	62	6	32	22	-	6	18	7	-	4	*222*
	Total	*924*	*486*	*496*	*227*	*197*	*512*	-	*207*	*180*	*106*	-	*81*	3416
IT	Pos	582	415	267	259	207	389	-	103	135	77	-	50	*2484*
	Neg	189	74	110	65	56	22	-	50	27	43	-	15	*651*
	Neu	102	30	59	10	52	49	-	1	32	32	-	100	*467*
	Total	*873*	*519*	*436*	*334*	*315*	*460*	-	*154*	*194*	*152*	-	*165*	3602

Table 2: Number of opinion expressions at the sentence level of the datasets.

		Other	Service	Rooms	Clean.	Food	Loc.	Check-in	Sleep-q.	Value	Build.	Busin.	*Avg*
EN	F_1	.607	.719	.793	.733	.794	.795	.464	-	.575	.553	.631	*.675*
	sMAE^M	.308	.219	.191	.114	.234	.259	.003	-	.202	.150	.029	*.171*
ES	F_1	.789	.911	.854	.933	.882	.896	-	.895	.829	.538	-	*.836*
	sMAE^M	.174	.093	.133	.303	.120	.293	-	.000	.150	.184	-	*.161*
IT	F_1	.676	.812	.788	.913	.884	.856	-	.789	.858	.532	-	*.790*
	sMAE^M	.292	.166	.242	.114	.204	.204	-	.067	.292	.114	-	*.188*

Table 3: Inter-annotator agreement. F_1 on sentence-level aspect identification (higher is better). sMAE^M on sentence-level sentiment agreement (only on matching aspects, lower is better).

there is, in general, a higher or equal agreement in ES and IT with respect to EN, indicating that the formers two were annotated in a more consistent way. The agreement on assignment of sentiment label is rather similar across the whole dataset.

4 Experiments

The experiments we present here are aimed at defining a shared baseline for future experiments. For this reason we chose a relatively simple setup that uses a simple learning model and minimal linguistic resources. We used a sentence-level Linear Chain (LC) Conditional Random Field (Lafferty et al., 2001) as described by Marcheggiani et al. (2014). With respect to the features extracted from text, we used three simple features types: word unigrams, bigrams, and SentiWordNet-based features, which consist of a Positive and a Negative feature extracted every time the review contains a word that is marked as such in SentiWordNet (Baccianella et al., 2010). To use SentiWordNet on ES and IT, we used Multilingual Central Repository (Gonzalez-Agirre et al., 2012) and MultiWordNet (Pianta et al., 2002) to map sentiment labels to Spanish and to Italian, respectively.

Experiments were run separately on the EN, ES, and IT parts, leaving cross-lingual experiments to future work. On each part we built five 70%/30% train/test splits, randomly generated by sampling the reviews annotated by single reviewers (we left out reviews annotated by all the reviewers, as we consider that part of the dataset more useful as a validation set for the optimization of methods tested in future experiments). We then run the five experiments and averaged their results.

4.1 Evaluation Measures

As for the agreement evaluation (Section 3), we split the evaluation of experiments into two parts, aspect detection and sentiment labeling. For the sentiment labeling part we used simple Macro-averaged Mean Absolute Error (MAE^M, not the symmetric version) as the true dataset labels are the reference ones in this case, while in the annotator agreement case the two sets of labels have equal importance.

4.2 Results

Experiments on ES and IT obtain better F_1 values than on EN, indicating that the observed higher human agreement can be also explained by a lower hardness of the task when working with Spanish

		Other	Service	Rooms	Clean.	Food	Loc.	Check-in	Sleep-q.	Value	Build.	Busin.	*Avg*
EN	F_1	.482	.595	.626	.729	.541	.616	.230	-	.331	.281	.222	*.465*
	MAE^M	.549	.822	.641	.968	.585	.959	.264	-	.598	.471	.000	*.586*
ES	F_1	.520	.668	.766	.782	.567	.730	-	.416	.593	.215	-	*.584*
	MAE^M	.839	.737	.515	.377	.516	1.002	-	.395	.564	.000	-	*.549*
IT	F_1	.576	.747	.646	.770	.697	.757	-	.254	.630	.087	-	*.574*
	MAE^M	.707	.781	.809	.887	.829	.746	-	.053	.403	.000	-	*.579*

Table 4: Linear Chain CRFs experiments. F_1 on sentence-level aspect identification (higher is better). MAE^M on sentence-level sentiment assignment (only on correctly identified aspects, lower is better).

and Italian.

MAE^M values are all similar across languages, again confirming what has been observed on agreement. However, MAE^M values on experiments are sensibly worse than those measured on agreement, possibly due to the fact that we used very basic features, with limited use of sentiment-related information.

5 Conclusion

We have presented Trip-MAML a multi-lingual extension of Trip-MA, originally presented in (Marcheggiani et al., 2014). The extension process involved crawling and selecting the reviews for the two new languages, Spanish and Italian, and their annotation by a total of six native language speakers. We measured dataset statistics and inter-annotator agreement, which show that the new ES and IT parts we produced are consistent with the original EN part. We also presented experiments on the dataset, based on a linear chain CRFs model for the automatic detection of aspects and their sentiment labels, establishing a baseline for future research. Trip-MAML enables the exploration of cross-lingual approaches to the problem of multi-aspect sentiment classification.

Acknowledgments

This work has been partially supported by ATTOS project (TIN2012-38536-C03-0) from the Spanish Government.

References

Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. 2009. Evaluation measures for ordinal regression. In *Proocedings of the 9th Conference on Intelligent Systems Design and Applications (ISDA 2009)*, pages 283–287.

Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. 2010. Sentiwordnet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In *LREC*, volume 10, pages 2200–2204.

Carmen Banea, Rada Mihalcea, and Janyce Wiebe. 2010. Multilingual subjectivity: Are more languages better? In *Proceedings of the 23rd International Conference on Computational Linguistics*, COLING '10, pages 28–36, Stroudsburg, PA, USA. Association for Computational Linguistics.

Judith A Chevalier and Dina Mayzlin. 2006. The effect of word of mouth on sales: Online book reviews. *Journal of marketing research*, 43(3):345–354.

Kerstin Denecke. 2008. Using sentiwordnet for multilingual sentiment analysis. In *Data Engineering Workshop, 2008. ICDEW 2008. IEEE 24th International Conference on*, pages 507–512. IEEE.

Aitor Gonzalez-Agirre, Egoitz Laparra, and German Rigau. 2012. Multilingual central repository version 3.0. In *Proceedings of the 8th International Conference on Language Resources and Evaluation (LREC 2012)*, pages 2525–2529.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, pages 168–177, New York, NY, USA. ACM.

Roman Klinger and Philipp Cimiano. 2014. The usage review corpus for fine-grained, multi-lingual opinion analysis. In *Proceedings of the Language Resources and Evaluation Conference (LREC)*, Reykjavik, Iceland, May. ELRA.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning (ICML 2001)*, pages 282–289.

Angeliki Lazaridou, Ivan Titov, and Caroline Sporleder. 2013. A bayesian model for joint unsupervised induction of sentiment, aspect and

discourse representations. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL 2013)*, pages 1630–1639.

Diego Marcheggiani, Oscar Täckström, Andrea Esuli, and Fabrizio Sebastiani. 2014. Hierarchical multi-label conditional random fields for aspect-oriented opinion mining. In *Proceedings of the 36th European Conference on IR Research (ECIR 2014)*, pages 273–285.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Foundations and trends in information retrieval*, 2(1-2):1–135.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 79–86. Association for Computational Linguistics.

Emanuele Pianta, Luisa Bentivogli, and Christian Girardi. 2002. Multiwordnet: developing an aligned multilingual database. In *Proceedings of the first international conference on global WordNet*, volume 152, pages 55–63.

Peter Prettenhofer and Benno Stein. 2010. Cross-language text classification using structural correspondence learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL 2010)*, pages 1118–1127.

Peter Prettenhofer and Benno Stein. 2011. Cross-lingual adaptation using structural correspondence learning. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 3(1):13.

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP 2013)*, volume 1631, page 1642.

Oscar Täckström and Ryan McDonald. 2011. Discovering fine-grained sentiment with latent variable structured prediction models. In *Proceedings of the 33rd European Conference on Advances in Information Retrieval (ECIR 2011)*, pages 368–374.

Xiaojun Wan. 2009. Co-training for cross-lingual sentiment classification. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pages 235–243.

Hongning Wang, Yue Lu, and Chengxiang Zhai. 2010. Latent aspect rating analysis on review text data: A rating regression approach. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2010)*, pages 783–792.

Janyce Wiebe, Theresa Wilson, and Claire Cardie. 2005. Annotating expressions of opinions and emotions in language. *Language resources and evaluation*, 39(2-3):165–210.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2009. Recognizing contextual polarity: An exploration of features for phrase-level sentiment analysis. *Computational linguistics*, 35(3):399–433.

Deep Convolutional Neural Network Textual Features and Multiple Kernel Learning for Utterance-Level Multimodal Sentiment Analysis

Soujanya Poria
Temasek Laboratory, Nanyang Technological University, Singapore
sporia@ntu.edu.sg

Erik Cambria
School of Computer Engineering, Nanyang Technological University, Singapore
cambria@ntu.edu.sg

Alexander Gelbukh
Centro de Investigación en Computación, Instituto Politécnico Nacional, Mexico
www.gelbukh.com

Abstract

We present a novel way of extracting features from short texts, based on the activation values of an inner layer of a deep convolutional neural network. We use the extracted features in multimodal sentiment analysis of short video clips representing one sentence each. We use the combined feature vectors of textual, visual, and audio modalities to train a classifier based on multiple kernel learning, which is known to be good at heterogeneous data. We obtain 14% performance improvement over the state of the art and present a parallelizable decision-level data fusion method, which is much faster, though slightly less accurate.

1 Introduction

The advent of the Social Web has enabled anyone with a smartphone or computer to easily create and share their ideas, opinions and content with millions of other people around the world. Much of the content being posted and consumed online is video. With billions of phones, tablets and PCs shipping today with built-in cameras and a host of new video-equipped wearables like Google Glass on the horizon, the amount of video on the Internet will only continue to increase.

It has become increasingly difficult for researchers to keep up with this deluge of video content, let alone organize or make sense of it. Mining useful knowledge from video is a critical need that will grow exponentially, in pace with the global growth of content. This is particularly important in sentiment analysis (Cambria et al., 2013a; 2013b; 2014), as both service and product reviews are gradually shifting from unimodal to multimodal. We present a method for detecting sentiment polarity in short video clips of a person uttering a sentence.

We do it using all three modalities: visual, such as facial expression, audio, such as pitch, and textual, the contents of the uttered sentence. While the visual and the audio modalities provide additional evidence that improves classification accuracy, we found the textual modality to have the greater impact on the result (Cambria and Hussain, 2015; Cambria et al., 2013c; Poria et al., 2015a; 2015b).

In this paper, we propose a novel way for feature extraction from text. Given a training corpus with hand-annotated sentiment polarity labels, following Kim (2014), we train a deep convolutional neural network (CNN) on it. However, instead of using it as a classifier, as Kim did, we use the values from its hidden layer as features for a much more advanced classifier, which gives superior accuracy. Similar ideas have been suggested in the context of computer vision for dealing with images, but have not been applied in the context of NLP to textual data, and, specifically, for sentiment polarity classification.

2 Overview of the Method

In this paper, we present two different methods for dealing with multimodal data: feature-level fusion and decision-level fusion, each one having its advantages and disadvantages.

We extracted features from the data for each modality independently. In the case of feature-level fusion, we then concatenated the obtained feature vectors and fed the resulting long vector into a supervised classifier. In the case of decision-level fusion, we fed the features of each modality into separate classifiers, and then combined their decisions. Our experimental results show that both of these methods outperform the state of the art by a large margin.

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 2539–2544,
Lisbon, Portugal, 17-21 September 2015.

3 Textual Features

We used a CNN as a trainable feature extractor to extract features from the textual data. Utterances in the original dataset are in Spanish. While usually it is better to work directly with the source language (Wang et al., 2013), in this work we translated utterances into English using Google translator. Without the translation into English, 68.56% accuracy was obtained.

The choice of CNN for feature extraction is justified by the following considerations:

1. The convolution layers of CNN can be seen as a feature extractor, whose output is then fed into a rather simplistic classifier useful for training the network but not the best at actual classification. CNN forms local features for each word and combine them to produce a global feature vector for the whole text. However, the features that CNN builds internally can be extracted and used as input for another, more advanced classifier. This turns CNN, originally a supervised classifier, into a trainable feature extractor.
2. As a feature extractor, CNN is automatic and does not rely on handcrafted features. In particular, it adapts well to the peculiarities of the specific dataset, in a supervised manner.
3. The features it gives are based on a hierarchy of local features, reflecting well the context.

A drawback of CNN as a classifier is that it finds only a local optimum, since it uses the same backpropagation technique as MLP. However, inspired by ideas introduced in the context of computer vision (Bluche et al., 2013), we, for the first time in the context of NLP, extract the features that CNN builds internally and feed them into a much more advanced classifier. In our experiments, this was SVM, or roughly its multi-kernel version MKL, which is good at finding the global optimum. Thus, the properties of CNN and SVM complement each other in such a way that their advantages are combined.

To form the input for the CNN feature extractor, for each word in the text we built a 306-dimensional vector by concatenating two parts:

1. *Word embeddings*. We used a publicly available word2vec dictionary (Mikolov et al., 2013a; 2013b; 2013c), trained on a 100 million word corpus from Google News using the continuous bag of words architecture. This dictionary provides a 300-dimensional vector for each word. For words not found in this dictionary, we used random vectors.
2. *Part of speech*. We used 6 basic parts of speech (noun, verb, adjective, adverb, preposition, conjunction) encoded as a 6-dimensional binary vector. We used Stanford Tagger as a part of speech tagger.

For each input text, the input vectors for the CNN were a concatenation of three parts:

1. Left padding. Two dummy "words" with zero vectors were added at the beginning of each text, in order to provide space for convolution, since at the convolution layers we used the kernel size of at most 3.
2. Text. All 306-dimensional vectors corresponding to each word were concatenated, preserving the word order.
3. Right padding. Again, at least 2 dummy "words" with zero vectors were added after each sentence to provide space for convolution. To form vectors for all texts in the corpus of the same dimensionality, they were also padded at the right with the necessary amount of additional dummy "words."

In our experiments, all texts were very short, consisting of one sentence, the longest one being of 65 words. Thus all input vectors were of dimension $306 \times (2 + 65 + 2) = 21{,}114$.

The CNN we used consisted of 7 layers:

1. Input layer, of 21,114 neurons.
2. Convolution layer, with a kernel size of 3 and 50 feature maps. The output of this layer was computed with a non-linear function; we used the hyperbolic tangent.
3. Max-pool layer with max-pool size of 2.
4. Convolution layer: kernel size of 2, 100 feature maps, also using the hyperbolic tangent.
5. Max-pool layer with max-pool size of 2.
6. Fully connected layer of 500 neutrons, whose values were later used as the extracted features. For regularization, we employed dropout on the penultimate layer with a constraint on L2-norms of the weight vectors.
7. Output softmax layer of 2 neurons, by the number of training labels—the sentiment polarity values: positive or negative. This layer was used only for training the CNN.

The CNN was trained using a standard backpropagation procedure. The training data for the output layer were the known sentiment polarity labels present in the training corpus for each text.

As features of the given text, we used the values of the penultimate, fully connected, layer of the CNN. In this way, we used the last output

		Text	Visual	Audio	Pérez-Rosas et al. (2013)	Our method	
						without feature selection	with feature selection
# features, without selection		500	4568	6373		↰	
Unimodal	# features, with selection	437	–	–	70.94%	79.14%	79.77%
		–	398	–	67.31%	75.22%	76.38%
		–	–	325	64.85%	74.49%	74.22%
Bimodal		379	109	–	72.39%	84.97%	85.46%
		384	–	81	72.88%	83.85%	84.12%
		–	242	209	68.86%	82.95%	83.69%
Multimodal		305	74	58	74.09%	87.89%	**88.60%**

Table 1. Accuracy of state-of-the-art method compared with our method with feature-level fusion. The number of features is for our experiments, not for [16]. Shaded cells are shared with Table 2.

layer of the CNN only for training, but for actual decision-making, we replaced it with much more sophisticated classifiers, namely, with SVM or MKL. Using only CNN as a classifier, 75.50% was obtained which is in fact lower than the result (79.77%) obtained when CNN was used to extract trainable features for the SVM classifier.

We also tried other word vectors having different dimensions, e.g., Glove word vectors and Collobart's word vectors. However, the best accuracy was obtained using Google word2vec.

4 Visual Features

We split each clip into frames (still images). From each frame, we extracted 68 facial characteristic points (FCPs), such as the position of the left corner of the left eye, etc., using the facial recognition library CLM-Z (Baltrušaitis et al., 2012). For each pair of FCPs, we calculated the distance. Thus, we characterized each facial expression by $68 \times 67 / 2 = 2{,}278$ distances. In addition, for each frame we extracted 6 face position coordinates (3D-dimensional displacement and angular displacement of face and head) using the GAVAM software. This gave $2{,}278 + 6 = 2{,}284$ values per frame.

For each of these values, we calculated its mean value and standard deviation over all frames of the clip; 4568 features in total.

5 Audio Features

We used the openSMILE software (Eyben et al., 2010) to extract audio features related to the pitch and voice intensity. This software extracts the so-called low-level descriptors, such as Mel frequency cepstral coefficients, spectral centroid, spectral flux, beat histogram, beat sum, strongest beat, pause duration, pitch, voice quality, perceptual linear predictive coefficients, etc., and their statistical functions, such as amplitude mean, arithmetic mean, root quadratic mean, standard deviation, flatness, skewness, kurtosis, quartiles, inter-quartile ranges, linear regression slope, etc. This gave us 6373 audio features in total.

6 Feature-Level Fusion

Feature-level fusion consisted in concatenation of the feature vectors obtained for each of the three modalities. The resulted vectors and along with the sentiment polarity labels from the training set, were used to train a classifier with a multiple kernel learning (MKL) algorithm; we used the SPF-GMKL implementation (Jain et al., 2012) designed to deal with heterogeneous data. Clearly, feature vectors resulted from concatenating so different data sources are heterogeneous.

The parameters of the classifier were found by cross validation. We chose a configuration with 8 kernels: 5 RBF with gamma from 0.01 to 0.05 and 3 polynomial with powers 2, 3, 4. We also tried Simple-MKL; it gave slightly lower results.

7 Feature Selection

We significantly reduced the number of features using feature selection. We used two different feature selectors: one based on the cyclic correlation-based feature subset selection (CFS) and another based on principal component analysis (PCA) with top K features, where K was experimentally selected and varied for different experiment. For example, in case of audio, visual and textual fusion, K was set to 300.

The union of the features selected by the two methods was used. For each unimodal, each bimodal, and the multimodal experiment, separate feature extraction was performed. The number of

selected features for each experiment is given in Table 1. In all cases except for the unimodal experiment with audio modality, feature selection slightly improved the results, in addition to the improvement in processing time. In the only case where feature selection slightly deteriorated the result, the difference was rather small.

8 Unimodal Classification and Decision-Level Fusion

For unimodal experiments and for decision-level fusion, we used one classifier per each modality; specifically, we used SVM. For each modality, in this way we obtained the probabilities of the labels. In unimodal experiments, we chose the label with the greater probability.

For decision-level fusion, we added these probabilities with weights, which were chosen experimentally, and, again, used the most probable label. The weights we used for decision-level fusion were chosen using detailed search with an intuition that best performing unimodal classifier has higher importance in the fusion. We do not claim that these weights are optimal. They are indeed sub-optimal and hence encourage the scope of future research.

Knowing a specific decision for the text modality allowed us to use evidence from a separate classifier; we used the one based on the Sentic Patterns (SP) (Poria et al., 2014a). It structures natural language clauses into a sentiment hierarchy used to infer the overall polarity label (positive vs. negative) for the input sentence. E.g., a sentence "The car is very old but it is rather not expensive", is positive, expressing a favorable sentiment of the speaker, who recommends purchasing the product. However, "The car is very old though it is rather not expensive" is negative, expressing reluctance of the speaker to purchase the car. Despite the latter contains exactly the same concepts as the former, the polarity is opposite because of the adversative dependency.

On benchmark datasets, SP perform better than state of the art sentiment classifiers, which outperforms the textual classifier described in Section 3. Since SP are a superior classifier, we used it as a bias to modify the weight of the textual modality. However, SP do not report a probability, but only a binary decision, so we only used them to tweak the weights in the probability mix: when the text-based unimodal classifier agreed with SP, we increased the weight of the text modality. Another benefit of the decision-level fusion is its speed, since fewer features are used for each classifier and since SVM, used as a unimodal classifier, is faster than MKL. In addition, separate classifiers can be run in parallel.

9 Experimental Results

We report results for tenfold cross-validation.

9.1 Dataset

We experimented on the dataset described by Morency et al. (2011). The dataset consists of 498 short video fragments where a person utters one sentence. The items are manually tagged for sentiment polarity, which can be positive, negative, or neutral. We discarded the neutral items from the dataset, which gave us a dataset of 447 clips tagged as positive or negative.

The video in the dataset is present in MP4 format with the resolution of 360 × 480, to which the developers converted all videos originally collected in different formats with different resolution. The duration of the clips is about 5 seconds on average. About 80% of the clips present female speakers. The developers provided transcription of the text of the sentences, which we used in our textual modality processing.

9.2 Results for Each Modality Separately

As a baseline, we used classifiers trained on features extracted from each modality separately. The results are shown in Table 1, unimodal section. The number of features after feature selection is indicate for the modality used.

The table shows that the best results were obtained for textual modality; the visual modality performed worse, and the audio was least useful. However, even the worst of our results is much better than the state-of-the-art (Pérez-Rosas et al., 2013). In each modality separately, our results outperform the state of the art by about 9%, which is about 30% reduction in error rate.

9.3 Results with Feature-level Fusion

As a yet another baseline, we tried feature-level fusion of only two modalities.

The results are shown in Table 1, bimodal section. Again, the number of features after feature selection is indicated for the two modalities used. As expected, missing the audio features was the least important, missing the video features was more significant, and missing the text features was most painful for the accuracy.

Even the worst result obtained with fusion of two modalities outperformed our best unimodal result, as well as the best result of the state of the

	Sentic Patterns	Weights			Fusion Accuracy	
		Text	Visual	Audio	Feature	Decision
Unimodal accuracy	81.73%	79.77%	76.38%	74.22%		
Unimodal 3-way		3-way majority voting				72.83%
	no	0.45	0.3	0.25		81.24%
	yes	0.5 / 0.25	0.3 / 0.4	0.2 / 0.35		82.06%
Bimodal with unimodal		+	+	0.3	85.46%	85.53%
		+	0.23	+	84.12%	84.86%
	no	0.4	+	+	83.69%	84.48%
	yes	0.45 / 0.3	+	+	same	**86.27%**

Table 2. Accuracy of our method with decision-level fusion and feature selection.

art. Finally, the best result, shown in the multimodal section of Table 1, was obtained when all three modalities were fused. This result outperforms the corresponding result of the state of the art by 14%, which gives 56% of error reduction.

9.4 Results with Decision-level Fusion

The results for decision-level fusion are shown in Table 2, last column. The shaded cells are shared with Table 1. In the second section, three classifiers were fused at the decision level. In the third section, two modalities indicated with the plus sign were fused at the feature level (giving the accuracy indicated in the penultimate column) and then this classifier was fused at the decision level with the third modality. The weights correspond to the share of each modality. In the last section, the weight for the unimodal classifier is shown, and the weight for the bimodal classifier was its complement to 1.

For the experiments that involved tweaking of the weights with the SP oracle, pairs of weights are shown: the weight used when the text modality results corresponded (left) with the SP prediction and the weight used when they did not (right). The accuracy with at least partial feature-level fusion was better than that for no feature-level fusion at all (3-way). As in the bimodal section of Table 1, excluding audio from feature-level fusion was least problematic and excluding text was most problematic.

In all cases, decision-level fusion did not significantly improve the accuracy of the best summand. However, separating text-based classifier permitted us to use the Sentic Patterns tweak, which cannot be used if the text-only results are not known. With this tweak, the best result was obtained. Even with this improvement, the accuracy of decision-level fusion was slightly lower than that of feature-level fusion; in exchange for much about twice better processing speed.

A baseline decision level evaluation strategy was taken which allowed us to take majority voting among the predicted class labels by unimodal classifiers. Based on this strategy the final class label was chosen by the maximum of the three unimodal models' votes. For the multimodal fusion using this baseline method only 72.83% accuracy was obtained. As expected the proposed feature and decision level fusion outperformed this baseline method by a large margin.

10 Conclusion

We have presented a novel method for determining sentiment polarity in video clips of people speaking. We combine evidence from the words they utter, the facial expression, and the speech sound. The main novelty of this paper consists in using deep CNN to extract features from text and in using MKL to classify the multimodal heterogeneous fused feature vectors.

We also presented a faster variant of our method, based on decision-level fusion. In case of the decision level fusion experiment, the coupling of Sentic Patterns to determine the weight of textual modality has enriched the performance of multimodal sentiment analysis framework considerably. However, the parameter selection for decision level fusion produced suboptimal results. A systematic mathematical approach for decision level fusion is an important future work.

Our future work will focus on extracting more relevant features from the visual modality. We will employ deep 3D convolutional neural networks on this modality for feature extraction. We will use a feature selection method to obtain key features; this will ensure the scalability as well as stability of the framework. We will continue our study of reasoning over text (Jimenez et al., 2015; Pakray et al., 2011; Sidorov et al., 2014; Sidorov, 2014) and in particular of concept-based sentiment analysis (Poria et al., 2014b).

References

1. Tadas Baltrušaitis, Peter Robinson, and Louis-Philippe Morency. 2012. 3D constrained local model for rigid and non-rigid facial tracking. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2610–2617.

2. Théodore Bluche, Hermann Ney, and Christopher Kermorvant. 2013. Feature extraction with convolutional neural networks for handwritten word recognition. In *12th International Conference on Document Analysis and Recognition (ICDAR)*, pp. 285–289.

3. Erik Cambria and Amir Hussain. 2015. Sentic Computing: A Common-Sense-Based Framework for Concept-Level Sentiment Analysis. Springer, Cham, Switzerland.

4. Erik Cambria, Björn Schuller, Bing Liu, Haixun Wang, and Catherine Havasi. 2013a. Knowledge-based approaches to concept-level sentiment analysis. *IEEE Intelligent Systems* 28(2):12–14.

5. Erik Cambria, Björn Schuller, Bing Liu, Haixun Wang, and Catherine Havasi. 2013b. Statistical approaches to concept-level sentiment analysis. *IEEE Intelligent Systems* 28(3):6–9.

6. Erik Cambria, Haixun Wang, and Bebo White. 2014. Guest editorial: Big social data analysis. Knowledge-Based Systems 69:1–2.

7. Erik Cambria, Newton Howard, Jane Hsu, and Amir Hussain. 2013c. Sentic blending: Scalable multimodal fusion for the continuous interpretation of semantics and sentics. In: *IEEE SSCI*, pp. 108–117, Singapore.

8. Florian Eyben, Martin Wöllmer, and Björn Schuller. 2010. OpenSMILE: The Munich versatile and fast open-source audio feature extractor. In *Proceedings of the International Conference on Multimedia*, pp. 1459–1462.

9. Ashesh Jain, Swaminathan V. N. Vishwanathan, and Manik Varma. 2012. SPF-GMKL: Generalized multiple kernel learning with a million kernels. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 750–758.

10. Sergio Jimenez, Fabio A. Gonzalez, and Alexander Gelbukh. 2015. Soft Cardinality in Semantic Text Processing: Experience of the SemEval International Competitions. *Polibits* 51:63–72.

11. Yoon Kim. 2014. Convolutional Neural Networks for Sentence Classification. In *EMNLP*, pp. 1746–1751.

12. Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. In *Intern. Conference on Learning Representations (ICLR)*.

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

14. Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013c. Linguistic regularities in continuous space word representations. In *NAACL-HLT*.

15. Louis-Philippe Morency, Rada Mihalcea, and Payal Doshi. 2011. Towards multimodal sentiment analysis: Harvesting opinions from the web. In *13th International Conference on Multimodal Interfaces*, pp. 169–176.

16. Verónica Pérez-Rosas, Rada Mihalcea, and Louis-Philippe Morency. 2013. Utterance-Level Multimodal Sentiment Analysis. In *ACL*, pp. 973–982.

17. Partha Pakray, Snehasis Neogi, Pinaki Bhaskar, Soujanya Poria, Sivaji Bandyopadhyay, and Alexander Gelbukh. 2011. A textual entailment system using anaphora resolution. In: *System Report. Text Analysis Conference, Recognizing Textual Entailment Track. Notebook*.

18. Soujanya Poria, Erik Cambria, Gregoire Winterstein, and Guang-Bin Huang. 2014a. Sentic patterns: Dependency-based rules for concept-level sentiment analysis. *Knowledge-Based Systems* 69:45–63.

19. Soujanya Poria, Erik Cambria, Amir Hussain, and Guang-Bin Huang. 2015a. Towards an intelligent framework for multimodal affective data analysis. *Neural Networks* 63:104–116.

20. Soujanya Poria, Erik Cambria, Newton Howard, Guang-Bin Huang, and Amir Hussain. 2015b. Fusing audio, visual and textual clues for sentiment analysis from multimodal content. *Neurocomputing*, DOI: 10.1016/j.neucom.2015.01.095

21. Soujanya Poria, Alexander Gelbukh, Erik Cambria, Amir Hussain, and Guang-Bin Huang. 2014b. EmoSenticSpace: A novel framework for affective common-sense reasoning. *Knowledge-Based Systems* 69:108–123.

22. Grigori Sidorov, Alexander Gelbukh, Helena Gómez-Adorno, and David Pinto. 2014. Soft Similarity and Soft Cosine Measure: Similarity of Features in Vector Space Model. *Computación y Sistemas* 18(3):491–504.

23. Grigori Sidorov. 2014. Should Syntactic N-grams Contain Names of Syntactic Relations? *International Journal of Computational Linguistics and Applications* 5(2):23–46.

24. Qiu-Feng Wang, Erik Cambria, Cheng-Lin Liu, and Amir Hussain. 2013. Common sense knowledge for handwritten Chinese recognition. *Cognitive Computation* 5(2):234–242.

SLSA: A Sentiment Lexicon for Standard Arabic

Ramy Eskander and Owen Rambow
Center for Computational Learning Systems
Columbia University
{rnd2110,ocr2101}@columbia.edu

Abstract

Sentiment analysis has been a major area of interest, for which the existence of high-quality resources is crucial. In Arabic, there is a reasonable number of sentiment lexicons but with major deficiencies. The paper presents a large-scale Standard Arabic Sentiment Lexicon (SLSA) that is publicly available for free and avoids the deficiencies in the current resources. SLSA has the highest up-to-date reported coverage. The construction of SLSA is based on linking the lexicon of AraMorph with SentiWordNet along with a few heuristics and powerful back-off. SLSA shows a relative improvement of 37.8% over a state-of-the-art lexicon when tested for accuracy. It also outperforms it by an absolute 3.5% of F1-score when tested for sentiment analysis.

1 Introduction

Sentiment analysis is the process of identifying and extracting subjective information using Natural Language Processing (NLP). It helps identifying opinions and extracting relevant information that lies behind the analyzed data. Sentiment analysis has received enormous interest in NLP, and in particular in the context of web content. This includes social media, blogs, discussions, reviews and advertisement.

While there has been extensive work on sentiment analysis in English and other languages of interest, less work has been done for Arabic. A major concern in Arabic NLP is the morphological complexity of the language along with the limited number of resources, corpora in particular.

The goal of this work is to build a publicly available large-scale Sentiment Lexicon for Standard Arabic (SLSA). For every lemma and part-of-speech (POS) combination that exists in a large Standard Arabic lexicon, SLSA assigns the scores of three sentiment labels: positive, negative and objective, in addition to the English gloss. The positive and negative scores range between zero and one, while the objective score is defined as *1 - (positive score + negative score)*.

The existence of SLSA is valuable to the field of Arabic sentiment analysis, which is expected to receive considerable focus during the current decade. SLSA is the first sentiment lexicon for Arabic to combine the following four strengths.

High coverage SLSA lists the sentiment of about 35,000 lemma and POS combinations, which is the highest coverage reported for Standard Arabic sentiment lexicons.

High quality Unlike many of the current lexicons whose construction is based on semi-supervised learning and heuristic-based approaches, SLSA is not constructed via machine learning models, while the use of heuristics is minimal.

Richness As opposed to sparse surface-based lexicons, SLSA is a lemma-based resource that attaches POS and English gloss information to each lemma, where the information of a lemma is applicable to its inflected forms. This makes the lexicon more useful when used by other research.

Public Availability SLSA is based on free resources and is publicly available for free.[1]

2 Related Work

Work on building Arabic sentiment lexicons mainly falls into two categories: 1) linking an Arabic sentiment lexicon with an English one, and 2) applying semi-supervised or supervised learning techniques on Arabic resources. We summarize these two types in turn.

We start with a survey of work based on translation, which our work falls into as well. The most similar work to the one presented in this paper is ArSenL (Badaro et al., 2014). ArSenL is considered the first publicly available large-scale Standard Arabic sentiment lexicon. It was constructed using a combination of SentiWordNet (Baccianella et al., 2010), Arabic WordNet (Black et al., 2006) and SAMA (Graff et al., 2009). Ar-

[1]The lexicon is available at http://volta.ldeo.columbia.edu/~rambow/slsa.html

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 2545–2550,
Lisbon, Portugal, 17-21 September 2015.

SenL outperforms the state-of-the-art Arabic sentiment lexicons. However, we show that SLSA has better coverage and quality. Moreover, ArSenL uses SAMA which is not publicly available for free, as opposed to SLSA which is based on free resources.

Another similar work to ArSenL is the resource developed by Alhazmi et al. (2013). They linked the Arabic WordNet to SentiWordNet via the provided synset offset information. However, the constructed lexicon has a limited coverage of nearly 10K lemmas, which makes it not very useful for further applications.

Abdul-Mageed and Diab (2014) presented SANA, a subjectivity and sentiment lexicon for Arabic. The lexicon combines pre-existing lexicons and involves automatic machine translation, manual annotations and gloss matching across several resources such as THARWA (Diab et al., 2014) and SAMA. SANA includes about 225K entries, where many of them are duplicates, inflected or not diacritized, which makes the resource noisy and less useable. Additionally, the automatic translation does not utilize the POS information, which affects the quality of the resource.

Other work that follows the translation approach includes the one presented by El-Halees (2011) where SentiStrength (Thelwall et al., 2010) was translated using a dictionary along with manual correction. Another instance is SIFAAT (Abdul-Mageed and Diab, 2012), an earlier version of SANA but with more reliance on translation. Another lexicon was built by Elarnaoty et al. (2012) who manually translated the MPQA lexicon (Wilson et al., 2005). The common aspect among those resources is the lack of adequate coverage and quality.

Mobarz et al. (2011) created a sentiment Arabic lexical Semantic Database (SentiRDI) by using a dictionary-based approach. The database has many inflected forms, i.e., it is not lemma-based. Moreover, the authors reported insufficient quality and plan to try other alternatives.

We now turn to work based entirely on Arabic resources. Mahyoub et al. (2014) created an Arabic sentiment lexicon that assigns sentiment scores to the words in Arabic WordNet using a lexicon-based approach. The lexicon was initially based on a few words and then expanded by exploiting synset relations in a semi-supervised learning manner. However, the lexicon is limited to about 23k lemmas and is not publicly available.

Another Arabic sentiment lexicon was created by Elhawary and Elfeky (2010). The lexicon was built using a similarity graph where the edges have similarity scores. A major drawback is the low coverage of the lexicon. Moreover, expanding the graph requires a huge corpus with polarity and semantic annotations and adds more sparsity.

3 Approach

Following the example of ArSenL (Badaro et al., 2014), SLSA is constructed by linking the lexicon of an Arabic morphological analyzer with SentiWordNet (Baccianella et al., 2010). Unlike ArSenL, SLSA uses AraMorph (Buckwalter, 2004), a morphological analyzer for Standard Arabic. An AraMorph entry represents a morpheme and contains the surface, lemma, part of speech (POS), and gloss information. The gloss information consists of a list of gloss terms, each of which contains one or more words (such as "*time limit / end*"). On the other side, SentiWordNet is a large-scale sentiment lexicon for English that assigns sentiment scores (positive, negative and objective) to the synsets in English WordNet (Miller et al., 1990) along with the POS and gloss information. Upon linking the two resources, the sentiment scores in SentiWordNet are applied to the entries of AraMorph to construct SLSA. The question this paper addresses is how to link these two resources, and we present a new linking algorithm compared to that used by ArSenL, with improved performance.

3.1 Preparing the Resources

It might seem intuitive to join the entries of AraMorph and SentiWordNet based on their glosses, but this does not work as expected. AraMorph and SentiWordNet were developed for different reasons and have different gloss structures (synonyms in AraMorph versus detailed descriptions in SentiWordNet). Mapping the glosses is one of the major bottlenecks in ArSenL, which is not able to find a match for 24% of the entries in SAMA. Instead, we link the two resources by relating the glosses of AraMorph to the synset terms in SentiWordNet. Additionally, we take POS into consideration as the glosses and synset terms might not be enough to disambiguate an entry. Next, we discuss the preparation steps that allow for the linking of the resources.

Cleaning-up AraMorph Some POS and lemma decisions in AraMorph are erroneous or not optimal. For example, some entries are assigned wrong POS tags, such as the NO_FUNC cases, or have inconsistent spellings of the lemmas. Also, some adverbs are redundant as they have the same lemma as an adjective. Accordingly, we cleaned up AraMorph in a way that allows for a better linking with SentiWordNet. The

cleaned-up AraMorph is closer to SAMA (used in ArSenL), which is itself a modified version of AraMorph. Practically, SAMA can replace AraMorph. However, the integration of AraMorph allows the lexicon to be publicly available for free, which SAMA prohibits.

Gloss Normalization Since the entries in AraMorph are bound to stems, the English glosses are inflected for number. As a result, we lemmatize the English glosses in AraMorph in order to be able to match to the synset terms in SentiWordNet. The lemmatization is done using Stanford CoreNLP Natural Language Processing Toolkit (Manning et al., 2014). Additionally, we remove from the glosses any descriptive text between parentheses, as well as the stop words *be*, *a*, *an* and *the* (unless *be* is the actual lemma of the AraMorph gloss). Moreover, if any of the lemmatized words in an AraMorph gloss does not match any of the synset terms in SentiWordNet and has a regular morphological derivation, the effect of the derivation is removed if the removal results in an existing synset term, e.g., *voluntariness* is converted to *voluntary* and *orientalization* becomes *orientalize*. We created the list of the derivational patterns manually by examining AraMorph glosses.

POS Mapping AraMorph has a rich POS tagset, while SentiWordNet has only four tags corresponding to nouns, adjectives, adverbs and verbs. Accordingly, AraMorph POS tags are mapped to the four tags in SentiWordNet. Some AraMorph POS types, such as particles, pronouns and prepositions, do not map to any SentiWordNet tags, and we exclude them as they have zero polarity scores, by definition.

AraMorph Rearrangement We collapse all the entries in AraMorph that have the same (lemma, POS) pair, and the English glosses become the union of the normalized glosses before the collapse. For example, a lemma might appear in two entries with two POS tags; *VERB_PERFECT* and *VERB_IMPERFECT* . After the preparation, the POS tags in both entries become the same (*VERB*), and the two entries collapse into one entry whose gloss is the union of the lemmatized past-tense and present-tense glosses. Figure 1 shows a sample of AraMorph before and after the preparation process (the Arabic transliteration is in the Buckwalter scheme (Buckwalter, 2004)).

SentiWordNet Rearrangement We extract all the unique combinations of synset terms and POS tags in SentiWordNet, while the indices of the synset terms are stripped off. However, since a synset term might appear in different synsets under the same POS with different indices and sentiment scores, the sentiment scores of an extracted entry is calculated as the average of all the sentiment scores that appear with the corresponding synset term and POS. Figure 2 shows a sample of SentiWordNet before and after the preparation process.

Stem	English Gloss	POS	Lemma
baliy~	tribulation/affliction	NOUN	baliy~ap_1
balAyA	tribulations/afflictions	NOUN	baliy~ap_1
balA	afflict/test	VERB_PERFECT	balA-u_1
balaw	afflict/test	VERB_PERFECT	balA-u_1
bal	afflict/test	VERB_PERFECT	balA-u_1
boluw	afflict/test	VERB_IMPERFECT	balA-u_1
bol	afflict/test	VERB_IMPERFECT	balA-u_1
bolaY	be afflicted/be tested	VERB_IMPERFECT	balA-u_1

Original AraMorph

Lemma	POS	English Gloss
baliy~ap_1	NOUN	affliction/tribulation
balA_1	VERB	afflict/test

Processed AraMorph

Figure 1: A sample of the original AraMorph (the upper table) and the processed version (the lower table). The glosses are normalized, while the POS tags are mapped to the tags in SentiWordNet. The entries of the same POS and lemma combinations are then collapsed, where their gloss becomes the union of the normalized glosses in the collapsed entries.

POS	ID	+ve	-ve	Synset Terms	Gloss
n	07305234	0	0.625	affliction#3	a cause of great suffering and distress
n	14213199	0	0.625	affliction#2	a condition of suffering or distress...
n	14477342	0	0.625	affliction#1	a state of great suffering and distress...
v	00259927	0	0.875	smite#3 afflict#2	cause physical pain or suffering in
v	01797730	0.125	0.625	afflict#1	cause great unhappiness for

Original SentiWordNet

Synset Term	POS	+ve	-ve
affliction	NOUN	0	0.625
afflict	VERB	0.0625	0.75

Processed SentiWordNet

Figure 2: A sample of the original SentiWordNet (the upper table) and the processed version (the lower table). The unique combinations of the synset terms and POS tags are extracted, where the sentiment scores of the extracted entries are the average of the scores in the contributing ones.

3.2 Linking the Resources

We start out by creating a link between an AraMorph entry and a SentiWordNet entry if any of the AraMorph one-word gloss terms and the POS match the SentiWordNet. Upon linking, we assign the AraMorph entry the sentiment scores of the matching one in SentiWordNet. The linking condition above applies successfully to 83.6% of the entries in AraMorph. If the condition does not apply, we relax it to allow for a more lenient POS agreement where *NOUN* and *ADJ* POS tags are used interchangeably, while the *VERB* entries in AraMorph become matchable with the *ADJ* ones in SentiWordNet. The reasons behind the decisions above are that AraMorph has hundreds of cases where the same lemma appears as *NOUN* and *ADJ*, while it is frequent that AraMorph assigns an adjectival gloss (preceded

by *be*) to *VERB* entries. The relaxed condition enables linking an additional 6.7% of AraMorph entries. If the relaxed condition is still not applicable for an AraMorph entry, the linking condition becomes more lenient by completely ignoring the POS agreement. The sentiment scores in that case become the average of the sentiment scores of the corresponding synset term across all the POS types. The latter condition allows matching additional 0.6% of AraMorph entries.

It might happen that none of the one-word gloss terms matches a synset term, or the gloss does not have any one-word gloss terms. In such a case, we consider multi-word gloss terms. We first remove the stop words, and then we test the relaxed condition on each word separately, starting with the shortest terms first. The process succeeds if a match could be established for all the words in a gloss term, and the sentiment scores become the average sentiment scores of the matching synset terms. The relaxed condition on multi-word terms solves additional 7.9% of the cases. Finally, if no match could be established across all the different gloss terms (1.2% of the entries), default neutral sentiment scores are assigned. The analysis of such cases is discussed in section 4.

Sometimes, a multi-word gloss term consists of words that denote excess (e.g., *most* and *more*), scarcity (e.g., *less* and *few*) or negation (e.g., *not*). We do not match such words to synset terms. Instead, they affect the polarity scores; we double the score, halve the score and swap the sign, respectively. We created the list of such words manually by examining AraMorph glosses.

Figure 3 illustrates the linking process between a sample of the processed AraMorph with a sample of the processed SentiWordNet, resulting in the construction of SLSA. The final SLSA lexicon consists of 34,821 entries. The counts of the different POS tags in SLSA along with the percentages of the different sentiment classes are reported in Table 1, while examples from the final lexicon are listed in Table 2.

POS	Count	Neutral %	+ve %	-ve %	Mixed %
NOUN	20,263	58.1	12.7	15.4	13.8
VERB	9,117	42.5	19.3	21.9	16.4
ADJ	5,395	36.5	18.7	17.0	27.8
ADV	46	73.9	10.9	2.2	13.0
ALL	34,821	50.7	15.3	17.3	16.7

Table 1: Statistics of SLSA: The counts of the different POS tags and the percentages of the different sentiment classes.

Lemma	POS	English Gloss
baliy~ap_1	NOUN	affliction/tribulation
balA_1	VERB	afflict/test

Processed AraMorph

Synset Term	POS	+ve	-ve
affliction	NOUN	0	0.625
afflict	VERB	0.0625	0.75

Processed SentiWordNet

Lemma	POS	English Gloss	+ve	-ve	Objective
baliy~ap_1	NOUN	affliction/tribulation	0	0.625	0.375
balA_1	VERB	afflict/test	0.0625	0.75	0.1875

SLSA

Figure 3: The linking between SentiWordNet and AraMorph by matching the AraMorph normalized glosses to the synset terms in SentiWordNet with respect to POS. The upper two tables are samples of the processed AraMorph and SentiWordNet, respectively, while the lower table represents a sample of SLSA based on the linking process. The objective score is calculated as *1 -(positive score+negative score)*.

Lemma	POS	English Gloss	+ve	-ve	Obj.
niEom_1	NOUN	wonderful	0.8	0	0.2
tawaE~aY_1	VERB	be attentive/cautious	0.4	0	0.6
AiHotiyAj_1	NOUN	need;requirement	0.1	0.2	0.7
$ahoriy~_1	ADJ	monthly	0	0	1
katab_1	VERB	write	0	0	1
mulaT~ax_1	ADJ	stained/sullied	0	0.3	0.7
dana>_1	VERB	be vile;be despicable	0	0.5	0.5
kamod_1	NOUN	swarthiness;sadness	0	0.8	0.2

Table 2: Examples of SLSA entries; Obj. = Objective. All scores are rounded for readability.

4 Evaluation

4.1 Intrinsic Evaluation

As mentioned in section 3, no match could be established for 1.2% of AraMorph entries. We manually investigate these cases more closely. About 75% of the entries that are not covered in SLSA have lemmas that express Arabic or Islamic subjects that do not have English counterparts such as *hamozap* (an Arabic name) and *kunAfap* (an Arabic food). Another 5% of the cases are countries or nationalities that are not listed in SentiWordNet such as *EAjiy* (Ivorian). Additional 2% of the cases are due to misspelled or non-English glosses in AraMorph such as *bon appetit*. The remaining cases (around 18%) have glosses that do not match any of the synset terms in SentiWordNet.

We then conduct an intrinsic evaluation of SLSA where the performance is compared to that of ArSenL, which is the most similar state-of-the-art lexicon (see Section 2). First, we randomly select 400 (lemma, POS) pairs for the evaluation. Only four pairs (1%) are not covered in SLSA. On the other side, 103 pairs (26%) are absent in ArSenL, which is consistent with the claim of the authors of ArSenL that only 76% of SAMA entries are matched in SentiWordNet. We then evaluate the random entries that exist in both SLSA and ArSenL (297 entries). We ask human anno-

tators to judge the correctness of the values in the two lexicons. ArSenL may have several sentiment values for the same entry, each with its own confidence score, so we used the sentiment values with the highest confidence score (averaged in the case of multiple answers). Since judging the values as real numbers is hard for humans, we map the sentiment scores into three classes of intensity (zero, up to 0.55 and above 0.55). An entry is correct only if the values of the positive and negative polarity classes are both correct. Each entry was judged by two annotators (without knowing its origin). They had to discuss and come to an agreement in the cases of disagreement (about 15% of the cases). SLSA and ArSenL have the exact same scores in 58.2% of the cases, which increases to 83.5% when mapping to the intensity classes.

Table 3 lists the accuracy of a majority baseline (neutral), SLSA and ArSenL for the different POS types[2]. SLSA gives error reductions of 58.7% and 37.8% over the baseline and ArSenL, respectively.

About 93% of SLSA errors are cases where the sentiment scores are doubtful in SentiWordNet, while the other errors are due to incorrect glosses in AraMorph. It might happen that an AraMorph entry is incorrectly linked to a SentiWordNet entry causing an error, but we do not see this in any of the manually analyzed data.

POS	Count	Baseline %	ArSenL %	SLSA %
NOUN	183	57.4	71.6	81.4
ADJ	50	42.0	66.0	74.0
VERB	62	43.5	58.1	80.6
ADV	2	50.0	100.0	100.0
All	293	51.9	68.0	80.1

Table 3: Accuracy results of a majority baseline (neutral), SLSA and ArSenL, evaluated on a test set that is covered in both SLSA and ArSenL

4.2 Extrinsic Evaluation

We conduct an extrinsic evaluation of SLSA on the task of sentiment analysis where a subjective sentence is classified to be either positive or negative. The performance is compared to that of ArSenL. We use an evaluation setup similar to the one described in (Badaro et al., 2014) using the corpus developed by Abdul-Mageed et al. (2011). The corpus involves 400 documents from the Penn Arabic Treebank (part 1 version 3) (Maamouri et al., 2004) where the sentences are tagged as objective, subjective-positive, subjective-negative and subjective-neutral. The evaluation only involves the sentences tagged as subjective-positive and subjective-negative. Random 80% of the sentences are used for training, while the rest are left for testing.

We train a Support Vector Machines classifier, through LIBSVM (Chang and Lin, 2011), using sentence vectors of three features representing the averages of the positive scores, negative scores and objective scores of the non-stop words in the sentence divided by the count of the underlying words. The scores are obtained by querying the lexicon using the lemma and POS information.

We optimize the classification to obtain the best F1-score based on five-fold cross validation on the training set using different SVM kernels and parameters. Polynomial kernels give the best weighted-average F1-score[3] of 68.6% (using SLSA), which is an absolute 0.2% improvement over linear kernels. Table 3 lists the precision, recall and F1-score of a majority baseline (subjective-negative), SLSA and ArSenL. SLSA provides absolute weighted-average F1-score improvements of 22.9% and 3.5% over the baseline and ArSenL, respectively.

		Baseline %	ArSenL %	SLSA %
Positive	F1	0.0	54.5	58.5
	Precision	0.0	56.5	61.8
	Recall	0.0	52.5	55.6
Negative	F1	75.4	72.0	75.2
	Precision	60.6	70.4	72.8
	Recall	100.0	73.7	77.6
Weighted-Ave. F1		45.7	65.1	68.6

Table 4: Sentiment analysis results of a majority baseline (subjective-negative), SLSA and ArSenL

5 Conclusion and Future Work

We have presented a publicly available large-scale Standard Arabic Sentiment Lexicon (SLSA) that avoids the deficiencies in the current lexicons. The construction of SLSA is based on linking the lexicon of AraMorph with SentiWordNet along with a few heuristics and powerful back-off. SLSA has the highest up-to-date reported coverage. SLSA shows a relative improvement of 37.8% over a state-of-the-art lexicon when tested for accuracy. It also outperforms it by an absolute 3.5% of F1-score when tested for sentiment analysis.

The future plans include manually correcting SLSA to reach a nearly 100% accuracy. Additionally, the work will be extended to the Arabic dialects for which AraMorph-like morphological analyzers are available. We also plan to study the cases where English and Arabic translations have different sentiments due to cultural differences.

[2]There are only few adverbs in the test set because they are rare in Arabic, where only 0.1% of the lexicon entries are adverbs.

[3]The weighted-average F1-score is the sum of the F-1 score of the positive class and the F1-score of the negative class, each multiplied by its percentage.

References

Muhammad Abdul-Mageed and Mona Diab. 2012. Toward building a large-scale Arabic Sentiment Lexicon. In *The Sixth International Global WordNet Conference*, Matsue, Japan.

Muhammad Abdul-Mageed and Mona Diab. 2014. SANA: A Large Scale Multi-Genre, Multi-Dialect Lexicon for Arabic Subjectivity and Sentiment Analysis. In *The Ninth international conference on Language Resources and Evaluation (LREC)*, Reykjavik, Iceland.

Muhammad Abdul-Mageed, Mona Diab, and Mohammed Korayem. 2011. Subjectivity and Sentiment Analysis of Modern Standard Arabic. In *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, Oregon, USA.

Samah Alhazmi, William Black, and John McNaught. 2013. SentiWordNet in Relation to SentiWordNet 3.0. *International Journal of Computational Linguistics*, 4.

Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. 2010. SentiWordNet 3.0: An Enhanced Lexical Resource for Sentiment Analysis and Opinion Mining. In *The Seventh international conference on Language Resources and Evaluation (LREC)*, Malta.

Gilbert Badaro, Ramy Baly, Hazem Hajj, Nizar Habash, and Wassim El-Hajj. 2014. A Large Scale Arabic Sentiment Lexicon for Arabic Opinion Mining. In *Arabic Natural Language Processing Workshop (WANLP), EMNLP*, Doha, Qatar.

William Black, Sabri Elkateb, Horacio Rodriguez, Musa Alkhalifa, Piek Vossen, Adam Pease, and Christiane Fellbaum. 2006. Introducing the Arabic WordNet Project. In *The Third International WordNet Conference (GWC)*, Jeju Island, Korea.

Tim Buckwalter. 2004. *Buckwalter Arabic Morphological Analyzer Version 2.0*. Linguistic Data Consortium, Philadelphia.

Chih-Chung Chang and Chih-Jen Lin. 2011. Libsvm: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2.

Mona Diab, Mohamed Al-Badrashiny, Maryam Aminian, Mohammed Attia, Pradeep Dasigi, Heba Elfardy, Ramy Eskander, Nizar Habash, Abdelati Hawwari, and Wael Salloum. 2014. Tharwa: A Large Scale Dialectal Arabic - Standard Arabic - English Lexicon. In *The Ninth international conference on Language Resources and Evaluation (LREC)*, Reykjavik, Iceland.

Alaa El-Halees. 2011. Arabic Opinion Mining Using Combined Classification Approach. In *The International Arab Conference on Information Technology (ACIT)*, Amman, Jordan.

Mohamed Elarnaoty, Samir AbdelRahman, and Aly Fahmy. 2012. A Machine Learning Approach For Opinion Holder Extraction in Arabic Language. In *Corr*, Amman, Jordan.

Mohamed Elhawary and Mohamed Elfeky. 2010. Mining Arabic Business Reviews. In *IEEE International Conference on Data Mining Workshops*, Sydney, Australia.

David Graff, Mohamed Maamouri, Basma Bouziri, Sondos Krouna, Seth Kulick, and Tim Buckwalter. 2009. Standard Arabic Morphological Analyzer (SAMA) Version 3.1. Linguistic Data Consortium LDC2009E73.

Mohamed Maamouri, Ann Bies, Tim Buckwalter, and Wigdan Mekki. 2004. The penn arabic treebank: Building a large-scale annotated arabic corpus. In *NEMLAR Conference on Arabic Language Resources and Tools*, Cairo, Egypt.

Fawaz H.H. Mahyoub, Muazzam A. Siddiqui, and Mohamed Y. Dahab. 2014. Building an Arabic Sentiment Lexicon Using Semi-supervised Learning. *Journal of King Saud University - Computer and Information Sciences*, 26.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In *The 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, Sydney, Australia.

George A. Miller, Richard Beckwith, and Derek Gross Christiane Fellbaum, and Katherine Miller. 1990. Introduction to WordNet: An On-line Lexical Database. *International Journal of Lexicography*, 3.

Hanaa Mobarz, Mohsen Rashwan, and Ibrahim Abdel-Rahman. 2011. Generating lexical Resources for Opinion Mining in Arabic Language Automatically. In *The Eleventh Conference on Language Engineering (ESOLE)*, Cairo-Egypt.

Mike Thelwall, Kevan Buckley, Georgios Paltoglou, and Di Cai. 2010. Sentiment Strength Detection in Short Informal Text. *Journal of the American Society for Information Science and Technology*, 61.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing Contextual Polarity in Phrase-Level Sentiment Analysis. In *The First conference on Human Language Technology and Empirical Methods in Natural Language Processing (EMNLP)*, Vancouver, Canada.

Reinforcing the Topic of Embeddings with Theta Pure Dependence for Text Classification*

Ning Xing[1], **Yuexian Hou**[1], **Peng Zhang**[1], **Wenjie Li**[2], **Dawei Song**[1]
[1]School of Computer Science and Technology, Tianjin University, China
[2]Department of Computing, The Hong Kong Polytechnic University, Hong Kong
{xingning,yxhou,pzhang,dwsong}@tju.edu.cn,cswjli@comp.polyu.edu.hk

Abstract

For sentiment classification, it is often recognized that embedding based on distributional hypothesis is weak in capturing sentiment contrast–contrasting words may have similar local context. Based on broader context, we propose to incorporate Theta Pure Dependence (TPD) into the Paragraph Vector method to reinforce topical and sentimental information. TPD has a theoretical guarantee that the word dependency is pure, i.e., the dependence pattern has the integral meaning whose underlying distribution can not be conditionally factorized. Our method outperforms the state-of-the-art performance on text classification tasks.

1 Introduction

Word embeddings can be learned by training a neural probabilistic language model or a unified neural network architecture for various NLP tasks (Bengio et al., 2003; Collobert and Weston, 2008; Collobert et al., 2011). In global context-aware neural language model (Huang et al., 2012), the global context vector is a weighted average of all word embeddings of a single document/paragraph. After trained with all word embeddings belonging to the current paragraph, a resulting Paragraph Vector can be obtained. Actually, Le and Mikolov's Paragraph Vector (Le and Mikolov, 2014) is trained based on the log-linear neural language model (Mikolov et al., 2013a).

For text classification, using a straightforward extension of language model (e.g. Le and Mikolov's Paragraph Vector) is considered not to be sensible. Embeddings learned for text classification should be very different from that learned for language modeling. For example, language models often calculate the probability of a sentence, therefore *this is a good movie* and *this is a bad movie* may not be discriminated from each other. In sentiment analysis task, the semantic representation of words needs to tell word *good* from *bad*, even if the two words have the same local context. For this reason, the local dependency is insufficient to model topical or sentiment information. Fortunately, if we have the global context of *good* like *interesting* or *amazing*, the sentiment meaning of the embedding will be explicit. However, the training of log-linear neural language model is based on local word dependencies (e.g., the co-occurrence of the words in a local window). Thus, Paragraph Vector can not explicitly model the word dependencies for those words that do not frequently appear in a local window but are actually closely dependent on each other.

In this paper, our aim is to extend the Paragraph Vector with global context which can capture topical or sentiment information effectively. However, if one explicitly considers the dependency patterns that are beyond the local window level, there is a possibility that the noisy dependency patterns can be involved and modeled in the distributed representation methods. Moreover, there should be an unique and explicit topical meaning in the patterns to guarantee no ambiguity in the global context. Therefore, we need a dependency mining method that not only models the long range dependency patterns, but also provides a theoretical guarantee that the dependency patterns are pure. Here, the "pure" dependency pattern is an integral semantic meaning/concept that cannot be factorized into sub dependency patterns.

In the language of statistics, Conditional Pure Dependence (CPD) means that the underlying distribution of the dependency patterns cannot be factorized under certain conditions (e.g., priors, observed words, etc.). It has been proved that CPD is the high-level pure dependence in (Hou et al.,

*Corresponding authors: Yuexian Hou and Peng Zhang.

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 2551–2556,
Lisbon, Portugal, 17-21 September 2015.

2013). However, judging CPD is NP-hard (Chickering et al., 2004). Fortunately, Theta Pure Dependence (TPD) is the sufficient criteria of CPD and can be identified in O(N) time, where N is the number of words (Hou et al., 2013). This finding motivates us to adopt TPD as the global context. Moreover, compared with other conventional co-occurrence-based methods, such as the Apriori algorithm (Agrawal et al., 1993), TPD based on the Information Geometry (IG) framework has a solid theoretical interpretations in statistics to guarantee the dependence is pure.

2 Modeling Topic with TPD

Compared with local context, global context can usually capture the text topic more precisely. It is easy to get local context by a sliding window. We define the centered word as the *current word* and the other words in the window as *local context words*. Global context words are extracted from all the documents in the corpus and can be divided into two parts: a) the words in the current document but outside of the local context window; b) the words never appeared in the document but in the corpus. The following example shows the words mentioned above, and the topic (the scene of filming) is easily captured by TPD:

- TPD: **scene** camera acting movie
  Text: there [*is great atmosphere in the* **scene** *from the location , the*] lighting , the fog and such , but the camera should be slowly following the killer...

The bracket stands for the local context window, and the size of window is 5, i.e. there are five local context words (in italics) in both sides of the current word (in bold). Global context words are underlined in the example.

In order to model the topic explicitly, the dependence pattern should report one and only one topical meaning. TPD has a theoretical guarantee that the dependency has an integral meaning whose underlying distribution can not be conditionally factorized. Formally, given a set of binary random variables $\mathbb{X} = \{X_1, \ldots, X_n\}$, where X_i denotes the occurrence $(X_i = 1)$ or absence $(X_i = 0)$ of the i-th word. Then the n-order TPD over $\mathbb{X}$ can be defined as follows.

DEFINITION 1. (TPD): $\mathbb{X} = \{X_1, \ldots, X_n\}$ is of n-order Theta Pure Dependence (TPD), iff the n-order θ coordinate $\theta_{12\ldots n}$ is significantly different from zero. (Hou et al., 2013)

TPD can be effectively identified by an explicit statistical test procedure: Log Likelihood Ratio Test (LLRT) (Nakahara and Amari, 2002) for θ-coordinate of IG. (Hou et al., 2013)

Here, we introduce two negative examples to further emphasize the importance of utilizing TPD. Example 1: *can, with, of*. The joint distribution of this words combination can be unconditionally factorized directly, since the occurrence of any word does not necessarily imply the occurrence of others. Example 2: *London, Chelsea, Sherlock Holmes*. As we all know, both *Chelsea* and *Sherlock Holmes* are closely related to *London*. *Chelsea* and *Sherlock Holmes* are two relatively independent topics, i.e. they are conditional independent given *London*. Although the three phrases are unconditionally dependent, their joint distribution can be conditionally factorized. Thus the dependency in both two examples can not be pure.

To explain TPD and the characteristic "pure" intuitively, let us look at a typical example of TPD: *climate, conference, Copenhagen*. The co-occurrence of the three words implies an unseparable high-level semantic entity compared with the two negative examples, introduced above. In negative examples, the high frequency of words co-occurrence can be explained as some kind of "coincidence", because each of them or their pairwise combinations has a high frequency, independently. However, the co-occurrence of TPD words cannot be fully explained as the random coincidence of, e.g., the co-occurrence of *Copenhagen* and *conference* (which can be any other conferences in Copenhagen) and the occurrence of *climate*.

The word "pure" in Hou et al. (2013) means that the joint probability distribution of these words is significantly different from the product of lower-order joint distributions or marginal distributions, w.r.t all possible decompositions. More formally, it requires that the joint distribution cannot be factorized unconditionally (UPD) or conditionally (CPD) in the language of graphical model. Let $x_i \in \{0, 1\}$ denote the value of X_i. Let $p(x)$, $x = [x_1, x_2, \ldots, x_n]^T$, be the joint probability distribution over $\mathbb{X}$. Then the definitions of UPD and CPD are as follows:

DEFINITION 2. (UPD): $\mathbb{X} = \{X_1, \ldots, X_n\}$ is of n-order Unconditional Pure Dependence (UPD), iff it can NOT be unconditionally factorized, i.e., there does NOT exist a k-partition $\{\mathbb{C}_1, \mathbb{C}_2, \ldots, \mathbb{C}_k\}$ of $\mathbb{X}$, $k > 1$, such at $p(x) =$

$p(c_1) * p(c_2)...p(c_k)$, where $p(c_i)$, $i = 1, \ldots, k$, is the joint distribution over $\mathbb{C}_i$. (Hou et al., 2013)

DEFINITION 3. (CPD): $\mathbb{X} = \{X_1, \ldots, X_n\}$ is of n-order Conditional Pure Dependence (CPD), iff it can NOT be conditionally factorized, i.e., there does NOT exist $\mathbb{C}_0 \subset \mathbb{X}$ and a k-partition $\{\mathbb{C}_1, \mathbb{C}_2, \ldots, \mathbb{C}_k\}$ of $\mathbb{V} = \mathbb{X} - \mathbb{C}_0$, $k > 1$, such at $p(v|c_0) = p(c_1|c_0) * p(c_2|c_0)...p(c_k|c_0)$, where $p(v|c_0)$ is the conditional joint distribution over $\mathbb{V}$ given $\mathbb{C}_0$, and $p(c_i|c_0)$, $i = 1, 2, \ldots, k$, is the conditional joint distribution over $\mathbb{C}_i$ given $\mathbb{C}_0$. In case that $\mathbb{C}_0$ is an empty set, we define $p(c_0) = 1$. (Hou et al., 2013)

Actually, CPD is stricter than UPD, and the dependence which just satisfies UPD is not pure enough to model the global context. Therefore, "pure" in our paper refers to the characteristic of CPD. However judging CPD is NP-hard. It is proved that a significant nonzero n-order θ parameter (TPD) entails the n-order CPD/UPD in Hou et al. (2013). The highest-order coordinate parameter in IG is a proper metric for the purity (i.e., the unique semantics) of high-order dependence. A pattern is TPD, iff the n-order θ coordinate $\theta_{12...n}$ is significantly different from zero. Moreover, The Log Likelihood Ratio Test implemented in the mixed coordinates can test whether $\theta_{12...n}$ is significantly different from zero.

Contrasting to TPD, the semantic coupling among the associations in the two negative examples is much weaker. In conclusion, *can, with, of* cannot give an explicit topic and *London, Chelsea, Sherlock Holmes* includes at least two topics. the co-occurrence of words in TPD (e.g. *climate, conference, Copenhagen*) implies an un-separable (pure) high-level semantic entity. A sufficient and unbroken meaning of dependence can not only supply the context but also avoid the ambiguity (or noise) in global context. Therefore, the meaning of pure is important in such a global context modeling method.

3 Global PV-DBOW and Dependence Vectors

A version of Paragraph Vector in Le and Mikolov (2014) PV-DBOW is extended with TPD to a new model: Global PV-DBOW (Glo-PV-DBOW). TPD has been extracted from the corpus before training. Given a sequence of training words w_1, w_2, w_3, ..., w_T and the global context glo_t of w_t, the objective of Glo-PV-DBOW is to maximize the average log probability:

$$L = \frac{1}{T}\sum_{t=1}^{T}\Big[\sum_{-c\le j\le c, j\neq 0} \log p(w_t|w_{t+j}) + \log p(w_t|glo_t) + \log p(w_t|doc_t)\Big] \quad (1)$$

where c is the local context window size. The indicator of the document that the current word w_t belongs to is denoted by doc_t. Further, we define $p(w_t|glo_t)$ in equation (2):

$$p(w_t|glo_t) = \prod_{a}^{A}\Big[p(w_t|dep_t^a)p(w_t|w_1^a, w_2^a, \ldots, w_N^a)\Big] \quad (2)$$

The indicator of the a-th w_t's TPD pattern is denoted as dep_t^a and can be trained to be a distributed representation of TPD: dependence vector $v_{dep_t^a}$. This (N+1)-order TPD consists of N+1 words: w_1^a, $w_2^a...w_N^a$ and w_t. The energy function of w_t and $w_i = (w_{t+j}, doc_t, dep_t^a)$ is uniform as follows:

$$E(w_t, w_i) = -{v_{w_t}}^T v_{w_i} \quad (3)$$

We define the energy function of TPD words:

$$E(w_t, w_1^a, w_2^a, \ldots w_N^a) = -\frac{1}{N}\sum_{n=1}^{N}{v_{w_t}}^T v_{w_n^a} \quad (4)$$

The resulting predictive distributions are given by

$$p(w_t|w_i) = \frac{exp({v_{w_t}}^T v_{w_i})}{\sum_{m=1}^{W} exp({v_{w_m}}^T v_{w_i})} \quad (5)$$

$$p(w_t|w_1^a, w_2^a, \ldots, w_N^a) = \frac{exp(\frac{1}{N}\sum_{n=1}^{N}{v_{w_t}}^T v_{w_n^a})}{\sum_{m=1}^{W} exp(\frac{1}{N}\sum_{n=1}^{N}{v_{w_m}}^T v_{w_n^a})} \quad (6)$$

Hierarchical softmax (Morin and Bengio, 2005) is adopted to reduce the cost of computation. The binary tree is specified with a Huffman tree, and the Huffman code of pseudo words m_i in w_t's Huffman path is denoted as x_{m_i}. For more about hierarchical softmax we used, please refer to (Mikolov et al., 2013b). Using stochastic gradient descent (SGD), distributed representations of the word, dependence and document have been trained. The update procedure of $v_{w_i} = (v_{w_{t+j}}, v_{doc_t}, v_{dep_t^a})$ is as same as the procedure described in (Mikolov et al., 2013b). Thus, the pseudo code for training TPD words is listed individually:

SGD FOR TRAINING THE TPD WORDS

$v_{w_t} \leftarrow current\_word$
$v_{w^a_{ave}} \leftarrow \frac{1}{N}\sum_{n=1}^{N} v_{w^a_n}$
$err \leftarrow 0$
for $\forall m_i$
    do $g \leftarrow (1 - x_{m_i} - \sigma({v_{m_i}}^T v_{w^a_{ave}})) * \alpha$
        $err+ = g * \frac{1}{N} * v_{m_i}$
        $m_i+ = g * v_{w^a_{ave}}$
for $n \leftarrow 1$ **to** N
    do $v_{w^a_n}+ = err$

4 Experiments

Apriori (not a pure dependency method) is contrastively adopted to implement Glo-PV-DBOW. Glo-PV-DBOW-TPD and Glo-PV-DBOW-Apri are all evaluated in two text classification tasks: sentiment analysis and topic discovery. The suffix (e.g., -2, -5) of our global method name denotes the order of dependency (the number of words in a dependence pattern). The order of dependency is changed because we want to show the superiority of the high-order TPD. The high-order TPD provides the more rich and explicit global context than the lower-order one since the high-order TPD cannot be reduced to the random coincidence of lower-order dependencies.

We cross-validate the hyperparameters and set the local context window size as 10, the dimension of embeddings as 100. In sentiment analysis task, Apriori's minimum support and TPD's theta_0 is respectively set as 0.004 and 1.4. While in topic discovery task, Apriori's minimum support and TPD's theta_0 is around 0.020 and 2.0 respectively. Since the classification accuracy of the approaches compared is a single result, we do not include any results for test of significance in our method and only report the average accuracy.

4.1 Sentiment Analysis on Movie Reviews

The binary sentiment classification on the IMDB dataset proposed by (Maas et al., 2011) is conducted. Results in Fig.1 show that global methods' performance is more stable than PV-DBOW's. Moreover, TPD works much better than Apriori, especially in the high-order dependence. Note that TPD-5 works better than TPD-2, while Apri-5 works worse than Apri-2. It can be explained that the Apriori algorithm is short of an explicit statistical test procedure to guarantee the pure dependence. Therefore, the Apriori algorithm is not suitable for generating the high-order dependence.

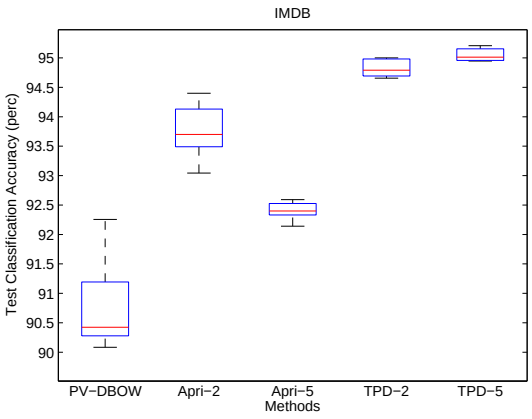


Figure 1: Box plot of classification accuracy over a local method (PV-DBOW) and 4 global methods (Apri-2/5, TPD-2/5).

Instead, the high-order TPD can provide the rich and explicit global context for the model. Meanwhile, it is verified that our method is good at capturing sentiment contrast.

Table 1 shows that Glo-PV-DBOW with 5-order TPD achieves the state-of-the-art performance. A promising result is an improvement of more than 2% over result published in Le and Mikolov (2014). Note that the algorithm process of Paragraph Vector (Le and Mikolov, 2014) is much more complex than PV-DBOW's. Paragraph Vector includes an extra inference stage. In addition, Paragraph Vector's document vector is a combination of two vectors: one learned by PV-DBOW and the other learned by Distributed Memory Model of Paragraph Vectors (PV-DM) (Le and Mikolov, 2014). The combined document vector has 800 dimensions, while all vectors in our experiments only have 100 dimensions.

4.2 Topic Discovery on News

The 20 Newsgroups dataset is a collection of approximately 20,000 newsgroup documents, partitioned across 20 different newsgroups. We follow (Crammer et al., 2012) to create binary problems from the dataset by creating binary decision problems of choosing between two similar groups. Therefore, the dataset is split into two sub-datasets as follows: *comp: comp.sys.ibm.pc.hardware vs. comp.sys.mac.hardware* and *sci: sci.electronics vs. sci.med*. Similarly, 1800 examples balanced between the two labels were selected for each problem.

The classification accuracy on each sub-dataset

Table 1: The performance of our method compared with other approaches on the IMDB dataset.

Model	Accuracy rate
BoW (bnc) (Maas et al., 2011)	87.80%
Full+Unlabeled+BoW (Maas et al., 2011)	88.89%
WRRBM (Dahl et al., 2012)	87.42%
WRRBM + BoW (bnc) (Dahl et al., 2012)	89.23%
SVM-bi (Wang and Manning, 2012)	89.16%
NBSVM-bi (Wang and Manning, 2012)	91.22%
PV-DBOW (Le and Mikolov, 2014)	90.79%
Paragraph Vector (Le and Mikolov, 2014)	92.58%
Sentence Vector + RNN-LM + NB-LM (Mesnil et al., 2014)	92.57%
mvCNN**o&w** (Johnson and Zhang, 2015)	93.34%
Glo-PV-DBOW-Apri-2	93.76%
Glo-PV-DBOW-Apri-5	92.41%
Glo-PV-DBOW-TPD-2	94.83%
Glo-PV-DBOW-TPD-5	**95.05%**

Table 2: The performance of our method compared to other approaches on 20 Newsgroup.

Model	Comp	Sci
Confidence-weighted (Crammer et al., 2012)	94.39%	97.56%
PV-DBOW (Le and Mikolov, 2014)	92.60%	98.02%
Glo-PV-DBOW-Apri-2	94.56%	98.42%
Glo-PV-DBOW-Apri-5	94.43%	98.13%
Glo-PV-DBOW-TPD-2	94.59%	**99.20%**
Glo-PV-DBOW-TPD-5	**95.47%**	98.74%

is recorded in Table 2. Compared with Confidence-weighted (Crammer et al., 2012) and PV-DBOW (Le and Mikolov, 2014), our extended models achieve the highest accuracy on each subdataset. Moreover, TPD as a pure dependence works better than Apriori when they provide the global context for our model. The topical information is effectively reinforced in embeddings by incorporating TPD.

4.3 Analysis on Word Embeddings

The cosine similarity of each word pair in 20 Newsgroups is computed. We list four center words and their nearest neighbors in PV-DBOW and Glo-PV-DBOW groups respectively. The rankings are labeled in front of neighbor words, and some notable neighbor words are in bold.

From Table 3, we can see that the statistical information of corpus like words co-occurrence can be mined by TPD. Therefore, the Glo-PV-DBOW's embeddings are context-aware and it can help a lot for classification tasks. The top 40 nearest neighbors of *ibm* are investigated, and we find *macintosh* and *mac* appeared in the PV-DBOW group but not in the Glo-PV-DBOW group. In

Table 3: Nearest neighbors of words ranking list based on cosine similarity.

Center word	PV-DBOW	Glo-PV-DBOW
ibm	1:aix	1:aix
	2:pc	2:pc
	...	3:pc's
	23:**macintosh**	4:**austin**
	34:**mac**	5:**workstations**
mac	1:macintosh	1:macintosh
	2:quicktime	2:**apple's**
	3:portable	3:quicktime
	4:utilities	4:**apple**
	5:**macs**	5:**macs**
486	1:386	1:386
	2:486dx	2:**cpu**
	3:33mhz	3:486dx
	4:486dx2	4:486dx2
	5:**cpu**	5:33mhz
Kingston	1:aix	1:aix
	2:mike	2:**ibm**
	3:sharks	3:jones
	4:jones	4:sharks
	5:**ibm**	5:mike

the corpus, the topic of documents is either *ibm* or *mac*. If we perform a classification task on "*ibm* versus *mac*", it will be hard to classify in the PV-DBOW group. That is because PV-DBOW tends to regard *ibm* and *mac* both as computers. However, the two different computer brands are distinguished in Glo-PV-DBOW. Further, *ibm* and *mac* co-occur rarely in one document, and the statistical information is noted by TPD.

5 Conclusion

This paper proposes to incorporate Theta Pure Dependence into Paragraph Vector to capture more topical and sentimental information in the context. The extended model is applied to a sentiment classification task and a topical detection task. Our accuracy outperforms the state-of-the-art result on the movie and news datasets. The approach can be improved further to fully leverage the un-factorized sense of high-order Theta Pure Dependence. In future, we will explore the applications of dependence distributed representation.

Acknowledgments

This work is funded in part by the Chinese 863 Program (grant No. 2015AA015403), the Key NSF Project of Chinese Tianjin (grant No. 15JCZDJC31100), the Chinese 973 Program (grant No. 2013CB329304 and 2014CB744604), the Chinese NSF Project (grant No. 61272291, 61402324 and 61272265).

References

Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. 1993. Mining association rules between sets of items in large databases. In *ACM SIGMOD Record*, volume 22, pages 207–216. ACM.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *The Journal of Machine Learning Research*, 3:1137–1155.

David Maxwell Chickering, David Heckerman, and Christopher Meek. 2004. Large-sample learning of bayesian networks is np-hard. *The Journal of Machine Learning Research*, 5:1287–1330.

Ronan Collobert and Jason Weston. 2008. A unified aagrawal1993miningrchitecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM.

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

Koby Crammer, Mark Dredze, and Fernando Pereira. 2012. Confidence-weighted linear classification for text categorization. *The Journal of Machine Learning Research*, 13(1):1891–1926.

George E Dahl, Ryan P Adams, and Hugo Larochelle. 2012. Training restricted boltzmann machines on word observations. *Proceedings of International Conference on Machine Learning*.

Yuexian Hou, Xiaozhao Zhao, Dawei Song, and Wenjie Li. 2013. Mining pure high-order word associations via information geometry for information retrieval. *ACM Transactions on Information Systems (TOIS)*, 31(3):12.

Eric H Huang, Richard Socher, Christopher D Manning, and Andrew Y Ng. 2012. Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 873–882. Association for Computational Linguistics.

Rie Johnson and Tong Zhang. 2015. Semi-supervised learning with multi-view embedding: Theory and application with convolutional neural networks. *arXiv preprint arXiv:1504.01255*.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1188–1196.

Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 142–150. Association for Computational Linguistics.

Grégoire Mesnil, Marc'Aurelio Ranzato, Tomas Mikolov, and Yoshua Bengio. 2014. Ensemble of generative and discriminative techniques for sentiment analysis of movie reviews. *arXiv preprint arXiv:1412.5335*.

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

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

Frederic Morin and Yoshua Bengio. 2005. Hierarchical probabilistic neural network language model. In *Proceedings of the international workshop on artificial intelligence and statistics*, pages 246–252. Citeseer.

Hiroyuki Nakahara and Shun-ichi Amari. 2002. Information-geometric measure for neural spikes. *Neural Computation*, 14(10):2269–2316.

Sida Wang and Christopher D Manning. 2012. Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pages 90–94. Association for Computational Linguistics.

That's So Annoying!!!: A Lexical and Frame-Semantic Embedding Based Data Augmentation Approach to Automatic Categorization of Annoying Behaviors using *#petpeeve* Tweets *

William Yang Wang and **Diyi Yang**
Language Technologies Institute
School of Computer Science
Carnegie Mellon University
{yww,diyiy}@cs.cmu.edu

Abstract

We propose a novel data augmentation approach to enhance computational behavioral analysis using social media text. In particular, we collect a Twitter corpus of the descriptions of annoying behaviors using the *#petpeeve* hashtags. In the qualitative analysis, we study the language use in these tweets, with a special focus on the fine-grained categories and the geographic variation of the language. In quantitative analysis, we show that lexical and syntactic features are useful for automatic categorization of annoying behaviors, and frame-semantic features further boost the performance; that leveraging large lexical embeddings to create additional training instances significantly improves the lexical model; and incorporating frame-semantic embedding achieves the best overall performance.

1 Introduction

In the ever-expanding era of social media, many scientific disciplines, such as health and healthcare, biology, and learning sciences, have adopted computational approaches to exploit patterns and behaviors in large datasets (Wang et al., 2015; Chen and Lonardi, 2009; Baker and Yacef, 2009). In contrast, the primary methods for behavioral sciences still rely on lab experiments with limited amount of subjects, which are time consuming and financially expensive. In addition to this, it is also difficult to obtain a set of samples with geographical variations in traditional lab-based behavioral experiments.

I really hate being interrupted while I'm talking to someone #petpeeve #canyounot 🤦

Figure 1: An anonymized example of *#petpeeve* tweets.

While the social media data are abundantly available, computational approaches to behavioral sciences using Twitter are not well-studied. Even when statistical techniques are applied to these tasks, their concentration has been on simple statistical significance tests and descriptive statistics (De Charms, 2013; Zhang et al., 2013). Therefore, we believe that statistical natural language processing techniques are needed for insightful analysis and interpretation in behavioral studies.

In this paper, we use Twitter as a corpus for computational behavioral science. More specifically, we focus on a case study of analyzing annoying behaviors. To do this, we exploit a corpus of 9 million tweets (Cheng et al., 2010), and extract the tweets that describe these behaviors using the *#petpeeve* hashtags. *#petpeeve* is a popular Twitter hashtag, which describes behaviors that might be annoying to others. An example of *#petpeeve* tweets is shown in Figure 1. To facilitate the analysis, we manually annotate 3,375 tweets with 60 fine-grained categories, which will be described in Section 3. We use a sparse mixed-effects topic model to analyze the salient words in each category, as well as the geographic variations. We show that lexical, syntactic, and semantic features enhance the automatic categorization of annoying behaviors; and that the performance is further improved with a novel lexical and frame-semantic embedding based data augmentation ap-

*We understand that many people find long titles annoying, so we intentionally use a very long one to help people understand what "pet peeve" means.

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 2557–2563,
Lisbon, Portugal, 17-21 September 2015.

proach. Our main contributions are three-fold:

- We provide a Twitter corpus with fine-grained annotations for computational behavior studies;
- We qualitatively analyze the Twitter language concerning annoying behaviors, with a focus on the topics and geographical variations;
- We propose various linguistic features and a novel data augmentation approach for automatic categorization of annoying behaviors.

We outline related work in the next section. The dataset is described in Section 3. We introduce the approach for analyzing *#petpeeve* Tweets in Section 4. Experimental results are shown in Section 5. We discuss possible applications in Section 6, and conclude in Section 7.

2 Related Work

Psychologists, behavioral scientists, and computer scientists have studied a wide-range of methods for behavior extraction (Mast et al., 2015). For example, in lab experiments, arm and body postures (Marcos-Ramiro et al., 2013) are often used to extract self-touch and gestures, while eye gaze (Funes Mora and Odobez, 2012), head pose (Ba and Odobez, 2011), face location and motion (Nguyen et al., 2012), and full-body pose (Shotton et al., 2013) can also be used as cues to extract gazing, nodding, and arm-related behaviors. There are also significant amount of studies of extracting facial and speech features to understand smiling (Bartlett et al., 2008), eye contact (Marin-Jimenez et al., 2014), and verbal behaviors (Basu, 2002).

With the surge of interest in computational social science (Lazer et al., 2009), Twitter has become a popular resource to study data-driven methods in social science (Miller, 2011). For example, O'Connor et al. (2010a) align the Twitter messages with public opinion time series to study computational political science. Ritter et al. (2010) study Twitter dialogues using a clustering approach. Bollen et al. (2011) use a sentiment analysis approach to predict the American stock market via Twitter. Li et al. (2014b) have investigated the alignment of Twitter mood with weather for sentiment analysis. In recent years, language technology researchers have focused on developing genre-specific Twitter part-of-speech tagging (Gimpel et al., 2011), named entity recognition (Ritter et al., 2011), summarization (O'Connor et al., 2010b), sentiment analysis (Agarwal et al., 2011), event extraction (Ritter et al., 2012; Li et al., 2014a), paraphrasing (Xu et al., 2014), machine translation (Ling et al., 2013), and dependency parsing (Kong et al., 2014) methods. To the best of our knowledge, even though there have been studies on using Twitter hashtags to study language-related behaviors (González-Ibánez et al., 2011; Bamman and Smith, 2015), Twitter NLP approaches to non-linguistic behaviors are not well studied in general.

Label	%	Label	%
appearance	.14	services	.02
disrespect	.06	traffic	.02
language	.06	advertisement	.01
hygiene	.05	bragging	.01
relationship	.05	children	.01
dishonesty	.03	complaining	.01
hypocrisy	.03	indolence	.01
incompetence	.03	physical	.01
interruption	.03	punctuality	.01
monetary	.03	racial	.01
sexual	.03	religious	.01
arrogance	.02	selfishness	.01
celebrity	.02	silence	.01
ignorance	.02	smoking	.01
privacy	.02	talkative	.01
products	.02	weather	.01

Table 1: The categories and percentages of annoying behaviors in *#petpeeve* tweets in our dataset. Note that 17% of the *#petpeeve* tweets are identified as *other* unrelated behaviors (not shown).

3 The Dataset

We use the Twitter corpus with 9 million sampled messages collected in prior work (Cheng et al., 2010), which includes a total of 121K users. The dataset includes latitude and longitude information.

We extract 3,375 tweets[1] with *#petpeeve* hashtags. We follow past work to annotate the tweets (Ritter et al., 2012; Li et al., 2014a): we apply the *LDA clustering + human-identification* approach to label the categories of the described annoying behaviors in these tweets. The human annotation process includes two stages: first, the annotators identify the 50 categories from the clustering process, and use these topics as a candi-

[1] http://www.cs.cmu.edu/~yww/data/petpeeves.zip

date label set to annotate the data; in the second stage, the categories are refined (to 60 classes) from the first pass, and the data is re-annotated with the refined human-specified category labels. Due to the complexity of this fine-grained annotation task, the inter-annotator agreement rate between two annotators is moderate (0.445).

The annotated categories and label distribution[2] of the dataset are shown in Table 1. In our random samples, the states that post the most *#petpeeve* tweets are NY, MD, CA, NJ, FL, GA, VA, TX, NC, PA, and DC. In our predictive experiments, we randomly select 60% of tweets for training, and 40% for testing.

4 Our Approach

In this section, we describe our methods for the qualitative and quantitative analyses. In particular, we briefly review a supervised approach of using sparse mixed-effects topic model to visualize the topical words to analyze this behavior data. For the quantitative task of automatic categorization of tweets, we propose a novel approach to create additional training data, using continuous lexical and semantic representations.

4.1 Supervised Topic Modeling

To analyze the salient words for each category of annoying behaviors, we utilize SAGE (Eisenstein et al., 2011), a state-of-the-art mixed-effect topic model, which has been used in several NLP applications (Sim et al., 2012; Wang et al., 2012). SAGE is ideal for our text analytic purposes, because it is supervised, and it builds relatively clean topic models by considering the additive effects and the background distribution of words. Therefore, we can use SAGE to visualize the salient words for each category of annoying behaviors using the 3,375 *#petpeeve* tweets. Each tweet is treated as a document, and we use Markov Chain Monte Carlo for inference. To facilitate the geographical analysis, we use Google's reverse geocoding service to extract the state information from coordinates, and apply SAGE for visualization.

[2]The categories that are not shown in the table are *backstabbing, boring, copycat, drinking, drug, empty promise, impoliteness, inconsiderate, indirect, insecurity, interference, irresponsible, jealous, judge, loneliness, misunderstanding, negativity, noisy, parents, politics, repetition, showoff, snobbish, stability, swearing, time-wasting, ungratefulness, and others*.

4.2 Embedding-Based Data Augmentation for Automatic Categorization of Tweets

In addition to the visualization task, we also ask the question: can we use linguistic cues to predict tweets that describe different annoying behaviors? We formulate the problem as a multiclass classification task, and consider the following feature sets:

- **Lexical Features**: we extract unigrams as surface-level lexical features.
- **Part-of-Speech Features**: to model shallow syntactic cues, we extract lexicalized part-of-speech features using the Stanford part-of-speech tagger (Toutanova et al., 2003).
- **Dependency Triples**: to better understand the deeper syntactic dependencies of keywords in tweets, we have also extracted typed dependency triples (e.g., *nsubj(hate,I)*) using the MaltParser (Nivre et al., 2007).
- **Frame-Semantics Features**: SEMAFOR (Das et al., 2010) is a state-of-the-art frame-semantics parser that produces FrameNet-style semantic annotation. We use SEMAFOR to extract frame-level semantic features.

Embeddings for Data Augmentation Since the Twitter messages are often short and noisy, and the training data is relatively scarce for each class, we consider the feasibility of leveraging external resources, in particular, continuous word embeddings (Mikolov et al., 2013a) to enhance the multiclass text categorization model.

Two major challenges for leveraging word embeddings for tweet classification are: 1) because word embeddings are continuous, it is difficult to fuse them with other discrete syntactic and semantic features; 2) it is not straightforward how one should transform the word-level representation to the tweet-level representation. In our preliminary experiments, we have evaluated the continuous word representation method (Turian et al., 2010), as well as incorporating neighboring words in the embeddings as additional features, but both methods fail to outperform the lexical baseline that uses only bag-of-word unigrams.

To solve this problem, we propose the use of neighboring words in continuous representations to create new instances to augment the training

weather	**ungratefulness**	**traffic**	**timewasting**	**talkative**	**swearing**	**stability**	**snobbish**
rains	helped	cop	wastingmytime	Tweeters	curse	mood	smut
STORM	ungrateful	lane	colleagues	Xs	teary	sensitive	intellectual
Blizzarad	clearly	pulled	Wen	wht	qweet91	dudes	moneycars
snowed	r	speed	BruklynFinest	sheesh	swears	nigga	LoWQUI
SNOW	them	Slow	hold	TwitterJail	10	up	lifestyle
smoking	**silence**	**showoff**	**sexual**	**services**	**selfishness**	**repetition**	**religious**
JAYECANE	guilty	louis	box	fil	ONLY	dislike	sinners
reggie	R	rims	wonder	requests	Selfish	repeat	IAmKevinTerrell
smoking	response	seein	Preach	convos	selfish	myself	spiritual
smoke	conversation	makin	suck	TIP	stay	same	CHURCH
smokers	sending	bag	pussy	products	hit	over	FOLK

Table 2: The salient words for categories of annoying behaviors learned by the sparse additive generative model of text.

State	Top Topical Words
NY	stalkers niqqas der den part dats liek havin
MD	fuckouttahere missing ima dmv fan situation tongue
CA	pocket clown phones football fit acting lip
NJ	nite blame p hips pum summer elses seein
FL	daddy both chipped pum rims nappy foh children
GA	oo affioncrockett season cigarettes year tatoos
VA	lane language middle might check winter past duke
TX	drama lmaoooo gtfoh nappy two jk stare unfollow
NC	everyday ear chic during hello wayansjr tryn nicca
PA	10 huh killyaself lifestyle shades round texts fucc
DC	dmv uncle nosey stare cares bish 1st lips

Table 3: The geographical variation of the annoying behaviors.

dataset. More specifically, in the embedding vocabulary $\mathcal{W}$, we search for the k-nearest-neighbor (knn) word w for a query term using cosine similarity between query $\vec{Q}$ and target word vectors $\vec{W}$:

$$\arg\max_{w \in \mathcal{W}} cosine(\vec{Q}, \vec{W}) \quad (1)$$

For each word in a tweet, we query the external embeddings, and replace them with their knn words to create a new training instance. For example, consider the tweet *"Being late is terrible"* with the *punctuality* label, after searching for knn words for each token, we create a new training instance: *"Be behind are bad"* with the same label.

Frame-Semantic Embeddings Although lexical (Mikolov et al., 2013a) and dependency based embeddings (Levy and Goldberg, 2014) have been studied, semantic-based embedding is still less understood. We consider the continuous embedding of semantic frames (Baker et al., 1998). To do this, we semantically parsed 3.8 million tweets using SEMAFOR (Das et al., 2010), and built a continuous bag-of-frame model to represent each semantic frame using Word2Vec[3]. We then use the same data augmentation approach to create additional instances with these semantic frame embeddings.

[3]https://code.google.com/p/word2vec/

Features	Precision	Recall	F1
Lexical	.341	.342	.341
+POS	.345	.346	.346
+Dependency*	.349	.350	.350
+Semantic Frames*	**.365**	**.367**	**.366**

Table 4: Comparing linguistic features for categorizing annoying behaviors. The best results are highlighted in **bold**.* indicates that the result is significantly better than the lexical baseline ($p < .0001$).

5 Experiments

5.1 Qualitative Analysis

We show the results of the visualization of salient words for each category of tweets in Table 2. SAGE clearly does a good job identifying annoying specific behaviors in each category. For example, in the *traffic* category, we see that the keywords *"cop"* and *"pulled"* that associate with traffic stop are identified. Also, *"slow"* and *"speed"* are also recognized as annoying behaviors during traffic. In the *selfishness* category, the word *"ONLY"* and *"Selfish"* are corrected identified. In the *silence* category, we see that the word *"R"* is promising, because it indicates the behavior when someone reads a blackberry message without reply. We see that many slang expressions are associated with various labels.

In Table 3, we show the geographical variation of tweets. The word *"dmv"* (DC-Maryland-Virginia) is correctly associated with MD and DC, and when we search the database, these *#petpeeve* tweets mainly refer to the 2010 snowstorm in the Winter affecting these areas. The *"daddy"* is prominent in the state of Florida, while the word *"rims"* is also identified, showing the unique car culture of this southern state.

5.2 Quantitative Evaluation

Experimental Setup We use the logistic regression model from LibShortText (Yu et al., 2013)

Methods	Prec.	Rec.	F1	Imp.
Lexical Baseline (No Data Augmentation)	.341	.342	.341	—
+ UrbanDictionary Embeddings	.343	.344	.344	0.9%
+ Twitter Embeddings*	.357	.358	.358	4.7%
+ GoogleNews Embeddings*	**.364**	**.366**	**.365**	**6.1%**
All Features Baseline (No Data Augmentation)	.365	.367	.366	—
+ Lexical (GoogleNews) and Frame-Semantic Embeddings*	.376	.377	.376	2.7%
+ Lexical (Twitter) and Frame-Semantic Embeddings*	**.379**	.380	.379	3.6%
+ Lexical (UD) and Frame-Semantic Embeddings*	**.379**	**.381**	**.380**	**3.8%**

Table 5: The effectiveness of leveraging continuous embeddings to create additional training instances. Imp.: relative improvement to the baseline without data augmentation. The best results for each section are highlighted in **bold**.* indicates that the result is significantly better than the baseline without data augmentation ($p < .0001$).

as the classifier in our 60-way multi-class classification experiments. Grid search is used to select the best hyper-parameter using the training data only. A final classifier is then trained using the best hyper-parameters and test set results are reported. We set $k = 5$ for knn in our data augmentation experiments: the training data is expanded to 5 times of the original size. We use a paired two-tailed student's t test to assess the statistical significance.

Word2Vec is used to train various lexical and semantic embedding models. We consider three lexical embeddings and one frame-semantic embeddings for data augmentation: 1) GoogleNews Lexical Embeddings trained with 100 billion words (Mikolov et al., 2013b); 2) Twitter Lexical Embeddings trained with 51 million of words; 3) Urban Dictionary lexical embeddings trained with 53 million of words from slang definitions and examples; 4) Twitter Semantic Frame Embeddings trained with 27 million frames.

Varying Feature Sets We compare various features in Table 4. We see that adding shallow part-of-speech features does not have a strong effect on the performance, but adding the dependency triples significantly outperforms the lexical baseline. We see that the semantic frames are particular useful, showing a 7% relative improvement over the baseline.

The Effectiveness of Data Augmentation Table 5 shows the results of data augmentation. We see that using the Google News lexical embeddings to augment the training data brings a 6.1% relative F1 improvement over the lexical baseline. When considering the additional frame-semantic embeddings from Twitter, our system obtains the best F1 of 0.380, bringing a 3.8% improvement over the no data augmentation baseline with all linguistic features.

6 Discussion

We provide a case study of automatically categorizing annoying behaviors using *#petpeeve* Tweets. We hope that this study can further solicit relevant research on fine-grained analysis of annoying behaviors in different dimensions, and use computational approaches to improve social good. For example, by using coordinates and other APIs, one might analyze the annoying behaviors in the public working environments (e.g., office, meeting rooms, etc.). By understanding what annoys their employees, companies can renovate their working setups, refine their policies, and improve the satisfaction and productivity of their employees.

In addition to *#petpeeve* Tweets, there are many other interesting hashtags that align well with traditional topics in behavior sciences. For example, hashtags like *#occupywallstreet* can be used to study crowd behaviors in terms of a political unrest. The *#ALS* hashtag can be used to study public behaviors in reaction to philanthropic campaigns. Overall, Tweets from carefully selected hashtags can be inexpensive to obtain, and facilitate significant amount of behavioral studies.

7 Conclusion

In this paper, we have presented a case study of the annoying behaviors using Twitter as a corpus. Our fine-grained visualization approach shows insights of different categories of these behaviors, with the geographical effects. We also show that linguistic cues are useful to categorize these behaviors automatically, and that using lexical and semantic embeddings as a data augmentation method significantly improves the performance.

References

Apoorv Agarwal, Boyi Xie, Ilia Vovsha, Owen Rambow, and Rebecca Passonneau. 2011. Sentiment analysis of twitter data. In *Proceedings of the Workshop on Languages in Social Media*, pages 30–38. Association for Computational Linguistics.

Sileye O Ba and Jean-Marc Odobez. 2011. Multiperson visual focus of attention from head pose and meeting contextual cues. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(1):101–116.

Ryan SJD Baker and Kalina Yacef. 2009. The state of educational data mining in 2009: A review and future visions. *JEDM-Journal of Educational Data Mining*, 1(1):3–17.

Collin F Baker, Charles J Fillmore, and John B Lowe. 1998. The berkeley framenet project. In *Proceedings of the 17th international conference on Computational linguistics-Volume 1*, pages 86–90. Association for Computational Linguistics.

David Bamman and Noah A Smith. 2015. Contextualized sarcasm detection on twitter. In *Ninth International AAAI Conference on Web and Social Media*.

Marian Bartlett, Gwen Littlewort, Tingfan Wu, and Javier Movellan. 2008. Computer expression recognition toolbox. In *Automatic Face & Gesture Recognition, 2008. FG'08. 8th IEEE International Conference on*, pages 1–2. IEEE.

Sumit Basu. 2002. *Conversational scene analysis*. Ph.D. thesis, MaSSachuSettS InStitute of Technology.

Johan Bollen, Huina Mao, and Xiaojun Zeng. 2011. Twitter mood predicts the stock market. *Journal of Computational Science*, 2(1):1–8.

Jake Y Chen and Stefano Lonardi. 2009. *Biological data mining*. CRC Press.

Zhiyuan Cheng, James Caverlee, and Kyumin Lee. 2010. You are where you tweet: a content-based approach to geo-locating twitter users. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 759–768. ACM.

D. Das, N. Schneider, D. Chen, and N.A. Smith. 2010. Probabilistic frame-semantic parsing. In *HLT-NAACL 2010*, page 948956, Los Angeles, California, USA, June.

Richard De Charms. 2013. *Personal causation: The internal affective determinants of behavior*. Routledge.

Jacob Eisenstein, Amr Ahmed, and Eric P Xing. 2011. Sparse additive generative models of text. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 1041–1048.

Kenneth Alberto Funes Mora and J Odobez. 2012. Gaze estimation from multimodal kinect data. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2012 IEEE Computer Society Conference on*, pages 25–30. IEEE.

Kevin Gimpel, Nathan Schneider, Brendan O'Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A Smith. 2011. Part-of-speech tagging for twitter: Annotation, features, and experiments. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 42–47. Association for Computational Linguistics.

Roberto González-Ibánez, Smaranda Muresan, and Nina Wacholder. 2011. Identifying sarcasm in twitter: a closer look. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 581–586. Association for Computational Linguistics.

Lingpeng Kong, Nathan Schneider, Swabha Swayamdipta, Archna Bhatia, Chris Dyer, and Noah A Smith. 2014. A dependency parser for tweets. In *EMNLP*.

David Lazer, Alex Sandy Pentland, Lada Adamic, Sinan Aral, Albert Laszlo Barabasi, Devon Brewer, Nicholas Christakis, Noshir Contractor, James Fowler, Myron Gutmann, et al. 2009. Life in the network: the coming age of computational social science. *Science (New York, NY)*, 323(5915):721.

Omer Levy and Yoav Goldberg. 2014. Dependencybased word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, volume 2, pages 302–308.

Jiwei Li, Alan Ritter, Claire Cardie, and Eduard Hovy. 2014a. Major life event extraction from twitter based on congratulations/condolences speech acts. In *Proceedings of Empirical Methods in Natural Language Processing*.

Jiwei Li, Xun Wang, and Eduard Hovy. 2014b. What a nasty day: Exploring mood-weather relationship from twitter. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 1309–1318. ACM.

Wang Ling, Guang Xiang, Chris Dyer, Alan Black, and Isabel Trancoso. 2013. Microblogs as parallel corpora. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 176–186, Sofia, Bulgaria, August. Association for Computational Linguistics.

Alvaro Marcos-Ramiro, Daniel Pizarro-Perez, Marta Marron-Romera, Laurent Nguyen, and Daniel

Gatica-Perez. 2013. Body communicative cue extraction for conversational analysis. In *Automatic Face and Gesture Recognition (FG), 2013 10th IEEE International Conference and Workshops on*, pages 1–8. IEEE.

Manuel Jesús Marin-Jimenez, Andrew Zisserman, Marcin Eichner, and Vittorio Ferrari. 2014. Detecting people looking at each other in videos. *International Journal of Computer Vision*, 106(3):282–296.

Marianne Schmid Mast, Daniel Gatica-Perez, Denise Frauendorfer, Laurent Nguyen, and Tanzeem Choudhury. 2015. Social sensing for psychology automated interpersonal behavior assessment. *Current Directions in Psychological Science*, 24(2):154–160.

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

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

Greg Miller. 2011. Social scientists wade into the tweet stream. *Science*, 333(6051):1814–1815.

Laurent Nguyen, Jean-Marc Odobez, and Daniel Gatica-Perez. 2012. Using self-context for multimodal detection of head nods in face-to-face interactions. In *Proceedings of the 14th ACM international conference on Multimodal interaction*, pages 289–292. ACM.

Joakim Nivre, Johan Hall, Jens Nilsson, Atanas Chanev, Gülsen Eryigit, Sandra Kübler, Svetoslav Marinov, and Erwin Marsi. 2007. Maltparser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(02):95–135.

Brendan O'Connor, Ramnath Balasubramanyan, Bryan R Routledge, and Noah A Smith. 2010a. From tweets to polls: Linking text sentiment to public opinion time series. *ICWSM*, 11:122–129.

Brendan O'Connor, Michel Krieger, and David Ahn. 2010b. Tweetmotif: Exploratory search and topic summarization for twitter. In *ICWSM*.

Alan Ritter, Colin Cherry, and Bill Dolan. 2010. Unsupervised modeling of twitter conversations. In *Proc of NAACL*.

Alan Ritter, Sam Clark, Oren Etzioni, et al. 2011. Named entity recognition in tweets: an experimental study. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1524–1534. Association for Computational Linguistics.

Alan Ritter, Oren Etzioni, Sam Clark, et al. 2012. Open domain event extraction from twitter. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1104–1112. ACM.

Jamie Shotton, Toby Sharp, Alex Kipman, Andrew Fitzgibbon, Mark Finocchio, Andrew Blake, Mat Cook, and Richard Moore. 2013. Real-time human pose recognition in parts from single depth images. *Communications of the ACM*, 56(1):116–124.

Yanchuan Sim, Noah A. Smith, and David A. Smith. 2012. Discovering factions in the computational linguistics community. In *Proceedings of the ACL-2012 Special Workshop on Rediscovering 50 Years of Discoveries*, ACL '12 Special Workshop on Rediscovering 50 Years of Discoveries.

Kristina Toutanova, Dan Klein, Christopher D Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 173–180. Association for Computational Linguistics.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 384–394. Association for Computational Linguistics.

William Yang Wang, Elijah Mayfield, Suresh Naidu, and Jeremiah Dittmar. 2012. Historical analysis of legal opinions with a sparse mixed-effects latent variable model. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 740–749. Association for Computational Linguistics.

Shiliang Wang, Michael J Paul, and Mark Dredze. 2015. Social media as a sensor of air quality and public response in china. *Journal of medical Internet research*, 17(3).

Wei Xu, Alan Ritter, Chris Callison-Burch, William B. Dolan, and Yangfeng Ji. 2014. Extracting lexically divergent paraphrases from Twitter. *Transactions of the Association for Computational Linguistics (TACL)*, 2(1).

H Yu, C Ho, Y Juan, and C Lin. 2013. Libshorttext: A library for short-text classification and analysis. Technical report, Technical Report. http://www. csie. ntu. edu. tw/˜ cjlin/ papers/libshorttext. pdf.

Ni Zhang, Shelly Campo, Kathleen F Janz, Petya Eckler, Jingzhen Yang, Linda G Snetselaar, and Alessio Signorini. 2013. Electronic word of mouth on twitter about physical activity in the united states: exploratory infodemiology study. *Journal of medical Internet research*, 15(11).

Detection of Steganographic Techniques on Twitter

Alex Wilson and **Phil Blunsom** and **Andrew D. Ker**
Department of Computer Science
University of Oxford
Oxford, OX1 3QD, UK
{alex.wilson,phil.blunsom,andrew.ker}@cs.ox.ac.uk

Abstract

We propose a method to detect hidden data in English text. We target a system previously thought secure, which hides messages in tweets. The method brings ideas from image steganalysis into the linguistic domain, including the training of a feature-rich model for detection. To identify Twitter users guilty of steganography, we aggregate evidence; a first, in any domain. We test our system on a set of 1M steganographic tweets, and show it to be effective.

1 Introduction

Consider this: two isolated prisoners, communicating by letters scrutinised by the prison warden. They cannot write openly about escape plots, and the warden destroys any written in code. They must *hide* the true message within the letter, using *steganography*: the art of hiding information.

In *cover modification*[1] steganography, a *cover* object is tweaked so that it carries a hidden message: this is called *embedding*, the tweaked version is the *stego* object, and the message is the *payload* (Fridrich, 2009). The message should not be detectable to any observer (the standard terminology for this is the *warden*, taken from the prisoner metaphor) who knows the system is deployed, but does not know the original cover.

We are concerned here with *linguistic steganography*, in which the cover is a piece of text, and the message is embedded using textual transformations intended to preserve the meaning of the original: synonym substitutions, syntactical transformations, etc. Note that we are not concerned with the subset of linguistic steganography that hides in file formatting (e.g. white space, as in Por et al. (2008)), which has no security against an informed warden (in the case of hiding information by adding extraneous white space, the warden simply has to look for consistent irregular use of spaces to spot an active steganographer).

The field suffers from a number of issues: compared to images, text covers have low capacity (Chang and Clark, 2010); certain methods are weak against human attackers (Grosvald and Orgun, 2011) (most paraphrase systems cannot guarantee perfectly fluent stego objects); finally, authors are generally concerned with the performance of the transformation (whether they produce grammatically/semantically correct transformations), rather than whether the generated stego objects are detectable or not (e.g. Chang and Clark (2010)).

However, there is a new challenger in the field. We proposed a new linguistic stegosystem (Wilson et al., 2014) and verified its security against human judges, who were unable to distinguish genuine covers from manipulated stego objects.

This paper aims to attack CoverTweet statistically. We are in the shoes of the warden, attempting to classify stego objects from innocent[2] cover objects. We propose techniques new to linguistic steganalysis, including a large set of features that detect unusual and inconsistent use of language and the aggregation of evidence from multiple sentences. This last development, known in the steganographic literature as *pooled steganalysis* (Ker, 2007), represents a first in both linguistic and image steganalysis.

[1] There are other steganographic paradigms, not in scope. *Translation* based methods hide information in the automatic translation of the cover (e.g. Meng et al. (2011)). Cover *generation* methods automatically produce text containing the payload (e.g. Chapman et al. (2001)).

[2] It is usual to call steganographers and their output 'guilty' (with non-steganographers and unchanged cover objects being 'innocent'). This has the possibility of seeming politically charged, so we will use the term 'active' instead: be aware that this is not the usual terminology.

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 2564–2569,
Lisbon, Portugal, 17-21 September 2015.

2 Linguistic Stegosystems

T-Lex (Winstein, 1998) is the oldest available cover-modification based linguistic stegosystem. It uses a dictionary containing a small number of disjoint synonym sets extracted from WordNet (Miller, 1995). Each set is unambiguously ordered (e.g. alphabetically), then values are embedded by changing cover words for their synonyms. Due to the the small dictionary, the *capacity* of covers is only ~ 0.1 bits per sentence.

CoverTweet is a modern evolution of T-Lex. It hides information in tweets by applying paraphrase rules taken from the Paraphrase Database (PPDB) (Ganitkevitch et al., 2013), a set of 169M rules. The system applies suitable rules to a given cover, generating a set of possible stego objects. These are ranked by a distortion measure (derived from the probabilities of applied rules and from sentence probabilities given by a language model), and assigned a keyed hash. A human operator filters the options for fluency, and chooses the best stego object with the desired hash.

CoverTweet uses a subset of the PPDB, restricted to lexical and phrasal substitutions. Even with the reduced set of rules, 4 bits can be embedded per tweet, and it was proven secure against human judges (Wilson et al., 2014).

Twitter is a realistic setting for steganography. There is precedent for information hiding and mass monitoring on micro-blogging sites, such as the use of code words and government censorship on the Chinese website Sina Weibo (Chen et al., 2013). For this reason, we are attacking this setting.

There are many other linguistic stegosystems, using an array of different hiding methods (e.g. adjective deletion, word order, anaphora resolution: (Chang and Clark, 2012a), (Chang and Clark, 2012b), (Vybornova and Macq, 2007)). Approximately 1 bit of payload per cover sentence is usual, making CoverTweet the exception. Unfortunately for steganalysis literature, the vast majority of these require data that is unavailable (and too expensive to reproduce); beyond CoverTweet, the only system that can be evaluated is T-Lex.

3 Related Work

To our knowledge, there have been only five prior attempts at linguistic steganalysis on cover modification based systems; of these, four attack T-Lex, the other attacks an equivalent proprietary system. Taskiran et al. (2006) was the first, using n-gram language models to extract features from stego text, before training a support vector machine (SVM) on the features. We will adopt some of these features for our attack.

Subsequent work (Xin-guang et al. (2006); Yu et al. (2009); Chen et al. (2011); Xiang et al. (2014)) has used smaller models: they have all designed a *single* feature to exploit a weakness, and used this (or the mean and variance of it) to train a classifier for attack. Analysis of results, especially the effect of embedding rate on detection, has been lacking or non-existent. This focus on individual features echoes the early work on image steganalysis, which has since shifted towards feature-rich models. We will be utilising the latter here, in addition to the pooled steganalysis paradigm.

4 Proposed features

Below we describe four classes of proposed features for individual tweets. Each class will be evaluated individually, and in combination.

Basic features Including: word count (including tokenised punctuation); the mean and variance of the number of characters in each word; the total stop word count[3]; and the counts for each individual stop word. (131 features)

n-gram features Using a 5-gram model, for n from 1 to 5, the mean, variance and total log likelihood of the n-grams in the tweet. (15 features)

Word length Equivalent to the n-gram features, using a 10-gram model of word length. We expect the PPDB to replace common words with uncommon longer words, or multiple shorter words. (30 features)

PPDB features *Kerckhoffs' principle* (Fridrich, 2009), which states that a stegosystem must be secure even when the attacker (*us*) knows how the system works, introduces an interesting opportunity for linguistic steganalysis. As all linguistic stegosystems rely on automatic paraphrasing, and generally require a source of data, it is conservative to assume that the attacker has access to this data source. As the attacker, we can derive features from this data.

Applying this, we extracted features using the same subset of the PPDB that CoverTweet uses. To our knowledge, this is the first work to explicitly apply Kerckhoffs' principle to linguistic

[3]We use the Natural Language Toolkit (Bird, 2006) for the list of 127 stop words.

steganalysis. This set includes: a repeat of the n-gram features, but only taking into account n-grams that contain at least one word in[4] the PPDB; the mean and variance of character count for words in the PPDB; for phrase lengths j from 1 to 5, the proportion of j-grams in the tweet that are in the PPDB; likelihood of the most probable paraphrased sentence, extracted by generating sentences in the same way as CoverTweet. Finally, a substitution score: each word or phrase in the PPDB is replaced by its most likely substitute, according to the language model. The max, min, mean and variance of the log likelihoods for the resulting sentences are used. (27 features)

4.1 Pooling the features

Individual linguistic stego objects carry a tiny payload, making detection an incredibly difficult task. In some cases, the stego tweet will be identical to the cover, making the task impossible. Although we do not hope to reliably identify stego tweets individually, the warden can still try to catch the active steganographers: through *pooled steganalysis* (Ker, 2007). Here, the warden considers all the evidence for each user, and aims to identify those actively using steganography. We combine features from batches of tweets, by the same user: taking the max, min, mean and variance of each feature. If performing steganography causes small consistent changes to a feature, this will be apparent in the mean. If it causes inconsistent changes, it may be apparent in the variance. If it occasionally produces an extreme result, the max and min will capture this. (812 features)

5 Method and Results

Here we take the role of an automatic censor, attempting to identify users on Twitter who are sending secret messages. We imagine a scenario where CoverTweet is openly available, and so can assume this is the system utilised by the active users.

5.1 Data and Embedding

From the Harvard TweetMap (Mostak, 2013), we gathered 72M English language tweets, from 1.2M users. Each tweet was canonicalised as follows: tokenised; made lowercase; usernames replaced by a uniform token; URLs replaced by a uniform token. We randomly selected 1000 users with 1000-2000 tweets to train and test a classifier. On the remaining tweets, we trained a 10-gram word length model, and a 5-gram language model, using SRILM (Stolcke, 2002) with Kneyser-Ney smoothing. For vocabulary, the language model was given every word in the PPDB, and every word in the set of tweets, including the removed tweets. We do not expect this to provide an unrealistic advantage to the censor: any word not in the PPDB cannot hide information.

We randomly took 10 users from the set of 1000, and produced 100 stego tweets for each, for three payload sizes: 1, 2 and 4 bits. The stego tweets were generated using CoverTweet, with a human operator selecting the most fluent option containing a desired (randomly generated) payload. If there were no fluent options, the tweet was skipped. If the tweet already contained the desired payload (if the hash value of the tweet already matched the message), it was left unchanged. We refer to this data as Manual CoverTweet (M-CT).

Due to the expensive nature of generating data with a human judge, and to assess the value of the human in-the-loop, we also automatically generated 1M stego tweets, by embedding data in 1000 tweets for each of the 1000 users. The same three payload sizes were used. Here, the tweet with the highest probability (provided by the language model and the PPDB) was selected. Again, the tweet was left unchanged if the tweet already contained the desired payload. Tweets were only skipped if there were no options with the correct payload. We refer to this as Automatic CoverTweet (A-CT). Finally, we embedded 1 bit in the same tweets using T-Lex, rejecting any that contained no words in the T-Lex dictionary; the result was approximately 100 tweets on average per user.

We split the users in half, training a linear ensemble classifier (Kodovskỳ et al., 2012) (designed to work with large feature sets for steganalysis) on feature instances from one half, testing it on the other. We only classified tweets by users for whom the classifier had no prior knowledge. All error rates are averaged over 10 different random user splits. For the M-CT data, where we have a much smaller set of data, we performed 10-fold cross validation, leaving one user out for testing each fold.

In each experiment we matched training and testing data: the training data was produced by the

[4]Where we say a word or phrase is 'in' the PPDB, we mean it features in one or more paraphrase rules that could have been applied to the original tweet.

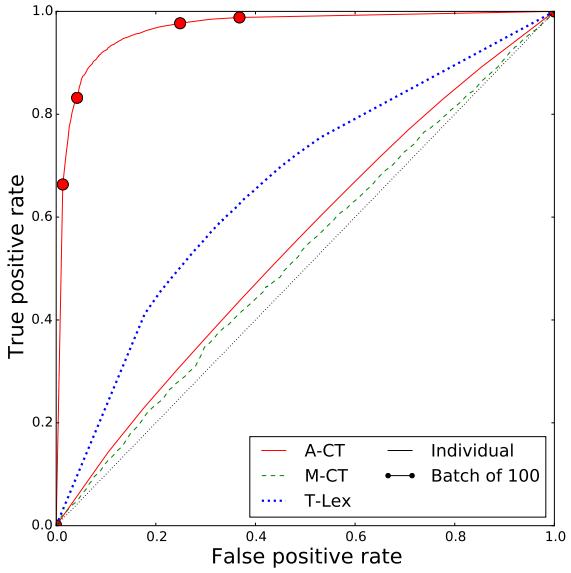


Figure 1: ROCs for each embedding method, for individual tweets or batches containing 1 bit.

	Batch size	AUC
A-CT	1	0.551
M-CT	1	0.509
T-Lex	1	0.667
A-CT	100	0.9631

Table 1: AUC values for the ROCs shown in Figure 1

same method as the testing data. For M-CT we tried an alternative scheme, by training on A-CT data, but testing on M-CT; this did not work reliably, and the accuracy of the resulting model went down as pooled batch size increased. The exploration of this phenomenon is left for later work.

5.2 A note on unchangeable covers

Some tweets cannot hide the payload, either when there are no paraphrase options for that hash, or when the human (for M-CT) vetoes all the options. This is the *non-shared selection channel* problem in steganography. Methods such as *syndrome trellis codes* (Filler et al., 2011) allow the unchangeable cover elements to be sent without compromising the message. These have not been applied to linguistic steganography, but we simulate their use: we remove the unchangeable tweets from the cover and stego sets, essentially giving the steganographer and detector the ability to ignore such tweets.

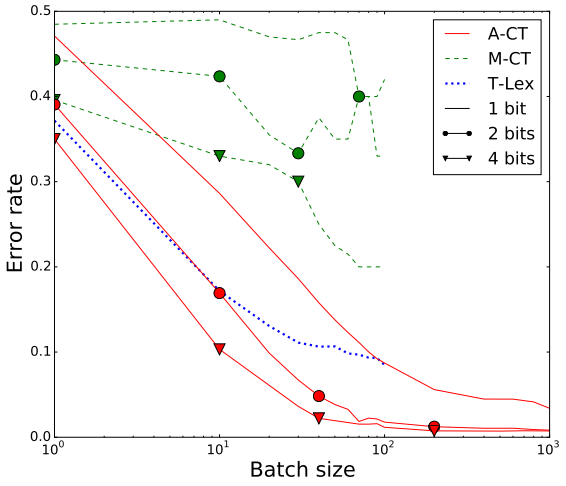


Figure 2: The effect of batch size on error rate. Batches of 100 are the maximum for M-CT and T-Lex, but we go up to 1000 with A-CT.

5.3 Results

As expected, the performance of the classifiers trained on individual tweets is poor (see Figure 1). In particular, the models trained on A-CT and M-CT data have very low accuracy on data with 1 bit payload, performing only slightly better than random guessing. T-Lex tweets prove slightly easier to detect. This does not mean that the systems are secure however. Though we cannot identify individual stego tweets, when we pool evidence we find we are able to train a model that can identify active users with high accuracy, for all data sets.

Figure 2 shows the change in error rate as the batch size (the number of tweets we pool) is increased. We can clearly see that increasing batch size improves accuracy. We also see that increasing payload size makes active users easier to spot. This is unsurprising: CoverTweet is forced to choose from fewer options when payload increases.

The experiment showed us that M-CT is the most secure stegosystem, though not immune to the effects of pooling. When combining features from 100 tweets, the classifier had an error rate of 0.21 on 4 bit M-CT data; data generated in the same way, with the same payload, was previously shown to be secure against human judges. With large batches, detection of A-CT is almost perfect.

To establish which class of features had the biggest effect on detection, we trained models on each combination of features described in Section 4. A subset of these combinations are shown

	b	n	p	w	b+n	n+p	p+w	b+n+p	p+n+c	all
A-CT (2 bits)	0.345	0.311	0.217	0.376	0.266	0.204	0.203	0.180	0.191	0.169
T-Lex (1 bit)	0.448	0.226	0.273	0.317	0.226	0.202	0.218	0.198	0.168	0.168

Table 2: Error rates for models trained on different combinations of feature class, for a batch of 10. The feature sets are as follows: basic (b), n-gram (n), PPDB (p) and word length (w).

in Table 2, for A-CT and T-Lex.We can see that for A-CT, the PPDB features easily outperform others, with an error rate of 0.217 when used alone; the warden's knowledge of the system is powerful. The second best is the n-gram set, though it performs better on T-Lex than on A-CT. This is most likely due to CoverTweet's use of the language model in the embedding stage: the system is attempting to minimise the distortion that these features are looking for. T-Lex has no such distortion measure, leaving it open to this sort of attack.

Basic and PPDB feature sets fare worse on T-Lex. The basic features are aimed at changes in word count and stop word usage: neither of these are affected by T-Lex substitutions. The PPDB features are at a disadvantage with T-Lex, as they are designed for CoverTweet. If T-Lex's data source were used instead of CoverTweet's, the performance would likely improve significantly.

The combination of PPDB and n-gram features on T-Lex gives us some interesting insight: despite the mismatch of substitution source, we still see an improvement over the n-gram features used on their own. This suggests that the warden does not need the exact data source as the steganographer for these features to be useful.

6 Conclusion

It was believed that linguistic steganography was weak against humans, but CoverTweet disproved it. We have shown that individual stego objects are seemingly also strong against statistical attacks. However, by pooling multiple pieces of evidence against a user, the warden can drastically improve detection rate. With each steganographic tweet sent, the user creeps closer to being caught. This is the first steganalytic classifier, in any domain, that successfully exploits pooled evidence. The design of steganographic systems must now take this type of attacker into account. It would interesting to determine whether human judges are capable of pooling large amounts of scant evidence: we conjecture not.

Results suggest that detection is improved by utilising rich-feature models; we only scratched the surface with regards to this. There are many avenues to explore, such as using multiple language models from which to extract features (this is the analogue of filter banks used in contemporary image steganalysis (Fridrich and Kodovskỳ, 2012)).

The security of a system should first be measured against a powerful (informed) attacker. We played this role by using features extracted using exact knowledge of the CoverTweet system (the PPDB features); this class of features was particularly effective against CoverTweet. The system should now be evaluated against a weaker attacker. We have seen that detection is still reliable when the warden knows the wrong system (T-Lex), but further experiments are required to determine exactly how detection rate is affected by mismatches in paraphrase sources, or language model.

References

Steven Bird. 2006. NLTK: the natural language toolkit. In *Proceedings of the COLING/ACL on Interactive presentation sessions*, pages 69–72. Association for Computational Linguistics.

Ching-Yun Chang and Stephen Clark. 2010. Linguistic steganography using automatically generated paraphrases. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 591–599. Association for Computational Linguistics.

Ching-Yun Chang and Stephen Clark. 2012a. Adjective deletion for linguistic steganography and secret sharing. In *COLING*, pages 493–510.

Ching-Yun Chang and Stephen Clark. 2012b. The secret's in the word order: Text-to-text generation for linguistic steganography. In *COLING*, pages 511–528.

Mark Chapman, George I Davida, and Marc Rennhard. 2001. A practical and effective approach to large-scale automated linguistic steganography. In *Information Security*, pages 156–165. Springer.

Zhili Chen, Liusheng Huang, Haibo Miao, Wei Yang, and Peng Meng. 2011. Steganalysis against

substitution-based linguistic steganography based on context clusters. *Computers & Electrical Engineering*, 37(6):1071–1081.

Le Chen, Chi Zhang, and Christo Wilson. 2013. Tweeting under pressure: analyzing trending topics and evolving word choice on sina weibo. In *Proceedings of the first ACM conference on Online social networks*, pages 89–100. ACM.

Tomáš Filler, Jan Judas, and Jessica Fridrich. 2011. Minimizing additive distortion in steganography using syndrome-trellis codes. *Information Forensics and Security, IEEE Transactions on*, 6(3):920–935.

Jessica Fridrich and Jan Kodovskỳ. 2012. Rich models for steganalysis of digital images. *Information Forensics and Security, IEEE Transactions on*, 7(3):868–882.

Jessica Fridrich. 2009. *Steganography in digital media: principles, algorithms, and applications*. Cambridge University Press.

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

Michael Grosvald and C Orhan Orgun. 2011. Free from the cover text: a human-generated natural language approach to text-based steganography. *Journal of Information Hiding and Multimedia Signal Processing*, 2(2):133–141.

Andrew D Ker. 2007. Batch steganography and pooled steganalysis. In *Information Hiding*, pages 265–281. Springer.

Jan Kodovskỳ, Jessica Fridrich, and Vojtěch Holub. 2012. Ensemble classifiers for steganalysis of digital media. *Information Forensics and Security, IEEE Transactions on*, 7(2):432–444.

Peng Meng, Yun-Qing Shi, Liusheng Huang, Zhili Chen, Wei Yang, and Abdelrahman Desoky. 2011. Linl: Lost in n-best list. In *Information Hiding*, pages 329–341. Springer.

George A. Miller. 1995. Wordnet: A lexical database for english. *COMMUNICATIONS OF THE ACM*, 38:39–41.

Todd Mostak. 2013. Harvard TweetMap. accessed October 2013. `https://worldmap.harvard.edu/`.

Lip Y Por, TF Ang, and B Delina. 2008. Whitesteg: a new scheme in information hiding using text steganography. *WSEAS Transactions on Computers*, 7(6):735–745.

Andreas Stolcke. 2002. SRILM - an extensible language modeling toolkit. pages 901–904.

Cuneyt M Taskiran, Umut Topkara, Mercan Topkara, and Edward J Delp. 2006. Attacks on lexical natural language steganography systems. In *IS&T/SPIE Electronic Imaging*, pages 120–129. International Society for Optics and Photonics.

Olga Vybornova and Benoit Macq. 2007. Natural language watermarking and robust hashing based on presuppositional analysis. In *Information Reuse and Integration, 2007. IRI 2007. IEEE International Conference on*, pages 177–182. IEEE.

Alex Wilson, Phil Blunsom, and Andrew D Ker. 2014. Linguistic steganography on twitter: hierarchical language modeling with manual interaction. In *IS&T/SPIE Electronic Imaging*, pages 201–217. International Society for Optics and Photonics.

Keith Winstein. 1998. Lexical steganography through adaptive modulation of the word choice hash. `http://www.imsa.edu/~keithw/tlex`. *Unpublished.*

Lingyun Xiang, Xingming Sun, Gang Luo, and Bin Xia. 2014. Linguistic steganalysis using the features derived from synonym frequency. *Multimedia tools and applications*, 71(3):1893–1911.

Sui Xin-guang, Luo Hui, and Zhu Zhong-liang. 2006. A steganalysis method based on the distribution of characters. In *Signal Processing, 2006 8th International Conference on*, volume 4, pages 54–56. IEEE.

Zhenshan Yu, Liusheng Huang, Zhili Chen, Lingjun Li, Xinxin Zhao, and Youwen Zhu. 2009. Steganalysis of synonym-substitution based natural language watermarking.

#SupportTheCause: Identifying Motivations to Participate in Online Health Campaigns

Dong Nguyen[1] **Tijs van den Broek**[2]
Claudia Hauff[3] **Djoerd Hiemstra**[4] **Michel Ehrenhard**[2]
[1]Human Media Interaction, University of Twente, d.nguyen@utwente.nl
[2]NIKOS, University of Twente, {t.a.vandenbroek,m.l.ehrenhard}@utwente.nl
[3]Web Information Systems, Delft University of Technology, c.hauff@tudelft.nl
[4]Database Group, University of Twente, d.hiemstra@utwente.nl

Abstract

We consider the task of automatically identifying participants' motivations in the public health campaign Movember and investigate the *impact* of the different motivations on the amount of campaign donations raised. Our classification scheme is based on the *Social Identity Model of Collective Action* (van Zomeren et al., 2008). We find that automatic classification based on Movember profiles is fairly accurate, while automatic classification based on tweets is challenging. Using our classifier, we find a strong relation between types of motivations and donations. Our study is a first step towards scaling-up collective action research methods.

1 Introduction

Social media is a valuable source for studying health-related behaviors (De Choudhury, 2014). For example, Twitter was used for disease surveillance (Lamb et al., 2013; Aramaki et al., 2011), and was studied for its role in disseminating medical information (Desai et al., 2012) and organizing public health campaigns (Emery et al., 2014; Wehner et al., 2014). Social media data provides many opportunities to study social phenomena such as health campaigns, but statistics based on aggregating *across* social media users only provide a big picture of the phenomenon. A deeper analysis of such phenomena requires fine-grained information about the involved users. Since such information is often not readily available, numerous studies have appeared on automatically inferring user characteristics (Bamman et al., 2014; Eisenstein et al., 2010; Nguyen et al., 2013).

In the context of health campaigns, social scientists have been interested in the *motivations* of the participants (Cugelman et al., 2011). Knowledge about individual motivations helps to explain the emergence and effectiveness of collective action, such as volunteering (Bekkers and Wiepking, 2011) or mobilizing other people (van den Broek et al., 2015). The Social Identity Model of Collective Action (SIMCA) (van Zomeren et al., 2008) identifies three key motivations of participants: 1) *social identification* with the campaign organization and community, 2) a perception of *injustice* about the cause, and 3) *collective efficacy*, the collective belief that the campaign can make a difference. Taken together, these three motivations predict the chance that an individual will participate in collective action, such as participation in an online health campaign. Aggregating motivations to group-level may explain the effectiveness of online health campaigns. Social scientists, however, have not used computational methods to measure these motivations (Johnston et al., 2009), so that their analyses are often confined to small datasets.

Our study is a first step towards scaling-up collective action research methods. To do so, we explore automatic classification of the motivation types according to the SIMCA model. We analyze the global health campaign Movember (movember.com), which aims to raise funds and awareness of men-related health issues by engaging online conversations. Movember's fundraisers ask their friends to sponsor their moustache and their efforts in the month of November. The funds are donated to research concerned with men-related health issues, such as prostate cancer.

Movember participants provide their motivations in their Movember profile. For example, a participant writing '*In honor of my Grandfather*' could be considered having an injustice motivation, while '*To lead the brave men of Team [...] (and our exceptionally understanding significant others) in epic moustachery.*' indicates a social identification motivation. Because such explicit motivation statements are not available for many online health campaigns, we also explore motiva-

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 2570–2576,
Lisbon, Portugal, 17-21 September 2015.

tion classification based on the tweets of participants during the campaign instead.

Our paper makes the following contributions:

- We automatically classify the motivations of Movember participants and explore the use of *free-text* motivations provided in Movember profiles and *tweets* posted by the participants during the campaign (Section 3).
- We apply our classifier to all *US* and *UK* Movember profiles and find that participants with an injustice motivation raise significantly more funds (Section 4).

2 Dataset

In this section we discuss the collection and the annotation of the data.

2.1 Collection

We collect data from two different sources.

Movember Profiles We focus on participants from the two countries with the highest number of English speaking Movember participants: the United States and the United Kingdom. From Movember we obtained the identifiers of all participants of these two countries and we crawled all US and UK Movember profiles in May 2015. We extracted information such as the name, motivation (free-text), amount raised and whether the participant was part of a team. We collected 166,422 US and 138,546 UK profiles.

Twitter Data We link Movember participants to Twitter accounts based on tweets with a link to a Movember profile in 2013 and 2014 (e.g., '*please support my moustache [LINK]*'). If the Levenshtein distance between the name of the author of the tweet and the name in the Movember profile was 1 or less, we considered it a match (in total: 5,519 users). Manual inspection of 100 matches showed that this method was highly precise (100% precision). However, some matches were missed due to the low Levenshtein distance threshold. For each Twitter user in our dataset, we collected the last 3,600 tweets.

We kept all tweets written between October 18 and December 14 (2 weeks before and after the campaign). For each user, we used tweets from either 2013 or 2014, depending on whether the user posted a tweet with a Movember link at least once during the period, given preference to the year 2014.

2.2 Annotation

We annotated the campaign participants based on their provided motivations in their 'My motivation' section of their Movember profiles. The motivation categories in our codebook are based on the Social Identity Model of Collective Action (van Zomeren et al., 2008):

- *Injustice:* A shared emotion that includes both affective (e.g, anger) and cognitive perceptions (ideology) of an unfair situation (van Zomeren et al., 2008). It covers the ideological motivation to join a campaign, when potential participants compare the cause and the situation of patients with their personal values (Klandermans, 2004). For example, '*my dad*', '*I had testicular cancer*' or '*because men's health is important to me*'.
- *Social identity:* A sense of belonging together that emerges out of common attributes, experiences and external labels (van Zomeren et al., 2008). Participants may have social motivations to identify with the online health campaign, while not being interested in the cause (Kristofferson et al., 2014). This category includes psychological benefits, such as reputation or fun, that the social interactions of a campaign provide. For example, '*my friends asked me again to join them*', or '*a great excuse to grow a stache*'.
- *Collective efficacy:* The shared belief that ones group is capable of resolving its grievances through a campaign (Bandura, 2000; Klandermans, 2004; van Zomeren et al., 2008), for example by stating '*this campaign can make a difference!*'.

Multiple motivations may be assigned to a single campaign participant. Exactly recurring motivation texts that occurred frequently (more than 50 times, based on data analysis), were most likely prefilled texts. They were not annotated, because it was unclear whether participants used these 'default' motivations on purpose. For example, the most frequent motivation '*my motivation is to use the power of the moustache to have an everlasting impact on the face of mens health*' appeared in 104k profiles. The interrater reliability calculated using Cohen's Kappa was found to be satisfactory to good based on 200 double annotations: injustice (0.71), social identity (0.67) and collective efficacy (0.47) (Landis and Koch, 1977).

Features	Injustice				Social Identity				Collective Efficacy			
	P	R	F_1	AUC	P	R	F_1	AUC	P	R	F_1	AUC
Tokens	0.813	0.789	0.801	0.833	0.768	0.792	0.779	0.790	0.595	0.656	0.624	0.708
LDA	0.789	0.802	0.795	0.829	**0.809**	0.795	**0.802**	**0.815**	0.514	**0.688**	0.588	0.669
Length	0.644	0.615	0.629	0.693	0.526	0.632	0.574	0.564	0.419	0.642	0.507	0.582
Country	0.422	0.559	0.481	0.522	0.495	0.493	0.494	0.524	0.373	0.498	0.426	0.523
All	**0.823**	**0.810**	**0.816**	**0.846**	0.777	**0.799**	0.788	0.798	**0.597**	0.660	**0.627**	**0.710**

Table 1: Results free-text motivations: precision (P), recall (R), F_1 score and AUC.

From the set of Movember participants with matched Twitter accounts, we annotated a randomly selected subset of 2,108 participants. 21.8% of the participants had more than one motivation type assigned. We randomly split our dataset into a training and test set (Table 2). We have made our annotations available to other researchers[1].

	Train	Test
# Participants	1,494	614
% US / UK	54.8/45.2	53.3/46.7
% Injustice	37.6	40.2
% Social identity	48.7	46.9
% Collective efficacy	36.1	35.0

Table 2: Dataset statistics

3 Classification Experiments

In this section we present results on automatically identifying the motivations of Movember participants. Because participants may have multiple motivation types, we train binary classification models for each motivation type separately. We use logistic regression with L2 regularization, implemented using the Scikit-learn toolkit (Pedregosa et al., 2011). We report results on the test set using precision, recall, F_1 score and the Area Under Curve (AUC) metric. Note that a majority class classifier achieves an AUC of 0.5. Feature development and parameter tuning was done based on cross-validation on the training set. Based on the same set of Movember participants, we explore the use of two different types of data: the provided free-text motivations in Movember profiles (Section 3.1) and tweets of the participants (Section 3.2).

[1] `http://www.dongnguyen.nl/data.html`: The identifiers of the Movember and corresponding Twitter accounts, the country, text provided in the Movember profiles, and the annotations.

3.1 Free-text Movember Motivations

All text is lowercased and tokenized. We explore the following features: 1) Token unigrams and bigrams (frequency values), 2) LDA with 20 topics (Blei et al., 2003) trained on text from all US and UK Movember profiles (with the topic proportions as feature values), 3) Text length, and 4) Country (US=1, UK=0) to control for prior motivation distributions in the two countries.

The token features already lead to a high performance, and no notable increase in performance is observed by adding the other features (Table 1). The features with the highest weight are shown in Table 4. The performance numbers are in line with the obtained inter-annotator agreement. For example, the performance is highest on the injustice category, which also had the highest inter-annotator agreement (and vice versa for collective efficacy).

The lengths of texts alone have predictive power. The texts are short (on average 158.4 characters), but there are markable differences between motivation types. Participants with an injustice motivation write longer motivations: the average length of their texts is 213.74 chars in the training set, compared to 148.24 chars (social identity) and 130.93 chars (collective efficacy).

Injustice	Social Identity	Collective Efficacy
LDA topic[a]	fun	LDA topic[b]
cancer	team	beat
friend	moustache	and family
lost	mo	change
father	grow	yourself
had	mustache	all of
survivor	LDA topic[c]	awareness
prostrate	fuzz	for movember
for my	movement	awareness of
my	look	last

Table 4: Top-weighted features for free-text motivation experiments.

[a] topic about family/friends who had cancer
[b] topic about raising funds for research
[c] topic about the Movember campaign

Features	Injustice				Social Identity				Collective Efficacy			
	P	R	F_1	AUC	P	R	F_1	AUC	P	R	F_1	AUC
1: Tokens	0.456	0.445	0.451	0.544	**0.528**	0.563	0.545	**0.559**	**0.394**	0.465	0.426	**0.540**
2: URLs	0.421	0.304	0.353	0.511	0.469	**0.736**	**0.573**	0.500	0.360	0.209	0.265	0.504
3: Mentions	0.435	0.340	0.382	0.522	0.477	0.694	0.566	0.511	0.360	**0.721**	**0.480**	0.515
4: Effort	0.434	0.518	**0.472**	0.532	0.489	0.531	0.509	0.520	0.363	0.498	0.420	0.513
5: LDA	0.427	0.510	0.465	0.525	0.512	0.538	0.525	0.542	0.378	0.521	0.438	0.530
6: Behavior	0.415	**0.526**	0.464	0.514	0.463	0.410	0.435	0.495	0.360	0.581	0.445	0.513
1+3+4+5+cntry	**0.463**	0.453	0.458	**0.550**	0.520	0.542	0.531	0.550	0.381	0.419	0.399	0.526

Table 3: Results on tweets: precision (P), recall (R), F_1 score and AUC.

3.2 Tweets

In this section, we present experiments on identifying the motivations based on Twitter data.

Preprocessing Many of the tweets posted during the time of the campaign are not about the campaign itself. Based on manually selected character sequences[2], we separate relevant from non-relevant tweets. The tweets are tokenized using the CMU POS tagger (Owoputi et al., 2013). The average number of tweets per user during the studied period is 109.1 (median: 46.0) and the average number of relevant tweets is 8.0 (median: 4.0).

Features We explore the same features as with the free-text motivations and several new features:

- *Unigram and bigram tokens*: URLs and user mentions are replaced by generic tokens. We only keep tokens used by at least 10 Twitter users and we use their normalized frequency.
- *URLs*: We extract tokens from URLs by taking the hostname, and paths up to depth 2 (e.g., *us.movember.com/team/12345* results in *us.movember.com*, *us.movember.com/team* and *us.movember.com/team/12345*).
- *User mentions*: The Twitter accounts that are mentioned.
- *Effort*: Length (#characters), #tweets about Movember, #tweets about Movember/#total number of tweets.
- *LDA* with 20 topics (Blei et al., 2003). The model is trained on 1.5M tweets from 2013 and 2014 about the Movember campaign.
- *Country*: US=1, UK=0.
- *Behavior*: Fraction of retweets, tweets that contain a user mention, hashtag, URL, or are a reply. Number of days with a tweet about Movember. Fraction of tweets in each week.

Results The results are reported in Table 3. The URLs and behavior features were excluded from the run with the combined feature set, because their individual results suggest no predictive power (possibly due to the small training set). The results are fairly low and just above the 0.5 AUC value of a random classifier. To test whether the best performing classifiers for each motivation type (based on their AUC scores) are significantly better than a random classifier, we use permutation tests. We permute the labels to break the link with the features and calculate the AUC scores of the classifiers by training and testing on 1000 of such permutations. The best classifiers for the injustice and social identity motivation types are significantly better than random (p <0.01), but the performance of the collective efficacy classifier is only slightly significant (p <0.05).

To understand the low performance numbers, we took a closer look at the task and the data. First, we aimed to get a sense of the difficulty of the task. In a small experiment based on 100 Twitter users from the test set, one of the authors read the tweets and tried to identify the motivations. The results were also low (injustice: 0.488, social identity: 0.548, and collective efficacy: 0.590), suggesting that the task in itself is also difficult for humans.

The task is challenging because many users only post a few tweets about the campaign. In our data, 382 users have only one relevant tweet and 1,271 users have 5 relevant tweets or less. Furthermore, many of the tweets posted during the campaign focus on the Movember community (Bravo and Hoffman-Goetz, 2015; Dwi Prasetyo et al., 2015), making it hard to distinguish between the different motivations. For example, *instagram.com* is among the top three of hostnames for all motivation types. Sometimes participants

[2] '*movember*', '*mobro*', '*mosista*', '*cancer*', '*shave*', '*donat*', '*tache*', '*prostate*', '*mo*', '*testicular*', '*mental*' '*men's health*'

do explicitely mention their motivation (e.g., '*In honour of my dad, [..], I'm growing a horrible moustache for an incredible cause, #Movember. Donate here: [LINK]*'), but such instances are rare and in general the motivations of participants are much less visible through their tweets.

Social media plays a large role in mediating social relationships and users adapt their behavior to the online communities they are participating in (Danescu-Niculescu-Mizil et al., 2013; Nguyen and Rosé, 2011). This may explain why most participants, regardless of their motivation, emphasize the Movember community and its practices (such as the growing of moustaches) in their tweets. Various studies within the emerging field of Computational Social Science (Lazer et al., 2009) have found that Twitter tends to be a good reflection of society (Lamb et al., 2013; O'Connor et al., 2010). However, our results emphasize that the nature of the used platform influences how humans behave, and that this should be taken into account when interpreting the data. In the case of Movember, Twitter data alone could give a misleading view of the motivations of the campaign's participants.

4 Motivations and Campaign Behavior

In this section we present a linear regression analysis (n=90,484) of how motivations affect campaign donations by applying our classifier to all US and UK Movember profile texts. Participants of the Movember campaign can be part of a team. We therefore included actual team membership as a control variable, as we expect that team members increase fundraisers' effort due to peer pressure. In our analysis, we exclude all participants that have predefined motivations (214,484 out of the 304,968 profiles), because these may not reflect the actual motivation.

The social identity motivation is the most frequent in both countries, but the countries differ in their distributions regarding the injustice and collective efficacy motivations (Table 5).

	% Injustice	% Identity	% Efficacy
UK	31.0	49.7	46.1
US	37.6	50.3	32.1

Table 5: Motivation distribution based on automatic annotation (n=90,484). Note that participants may have multiple motivations.

On average, US participants donate more than UK participants (Table 6). US campaign participants with an injustice motivation raise significantly (coef = 91.525, $p < 0.001$) more money than participants with a social identity (coef = -5.479, $p =$ not significant) or collective efficacy motivation (coef = -5.765, $p < 0.1$). Participants that are part of a team raise significantly (coef = 75.849, $p < 0.001$) more money than participants without a team. Similar results were obtained for the UK. Furthermore, participants with a social identity motivation are more often part of a team (UK: 58% vs. 51% of the participants without a social identity motivation, US: 80% vs. 76%). The regression analysis reveals that being part of a team has a stronger and more positive effect on the amounts raised than the expression of identity as a motivation in the Movember profiles. Our findings are in line with recent Slacktivism research which proposes that people that express social motivations are reluctant to give more than token support due to a lack of interest in the campaigns cause (van den Broek et al., 2015; Kristofferson et al., 2014). Actual team membership, however, contributes to the effectiveness of online fundraising.

	Injustice	Identity	Efficacy
UK ($)	203.74	128.36	123.39
US ($)	234.47	156.07	169.03

Table 6: Average amount raised (n=90,484). British pounds were converted in dollars following the exchange rate in November 2013.

5 Conclusion

We explored the task of automatically identifying the motivations of Movember campaign's participants. A classifier based on Movember profile texts performed better than a classifier based on Twitter data, possibly due to the role of Twitter in building social relationships. Based on US and UK Movember data, we found a strong link between motivations and donations, and motivations and team membership. Classification of motivations might help campaign organizers to improve their communication strategies. Our study is limited to the Movember campaign. Future research might diverge to other types of online collective action, such as online petitions and open source communities. We also plan to explore larger datasets and features based on network structures.

Acknowledgements Thanks to Ariana Need and Anna Priante for helpful discussions and feedback on the article, Movember for support on collecting the Movember accounts and feedback on the article, and Twitter for providing part of the tweets used in this study through the Twitter Data Grant. Thanks also to the anonymous reviewers for their helpful comments. This research was funded in part by the 3TU Federation and the Dutch national project COMMIT. The first author was also supported by the Netherlands Organization for Scientific Research (NWO) grant 640.005.002 (FACT).

References

Eiji Aramaki, Sachiko Maskawa, and Mizuki Morita. 2011. Twitter catches the flu: Detecting influenza epidemics using Twitter. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*.

David Bamman, Jacob Eisenstein, and Tyler Schnoebelen. 2014. Gender identity and lexical variation in social media. *Journal of Sociolinguistics*, 18(2):135–160.

Albert Bandura. 2000. Exercise of human agency through collective efficacy. *Current Directions in Psychological Science*, 9(3):75–78.

René Bekkers and Pamala Wiepking. 2011. A literature review of empirical studies of philanthropy: Eight mechanisms that drive charitable giving. *Nonprofit and Voluntary Sector Quarterly*, 40(5):924–973.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Caroline A. Bravo and Laurie Hoffman-Goetz. 2015. Tweeting about prostate and testicular cancers: Do Twitter conversations and the 2013 Movember canada campaign objectives align? *Journal of Cancer Education*, pages 1–8.

Brian Cugelman, Mike Thelwall, and Phil Dawes. 2011. Online interventions for social marketing health behavior change campaigns: A meta-analysis of psychological architectures and adherence factors. *Journal of medical Internet research*, 13(1).

Cristian Danescu-Niculescu-Mizil, Robert West, Dan Jurafsky, Jure Leskovec, and Christopher Potts. 2013. No country for old members: User lifecycle and linguistic change in online communities. In *Proceedings of the 22nd international conference on World Wide Web (WWW '13)*.

Munmun De Choudhury. 2014. Opportunities of social media in health and well-being. *XRDS*, 21(2):23–27.

Tejas Desai, Afreen Shariff, Aabid Shariff, Mark Kats, Xiangming Fang, Cynthia Christiano, and Maria Ferris. 2012. Tweeting the meeting: An in-depth analysis of Twitter activity at Kidney Week 2011. *PLoS ONE*, 7(7):e40253.

Nugroho Dwi Prasetyo, Claudia Hauff, Dong Nguyen, Tijs A. van den Broek, and Djoerd Hiemstra. 2015. On the impact of Twitter-based health campaigns: A cross-country analysis of Movember. In *The Sixth International Workshop on Health Text Mining and Information Analysis*.

Jacob Eisenstein, Brendan O'Connor, Noah A. Smith, and Eric P. Xing. 2010. A latent variable model for geographic lexical variation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*.

Sherry L. Emery, Glen Szczypka, Eulàlia P. Abril, Yoonsang Kim, and Lisa Vera. 2014. Are you scared yet? Evaluating fear appeal messages in tweets about the tips campaign. *Journal of Communication*, 64(2):278–295.

Alastair Iain Johnston, Rawi Abdelal, Yoshiko Herrera, and Rose McDermott, editors. 2009. *Measuring Identity: A Guide for Social Scientists*. Cambridge University Press.

Bert Klandermans. 2004. The demand and supply of participation: Social-psychological correlates of participation in social movements. In *The Blackwell Companion to Social Movements*, pages 360–379. Blackwell Publishing Ltd.

Kirk Kristofferson, Katherine White, and John Peloza. 2014. The nature of slacktivism: How the social observability of an initial act of token support affects subsequent prosocial action. *Journal of Consumer Research*, 40(6):1149 – 1166.

Alex Lamb, Michael J. Paul, and Mark Dredze. 2013. Separating fact from fear: Tracking flu infections on Twitter. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

J. Richard Landis and Gary G. Koch. 1977. The measurement of observer agreement for categorical data. *Biometrics*, 33(1):159–174.

David Lazer, Alex Sandy Pentland, Lada Adamic, Sinan Aral, Albert Laszlo Barabasi, Devon Brewer, Nicholas Christakis, Noshir Contractor, James Fowler, Myron Gutmann, Tony Jebara, Gary King, Michael Macy, Deb Roy, and Marshall Van Alstyne. 2009. Life in the network: the coming age of computational social science. *Science (New York, NY)*, 323(5915):721–723.

Dong Nguyen and Carolyn P. Rosé. 2011. Language use as a reflection of socialization in online communities. In *Proceedings of the Workshop on Language in Social Media (LSM 2011)*, pages 76–85.

Dong Nguyen, Rilana Gravel, Dolf Trieschnigg, and Theo Meder. 2013. "How old do you think I am? A study of language and age in Twitter. In *Proceedings of the Seventh International AAAI Conference on Weblogs and Social Media*.

Brendan O'Connor, Ramnath Balasubramanyan, Bryan Routledge, and Noah Smith. 2010. From tweets to polls: Linking text sentiment to public opinion time series. In *Proceedings of the Fourth International AAAI Conference on Weblogs and Social Media*.

Olutobi Owoputi, Brendan O'Connor, Chris Dyer, Kevin Gimpel, Nathan Schneider, and Noah A. Smith. 2013. Improved part-of-speech tagging for online conversational text with word clusters. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Tijs A. van den Broek, David J. Langley, and Michel L. Ehrenhard. 2015. Activist versus slacktivist: A dual path model of online protest mobilization. In *Academy of Management Best Paper Proceedings, OCIS Division*.

Martijn van Zomeren, Tom Postmes, and Russell Spears. 2008. Toward an integrative social identity model of collective action: A quantitative research synthesis of three socio-psychological perspectives. *Psychological bulletin*, 134(4):504 –535.

Mackenzie R. Wehner, Mary-Margaret Chren, Melissa L. Shive, Jack S. Resneck Jr, Sherry Pagoto, Andrew B. Seidenberg, and Eleni Linos. 2014. Twitter: an opportunity for public health campaigns. *The Lancet*, 384(9938):131–132.

An Analysis of Domestic Abuse Discourse on Reddit

Nicolas Schrading[1] **Cecilia O. Alm**[2] **Ray Ptucha**[1] **Christopher M. Homan**[3]
[1] Kate Gleason College of Engineering, Rochester Institute of Technology
[2] College of Liberal Arts, Rochester Institute of Technology
[3] Golisano College of Computing and Information Sciences, Rochester Institute of Technology
$\{jxs8172^{\S}|coagla^{\S}|rwpeec^{\S}|cmh^{\dagger}\}@\{^{\S}rit.edu|^{\dagger}cs.rit.edu\}$

Abstract

Domestic abuse affects people of every race, class, age, and nation. There is significant research on the prevalence and effects of domestic abuse; however, such research typically involves population-based surveys that have high financial costs. This work provides a qualitative analysis of domestic abuse using data collected from the social and news-aggregation website reddit.com. We develop classifiers to detect submissions discussing domestic abuse, achieving accuracies of up to 92%, a substantial error reduction over its baseline. Analysis of the top features used in detecting abuse discourse provides insight into the dynamics of abusive relationships.

1 Introduction

Globally, 30% of women fifteen and older have experienced physical and/or sexual intimate partner violence at some point in their life (Devries et al., 2013). While domestic abuse tends to have greater prevalence in low-income and non-western countries, it is still endemic in regions like North America and Western Europe. In the United States, by an intimate partner, 9.4% of women have been raped, 16.9% of women and 8% of men have experienced sexual violence other than rape, and 24.3% of women and 13.8% of men have experienced severe physical violence (Black et al., 2011). This translates to an estimated economic cost of $5.8 billion for direct medical and mental health care services, along with lost productivity and reduced lifetime earnings (Craft, 2003). Economic costs are calculable and provide concrete metrics for policy makers, but the physical and psychological effects felt by victims of domestic abuse are the true costs. Domestic abuse is the 12^{th} leading cause of years of life lost (Murray et al., 2013), and it contributes to health issues including frequent headaches, chronic pain, difficulty sleeping, anxiety, and depression (Black et al., 2011).

The data used to calculate such statistics are often derived from costly and time-consuming population-based surveys that primarily seek to obtain insight into the prevalence and consequences of domestic abuse. Due to safety concerns for victims and researchers, these surveys follow strict guidelines set by the World Health Organization (Garcia-Moreno et al., 2001). Great care must be taken by the researchers to ensure the safety of the participants, and therefore the number of participants is often quite small (Burge et al., 2014). One way to avoid the cost of wide scale surveys while still maintaining appropriate research conditions is to leverage the abundance of data publicly available on the web. Of particular interest are moderated forums that allow discourse between users.

Reddit[1] is one such website, and the chosen source of data for this paper. This site has a wide range of forums dedicated to various topics, called *subreddits*, each of which are moderated by community volunteers. For subreddits dedicated to sensitive topics such as depression, domestic abuse, and suicide, the moderators tend to ensure that the anonymous submitter has access to local help hotlines if a life-threatening situation is described. They also enforce respectful behavior and ensure that the submissions are on topic by deleting disrespectful or off-topic posts. Finally, they ensure that site rules are followed, including the strict disallowal of *doxing*, the practice of using submission details to reveal user identities.

Reddit allows lengthy submissions, unlike Twitter, and therefore the use of standard English is more common. This allows natural language processing tools like semantic role labelers trained on standard English to function better. Finally, Red-

[1] See `www.reddit.com`.

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 2577–2583,
Lisbon, Portugal, 17-21 September 2015.

dit allows users to comment on submissions, providing them with the ability to ask questions, give advice, and provide support. This makes its data ideal for sensitive subjects not typically discussed in social media.

This work makes two contributions: classifiers for identifying texts discussing domestic abuse and an analysis of discussions of domestic abuse in several subreddits.

2 Related Work

Social media sites are an emerging source of data for public health studies, such as mental health, bullying, and disease tracking. These sites provide less intimidating and more accessible channels for reporting, collectively processing, and making sense of traumatic and stigmatizing experiences (Homan et al., 2014; Walther, 1996). Many researchers have focused on Twitter data, due to its prominent presence, accessibility, and the characteristics of tweets. For instance, De Choudhury et al. (2013) predicted the onset of depression from user tweets, while other studies have modeled distress (Homan et al., 2014; Lehrman et al., 2012). Most relevantly, Schrading et al. (2015) used the #WhyIStayed trend to predict whether a tweet was about staying in an abusive relationship or leaving, analyzing the lexical structures victims of abuse give for staying or leaving.

Reddit has been studied less in this area, with work mainly focusing on mental health. In Pavalanathan and De Choudhury (2015), a large number of subreddits on the topic of mental health were identified and used to determine the differences in discourse between throwaway[2] and regular accounts. They observed almost 6 times more throwaway submissions in mental health subreddits over control subreddits, and found that throwaway accounts exhibit considerable disinhibition in discussing sensitive aspects of the self. This motivates our work in analyzing Reddit submissions on domestic abuse, which can be assumed to have similar levels of throwaway accounts and discussion. Additionally, Balani and De Choudhury (2015) used standard ngram features, along with submission and author attributes to classify a submission as high or low self-disclosure.

[2]Anonymous, one-time accounts to submit a single (often personal or sensitive) submission or comment.

3 Dataset[3] and Data Analysis

Following the procedure in Balani and De Choudhury (2015) for subreddit discovery, we identified several subreddits that focus on domestic abuse. Additionally, we determined subreddits unrelated to domestic abuse, to be used as a control set. Table 1 shows the subreddits, the total number of unique posts (called *submissions*[4]), and total number of replies to those submissions (called *comments*) collected.

Domestic Abuse	**# Submissions**	**# Comments**
abuseinterrupted	1653	1069
domesticviolence	749	2145
survivorsofabuse	512	2172
Control	**# Submissions**	**# Comments**
casualconversation	7286	285575
advice	5913	31323
anxiety	4183	23300
anger	837	3693

Table 1: The domestic abuse subreddits and control subreddits with the total number of submissions and comments collected.

The *anger* and *anxiety* subreddits were chosen as control subreddits in order to help the classifier discriminate between the dynamics of abusive relationships and the potential effects of abuse on victims. For example, anxiety and anger may be affect caused by domestic abuse, but they are also caused by a wide variety of other factors. By including these subreddits in the control set, a classifier should utilize the situations, causes, and stakeholders in abusive relationships as features, not the affect particularly associated with abusive relationships. Similarly, the *advice* subreddit was chosen as a way to help the classifier understand that advice-seeking behavior is not indicative of abuse. The *casualconversation* subreddit allows discussion of anything, providing an excellent sample of general written discourse. The domestic abuse subreddits have far fewer active users, submissions, and comments in total.

3.1 Preprocessing

All experiments used the same preprocessing steps. From the collected subreddits, only submissions with at least one comment were chosen to be included for study. We then ran the submission text through the Illinois Curator (Clarke et

[3]Released on nicschrading.com/data/.

[4]Submission text is its title and selftext (an optional text body) concatenated together.

al., 2012) to provide semantic role labeling (SRL) (Punyakanok et al., 2008). A total of 552 domestic abuse submissions were parsed, and we randomly chose an even distribution of the control subreddits (138 each), yielding a total sample size of 1104. All submissions were normalized by lowercasing, lemmatizing, and stoplisting. External links and URLs were replaced with *url* and references to subreddits, e.g. */r/domesticviolence*, were replaced with *subreddit_link*.

3.2 Descriptive Statistics

We present basic descriptive statistics on the set of 552 abuse submissions and 552 non-abuse submissions in Table 2.

	Abuse	**Non-Abuse**
Avg comments/post	5.4 ± 6.1	13.2 ± 25.3
Avg score/post	6.1 ± 5.1	7.5 ± 16.4
Avg tokens/post	278 ± 170	208 ± 164
# unique submitters	482	535
Avg comment depth	0.96 ± 1.5	1.5 ± 1.9
Avg comment score	2.2 ± 2.7	2.1 ± 2.9
Avg tokens/comment	107 ± 128	53.4 ± 79.9
# comments	2989	6964
# unique commenters	1022	2519

Table 2: Basic descriptive statistics. The score is provided by users voting on submissions/comments they feel are informative. The depth of a comment indicates where in a reply chain it falls. A depth of 0 means it is in reply to the submission, a depth of 1 means it is in reply to a depth 0 comment, etc. The $\pm$ values are standard deviation metrics.

In general, the non-abuse subreddits have more discourse between commenters, as indicated by a larger comment depth, however, the abuse subreddits tend to have longer submissions and replies. The abuse subreddits perhaps also have a smaller more tight-knit, community as indicated by fewer numbers of unique submitters and commenters.

3.3 Ngram Attributes

To get a sense of the language used between the two sets of subreddits, the most frequent 1-, 2-, and 3-grams were examined. While there are many common and overlapping ngrams in the two sets, each set does have distinct ngrams. In the abuse set, distinct ngrams include the obvious *abuse* (1595 occurences), *domestic violence* (202), and *abusive relationship* (166). Additionally, unique 3-grams related to the agents and situations in abusive relationships like *local dv agency* (12) and *make feel bad* (11) appear. Also included are unique empathetic and helping discourse from comments, including *let know* (121), and *feel free pm*[5] (27). This indicates that comment data could improve classification results, as support and empathy may be more prevalent in the *abuse* set than in the control set.

3.4 Semantic Role Attributes

From the SRL tool, our dataset was tagged with various arguments of predicates. This data is particularly useful in our study, as we are interested in examining the semantic actions and stakeholders within an abusive relationship. By performing a lookup in Proposition Bank (Martha et al., 2005) with a given argument number, predicate, and sense, we retrieved unique role labels for each argument.

We determined the top 100 most frequent roles and predicates in the two sets, and took only the unique roles and predicates within each set to see what frequently occurring but unique roles and predicates exist within the abuse and control group.

Role Label	**Predicate**
caller, 175	abuse, 433
thing hit, 174	share, 167
agent, hitter - animate only!, 164	believe, 164
abuser, agent, 162	call, 151
entity abused, 139	remember, 149
utterance, 115	cry, 147
patient, entity experiencing hurt/damage, 113	!tell, 142
utterance, sound, 104	send, 127
belief, 104	thank, 127
benefactive, 103	realize, 124

Table 3: Top 10 unique roles and predicates with their frequency for the abuse data. An exclamation point on a predicate indicates negation.

Table 3 contains roles and predicates that are powerful indicators of an abusive relationship, including a *caller*, *hitter*, *thing hit*, *abuser*, and *entity experiencing hurt/damage*. Importantly, several predicates that appear in this data also appear in a study on discussions of domestic abuse in Twitter data, including *believe* and *realize*, which indicate cognitive manipulation in the victims of domestic abuse (Schrading et al., 2015).

[5]The initials *pm* stand for private message.

Classifer	N	P	R	N+P	N+R	P+R	N+P+R
Linear SVM	$\mathbf{90 \pm 3}$	72 ± 5	73 ± 4	88 ± 3	88 ± 3	73 ± 4	87 ± 3

Table 4: Classification accuracies of Linear SVM. N=Ngrams, P=Predicates, R=Roles.

4 Classification Experiments

In order to discover the semantic and lexical features salient to abusive relationships, we designed several classifiers. The subreddit category to which a submission was posted was used as the gold standard label of *abuse* or *non-abuse*. The labels were validated by examining the top ngrams, roles, and predicates in Section 3, and taking into account that these subreddits are moderated for on-topic content. We ran several experiments to study classifiers, the impact of features, and the effect of comments on prediction performance.

4.1 Combinations of Features

We used the 1-, 2-, and 3-grams in the submission text, the predicates, and the semantic role labels as features, using TF-IDF vectorization[6]. Perceptron, naïve Bayes, logistic regression, random forest, radial basis function SVM, and linear SVM classifiers were parameter optimized using 10-fold cross validation. Table 4 contains the results for the best classifier. The best features are the ngrams, achieving the highest accuracies alone. Predicates and semantic roles perform admirably, but bring the classifier accuracies down slightly when added to ngrams. To determine the top features for prediction, we examined the weights of the top performing classifier, Scikit-learn's (Pedregosa et al., 2011) Linear SVM with C=0.1, as in Guyon et al. (2002). These, along with their weights, are shown in Table 5.

4.2 Comment Data Only

We experimented with only comment data to predict if they were posted in an *abuse* or *non-abuse* subreddit. Because ngram features performed best in the previous experiment, a larger set of submissions (1336 per class) was used. A final held out testset was created from 10% of these submissions, giving 1202 submissions per class for the devset and 134 per class for the testset. Taking the comments from these submissions yielded 4712 abuse and 19349 non-abuse comments for the devset and 642 abuse and 2264 non-abuse comments for the testset. 10-fold cross-validation was used on the devset to tune the classifier. Using a Linear SVM with C=1 achieved an F1 score of $0.70 \pm .02$ on the devset. On the held out testset, it achieved a precision of 0.68, recall of 0.62, and F1 score of 0.65. Examining its weights gives features similar to those in Table 5, with additional empathetic discourse like *thank*, *hug*, and *safe* in the abuse class.

[6]Binary features and only unigrams were tried but these did not improve results.

Abuse	Non Abuse
abusive, 1.3	anxiety, 1.1
child, 0.93	anger, 1.1
abuser, 0.86	job, 0.52
relationship, 0.84	school, 0.46
therapy, 0.83	hour, 0.45
survivor, 0.83	week, 0.45
domestic, 0.73	fuck, 0.44
happen, 0.72	class, 0.42
violence, 0.68	college, 0.41
father, 0.67	fun, 0.40

Table 5: Top 10 features based on Linear SVM weights using only ngrams from submissions. The classifier may be relying heavily on the anxiety and anger subreddits to discriminate between abuse and non-abuse, as indicated by the sharp drop in SVM weight from *anger* to *job*. Abuse word weights are more evenly distributed.

4.3 Comment and Submission Text Combined

Concatenating the comments to their respective submissions may improve results, but because comments can be completely off-topic or in reply to other comments, we experimented with only the top-scoring comments and those most similar to the submission text. To compute similarity we used a sum of the word vector representations from Levy and Goldberg (2014) as included in spaCy (Honnibal, 2015) and used cosine similarity. Taking only the top 90^{th} percentile in user voting score and text similarity, we had 2688 abuse comments and 7852 non-abuse comments concatenated to the 1336 submissions of their class. This method achieves extremely high accuracy of $94\% \pm 2\%$ on the devset and 92% on the testset using a Linear SVM with C=1. A classifier trained only on the submission text data from the same devset/testset split obtains the lower accuracies of

$90\% \pm 2\%$ on the devset and 86% on the testset, indicating that comments can add predictive power. The top features are similar to those in Table 5.

4.4 Uneven Set of Submissions

Using the method in Section 4.3 to train the classifier, a larger, uneven set of data was examined (still using only ngrams). This set contained 1336 *abuse* and 17020 *non-abuse* instances. From this set, 15% were held out for final examination as a testset and the rest was used as a devset with 5-fold cross-validation. On the devset, an F1 score of 0.81 ± 0.01 was achieved while on the testset it had a precision of 0.84, recall of 0.74, and F1 score of 0.79. The best classifier was a Linear SVM with C=100. The confusion matrix of the testset is in Table 4.4.

		Predicted Class	
		Abuse	**Non-Abuse**
Actual Class	**Abuse**	152	53
	Non-Abuse	29	2520

Table 6: Confusion matrix on the testset of the Abuse/Non-Abuse classifier.

This classifier has good precision for the *abuse* class, and decent recall, meaning that there can be confidence that submissions flagged as *abuse* are indeed about *abuse*. By applying this classifier to a large held out set of data, these results suggest that many potentially relevant submissions would be flagged for examination, and they would mostly be about *abuse*.

4.5 Testing on Completely Held Out Subreddits

To get a sense of efficacy in the wild in detecting submissions about abuse, the best classifier from Section 4.4 was taken (trained on the devset data) and run on a large set of submissions from the *relationships* and *relationship_advice* subreddits. These subreddits are general forums for discussion and advice on any relationship (not necessarily intimate). Their submissions tend to be long, descriptive, and extremely personal.

After running the abuse classifier on the submissions from these subreddits with at least 1 comment (13623 in total, with their 90^{th} percentile comments concatenated), 423 submissions were flagged as being about abuse. 101 of these 423 were annotated by 3 annotators (co-authors), using the labels *A* (the submission discusses an abusive relationship), *M* (off-hand mention of abuse), *N* (not about abuse), and *O* (off-topic submission or other).

From the three annotators' annotations, on average 59% are *A*, 16% are *M*, 23% are *N*, and 2% are *O*. The percentage of overall agreement was 72% and Randolph's free-marginal multirater kappa (Warrens, 2010) score was 0.63. Annotators occasionally had a hard time distinguishing between *A* and *M*, as context may have been missing. Combining the two by considering all *M* as *A*, the average percent of *A* increases to 75%, the percentage of overall agreement improves to 86% and the kappa score improves to 0.79. Taking the statistic that on average 75% of the flagged submissions in the annotated subset are about abuse or have a mention of abuse indicates that this classifier should hopefully have a precision of around 0.75 on unseen Reddit data at large. Understandably, the precision drops by about .1 compared to its use on the subreddits it was trained and tested on. A precision of 0.75 on this set of data would mean that any statistics from this set may include some noise, but overall, the trends should reveal important results about abuse.

5 Conclusion

This work provides an analysis of domestic abuse using the online social site Reddit. Language analysis reveals interesting patterns used in discussing abuse, as well as initial data about the semantic actions and stakeholders involved in abusive relationships. Multiple classifiers were implemented to determine the top semantic and linguistic features in detecting abusive relationships. Simpler features such as ngrams performed above the more complex predicate and role labels extracted from a semantic role labeler, though the more complex structures contribute to interesting insights in data analysis. Future work could use a larger training set from multiple online sites to analyze the patterns of online abuse discourse across varied forums.

Acknowledgement

This work was supported in part by a GCCIS Kodak Endowed Chair Fund Health Information Technology Strategic Initiative Grant and NSF Award #SES-1111016.

References

Sairam Balani and Munmun De Choudhury. 2015. Detecting and characterizing mental health related self-disclosure in social media. In *Proceedings of the 33rd Annual Association for Computing Machinery Conference Extended Abstracts on Human Factors in Computing Systems*, CHI EA '15, pages 1373–1378, New York, NY, USA. ACM.

Michele C Black, Kathleen C Basile, Matthew J Breiding, Sharon G Smith, Mikel L Walters, Melissa T Merrick, and MR Stevens. 2011. National intimate partner and sexual violence survey. *Atlanta, GA: Centers for Disease Control and Prevention*, 75.

Sandra K. Burge, Johanna Becho, Robert L. Ferrer, Robert C. Wood, Melissa Talamantes, and David A. Katerndahl. 2014. Safely examining complex dynamics of intimate partner violence. *Families, Systems, & Health*, 32(3):259 – 270.

Munmun De Choudhury, Scott Counts, Eric Horvitz, and Michael Gamon. 2013. Predicting depression via social media. In *Proceedings of the Seventh Annual International Association for the Advancement of Artificial Intelligence Conference on Weblogs and Social Media*, pages 128–137. AAAI, July.

James Clarke, Vivek Srikumar, Mark Sammons, and Dan Roth. 2012. An NLP curator (or: How I learned to stop worrying and love NLP pipelines). In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 3276–3283, Istanbul, Turkey, may. European Language Resources Association (ELRA).

Carole Craft. 2003. Costs of intimate partner violence against women in the United States. Technical report, National Center for Injury Prevention and Control, Centers for Disease Control and Prevention, Atlanta, GA.

K.M. Devries, Joelle Y.T. Mak, C. García-Moreno, M. Petzold, J.C. Child, G. Falder, S. Lim, L.J. Bacchus, R.E. Engell, L. Rosenfeld, C. Pallitto, T. Voss, and C.H. Watts. 2013. The global prevalence of intimate partner violence against women. *Science*, 340(6140):1527–1528.

Claudia Garcia-Moreno, C Watts, and L Heise. 2001. Putting women first: Ethical and safety recommendations for research on domestic violence against women. *Department of Gender and Womens Health, World Health Organization. Geneva, Switzerland*.

Isabelle Guyon, Jason Weston, Stephen Barnhill, and Vladimir Vapnik. 2002. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46(1-3):389–422, March.

Christopher Homan, Ravdeep Johar, Tong Liu, Megan Lytle, Vincent Silenzio, and Cecilia Ovesdotter Alm. 2014. Toward macro-insights for suicide prevention: Analyzing fine-grained distress at scale. In *Proceedings of the Workshop on Computational Linguistics and Clinical Psychology: From Linguistic Signal to Clinical Reality*, pages 107–117, Baltimore, Maryland, USA, June. Association for Computational Linguistics.

Matthew Honnibal. 2015. SpaCy: Industrial strength NLP with Python and Cython. `https://github.com/honnibal/spaCy`.

Michael Thaul Lehrman, Cecilia Ovesdotter Alm, and Rubén A. Proaño. 2012. Detecting distressed and non-distressed affect states in short forum texts. In *Proceedings of the Second Workshop on Language in Social Media*, LSM '12, pages 9–18, Stroudsburg, PA, USA. Association for Computational Linguistics.

Omer Levy and Yoav Goldberg. 2014. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, volume 2, pages 302–308.

Palmer Martha, Gildea Dan, and Kingsbury Paul. 2005. The Proposition Bank: A corpus annotated with semantic roles. *Computational Linguistics Journal*, 31:71–106.

Christopher JL Murray, Jerry Abraham, Mohammed K Ali, Miriam Alvarado, Charles Atkinson, Larry M Baddour, David H Bartels, Emelia J Benjamin, Kavi Bhalla, Gretchen Birbeck, et al. 2013. The state of US health, 1990-2010: Burden of diseases, injuries, and risk factors. *JAMA*, 310(6):591–606.

Umashanthi Pavalanathan and Munmun De Choudhury. 2015. Identity management and mental health discourse in social media. In *Proceedings of WWW'15 Companion: 24th International World Wide Web Conference, Web Science Track*, Florence, Italy, May. WWW'15 Companion.

Fabian Pedregosa, Gaël Varoquaux., Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Vasin Punyakanok, Dan Roth, and Wen-tau Yih. 2008. The importance of syntactic parsing and inference in semantic role labeling. *Computational Linguistics*, 34(2):257–287.

Nicolas Schrading, Cecilia Ovesdotter Alm, Raymond Ptucha, and Christopher Homan. 2015. #WhyISTayed, #WhyILeft: Microblogging to make sense of domestic abuse. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1281–1286, Denver, Colorado, May–June. Association for Computational Linguistics.

Joseph Walther. 1996. Computer-mediated communication: Impersonal, interpersonal, and hyperpersonal interaction. *Communication Research*, 23(1):3–43, Feb.

Matthijs J. Warrens. 2010. Inequalities between multirater kappas. *Advances in Data Analysis and Classification*, 4(4):271–286.

Twitter-scale New Event Detection via K-term Hashing

Dominik Wurzer
School of Informatics
University of Edinburgh
d.s.wurzer
@sms.ed.ac.uk

Victor Lavrenko
School of Informatics
University of Edinburgh
vlavrenk
@inf.ed.ac.uk

Miles Osborne
Bloomberg
London
mosborne29
@bloomberg.net

Abstract

First Story Detection is hard because the most accurate systems become progressively slower with each document processed. We present a novel approach to FSD, which operates in constant time/space and scales to very high volume streams. We show that when computing novelty over a large dataset of tweets, our method performs 192 times faster than a state-of-the-art baseline without sacrificing accuracy. Our method is capable of performing FSD on the full Twitter stream on a single core of modest hardware.

1 Introduction

First Story Detection (FSD), also called New Event Detection, is the task of identifying the very first document in a stream to mention a new event[1]. FSD was introduced as port of the TDT[2] initiative and has direct applications in finance, news and government security. The most accurate approaches to FSD involve a runtime of $O(n^2)$ and cannot scale to unbounded high volume streams such as Twitter. We present a novel approach to FSD that operates in $O(1)$ per tweet. Our method is able to process the load of the average Twitter Firehose[3] stream on a single core of modest hardware while retaining effectiveness on par with one of the most accurate FSD systems. During the TDT program, FSD was applied to news wire documents and solely focused on effectiveness, neglecting efficiency and scalability. The traditional approach to FSD (Petrovic et al., 2010) computes the distance of each incoming document to all previously seen documents and the minimum distance determines the novelty score. Documents, whose minimum distance falls above a certain threshold are considered to talk about a new event and declared as first stories. Consequently, the computational effort increases with each document processed.

1.1 Related Work

Researchers have proposed a range of approaches to scale FSD to large data streams. Sankaranarayanan et al. (2009) were one of the first to apply FSD to Twitter. They reduced the volume by classifying documents into news/non-news and only compared to tweets within a 3-day window. They did not perform a quantitative evaluation of their approach. Sakaki et al. (2010) and Li et al. (2012) applied keyword filtering in conjunction with classification algorithms, which allowed them to efficiently detect certain events with high precision. These two approaches, although efficient and effective, require a user to explicitly define a set of keywords or to provide a set of examples that he wants to track. The approach cannot detect previously unknown events.

Phuvipadawat and Murata (2010), Ozdikis et al. (2012) and Cordeiro (2012), scale their systems by only considering tweets containing hashtags. Although efficient, this method don't consider 90% of the tweets (Petrovic, 2013), which limits their scope.

Cataldi et al. (2010), Weng et al.(2011) and Cordeiro (2012) use the degree of burstiness of terms during a time interval to detect new events. This approach is not suitable for FSD as events are detected with a time lag, once they grow in popularity.

Petrovic et al. (2010) were the first to demonstrate FSD on Twitter in constant time and space, while maintaining effectiveness comparable to those of pair-wise comparison systems. The key was to

[1] e.g. a natural disaster or a scandal

[2] TDT by NIST - 1998-2004. http://www.itl.nist.gov/iad/mig/tests/tdt/resources.html (Last Update: 2008)

[3] 5,700 tweets per second https://about.twitter.com/company (last updated: March 31, 2015)

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 2584–2589,
Lisbon, Portugal, 17-21 September 2015.

reduce the search space using Locality Sensitive Hashing (LSH). Each tweet was hashed, placing it into buckets that contain other similar tweets, which are subsequently compared. Operation in constant space was ensured by keeping the number of tweets per bucket constant. Because LSH alone performed ineffectively, Petrovic et al. (2010) additionally compared each incoming tweet with the k most recent tweets.

Allan et al. (2003) analysed scoring functions for novelty detection while focusing on their effectiveness. They presented a language-model (LM) based novelty measure using the KL divergence between the LM of a document and a single LM built on all previously scored documents, which they referred to as an aggregate measure language model. The idea of maintaining a single representation covering all previously seen documents, instead of performing pairwise comparisons with every document is closely related to the approach presented in this paper.

2 Approach

First Story Detection is a challenging task (Allan et al., 2000). The highest FSD accuracy is achieved by nearest-neighbour methods, where each incoming document (tweet) is compared to all documents that came before it, and the novelty score is determined by the most-similar documents in the past. This approach requires us to make $n-1$ comparisons[4] to determine the novelty of document d_n. The approach becomes progressively slower with each processed document, and cannot scale up to unbounded streams like Twitter. Prior attempts to speed up FSD involve organising previously seen documents $d_1 \ldots d_{n-1}$ into clusters (Allan et al., 1989) or LSH buckets (Petrovic et al., 2010). The document d_n is then compared only to past documents in the nearest cluster or LSH bucket, resulting in substantially fewer than n comparisons. While this approach is reasonably effective, it does lead to decreased accuracy, as potentially relevant past documents may exist in other clusters/buckets and would not be compared against.

2.1 First Story Detection in constant time

Our method computes the novelty of document d_n in a time that is constant with respect to n. The main difference from previous approaches is that we do not compare d_n to individual documents that came before it. Instead, we store the content of past documents $d_1 \ldots d_{n-1}$ in a single lookup table H_{n-1}. When d_n arrives, we count what fraction of its content is novel by looking it up in H_{n-1}. The number of lookups is polynomial in $|d_n|$ (the length of the document), and is independent of n.

Formally, let d_n denote the set of distinct words occurring in the n'th document in the stream. Let a k-term $t{=}\{w_1, w_2, \ldots\}$ denote a non-empty set of up to k distinct words. We define the content c_n to be the set of all k-terms that can be formed from the words in the document $d_n : c_n = \{\, t : t \subset d_n,\ |t| \leq k\}$. We estimate the novelty of document d_n as the proportion of novel k-terms, i.e. k-terms that do not appear in the history H_{n-1}:

$$\mathrm{N}(d_n) = \sum_{t \in c_n} \alpha_{|t|} \binom{|d_n|}{|t|}^{-1} \begin{Bmatrix} 1 : t \notin H_{n-1} \\ 0 : t \in H_{n-1} \end{Bmatrix} \quad (1)$$

Here $\alpha_{|t|}$ is the weight assigned to k-terms of size $|t|$, and $\binom{|d_n|}{|t|}$ is the total number of such k-terms formed from d_n. After the novelty is computed, we update the history H to include all k-terms formed from d_n:

$$H_n \leftarrow H_{n-1} \cup c_n \quad (2)$$

The computational cost of equations (1) and (2) is determined by the number of k-terms formed from the document d_n, and can be bounded at $O(|d_n|^k)$ operations. The complexity is manageable, as tweets are short and we keep k small.

2.2 Operating in constant time and space

We use a Bloom filter (Bloom, 1970) to maintain the history H_{n-1} of previously seen k-terms. For each k-term t we compute a 32-bit Murmur[5] hashcode, and use it as an index into a fixed-length bit-array. This ensures that both membership testing ($t{\in}H$) and history update can be performed in constant time. Constraining H to be a fixed-length array also means that our method operates in constant space, irrespective of the size of the stream and its vocabulary growth. In contrast to our method, previous approaches to FSD required more and more memory to maintain the history of the stream (see Figure 3).

[4] Each comparison requires $|d_n|$ scalar multiplications; $|d|$ denotes the number of distinct words in document d.

[5] https://en.wikipedia.org/wiki/MurmurHash

A potential downside of using a Bloom filter is that it introduces a small probability of false matches: a novel k-term t_i may collide with a previously observed k-term t_j and would be reported as non-novel. The probability of collision is directly proportional to the load factor of the Bloom filter, i.e. the fraction of non-zero bits in the array. By Heaps law (Egghe, 2007) the number of distinct words (and k-terms) will continue to grow and will eventually saturate the bit-array. To mitigate this problem, we introduce a deletion strategy: whenever the load factor exceeds a pre-determined threshold ρ, we zero out a random bit in H. This allows us to keep low the probability of false matches, at the cost of forgetting some previously-seen k-terms.

2.3 Parameter settings

We make the following parameter choices based on initial experiments on our training dataset. We set the maximum size of k-terms to be $k = 3$ and keep the Bloom filter load factor under $\rho = 0.6$. We tokenize the tweets on punctuation, treat all hashtags and mentions as words, stem them using the stemmer by Krovetz (1993), but do not remove stopwords. We optimise the weights $\alpha_1 \ldots \alpha_k$ using grid search on the same training data set.

3 Experiments

In a streaming setting, documents arrive one at a time on a continual basis. FSD requires computing a novelty score for each document in a single-pass over the data. High novelty scores indicate new topics. We use the standard TDT evaluation procedure (Allan, 2002) and the official TDT3 evaluation scripts with standard settings for evaluating FSD accuracy. The Detection Error Trade-off (DET) curve shows the trade-off between miss and false alarm probability for the full range of novelty scores. The normalized Topic Weighted Minimum Cost (C_{min}) is a linear combination of miss and false alarm probabilities, which allows comparing different methods based on a single value metric. Efficiency is measured by the throughput of tweets per second and the memory footprint. To ensure a fair comparison, all reported numbers are averaged over 5 runs on an idle machine using a single core (Intel-Xeon CPU with 2.27GHz).

3.1 Data set

We use the data set developed by Petrovic (2013), Petrovic et al. (2013b) as a test set, which consists of 27 topics and 116,000 tweets from the period of April till September 2011. Parameters were tuned using a sample of the data set annotated by Wurzer et al. (2015) as a training set.

3.2 Baselines

We compare our system (**k-term**) against 3 baselines.

UMass is a state-of-the-art FSD system, developed by Allan et al. (2000). It is known for its high effectiveness in the TDT2 and TDT3 competitions (Fiscus, 2001) and widely used as a benchmark for FSD systems (Petrovic et al., 2010; Kasiviswanathan et al., 2011; Petrovic 2013;). UMass makes use of an inverted index and k-nearest-neighbour clustering, which optimize the system for speed by ensuring a minimal number of comparisons. To maximise efficiency, we set-up UMass to operate in-memory by turning off its default memory mapping to disk. This ensures fair comparisons, as all algorithms operate in memory.

LSH-FSD is a highly-scalable system by Petrovic et al. (2010). It is based on Locality Sensitive Hashing (LSH) and claims to operate in constant time and space while performing on a comparable level of accuracy as UMass. We configure their system using the default parameters (Petrovic et al., 2010).

KL-FSD We also compare our approach with the aggregate measure language model (Allan et al., 2003) because it builds upon a similar principle.

3.3 Effectiveness and Efficiency

In Table 1, the UMass system shows state-of-the-art accuracy ($C_{min} = 0.79$, lower is better), but can only process 30 tweets per second. LSH-FSD operates 17 times faster, at the cost of a 13% decrease in accuracy ($C_{min} = 0.90$). Our system (k-term) operates on par with UMass in terms of accuracy, while being 197 times faster. KL-FSD, which is based on uni-grams, reveals the highest throughput at a considerable cost of efficiency ($C_{min} = 0.96$).

To further investigate accuracy we also compare the systems over the full range of the novelty thresholds illustrated by the DET plot in Figure 1.

Algorithm	Cmin	%-diff	tweets/sec	speed-up
UMass	0.7981	-	30	-
LSH-FSD	0.9061	-13.5%	500	17x
KL-FSD	0.9648	-21%	**6,600**	220x
k-term	**0.7966**	+0.2%	5,900	197x

Table 1: Comparing the effectiveness and efficiency of our system (k-term) with the 3 baselines

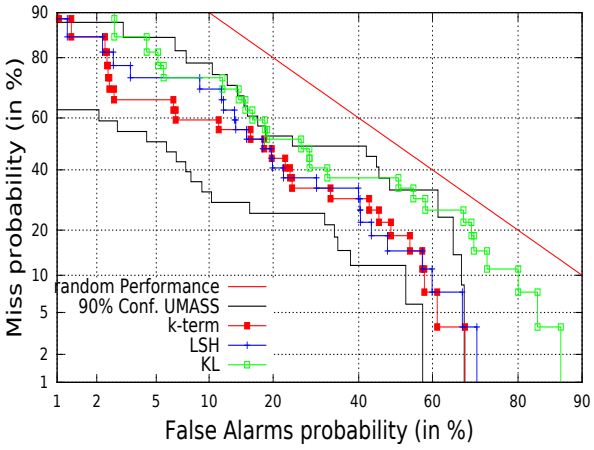


Figure 1: DET plot of UMass, Kl-FSD, LSH-FSD and k-term showing that LSH and k-term are statistically indistinguishable from UMass in terms of effectiveness;

Additionally we show the 90% confidence interval of UMass in two solid lines. We observer that both, FSD-LSH and our system (k-term) are statistically indistinguishable form UMass at any Miss-False Alarm trade-off point: their DET curves fall entirely within the 90% confidence interval of UMass. Note that DET curve of UMass is formed by the middle of it's 90% confidence interval curves. KL-FSD in contrast results in significantly worse accuracy than UMass in the mid-range and in particular the high recall area of the DET plot. We conclude that uni-grams are insufficient for determining the novelty of tweets.

3.4 FSD in constant time and space

High-volume streams require operation in constant time and space. Figure 2 compares the change in throughput of LSH-FSD, UMass and k-term as we process more and more tweets in the stream. Additionally, the plot also shows the average rate of tweets in the Twitter Firehose[6] at 5,787 tweets per second. Note that our system processes the equivalent of the full Twitter stream on a single core of modest hardware. This surpasses the throughput of LSH-FSD, a system known for high

[6]https://about.twitter.com/company (last updated: March 31, 2015)

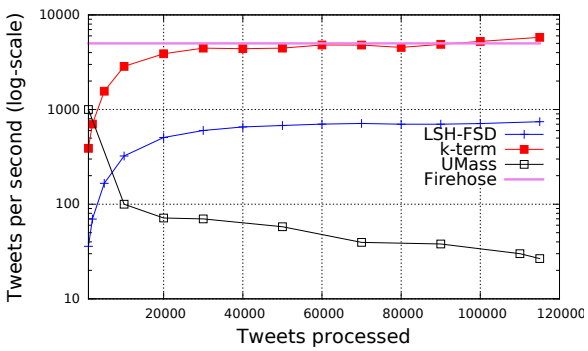


Figure 2: Throughput of UMass, LSH-FSD and k-term in comparison to the full Twitter stream (Firehose)

efficiency, by more than an order of magnitude. The throughput of LSH-FSD and k-term increases up until 20k documents because both approaches require initialisation of their data structures, which makes them slow when the number of documents is low. UMass has no initialization and performs the fastest when the number of documents is kept low. The pair-wise comparison of UMass causes it's throughput to decrease drastically with every new document. In Figure 2 we compare the memory requirements of k-term and LSH-FSD at different points in the stream. Although Petrovic et al. (2010) designed their system (LSH-FSD) to operate in constant space, we found that the memory requirement gradually increases with the number of documents processed, as seen in Figure 3. We hypothesise that this increase results from new terms added to the vocabulary. Our system has a strictly constant memory footprint.

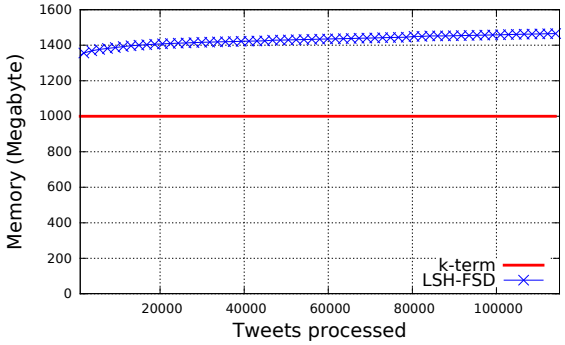


Figure 3: Space requirement for LSH-FSD and k-term; showing constant space for k-term

4 Conclusion

We presented an approach to FSD in a high volume streaming setting in constant time and space. Our approach computes novelty based on a single

lookup table that represents past documents. Shifting from direct comparisons with previous documents to comparisons with a single model that combines them, accounts for a great increase in efficiency. For the first time, we showed that it is possible to perform FSD on the full Twitter stream on a single core of modest hardware. This greatly outperforms state-of-the-art systems by an order of magnitude without sacrificing accuracy.

References

J. Allan, C. Wade, and A. Bolivar. Retrieval and novelty detection at the sentence level. In SIGIR 03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval, pages 314 - 321. ACM Press, 2003

James Allan. 2002. Topic Detection and Tracking: Event-Based Information Organization. Kluwer Academic Publishers, Norwell, MA, USA.

James Allan, Victor Lavrenko and Hubert Jin. 2000. First story detection in TDT is hard. In Proceedings of the ninth international conference on Information and knowledge management. ACM.

James Allan, Ron Papka and Victor Lavrenko. 1998. On-line new event detection and tracking. Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval. ACM.

Burton H. Bloom. 1970. Space/time trade-offs in hash coding with allowable errors. Communications of the ACM, 13(7), 422-426.

Cataldi, M., Caro, L. D., and Schifanella, C. (2010). Emerging topic detection on Twitter based on temporal and social terms evaluation. In Proceedings of the 10th International Workshop on Multimedia Data Mining, pages 4:1 - 4:10. ACM.

Cordeiro, M. (2012). Twitter event detection: Combining wavelet analysis and topic inference summarization. In Doctoral Symposium in Informatics Engineering, pages 123 - 138.

Leo Egghe. 2007. Untangling Herdan's law and Heaps' law: Mathematical and informetric arguments. Journal of the American Society for Information Science and Technology 58.5: 702-709.

Piotr Indyk and Rajeev Motwani. 1998. Approximate nearest neighbours: towards removing the curse of dimensionality. In Proceedings of the thirtieth annual ACM symposium on Theory of computing (STOC '98). ACM, New York, NY, USA.

Shiva Prasad Kasiviswanathan, Prem Melville, Arindam Banerjee, and Vikas Sindhwani. Emerging topic detection using dictionary learning. In Proceedings of the Twentieth ACM international conference on Information and knowledge management, 2011.

Robert Krovetz. 1993. Viewing morphology as an inference process. Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval. ACM.

Li, R., Lei, K. H., Khadiwala, R., and Chang, K. C.-C. (2012). TEDAS: A Twitter-based event detection and analysis system. In Proceedings of 28th International Conference on Data Engineering, pages 1273 - 1276. IEEE Computer Society.

Li, C., Sun, A., and Datta, A. (2012b). Twevent: Segment-based event detection from tweets. In Proceedings of ACM Conference on Information and Knowledge Management. ACM.

Jimmy Lin, Rion Snow, and William Morgan. 2011. Smoothing techniques for adaptive online language models: topic tracking in tweet streams. In Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '11). ACM, New York, NY, USA, 422-429.

S. Muthukrishnan. 2005. Data streams: Algorithms and applications. Now Publishers Inc.

Jeffrey Nichols, Jalal Mahmud, and Clemens Drews. 2012. Summarizing sporting events using twitter. In Proceedings of the 2012 ACM international conference on Intelligent User Interfaces (IUI '12). ACM, New York, NY, USA.

Ozdikis, O., Senkul, P., and Oguztuzun, H. (2012). Semantic expansion of hashtags for enhanced event detection in Twitter. In Proceedings of the 1st International Workshop on Online Social Systems.

Sasa Petrovic, Miles Osborne, and Victor Lavrenko. 2010. Streaming first story detection with application to Twitter. In Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics (HLT '10). Association for Computational Linguistics, Stroudsburg, PA, USA.

Sasa Petrovic. 2013. Real-time event detection in massive streams. Ph.D. thesis, School of Informatics, University of Edinburgh.

Sasa Petrovic, Miles Osborne, Richard McCreadie, Craig Macdonald, Iadh Ounis, and Luke Shrimpton. Can Twitter replace Newswire for breaking news? In Proc.of ICWSM, 2013b.

Raymond K. Pon, Alfonso F. Cardenas, David Buttler, and Terence Critchlow. 2007. Tracking multiple topics for finding interesting articles. In Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '07). ACM, New York, NY, USA.

Phuvipadawat, S. and Murata, T. (2010). Breaking news detection and tracking in Twitter. In Proceedings of the 2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, pages 120 - 123. IEEE Computer Society.

Deepak Ravichandran, Patrick Pantel, and Eduard Hovy. 2005. Randomized Algorithms and NLP: Using Locality Sensitive Hash Functions for High Speed Noun Clustering. In Proceedings of ACL.

Sankaranarayanan, J., Samet, H., Teitler, B. E., Lieberman, M. D., and Sperling, J. (2009). Twitterstand: news in tweets. In Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, pages 42 - 51. ACM.

Sakaki, T., Okazaki, M., and Matsuo, Y. (2010). Earthquake shakes Twitter users: real-time event detection by social sensors. In Proceedings of the 19th International Conference on World Wide Web, pages 851 - 860. ACM.

I. Soboroff, I. Ounis, and J. Lin. 2012. Overview of the trec-2012 microblog track. In Proceedings of TREC.

Jintao Tang, Ting Wang, Qin Lu, Ji Wang, and Wenjie Li. 2011. A Wikipedia based semantic graph model for topic tracking in blogosphere. In Proceedings of the Twenty-Second international joint conference on Artificial Intelligence - Volume Three (IJCAI'11).

TDT by NIST - 1998-2004. http://www.itl.nist.gov/iad/mig/tests/tdt/resources.html (Last Update: 2008)

Jianshu Weng, Erwin Leonardi, Francis Lee. Event Detection in Twitter. 2011. In Proceeding of ICWSM. AAAI Press.

Weng, J., Yao, Y., Leonardi, E., and Lee, F. (2011). Event detection in Twitter. In Proceedings of the 5th International Conference on Weblogs and Social Media, pages 401 - 408. The AAAI Press.

Dominik Wurzer, Victor Lavrenko, Miles Osborne. 2015. Tracking unbounded Topic Streams. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL) and the 7th International Joint Conference on Natural Language Processing, pages 1765 - 1773.

Xintian Yang, Amol Ghoting, Yiye Ruan, and Srinivasan Parthasarathy. 2012. A framework for summarizing and analysing twitter feeds. In Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '12). ACM, New York, NY, USA.

Classifying Tweet Level Judgements of Rumours in Social Media

Michal Lukasik,[1] **Trevor Cohn**[2] and **Kalina Bontcheva**[1]
[1]Computer Science
The University of Sheffield
{m.lukasik,k.bontcheva}@shef.ac.uk
[2]Computing and Information Systems
The University of Melbourne
t.cohn@unimelb.edu.au

Abstract

Social media is a rich source of rumours and corresponding community reactions. Rumours reflect different characteristics, some shared and some individual. We formulate the problem of classifying tweet level judgements of rumours as a supervised learning task. Both supervised and unsupervised domain adaptation are considered, in which tweets from a rumour are classified on the basis of other annotated rumours. We demonstrate how multi-task learning helps achieve good results on rumours from the 2011 England riots.

1 Introduction

There is an increasing need to interpret and act upon rumours spreading quickly through social media, especially in circumstances where their veracity is hard to establish. For instance, during an earthquake in Chile rumours spread through Twitter that a volcano had become active and that there was a tsunami warning in Valparaiso (Mendoza et al., 2010). Other examples, from the riots in England in 2011, were that rioters were going to attack Birmingham's children hospital and that animals had escaped from the zoo (Procter et al., 2013).

Social scientists (Procter et al., 2013) analysed manually a sample of tweets expressing different judgements towards rumours and categorised them manually in supporting, denying or questioning. The goal here is to carry out tweet-level judgement classification automatically, in order to assist in (near) real-time rumour monitoring by journalists and authorities (Procter et al., 2013). In addition, information about tweet-level judgements has been used as a first step for early rumour detection by (Zhao et al., 2015).

The focus here is on tweet-level judgement classification on unseen rumours, based on a training set of other already annotated rumours. Previous work on this problem either considered unrealistic settings ignoring temporal ordering and rumour identities (Qazvinian et al., 2011) or proposed regular expressions as a solution (Zhao et al., 2015). We expect posts expressing similar opinions to exhibit many similar characteristics across different rumours. Based on the assumption of a common underlying linguistic signal, we build a transfer learning system that labels newly emerging rumours for which we have little or no annotated data. Results demonstrate that Gaussian Process-based multi task learning allows for significantly improved performance.

text	position
Birmingham Children's hospital has been attacked. F***ing morons. #UKRiots	support
Girlfriend has just called her ward in Birmingham Children's Hospital & there's no sign of any trouble #Birminghamriots	deny
Birmingham children's hospital guarded by police? Really? Who would target a childrens hospital #disgusting #Birminghamriots	question

Table 1: Tweets on a rumour about hospital being attacked during 2011 England Riots.

The novel contributions of this paper are: 1. Formulating the problem of classifying judgements of rumours in both supervised and unsupervised domain adaptation settings. 2. Showing how a multi-task learning approach outperforms single-task methods.

2 Related work

In the context of rumour spread in social media, researchers have studied differences in informa-

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 2590–2595,
Lisbon, Portugal, 17-21 September 2015.

tion flows between content of varying credibility. For instance, Procter et al. (2013) grouped source tweets and re-tweets into information flows (Lotan et al., 2011), then ranked these by flow size, as a proxy of significance. Information flows were then categorised manually. Along similar vein, Mendoza et al. (2010) found that users deal with true and false rumours differently: the former are affirmed more than 90% of the time, whereas the latter are challenged (questioned or denied) 50% of the time. Friggeri et al. (2014) analyzed a set rumours from the *Snopes.com* website that have been matched to Facebook public conversations. They concluded that false rumours are more likely to receive a comment with link to *Snopes.com* website. However, none of the above attempted to automatically classify rumours.

With respect to automatic methods for detecting misinformation and disinformation in social media, Ratkiewicz et al. (2011) detect political abuse (a kind of disinformation) spread through Twitter. The task is defined in purely information diffusion settings and is not necessarily related with the truthfulness of the piece of information. Castillo et al. (2013) proposed methods for identifying newsworthy information cascades on Twitter and then classifying these cascades as credible and not credible. The main difference from our task is that credibility classification is carried out over the entire information cascade, classified objects are not necessarily rumours and no explicit judgement classification was performed in their approach.

Early rumour identification is the focus of Zhao et al. (2015), where regular expressions are used for finding questioning and denying tweets as a key pre-requisite step for rumour detection. Unfortunately, when we applied these regular expressions on our dataset, they yielded only 16% recall for questioning and 14% recall for denying tweets. Consequently, this motivated us to seek a better approach to tweet-level classification.

The work most relevant to ours is due to Qazvinian et al. (2011). Their method first carries out rumour retrieval, whereby tweets are classified into rumour related and non-rumour related. Next, rumour-related tweets are classified into supporting and not-supporting. The classifier is trained by ignoring rumour identities, i.e., pooling together tweets from all rumours, and ignoring the temporal dependencies between tweets. In contrast, we formulate the rumour classification problem as transfer learning, where unseen rumours (or rumours with few initial tweets observed) are classified using already known rumours – a much harder and more practical setting. Moreover, unlike Qazvinian et al. (2011), we consider the multi-class classification problem and do not collaps questioning and denying tweets into a single class, since they differ significantly.

Rumour	Supporting	Denying	Questioning
army bank	62	42	73
hospital	796	487	132
London Eye	177	295	160
McDonald's	177	0	13
Miss Selfridge's	3150	0	7
police beat girl	783	4	95
zoo	616	129	99

Table 2: Counts of tweets with supporting, denying or questioning labels in each rumour collection.

3 Data

We evaluate our work on several rumours circulating on Twitter during the England riots in 2011 (see Table 2). The dataset was analysed and annotated manually as supporting, questioning, or denying a rumour, by a team of social scientists studying the role of social media during the riots (Procter et al., 2013). The original dataset also included commenting tweets, but these have been removed from our experiments due to their small number (they constituted only 5% of the corpus).

As can be seen from the dataset overview in Table 2, different rumours exhibit varying proportions of supporting, denying and questioning tweets, which was also observed in other studies of rumours (Mendoza et al., 2010; Qazvinian et al., 2011). These variations in majority classes across rumours underscores the modeling challenge in tweet-level classification of rumour attitudes.

With respect to veracity, one rumour has been confirmed as true (Miss Selfridge's being on fire), one is unsubstantiated (police beat girl), and the remaining five are known to be false. Note, however, that the focus here is not on classifying truthfulness, but instead on identifying the attitude expressed in each tweet towards the rumour.

4 Problem formulation

Let R be a set of rumours, each of which consists of tweets discussing it, $\forall_{r \in R}\, T_r = \{t^r_1, \cdots, t^r_{r_n}\}$. $T = \cup_{r \in R} T_r$ is the complete set of tweets from all rumours. Each tweet is classified as supporting, denying or questioning with respect to its rumour: $y(t) \in \{0, 1, 2\}$, where 0 denotes supporting, 1 means denying and 2 denotes questioning.

First, we consider the Leave One Out (LOO) setting, which means that for each rumour $r \in R$, we construct the test set equal to T_r and the training set equal to $T \setminus T_r$. Therefore this is a very challenging and realistic scenario, where the test set contains an entirely unseen rumour, from those in the training set.

The second setting is Leave Part Out (LPO). In this formulation, a very small number of initial tweets from the target rumour is added to the training set $\{t^r_1, \cdots, t^r_{r_k}\}$. This scenario becomes applicable typically soon after a rumour breaks out and journalists have started monitoring and analysing the related tweet stream. The experiments section investigates how the number of initial training tweets influences classification performance on a fixed test set, namely: $\{t^r_{r_l}, \cdots, t^r_{r_n}\}$, $l > k$.

The tweet-level classification problem here assumes that tweets from the training set are already labelled with the rumour discussed and the attitude expressed towards that. This information can be acquired either via manual annotation as part of expert analysis, as is the case with our dataset, or automatically, e.g. using pattern-based rumour detection (Zhao et al., 2015). Afterwards, our method can be used to classify the attitudes expressed in each new tweet from outside the training set.

5 Gaussian Processes for Classification

Gaussian Processes are a Bayesian non-parametric machine learning framework that has been shown to work well for a range of NLP problems, often beating other state-of-the-art methods (Cohn and Specia, 2013; Lampos et al., 2014; Beck et al., 2014; Preotiuc-Pietro et al., 2015). We use Gaussian Processes as this probabilistic kernelised framework avoids the need for expensive cross-validation for hyperparameter selection.[1]

[1]There exist frequentist kernel methods, like SVMs, which additionally require extensive heldout parameter tuning.

The central concept of Gaussian Process Classification (GPC; (Rasmussen and Williams, 2005)) is a latent function f over inputs $\mathbf{x}$: $f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}'))$, where m is the mean function, assumed to be 0 and k is the kernel function, specifying the degree to which the outputs covary as a function of the inputs. We use a linear kernel, $k(\mathbf{x}, \mathbf{x}') = \sigma^2 \mathbf{x}^\top \mathbf{x}'$. The latent function is then mapped by the probit function $\Phi(f)$ into the range $[0, 1]$, such that the resulting value can be interpreted as $p(y = 1|\mathbf{x})$.

The GPC posterior is calculated as

$$p(f^*|X, \mathbf{y}, \mathbf{x}_*) = \int p(f^*|X, \mathbf{x}_*, \mathbf{f}) \frac{p(\mathbf{y}|\mathbf{f})p(\mathbf{f})}{p(\mathbf{y}|X)} d\mathbf{f},$$

where $p(\mathbf{y}|\mathbf{f}) = \prod_{j=1}^{n} \Phi(f_j)^{y_j}(1 - \Phi(f_j))^{1-y_j}$ is the Bernoulli likelihood of class y. After calculating the above posterior from the training data, this is used in prediction, i.e.,

$$p(y_* = 1|X, \mathbf{y}, \mathbf{x}_*) = \int \Phi(f_*)\, p(f_*|X, \mathbf{y}, \mathbf{x}_*)\, df_*\,.$$

The above integrals are intractable and approximation techniques are required to solve them. There exist various methods to deal with calculating the posterior; here we use Expectation Propagation (EP; (Minka and Lafferty, 2002)). In EP, the posterior is approximated by a fully factorised distribution, where each component is assumed to be an unnormalised Gaussian.

In order to conduct multi-class classification, we perform a one-vs-all classification for each label and then assign the one with the highest likelihood, amongst the three (supporting, denying, questioning). We choose this method due to interpretability of results, similar to recent work on occupational class classification (Preotiuc-Pietro et al., 2015).

Intrinsic Coregionalization Model In the LPO setting initial labelled tweets from the target rumour are observed as well. In this case, we propose to weight the importance of tweets from the reference rumours depending on how similar their characteristics are to the tweets from the target rumour available for training. To handle this with GPC, we use a multiple output model based on the Intrinsic Coregionalisation Model (ICM; (Álvarez et al., 2012)). It has already been applied successfully to NLP regression problems (Beck et al., 2014) and it can also be applied to classification

ones. ICM parametrizes the kernel by a matrix which represents the extent of covariance between pairs of tasks. The complete kernel takes form of

$$k((\mathbf{x}, d), (\mathbf{x}', d')) = k_{data}(\mathbf{x}, \mathbf{x}')B_{d,d'} ,$$

where B is a square coregionalization matrix, d and d' denote the tasks of the two inputs and k_{data} is a kernel for comparing inputs $\mathbf{x}$ and $\mathbf{x}'$ (here, linear). We parametrize the coregionalization matrix $B = \boldsymbol{\kappa} I + \boldsymbol{v}\boldsymbol{v}^T$, where $\boldsymbol{v}$ specifies the correlation between tasks and the vector κ controls extent of task independence.

Hyperparameter selection We tune hyperparameters $\mathbf{v}$, $\boldsymbol{\kappa}$ and σ^2 by maximizing evidence of the model $p(\mathbf{y}|X)$, thus having no need for a validation set.

Methods We consider GPs in three different settings, varying in what data the model is trained on and what kernel it uses. The first setting (denoted GP) considers only target rumour data for training. The second (GPPooled) additionally considers tweets from reference rumours (i.e. other than the target rumour). The third setting is GPICM, where an ICM kernel is used to weight influence from tweets from reference rumours.

6 Features

We conducted a series of preprocessing steps in order to address data sparsity. All words were lowercased; stopwords removed; all emoticons were replaced with words[2]; and stemming was performed. In addition, multiple occurrences of a character were replaced with a double occurrence (Agarwal et al., 2011), to correct for misspellings and lengthenings, e.g., *looool*. All punctuation was also removed, except for ., *!* and *?*, which we hypothesize to be important for expressing emotion. Lastly, usernames were removed as they tend to be rumour-specific, i.e., very few users comment on more than one rumour.

After preprocessing the text data, we use either the resulting bag of words (BOW) feature representation or replace all words with their Brown cluster ids (Brown), using 1000 clusters acquired from a large scale Twitter corpus (Owoputi et al., 2013). In all cases, simple re-tweets are removed from the training set to prevent bias (Llewellyn et al., 2014).

[2]We used the dictionary from: `http://bit.ly/1rX1Hdk` and extended it with: :o, : |, =/, :s, :S, :p.

method	acc
Majority	0.68
GPPooled Brown	0.72
GPPooled BOW	0.69

Table 3: Accuracy taken across all rumours in the LOO setting.

7 Experiments and Discussion

Table 3 shows the mean accuracy in the LOO scenario following the GPPooled method, which pools all reference rumours together ignoring their task identities. ICM can not use correlations to target rumour in this case and so can not be used. The majority baseline simply assigns the most frequent class from the training set.

We can observe that methods perform on a level similar to majority vote, outperforming it only slightly. This indicates how difficult the LOO task is, when no annotated target rumour tweets are available.

Figure 1 shows accuracy for a range of methods as the number of tweets about the target rumour used for training increases. Most notably, performance increases from 70% to around 80%, after only 10 annotated tweets from the target rumour become available, as compared to the results on unseen rumours from Table 3. However, as the amount of target rumour increases, performance does not increase further, which suggests that even only 10 human-annotated tweets are enough to achieve significant performance benefits. Note also how the use of reference rumours is very important, as methods using only the target rumour obtain accuracy similar to the Majority vote classifier (GP Brown and GP BOW).

The top performing methods are GPCIM and GPPooled, where use of Brown clusters consistently improves results for both methods over BOW, irrespective of the number of tweets about the target rumour annotated for training. Moreover, GPICM is better than GPPooled both with Brown and BOW features and GPCIM with Brown is ultimately the best performing of all.

In order to analyse the importance of Brown clusters, Automatic Relevance Determination (ARD) is used (Rasmussen and Williams, 2005) for the best performing GPICM Brown in the LPO scenario. Only the case where the first 10 tweets are used for training is considered, since it already

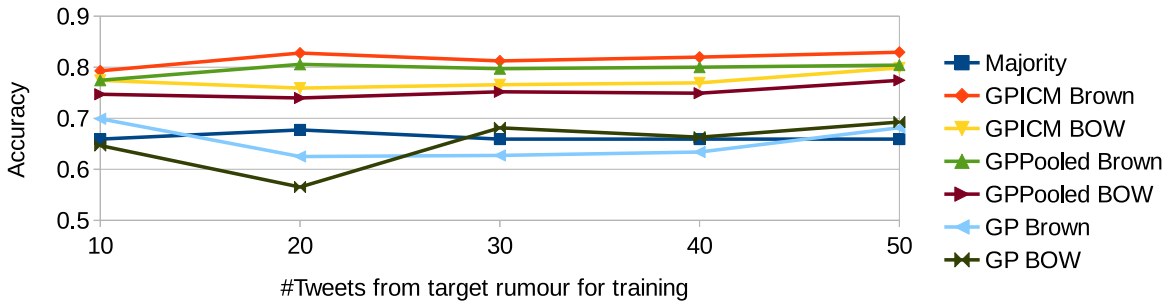


Figure 1: Accuracy measures for different methods versus the size of the target rumour used for training in the LPO setting. The test set is fixed to all but the first 50 tweets of the target rumour.

supporting	denying	questioning
?	fake	?
10001101	11111000001	10001101
!	not	!
10001100	001000	10001100
not	?	hope
001000	10001101	01000111110
fake	!	true
11111000001	10001100	111110010110
true	bullshit	searching
111110010110	11110101011111	01111000010

Table 4: Top 5 Brown clusters, each shown with a representative word. For further details please see the cluster definitions at `http://www.ark.cs.cmu.edu/TweetNLP/cluster_viewer.html`.

performs very well. Using ARD, we learn a separate length-scale for each feature, thus establishing their importance. The weights learnt for different clusters are averaged over the 7 rumours and the top 5 Brown clusters for each label are shown in Table 4. We can see that clusters around the words *fake* and *bullshit* turn out to be important for the denying class, and *true* for both supporting and questioning classes. This reinforces our hypothesis that common linguistic cues can be found across multiple rumours. Note how punctuation proves important as well, since clusters *?* and *!* are also very prominent.

8 Conclusions

This paper investigated the problem of classifying judgements expressed in tweets about rumours. First, we considered a setting where no training data from target rumour is available (LOO). Without access to annotated examples of the target rumour the learning problem becomes very difficult. We showed that in the supervised domain adaptation setting (LPO) even annotating a small number of tweets helps to achieve better results. Moreover, we demonstrated the benefits of a multi task learning approach, as well as that Brown cluster features are more useful for the task than simple bag of words.

Judgement estimation is undoubtedly of great value e.g. for marketing, politics and journalism, helping to target widely believed topics. Although the focus here is on classifying community reactions, Castillo et al. (2013) showed that community reaction is correlated with actual rumour veracity. Consequently our classification methods may prove useful in the broader and more challenging task of annotating veracity.

An interesting direction for future work would be adding non-textual features. For example, the rumour diffusion pattern (Lukasik et al., 2015) may be a useful cue for judgement classification.

Acknowledgments

Work partially supported by the European Union under grant agreement No. 611233 PHEME. The work was implemented using the GPy toolkit (GPy authors, 2015).

References

Apoorv Agarwal, Boyi Xie, Ilia Vovsha, Owen Rambow, and Rebecca Passonneau. 2011. Sentiment analysis of twitter data. In *Proceedings of the Workshop on Languages in Social Media*, LSM '11, pages 30–38.

Mauricio A. Álvarez, Lorenzo Rosasco, and Neil D. Lawrence. 2012. Kernels for vector-valued functions: A review. *Found. Trends Mach. Learn.*, 4(3):195–266.

Daniel Beck, Trevor Cohn, and Lucia Specia. 2014. Joint emotion analysis via multi-task Gaussian processes. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '14, pages 1798–1803.

Carlos Castillo, Marcelo Mendoza, and Barbara Poblete. 2013. Predicting information credibility in time-sensitive social media. *Internet Research*, 23(5):560–588.

Trevor Cohn and Lucia Specia. 2013. Modelling annotator bias with multi-task Gaussian processes: An application to machine translation quality estimation. In *51st Annual Meeting of the Association for Computational Linguistics*, ACL '13, pages 32–42.

Adrien Friggeri, Lada Adamic, Dean Eckles, and Justin Cheng. 2014. Rumor cascades. In *International AAAI Conference on Weblogs and Social Media*.

The GPy authors. 2015. GPy: A Gaussian process framework in Python. `http://github.com/SheffieldML/GPy`.

Vasileios Lampos, Nikolaos Aletras, Daniel Preotiuc-Pietro, and Trevor Cohn. 2014. Predicting and characterising user impact on twitter. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, EACL'14, pages 405–413.

Clare Llewellyn, Claire Grover, Jon Oberlander, and Ewan Klein. 2014. Re-using an argument corpus to aid in the curation of social media collections. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation*, LREC'14, pages 462–468.

Gilad Lotan, Erhardt Graeff, Mike Ananny, Devin Gaffney, Ian Pearce, and danah boyd. 2011. The Arab spring— the revolutions were tweeted: Information flows during the 2011 Tunisian and Egyptian revolutions. *International Journal of Communication*, 5(0).

Michal Lukasik, Trevor Cohn, and Kalina Bontcheva. 2015. Point process modelling of rumour dynamics in social media. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015*, pages 518–523.

Marcelo Mendoza, Barbara Poblete, and Carlos Castillo. 2010. Twitter under crisis: Can we trust what we RT? In *1st Workshop on Social Media Analytics*, SOMA'10, pages 71–79.

Thomas Minka and John Lafferty. 2002. Expectation-propagation for the generative aspect model. In *Proceedings of the Eighteenth Conference on Uncertainty in Artificial Intelligence*, UAI'02, pages 352–359.

Olutobi Owoputi, Chris Dyer, Kevin Gimpel, Nathan Schneider, and Noah A. Smith. 2013. Improved part-of-speech tagging for online conversational text with word clusters. In *Proceedings of NAACL*, pages 380–390.

Daniel Preotiuc-Pietro, Vasileios Lampos, and Nikolaos Aletras. 2015. An analysis of the user occupational class through twitter content. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015*, pages 1754–1764.

Rob Procter, Jeremy Crump, Susanne Karstedt, Alex Voss, and Marta Cantijoch. 2013. Reading the riots: What were the police doing on twitter? *Policing and society*, 23(4):413–436.

Vahed Qazvinian, Emily Rosengren, Dragomir R. Radev, and Qiaozhu Mei. 2011. Rumor has it: Identifying misinformation in microblogs. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 1589–1599.

Carl Edward Rasmussen and Christopher K. I. Williams. 2005. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press.

Jacob Ratkiewicz, Michael Conover, Mark Meiss, Bruno Gonalves, Alessandro Flammini, and Filippo Menczer. 2011. Detecting and tracking political abuse in social media. In *5th International AAAI Conference on Weblogs and Social Media*, ICWSM'11.

Zhe Zhao, Paul Resnick, and Qiaozhu Mei. 2015. Early detection of rumors in social media from enquiry posts. In *International World Wide Web Conference Committee (IW3C2)*.

Identification and Verification of Simple Claims about Statistical Properties

Andreas Vlachos and **Sebastian Riedel**
Department of Computer Science
University College London
{a.vlachos,s.riedel}@cs.ucl.ac.uk

Abstract

In this paper we study the identification and verification of simple claims about statistical properties, e.g. claims about the *population* or the *inflation rate* of a country. We show that this problem is similar to extracting numerical information from text and following recent work, instead of annotating data for each property of interest in order to learn supervised models, we develop a distantly supervised baseline approach using a knowledge base and raw text. In experiments on 16 statistical properties about countries from Freebase we show that our approach identifies simple statistical claims about properties with 60% precision, while it is able to verify these claims without requiring any explicit supervision for either tasks. Furthermore, we evaluate our approach as a statistical property extractor and we show it achieves 0.11 mean absolute percentage error.

1 Introduction

Statistical properties are commonly used to describe entities, e.g. *population* for countries, *net_value* for companies, *points_scored* for athletes, etc. Claims about such properties are very common in news articles and social media, however they can be erroneous, either due to author error or negligence at the time of writing or because they eventually become out of date. While manual verification (also referred to as fact-checking) is conducted by journalists in news organizations and dedicated websites such as `www.emergent.info`, the volume of the claims calls for automated approaches, which is one of the main objectives of computational journalism (Cohen et al., 2011; Flew et al., 2012).

In this paper we develop a baseline approach to identify and verify simple claims about statistical properties against a database. The task is illustrated in Figure 1. Given a sentence, we first identify whether it contains a claim about a property we are interested in (`population` in the example), which entity it is about and the value claimed (Lesotho and 2,000,000 respectively). We then proceed to verify the value claimed in text for the property of this entity against the value known in a knowledge base such as Freebase and return a score reflecting the accuracy of the claim (absolute percentage error in the example).

Text: Lesotho, a landlocked enclave of South Africa, has a population of nearly 2 million and covers an area slightly smaller than the U.S. state of Maryland.
Entity: Lesotho **Property:** `population` **Value claimed in text:** 2,000,000
Value in knowledge base: 2,193,843 **Absolute percentage error:** 0.09

Figure 1: Claim identification and verification.

Claim identification is essentially an instance of information extraction. While it would be possible to develop supervised models, this would require expensive manual data annotation for each property of interest. Instead, we follow the distant supervision paradigm (Craven and Kumlien, 1999; Mintz et al., 2009) using supervision obtained by combining triples from a knowledge base and raw text. However, statistical properties are more challenging in applying the distant supervision assumption than relations between named entities due to the fact that the numerical values are often approximated in text, as in the example of Figure 1. Consequently, linking the values mentioned in text with those in the knowledge base is not trivial and thus it is not straightforward to generate training instances for the property of interest.

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 2596–2601,
Lisbon, Portugal, 17-21 September 2015.

To address this issue, we propose a distantly supervised claim identification approach that relies on approximate instead of exact matching between values in text and the knowledge base. In experiments on 16 statistical properties about countries from Freebase we show that our approach identifies simple statistical claims with 60% precision, while it is able to verify these claims without requiring any explicit supervision for this task. In developing our approach, we also evaluate it as a statistical property extractor achieving 0.11 mean absolute percentage error. The code and the datasets developed are publicly available from `https://github.com/uclmr/simpleNumericalFactChecker`.

2 Claim identification algorithm

Our approach to claim identification relies on discovering textual patterns between an entity and a numerical value used to express the property of interest. For example, the first, second and fourth patterns in Table 1 express the *population* property, and we would like our approach to select them to identify claims about this property.

During training, we assume as input a set of entity-value pairs from the knowledge base for the property of interest (top-right part of Table 1) and a set of textual patterns (bottom-left part), each associated with entity-value pairs (bottom-right part). The patterns and the entity-value pairs associated with them are obtained by processing raw text, which we discuss in Section 3.

The key difficulty compared to other applications of distant supervision is that numerical values are often approximated in text. Thus, instead of looking for patterns that report the exact values for each entity, we develop an approach for finding the patterns that predict the values well. Intuitively, the algorithm ranks all text patterns according to how well they predict the values for the property at question and then greedily selects them till the accuracy of the aggregate predictions by the selected paterns stop improving. To compare the predicted entity-value pairs $\hat{EV}$ against the property entity-value pairs EV_{prop} we use the mean absolute percentage error (MAPE):

$$MAPE(EV_{prop}, \hat{EV}) = \frac{1}{|E|} \sum_{e \in E'} \frac{|v_e - \hat{v_e}|}{|v_e|} \quad (1)$$

Note that only the values predicted for entities in both EV_{prop} and $\hat{EV}$ (denoted by E') are taken into account in this equation, thus in calculating MAPE for the pattern "X has _ inhabitants" from Table 1 against the entity-values for *population* only the two values present are considered. MAPE is commonly used in measuring forecasting accuracy of algorithms in finance (Hyndman and Koehler, 2006). Unlike mean absolute error or mean squared error it adjusts the errors according to the magnitude of the correct value.

Initially (line 1) the algorithm decides on a default value (v_{def}) to return for the property at question among three options: the mean of the training values, their median or zero. The criterion for the choice is which one results in a better MAPE score on the training data. We refer to this prediction as the `InformedGuess`. This default value is used when predicting (lines 16-29) in case there are no values for an entity in the patterns selected, e.g. if only the pattern "X has _ inhabitants" is selected and the prediction for Iceland is requested.

Following this, the patterns are ranked according to the MAPE score of their entity values with respect to the entity values of the property at question (lines 2-4). We then iterate over the patterns in the order they were ranked. For every pattern, we add it to the set of patterns used in predicting (lines 8-9), and evaluate the resulting predictions using MAPE with respect to the training values (line 10). If MAPE is increased (predictions become worse), then we remove the newly added pattern from the set and stop. Otherwise, we continue with the next pattern in the queue.

In experiments with this algorithm we found that while it often identified useful patterns, sometimes it was misled by patterns that had very few entities in common with the property and the values of those entities happen to be similar to those of the property. For example, the pattern "_ tourists visited X" in Figure 1 has only one entity-value pair ("France:68,000,000") and the value is very close to the *population* value for "France". To ameliorate this issue, we adjusted the MAPE scores used in the ranking step (line 4) according to the number of values used in the calculation using the following formula:

$$\text{adjusted}MAPE = \frac{c}{c+N} MAPE \quad (2)$$

where N is the number of values used in calculating MAPE in Equation 1 and c is a parameter that regulates the adjustment. Lower c puts more importance on the number of values used,

population	France:66,028,467, Russia:143,700,000, Iceland:325,600
the population of X is _	France:66,000,000, Russia:140,000,000, Iceland:325,000
X's population is estimated at _	France:66,030,000, Russia:145,000,000
X's inflation rate is _	France:0.9, Iceland:4.0
X has _ inhabitants	Russia:140,000,000, Iceland:300,000
_ tourists visited X	France:68,000,000

Table 1: Property and text patterns associated with entity-value pairs.

Algorithm 1: Claim identification algorithm

Input: Entity-values for property $EV_{prop} = \{(e_1, v_1), (e_2, v_2), \ldots\}$, patterns $P = \{p1, p2, \ldots\}$, entity-values for pattern p: EV_p

Output: Selected patterns P_{sel}

v_{def} = `InformedGuess` (EV_{prop})
priorityQueue $Q = \emptyset$
foreach *pattern* $p \in P$ **do**
    `push` (Q, (p, `MAPE` (EV_{prop}, EV_p)))
$P_{sel} = \emptyset$
mp = `MAPE` (EV_{prop}, `predict` (E, P_{sel}))
while $Q \neq \emptyset$ **do**
    pattern p = `pop` (Q)
    P'_{sel} = $P_{sel} \cup \{p\}$
    mp' = `MAPE` (EV_{prop}, `predict` (E, P'_{sel}))
    if $mp' > mp$ **then**
        break
    else
        $mp = mp'$
        $P_{sel} = P'_{sel}$
function `predict`**(entities** E**, patterns** P_{sel}**)**
    $\hat{EV} = \emptyset$
    foreach $e \in E$ **do**
        $sum = 0$
        $count = 0$
        foreach $p \in P_{sel}$ **do**
            if $(e, v) \in EV_p$ **then**
                $sum+ = p\{e\}$
                $count+ = 1$
        if $count > 0$ **then**
            $\hat{EV} = \hat{EV} \cup (e, sum/count)$
        else
            $\hat{EV} = \hat{EV} \cup (e, v_{def})$
    return $\hat{EV}$

thus leading the algorithm to choosing patterns assessed with more values, and thus more reliably.

3 Data collection

To evaluate the claim identification approach developed we compiled a dataset of statistical properties from Freebase. We downloaded a snapshot[1] of all instances of the *statistical_region* entity type with all their properties with their most recent values, keeping only those were from 2010 onwards. From those we selected the 16 properties listed in Table 2, each property having values for 150-175 regions (mostly countries).

To collect texts from which the text patterns between entities and numerical values will be extracted we downloaded documents from the web. In particular, for each region combined with each property we formed a query consisting of the two and submitted it to Bing via its Search API. Following this we obtained the top 50 results for each query, downloaded the HTML pages corresponding to each result and extracted their textual content with BoilerPipe (Kohlschütter et al., 2010). We then processed the texts using the Stanford CoreNLP toolkit (Manning et al., 2014) and from each sentence we extracted textual patterns between all the named entities recognized as locations and all the numerical values. Two kinds of patterns were extracted for each location and numerical value: surface patterns (as the ones shown in Table 1) and lexicalized dependency paths.

This pattern extraction process resulted in a large set of triples consisting of a region, a pattern and a value. Different sentences might result in triples containing the same region and textual pattern but different value. Such variation can arise due to either the approximations of values in text or due to the pattern being highly ambiguous, e.g. "X is _". We distinguish between the two by requiring each region-pattern combination to have appeared at least twice and its values to have stan-

[1] Data was collected in May 2014.

dard deviation less than 0.1. In this case, then the region-pattern value is set to the mean of the values it is encountered with, otherwise is removed.

4 Information extraction experiments

We first evaluate our approach as a statistical property extractor for two reasons. First, while our main goal is to develop a claim identification approach, there is no data for this task to evaluate, thus making development difficult. On the other hand, we can evaluate statistical property extraction in a straightforward way, thus facilitating development and parameter tuning. Second, the algorithm described learns such an extractor, thus it is of interest to know its performance.

We split the values collected from Freebase into 2/3 for training and 1/3 for testing, ensuring that all regions are present in both datasets. The accuracy is evaluated using MAPE. When using adjusted MAPE we set the parameter c for each property using 4-fold cross-validation.

The performance of Algorithm 1 using the unadjusted MAPE was 0.72 averaged over all properties. Using the adjusted version this was greatly improved to 0.49. We also evaluated the `InformedGuess` prediction which returns the same value for all regions (it chooses the value that performs best among the mean, the median and 0), and its overall MAPE was 0.79. Recalling that Algorithm 1 returns the `InformedGuess` in case no pattern is found for an entity, we also evaluate the performance without returning a value for such entities, thus ignoring them in the evaluation. In that case the performance with unadjusted MAPE improves to 0.17 but 10% coverage, while with adjusted MAPE it improves to 0.11 with 43% coverage. Best performances were achieved for relations such as *population* which have a wide range of values that is well separated from the rest, while percentage rates were usually harder for the opposite reason. Thus we conclude that the algorithm with adjusted MAPE selects better patterns for each property that are encountered more frequently, which is important for the main goal of this paper, claim identification.

5 Claim identification and verification

We now evaluate our approach on claim identification. For each property, we run Algorithm 1 using adjusted MAPE and the parameter c as chosen in the experiments of the previous section to select patterns expressing it. We then process all texts and if a sentence contains one of the selected patterns between an entity and a value, it is returned for manual inspection as shown in Figure 1.

Freebase property	claims	precision
consumer_price_index	116	0.93
cpi_inflation_rate	464	0.92
diesel_price_liter	212	1.00
fertility_rate	307	0.99
gdp_growth_rate	39	0.31
gdp_nominal	308	0.98
gdp_nominal_per_capita	415	0.20
gni	413	0.62
gni_per_capita	795	0.49
health_expenditure	197	0.99
internet_users_%	93	0.00
life_expectancy	581	0.45
population	1583	0.9
population_growth_rate	1377	0.11
renewable_freshwater	105	1.00
undernourishment	87	0.13
OVERALL	7092	0.60

Table 2: Claim identification results.

The claims returned were labeled by the authors of the paper as correctly or incorrectly identified according to the following guidelines. A claim is extracted correctly only if both the entity and the value are extracted correctly and the sentence expresses the property at question. E.g. a claim identified in a sentence containing a country and its correct GDP growth rate without stating it as such (the same percentage rate can be true for multiple statistical properties) is considered incorrect. Furthermore, we considered claims referring to past measurements (e.g. results of a past census) to be correctly identified.

Results for each property are shown in Table. 2. Overall precision was 60% over 7,092 statements, and it varied substantially across properties. Perfect precision was found for claims of *renewable_freshwater* for which one textual pattern was responsible for all the claims identified and it was correct. On the other hand, the zero precision for claims of *internet_user_%* was due to identifying correctly sentences listing countries and their respective values for this property but not identifying the country-value pairs correctly. More representative of properties with precise claim identifi-

cation was *population*, for which the relatively few errors were due to the patterns learned not being able to distinguish between different types of population e.g. general vs working population. On the other hand, the claims for *gni_per_capita* had low accuracy because they were confused with those of *gdp_nominal_per_capita*, as their values tend to be relatively close. The claims identified and annotated manually are attached to our submission. Finally, some errors are due to the algorithm being constrained to extract a claim considering only the text pattern between the entity and the value, thus ignoring parts of the sentence that might be relevant. For example, the pattern "the population of X is _" is generally reliable, but in the sentence "The population of Tajikistan is 90 % Muslim" it extracts a claim incorrectly.

The verification stage of the simple claims we extract is rather simple; we just score the claims according to the absolute percentage error of the value claimed in text with respect to the value in known in Freebase. In the process of labeling the claims identified we did a qualitative analysis of the claims with high error. We found cases where our algorithm flagged cases of out of date estimates of populations used, e.g. the webpage `http://www.economywatch.com/world_economy/bolivia`[2] states that the population of Bolivia is 9 million, while it is estimated to be above 10 million.

6 Discussion - Related work

As explained, we tackle claim identification as an instance of information extraction, and propose a baseline able to perform both tasks. However, it is important to distinguish between them. In claim identification we are interested in all claims about a property, even inaccurate ones; in information extraction on the other hand, and especially its formulation as knowledge base population, we are interested in the accurate claims only, since extracting inaccurate ones will lead to erroneous information added to the knowledge base. The difference between the two tasks is captured by the verification task. In this paper our main goals are identification and verification, but we train our approach on information extraction, relying on the assumption that most claims made in the texts retrieved via the web search engine are accurate.

In related work, Nakashole and Mitchell (2014) developed an approach to verify subject-verb-object triples against a knowledge base, taking into account the objectivity of the language used in the sources stating the triple. Our approach is agnostic to the syntactic form of the claims, thus it can identify claims expressed in greater linguistic variety. Ciampaglia et al. (2015) fact-checked subject-predicate-object triples against a knowledge graph constructed from DBpedia, but they considered only the paths between the subject and the predicate in their algorithm thus ignoring the predicate itself. Dong et al. (2015) established the trustworthiness of a web source by comparing the subject-predicate-object triples extracted from it to the Knowledge Vault built by Google, but did not focus on claim identification and verification. Adar et al. (2009) developed an approach to detect inconsistencies between versions of Wikipedia in different languages, but they focused on manually extracted infoboxes. Finally, Vlachos and Riedel (2014) compiled a dataset of claims fact-checked by journalists, but the claims are much more complex than the ones we considered in this paper.

Other work that discussed the extraction of statistical properties includes the approaches of Hoffmann et al. (2010) and Intxaurrondo et al. (2015), both employing approximate matching to deal with the approximation of numerical values in text. In order to learn their model, Hoffmann et al. (2010) take advantage of the structure of the articles in Wikipedia developing a classifier that identifies the schema followed by each article, which is not straightforward to extend to texts beyond this source. Intxaurrondo et al. (2015) on the other hand focus on tweets and make the assumption that the entity discussed in each tweet is determined in advance, thus the extractor needs only to associate a numerical value with the property of interest, i.e. the task is reduced from triple extraction to labeling values.

7 Conclusions - Future work

In this paper we developed a distantly supervised approach for identification and verification of simple statistical claims. We evaluated both as statistical property extractor and as a claim identifier on 16 relations from Freebase. In future work we aim to improve our approach by taking into account continuous representations of the words in the patterns and to extend it to more complex claims, e.g. claims about change in financial indicators.

[2] Accessed in August 2015.

References

Eytan Adar, Michael Skinner, and Daniel S. Weld. 2009. Information arbitrage across multi-lingual wikipedia. In *Proceedings of the Second ACM International Conference on Web Search and Data Mining*, pages 94–103.

Giovanni Luca Ciampaglia, Prashant Shiralkar, Luis M. Rocha, Johan Bollen, Filippo Menczer, and Alessandro Flammini. 2015. Computational fact checking from knowledge networks. *PLoS ONE*, 10(6):e0128193, 06.

Sarah Cohen, Chengkai Li, Jun Yang, and Cong Yu. 2011. Computational journalism: A call to arms to database researchers. In *Proceedings of the Conference on Innovative Data Systems Research*, volume 2011, pages 148–151.

Mark Craven and Johan Kumlien. 1999. Constructing biological knowledge-bases by extracting information from text sources. In *Proceedings of the 7th International Conference on Intelligent Systems for Molecular Biology*, pages 77–86.

Xin Luna Dong, Evgeniy Gabrilovich, Kevin Murphy, Van Dang, Wilko Horn, Camillo Lugaresi, Shaohua Sun, and Wei Zhang. 2015. Knowledge-based trust: Estimating the trustworthiness of web sources. pages 938–949.

Terry Flew, Anna Daniel, and Christina L. Spurgeon. 2012. The promise of computational journalism. In *Proceedings of the Australian and New Zealand Communication Association Conference*, pages 1–19.

Raphael Hoffmann, Congle Zhang, and Daniel S. Weld. 2010. Learning 5000 relational extractors. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, page 286295.

Rob J. Hyndman and Anne B. Koehler. 2006. Another look at measures of forecast accuracy. *International Journal of Forecasting*, 22(4):679 – 688.

Ander Intxaurrondo, Eneko Agirre, Oier Lopez de Lacalle, and Mihai Surdeanu. 2015. Diamonds in the rough: Event extraction from imperfect microblog data. In *Proceedings of the 2015 Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*.

Christian Kohlschütter, Peter Fankhauser, and Wolfgang Nejdl. 2010. Boilerplate detection using shallow text features. In *Proceedings of the 3rd ACM International Conference on Web Search and Data Mining*, pages 441–450.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 1003–1011.

Ndapandula Nakashole and Tom M Mitchell. 2014. Languageaware truth assessment of fact candidates. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*.

Andreas Vlachos and Sebastian Riedel. 2014. Fact checking: Task definition and dataset construction. In *Proceedings of the ACL 2014 Workshop on Language Technology and Computational Social Science*, July.

Author Index